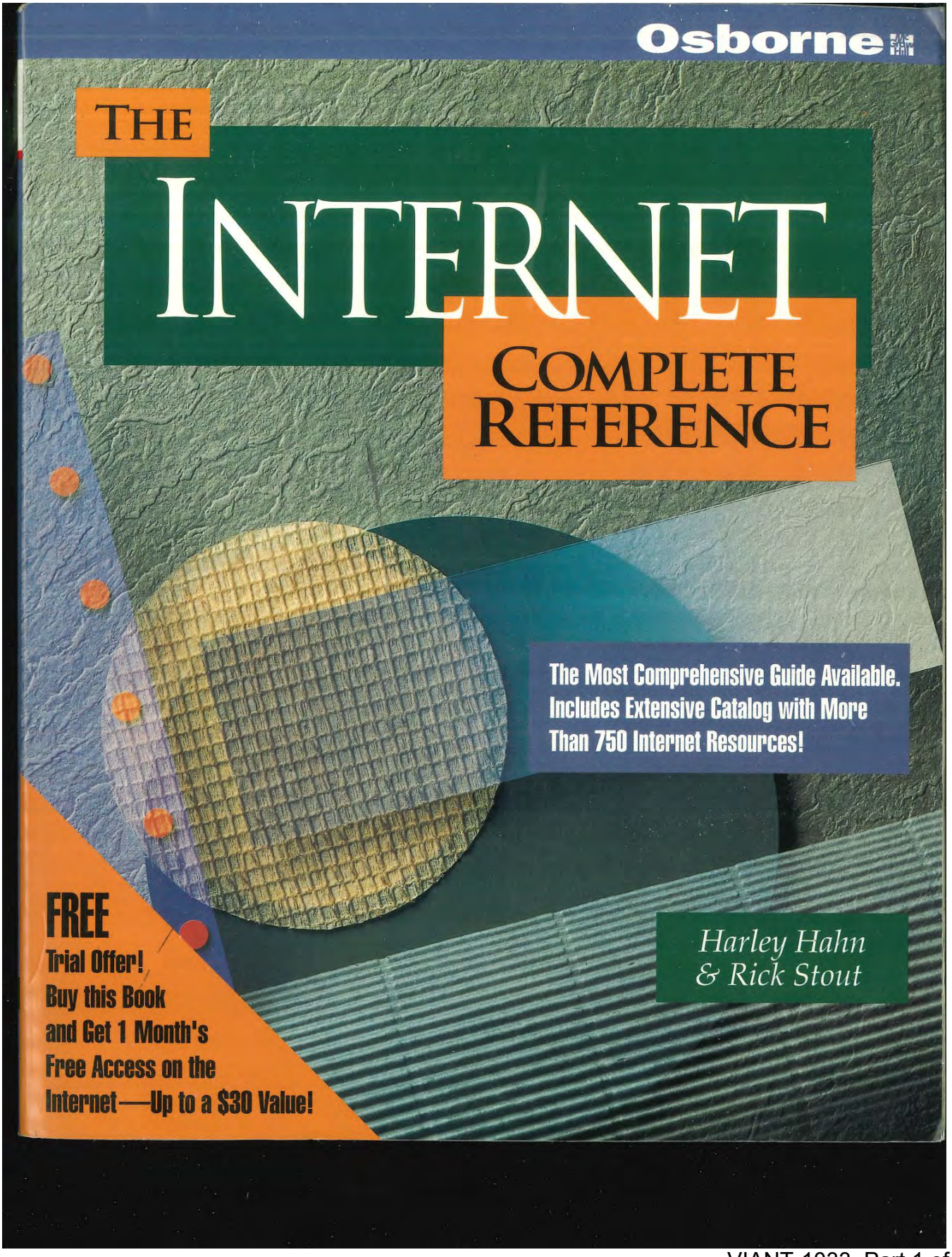


VIANT-1033, Part 1 of 3

"This is by far THE most comprehensive book on the Internet. A must for novices, even experts will learn something new from this book. The attention to detail is first-rate, yet its presentation understandable for everyone."

Scott Yanoff
Author of the "Yanoff List" of Internet services
distributed electronically throughout the Internet

"This book is exactly what I have needed for a long time! Even if you already own other Internet books, *The Internet Complete Reference* is a must-read. It is at once tremendously useful and thoroughly readable. Harley Hahn's irreverent sense of humor and clear, insightful explanations will both keep you laughing and make you an Internet expert. I give it an absolute, unqualified recommendation for anyone who uses or plans to use the Internet."

Kenn Nesbitt, Consultant
Microsoft Consulting Services, Microsoft Corporation

"This is a terrific reference. Quantifying and organizing the world of the Internet is an awesome task, and Harley Hahn has done it!"

Bob Rieger
Founder and Senior Vice President, Operations
Netcom On-Line Communications Services

"*The Internet Complete Reference* is the best Internet book ever written. Superb and authoritative, a must-read for personal and business users alike. No matter how many other Internet books you have, you NEED this book. "

Sam Albert, President
Sam Albert Associates
Scarsdale, New York

"Hahn has produced by far the most comprehensive and illuminating work on the Internet. This is indeed an indispensable guide for any serious user to the world's largest computer network."

Samuel Ko
Internet Book Reviewer, Vancouver, Canada

"At last, a single volume that not only explains what the Internet is, and why it is so exciting, but also provides detailed instructions on how to use all the major Internet services, and includes exhaustive listings of useful archive sites and a wealth of 'insider' information. Hahn writes with wit, clarity and depth that will entertain and inform both the curious novice as well as the experienced net dweller. Copies of this book should be in close proximity to any window onto
the Internet!"

Mark Schildhauer, Ph.D.
Technical Coordinator, Social Sciences Computing Facility
University of California, Santa Barbara

"Finally, a book that painlessly releases the many gifts of the Internet! *The Internet Complete Reference* empowers the reader with a complete understanding of the facilities, terminology, and concepts necessary to participate in the Internet. Unique in its approach, this book covers everything, even the fun stuff, representing the Internet as it should be portrayed. I strongly recommend this book to anyone who is curious about the Internet, and needs a thorough, hands-on-oriented explanation that can be easily understood without a technical degree."

Stewart I. Alpert
Member of Technical Staff, Networking Specialist
The Santa Cruz Operation (SCO)

"*The Internet Complete Reference* is the best in its class: a masterful exposition of everything you need to know to use the Internet. The writing is superb: the text easy to understand. The catalog of Internet resources is itself worth the price of the book. If you use or want to use the Internet, this book is indispensible."

Hal Topper, Director of Consulting Relations
IBM Corporation

"If you want to tap into the Internet's wealth of resources, this straight-forward guide will empower you. The Internet is vastly complex and difficult--which is why mortals need *The Internet Complete Reference*."

David Flack
Editor-In-Chief
Unixworld

"The Internet has promised to change the way people communicate on a global basis, and in the same breath threatened to do so in virtually undocumented fashion. Harley Hahn's *The Internet Complete Reference* changes all that by presenting more useful techniques and tips for using and understanding this global tangle of wires than anything we've seen in print. It's readable AND we can find stuff in it. Hint: This book will be the handbook for Internet users for many years to come."

Jack Rickard
Editor/Publisher
Boardwatch Magazine

"*The Internet Complete Reference* is an excellent tool for new Internet users and for anyone involved with training new users. Harley Hahn has succeeded in explaining Internet concepts in plain language, making it easy for the uninitiated to get started on the network. Just what every new user needs: a pro at their shoulder offering advice. The Internet resource catalog is great!"

Susan Calcari
Info Scout (Internet resource specialist)
InterNIC Information Services, General Atomics

"*The Internet Complete Reference* is an excellent introduction to the Internet for the novice. With the increasing access of K-12 schools on the Internet, this book should be a winner."

Bart Miller, Associate Professor
Computer Sciences Department
University of Wisconsin-Madison

The Internet Complete Reference

Harley Hahn
Rick Stout

Osborne **McGraw-Hill**
Berkeley New York St. Louis San Francisco
Auckland Bogotá Hamburg London Madrid
Mexico City Milan Montreal New Delhi Panama City
Paris São Paulo Singapore Sydney
Tokyo Toronto

Osborne **McGraw-Hill**
2600 Tenth Street
Berkeley, California 94710
U.S.A.

For information on software, translations, or book distributors outside of the U.S.A., please write to Osborne McGraw-Hill at the above address.

The Internet Complete Reference

Copyright © 1994 by Harley Hahn. All rights reserved. Printed in the United States of America. Except as permitted under the Copyright Act of 1976, no part of this publication may be reproduced or distributed in any form or by any means, or stored in a database or retrieval system, without the prior written permission of the publisher, with the exception that the program listings may be entered, stored, and executed in a computer system, but they may not be reproduced for publication.

234567890 DOC 9987654

ISBN 0-07-881980-6

Information has been obtained by Osborne **McGraw-Hill** from sources believed to be reliable. However, because of the possibility of human or mechanical error by our sources, Osborne **McGraw-Hill**, or others, Osborne **McGraw-Hill** does not guarantee the accuracy, adequacy, or completeness of any information and is not responsible for any errors or omissions or the results obtained from use of such information.

To my lovely wife and best friend, Kimberlyn.

— Harley Hahn

To my loving wife, Dawn.

— Rick Stout



Contents

Introduction xix
Acknowledgments xxiii

1 Introduction 1
What is the Internet? 2
Using the Internet 3
How to Use This Book 4
Do You Need to Know Unix? 5
Should You Learn Unix? 5

2 Understanding the Internet 9
Our Friend the Network 10
Hosts and Terminals 12
Client/Server Systems 13
X Window and X Clients 15
Four Important Services Provided by the Internet 19

A Quick Tour of the Internet Resources 20
Electronic Mail 21
Remote Login 21
Finger Service 22
Usenet 22
Anonymous FTP 23
Archie Servers 23
Talk Facility 24
Internet Relay Chat 24
Gopher 25
Veronica and Jughead 25
Wais Servers 25
World Wide Web 26
White Pages Directories 26
Electronic Magazines 27
Mailing Lists 27
Internet BBSs 27
Games 28
MUD 28
What is TCP/IP? 29

3 How to Connect to the Internet 33
The Internet Connection 34
Types of Internet Connections 34
Telephone Connections 35
Connecting a Terminal Over a Telephone Line 37
Connecting a Computer Over a Telephone Line 39
Are You Already On the Internet? 41
Using the Internet for Free 41
Community Computing and the Freenets 42
Public Internet Access 43

4 Internet Addressing 47
Standard Internet Addresses 48
Understanding a Domain Name: Sub-Domains 49
Variations on the Standard Internet Address Format 51
Top-Level Domains 53
Pseudo-Internet Addresses 56
IP Addresses: The **host** Command 57
UUCP 59
UUCP Addresses and Bang Paths 60

Simplified UUCP Addressing 61
Sending Mail to Other Networks 63

5 The Internet Mail System 67
The Basis for the Internet Mail System: The Transport Agent 68
Your Interface to the Mail System: The User Agent 69
Common Unix–Based Mail Programs 70
PC– and Macintosh–Based Mail Programs 72
Basic Terminology 74
Automatically Forwarding Your Mail 76
Mail Tips and Guidelines 77
Understanding Mail Headers 79
The Difference Between Text and Binary Data 82
Using MIME to Mail Binary Data 84

6 The Standard Unix Mail System Program 89
Orientation to the Unix mail Program 90
Specifying Mail Addresses 91
Composing a Message 92
The Tilde Escape Commands 94
A Shortcut When Sending Mail 99
How Can You Tell If a Message Has Been Read? 100
Checking to See If Mail Has Arrived: **from, biff** 101
Reading Your Mail: An Overview 104
The Current Message and Message Lists 107
Displaying a Message 109
Saving and Deleting Messages 110
Replying to a Message 113
Composing New Messages and Forwarding Mail 115

7 Using Telnet to Connect to Remote Hosts 119
An Overview of Telnet Connections 120
Starting the **telnet** Program 121
A Second Way to Start **telnet** 124
Connecting to a Specific Port Number 126
More About Port Numbers 127

The **telnet** Escape Character: CTRL-] 128
Using **telnet** Commands 128
Job Control 130

8 Fingering the World 135
What Is the Finger Service? 136
Displaying Information About a Person 138
Understanding Finger Information 140
Is There a Way to Find Out If Someone Has Fingered You? 143
The **.plan** and **.project** Files 143
Interpreting Information About Mail 145
Fingering a Remote Host 147
Changing Your Personal Finger Information 149
A Guide to Safe Fingering 150
Special-Purpose Finger Services 151
Coke Servers 152

9 Introduction to Usenet 157
What Is Usenet? 158
Basic Terminology 158
What Is It Like When You Use Usenet? 159
What Is the Difference Between Usenet and the Internet? 161
Who Runs Usenet? 162
How Is the News Transported? 163
How Big Is Usenet? 165
The Mainstream and Alternative Hierarchies 166
The Cultural, Organizational, and Regional Hierarchies 168
Clarinet: Real News 170
How Many Different Newsgroups Are There? 172
Newsgroup Naming Conventions 172
Moderated Newsgroups 175
What Are the Most Popular Newsgroups? 175
How Are New Newsgroups Started? 178
Frequently Asked Question (FAQ) Lists 179

10 Understanding Usenet Articles 183
The Format of a News Article 184
The Header 184
The Signature 193
Followup Articles 195
Usenet Acronyms 197

Usenet Slang 197
Foo, Bar, and Foobar 200
RTFM 201
Smileys 203

11 Reading and Posting Usenet Articles 207
Understanding Your Newsreader as a Client 208
How Does a News Server Keep Track of All the Articles? 209
How to Choose a Newsreader 210
What Do the Newsreader Names Mean? 212
Learning How to Use a Newsreader 213
Posting Your Own Articles 215
The Format of the **.newsrc** File 218
Rot-13 220

12 The **rn** Newsreader 225
Overview of the **rn** Newsreader 226
Starting and Stopping 227
Selecting a Newsgroup 229
Selecting and Reading Articles 230
Replying to an Article 232
Posting an Article 233
Commands to Use While Selecting a Newsgroup 235
Commands to Use While Selecting an Article 236
Commands to Use While Paging Through an Article 238

13 The **trn** Newsreader 241
Overview of the **trn** Newsreader 242
Starting and Stopping 242
Selecting a Newsgroup 244
Selecting a Thread 244
Selecting and Reading Articles 246
Commands to Use While Selecting a Newsgroup 250
Commands to Use While Selecting a Thread 251
Commands to Use While Selecting an Article 253
Commands to Use While Paging Through an Article 255

14 The **nn** Newsreader 257
Overview of the **nn** Newsreader 258
Starting and Stopping 259
Displaying Help Information 262
Selecting Articles 263
Moving to Another Newsgroup 266
Reading Articles 267
Posting an Article 268
The nn Family of Commands 269
Commands That Are Always Available 271
Commands to Use While Selecting Articles 272
Commands to Use While Reading Articles 274

15 The **tin** Newsreader 277
Overview of the **tin** Newsreader 278
Customizing Your Working Environment 280
Starting and Stopping 280
The Newsgroup Selection List 283
Selecting a Newsgroup 284
The Thread Selection List 285
Selecting a Thread to Read 287
Reading Articles 287
Posting an Article 289
Commands That Are Always Available 291
Commands to Use While Selecting a Newsgroup 292
Commands to Use While Selecting a Thread to Read 293
Commands to Use While Reading an Article 294

16 Anonymous FTP 297
Understanding FTP 298
Understanding Anonymous FTP 299
The Importance of Anonymous FTP 300
Starting the **ftp** Program 302
A Second Way to Start the **ftp** Program 306
Directories and Files 308
An Overview of the **ftp** Commands 311
The Basic **ftp** Commands 312
The Connection **ftp** Commands 314
The Directory **ftp** Commands 315
The File Transfer **ftp** Commands 318
Setting **ftp** Options 321
A Typical Anonymous FTP Session 323

Finding Your Way Around a New Computer 324
Requesting Anonymous FTP Services by Mail 325

17 Using Archie to Find Anonymous FTP Files 331
What Is Archie? 332
How Does Archie Work? 333
Getting Started with an Archie Server 333
Setting Variables with an Archie Server 335
Setting the search Variable 340
Performing a Search 341
Mailing the Results of an Archie Search 342
Commands to Use While Reading Archie Server Output 343
Using the Whatis Database 344
Displaying Help Information 345
A Summary of Archie Server Commands 348
A Typical Session with Archie 350
Understanding Archie Clients 354
Using an Archie Client 355
Using Archie by Mail 359

18 File Types Used on the Internet 363
Using Archie to Find File Utility Programs 364
File Types Used with Anonymous FTP 365
File Types Used with Usenet and Mail 367
Files That Hold Documents 369
Compressed Files (**.Z**) 370
Tarfiles (**.tar**) 373
Compressed Tarfiles (**.tar.Z**) 375
Uuencoded Files (**.uue**) 376
Btoa Files 377
Mailing Compressed Archives with Tarmail 379
Shar Files (**.shar**) 380
Zipfiles (**.zip**) 384
Uploading and Downloading Files to Your PC or Macintosh 386
Downloading and Viewing Picture Files 389

19 Talking 393
Introducing the **talk** Program 394
The **talk** Screen 395

Guidelines for Talking 397
Problems with Addresses and What to Do 399
The "Checking for Invitation" Problem and What to Do 401
Refusing to Talk 403
The **ytalk** Program 404

20 Internet Relay Chat 411
What Is Internet Relay Chat? 412
Learning About IRC 416
Starting Your IRC Client 417
Getting Started 418
Nicknames 421
Commands for Talking to Specific People 423
Displaying Information About People 426
Summary of Basic IRC Commands 427

21 The Gopher, Veronica, and Jughead 429
What Is the Gopher? 430
How Does the Gopher Work? 430
Travelling Through Gopherspace 431
Musing on the Gopher 434
Starting Your Gopher Client 435
Public Gopher Clients 437
Basic Gopher Commands 438
Moving Around a Gopher Menu 439
Saving a File 441
Customizing Your Gopher Environment 442
Summary of Gopher Commands 443
Types of Gopher Resources 445
Using Veronica to Search Gopherspace 450
Learning More About Veronica 454
Using Jughead to Search Gopherspace 455
The Names Veronica and Jughead 456

22 Finding Someone on the Internet 459
An Approach to Finding Someone 460
White Pages Directories 460
The Gopher 461
Whois Servers 461
The Knowbot 463
Fred and the X.500 Directories 463
The Usenet Address Server 464
Netfind: The Program 465
Netfind: The Server 468

Using Netfind to Find the Location of a Computer 470
What to Do When All Else Fails 471

23 Wais 475
The Motivation for Wais 476
What Is Wais? 477
Wais Clients 479
Musing on Wais 480
Starting to Use Wais 481
Selecting a Source 483
Performing a Search 485
A Summary of Wais Commands 488
A Real-Life Example of Using Wais 488
Tips for Choosing a Source 493

24 The World Wide Web 495
What Is the Web? 496
Why Would You Use the Web? 497
Using the Web 498
Choosing a Browser 499
An Example of Using the Web 501
Commands to Use with the Line Mode Browser 506
Finding a Browser 510

25 Mailing Lists 515
What Is a Mailing List? 516
How Are Mailing Lists Different from Usenet Newsgroups? 516
Moderated and Unmoderated Mailing Lists 518
How Mailing Lists Are Administered 519
Bitnet 520
Bitnet Mailing Lists and Usenet 521
Sending Mail to Bitnet 522
Finding Mailing Lists 526
Subscribing to a Bitnet Mailing List 529
Subscribing to an Internet Mailing List 533

Catalog of Internet Resources 537
Agriculture 540
Archaeology 541
Art 541
Astronomy 542

Automobiles 542
Aviation 543
BBSs 544
Biology 549
Bizarre 550
Books 552
Business and Finance 554
Calculators 556
Chemistry 556
Children 557
Computer Culture 557
Computer Hardware 558
Computer Literature 560
Computer Networks 564
Computer Technology 564
Computer Vendors 565
Consumer Services 565
Cyberpunk 566
Economics 567
Education 569
Environment 572
Food and Drink 573
Fun 574
Games 576
Games (Video) 579
Geoculture 580
Geography 581
Geology 582
Government 582
Health 587
Historical Documents 588
History 589
Hobbies 589
Humanities 592
Humor 593
Humor Archives 596
Internet 598
Intrigue 610
Language 611
Languages 612
Law 612
Libraries 614
Literature: Authors 616
Literature: Collection 619
Literature: Titles 620
Locks and Keys 628
Magazines 628

Mathematics 630
Medicine 631
Military 633
Movies 634
MUDs 634
Music 638
News 642
Oceanography 644
Operating Systems 644
Organizations 644
Pets 645
Physics 646
Pictures and Sound 647
Politics 648
Programming 649
Publications 653
Religion 655
Science 659
Sex 659
Software 662
Software: Archives 664
Software: Internet 666
Software: Macintosh 667
Software: Utilities 668
Space 669
Sports and Athletics 672
Technology 675
Telephones 676
Television 677
Travel 678
Trivia 679
Unix 680
Usenet 682
Weather 683

A Public Access to the Internet 687
The Purpose of This Appendix 688
Special Offer for Our Readers 688
Choosing an Internet Access Provider 689
How to Use This Appendix 690
Country Reference List 696
List of Public Internet Providers 696

B Public Archie Servers 717

C Public Gopher Clients 721

D Public Wais Clients 725

E Public World Wide Web Browsers 729

F List of Top-Level Internet Domains 733

G List of Usenet Discussion Groups 739
Mainstream Hierarchies 740
Alternative Hierarchies 764
Clarinet Hierarchies 789

Index 795



Introduction

The Internet is, by far, the greatest and most significant achievement in the history of mankind. What? Am I saying that the Internet is more impressive than the pyramids? More beautiful than Michelangelo's David? More important to mankind than the wondrous inventions of the industrial revolution? Yes, yes and yes.

Do I expect you to believe this? Of course not—not right now anyway. After all, the Internet is just a computer network and—let's face it—most of what we use computers for is pretty dull.

However, for years now, people have been connecting computers into networks. At first, the networks were a lot of trouble but, by the early 1990s, the engineers and scientists finally figured out how to make it all work most of the time. And now we have the Internet: a worldwide network connecting millions of computers and millions of people. What is amazing is that, within a few short years, the Internet has changed our civilization permanently and has introduced us to two completely unexpected ideas.

First, tens of thousands of people have been laboring to build the Internet. They have worked alone, in small groups, and within organizations, but always like so many ants in a global anthill. Most of these people are only doing their jobs but, just like the ants, they serve the common good while having no conception of the order and the compelling forces that drive their work.

xix

Is it possible that there is a biological urge for us to create computers in the first place? I think not. But, once we did develop these strange and wonderful slaves, there most certainly was an irresistible genetic program for us to follow. Deep inside us, there is a voice that we hear only subconsciously and only as a species; a voice that commands us to take these computers, connect them into networks and . . . communicate.

This leads to the second great idea: that when we connect computers we invariably create something that is much more than the sum of its parts. As we said, the Internet consists of millions of computers and millions of people, but the computers are not important: we use them only to run our programs (that is, to follow our instructions). What is important is something that we cannot yet understand, the answer to the question: What happens when millions of people gather in a safe place to talk and to share? That is what the Internet is all about and that is what we are just beginning to find out.

I sense that we are near the beginning of a great and important change in human affairs. Personally, I don't understand this change. Indeed, I suspect that it is beyond the capabilities of any of us to completely appreciate what is happening, and that the best we can do right now is ride the wave. What I do understand is that there is a reason why we built the Internet and that, as human beings, we have an obligation to learn how to use it and to participate.

Now, about this book.

Before we start, I want to make sure that you understand an important point: the Internet is easy to use, but is difficult to learn. I have filled this book with technical details, important details that you must understand and master in order to use and enjoy the Internet. Still, don't for a moment think that you need to be a computer expert. The Internet is not for nerds, but just as surely it is not for dummies: it is for those people who are willing to think and to learn.

If you ever have one of those days where everything seems to go wrong, and the computer just won't cooperate, and you can't understand what is happening, take a moment and try to see things in a larger sense. As a human being, you share the birthright of intelligence, curiosity and, above all, the ability to learn. However, these gifts are not free. By your very nature, you can not only learn, you are compelled to learn if you are to remain happy and fulfilled. Still, as the saying goes, there is no royal road to knowledge.

To use the Internet, you will have to expend some time and some effort (actually, a great deal of time and effort). But if you do, here is what I promise you:

I promise that I will stay with you.
I promise that I will guide you.
I promise that I will teach you the new words and the new ideas and the new skills that you need to use the Internet.

I want you to know that I have taken a great deal of time to organize this book so as to lead you from one topic to another in a way that will make sense to you (at least, in retrospect).

However, you do not need to read everything from start to finish (although I think you should). Actually, this book comes with only two simple instructions:

1. Start by reading Chapter 1.
2. When in doubt, do something enjoyable that requires effort.

I have not forgotten what it is like to be faced with a large, inconceivable mass of unknowable information. What you are holding in your hands is much more than a computer book. It is a connection—a real connection—between you and me. Your job is to put in the effort and the time. My job is to open the doors.

Buy this book. I am on your side.

— Harley Hahn



Acknowledgments

The writing and production of this book required the help of a large number of people. The most important is my co-author, Rick Stout, who worked a great many hours researching and creating the list of Internet access providers, the master list of Usenet newsgroups, and the extensive catalog of Internet resources. If you enjoy using the catalog, you may want to send Rick a small gift on his birthday (October 9th). Don't worry if you've missed it, Rick will accept presents all year round.

The second person I would like to thank is my senior technical reviewer, Michael Peirce, of the Computer Science Department at Trinity College in Dublin, Ireland. Michael read over every word of this book and tested every example, sending me back hundreds of "suggestions". Without Michael's careful reviewing and comments, this book would have been finished in half the time (and would have been about one third as good).

On the publisher's side of the aisle, the person with the most profound influence on this book was the Acquisitions Editor Scott Rogers. Throughout the course of the writing, Scott provided daily demonstrations of patience and skill that would be difficult to overpraise. In the chain of evolution, computer book editors generally lie somewhere between diamond-backed rattlesnakes and landlords, making Scott's exceptional enthusiasm and respect for writers all the more valuable.

Perhaps the most important of Scott Roger's contributions was his willingness to actually acquire an Internet account and learn how to send and receive electronic mail, proving, as he so properly observes, that "anyone can do it".

If you ever find yourself near the Osborne McGraw-Hill offices, we recommend that you pay them a visit. Walk up to the receptionist (her name is Ann) and ask nicely if you might be allowed to go in and shake Mr. Rogers hand. (However, don't be disappointed if he happens to be in a meeting and you are forced to settle for an autographed picture.)

Aside from Scott, there are a number of other people at Osborne McGraw-Hill who deserve special mention. The Project Editor, Kelly Barr, was always ready to do whatever was necessary to make this book as good as possible. As you can see, he succeeded. (Although, between us, it may not have actually been necessary for Kelly to sleep with the manuscript under his pillow each night for four months. Still, there are many tricks of the publishing trade that I still can't figure out, and one does not argue with success.)

The other Osborne people that deserve mention are Sherith Pankratz (Scott Roger's Editorial Assistant) for her excellent editorial assisting; Marcela Hancik (Production Supervisor) for all the extra work she did designing special formats for our tables and figures; Peter Hancik, also in production, who contributed significantly (and at short notice) to the design of the catalog; Ann Kameoka (Publicity Manager) who made sure that everyone in America knew about this book; and Ann Pharr, the most courteous receptionist west of the Mason-Dixon line. Finally, there is the big man in charge, Jeff Pepper (Editor-in-Chief), whose conception of this book was so immaculate as to assure his place in literary history.

By the way, as you can tell from their names, Marcela and Peter Hancik are married. If you are like me, you are probably wondering what the home life of two book design people must be like. To satisfy your curiosity, here is a typical dinner conversation:

Marcela: Thanks for setting the table Peter, but why did you place the salt shaker so far to one side?

Peter: Well, Marcela, I wanted to align both the salt and pepper shakers flush left in order to balance them with the napkins.

Marcela: But what about the forks?

Peter: Well, I decided to go with the regular dinner forks which, as you know, are three point sizes larger than the salad forks. But since all the cutlery is monospaced, I had to put the plates on the floor to make sure that everything else fit within the whitespace around the outside of the tablecloth.

Marcela: Looks good. Let's go with your design and if there is any problem, we can change it in the next reprinting at breakfast.

On a more punctuational note, I would like to take a moment to acknowledge the excellent copy editing of Lunaea Hougland, queen among copy editors. Time after time, Lunaea has always managed, under great pressure, to ensure that each comma was in its exact place. In fact, take a few minutes now and check out

some of the commas in this book. See what I mean? You will never see "hel,lo" or "ball,oon" or even "ranny,gazoo"—not in one of Lunaea's books.

Throughout the research and writing of this book, there were many Internet people from all over the world who contributed greatly. Here they are:

Andy Gruss: Carnegie-Mellon University, Pennsylvania
Bart Miller: University of Wisconsin, Madison
Bob Hawley: Bell Laboratories, New Jersey
Brian K. Reid: Digital Equipment Corp, Palo Alto, California
Brooke Jarrett: La Mesa, California
David Gaudines: Concordia University, Montreal, Canada
David Lemson: University of Illinois
Desiree Madison-Biggs: Netcom Online Communication Services
Dick Martin: TRW Corporation, San Diego, California
Hugh Evans: European Space Research & Tech Centre, The Netherlands
Iain Lea: Siemens AG, Erlangen, Germany
Jason Heimbaugh: University of Illinois
Jeff Inglis: Middlebury College, Middlebury, Vermont
Kai Rohrbacher: University of Karlsruhe, Germany
Kent C. Ritchie: University of Minnesota
Larry W. Virden: Chemical Abstracts Service, Ohio
Len Rose: Pagesat; Netsys, Palo Alto, California
Marc Tamsky: University of California at Santa Barbara
Mark Schildhauer: University of California at Santa Barbara
Mary A. Axford: Georgia Institute of Technology
Mary Leyva: GTE California, Whittier, California
Michael Mealling: Georgia Institute of Technology
Michael Peirce: Trinity College, Ireland
Michael Shuster: Technical University, Vienna, Austria
Paola Kathuria: ArcGlade Services, London, England
Paul Vixie: Vixie Enterprises, Redwood City, California
Peter ten Kley: Utrecht, The Netherlands
Peter Wemm: Demaas Proprietary Limited, Perth, Australia
Randy Bush: PSGnet, Portland, Oregon
Rhett Jones: University of Utah
Rick Broadhead: York University, Toronto, Canada
Robert W. Hill: University of Kansas, Lawrence, Kansas
Scott Yanoff: University of Wisconsin
Stewart Alpert: Santa Cruz Operation
Tom Gerstel: Ithaca College, Ithaca, New York

I would like to take a moment to offer special thanks to three wonderful people who helped Rick and me tremendously by sending us a great number of insightful comments, as well as hidden treasures, from around the Internet. These people are Mary Axford (ace reference librarian) at the Georgia Institute of Technology, Peter ten Kley in The Netherlands, and Rick Broadhead at York University in Toronto, Canada.

Finally, I would like to thank my wonderful wife Kimberlyn, whose beauty and grace are surpassed only by her intelligence and talents.



CHAPTER 1

Introduction

In this chapter we introduce you to the Internet and lay bare our intentions for this book.

We start by asking the seminal question, "What is the Internet?" From there, we advance from the philosophical to the practical by offering our suggestions on how to best use and enjoy this book.

Next, we tackle an oft-asked query: "Do I need to learn Unix?" (as well as the only slightly less-asked query, "What is this Unix thing anyway?").

What is the Internet?

The Internet is the name for a group of worldwide information resources. These resources are so vast as to be well beyond the comprehension of a single human being. Not only is there no one who understands all of the Internet, there is no one who even understands most of the Internet.

The roots of the Internet lie in a collection of computer networks that were developed in the 1970s. They started with a network called the Arpanet that was sponsored by the United States Department of Defense. The original Arpanet has long since been expanded and replaced, and today its descendents form the global backbone of what we call the Internet.



FUN TIP: Technology marches on: The first experimental network using Internet-like technology involved four computers and was built in 1969. This was 56 years after the invention of the zipper, 37 years after the introduction of the first parking meter, and 13 years prior to the development of the first IBM personal computer.

It would be a mistake, though, to think of the Internet as a computer network, or even a group of computer networks connected to one another. From our point of view, the computer networks are simply the medium that carries the information. The beauty and utility of the Internet lie in the information itself.

Take a moment and skim through the catalog that comprises the last part of this book. Notice the enormous variety. As we start to work together, this is how we want you to think of the Internet: not as a computer network, but as a huge source of practical and enjoyable information.

But this is only the beginning. We would also like you to develop an appreciation of the Internet as a people-oriented society. Put simply, the Internet allows millions of people all over the world to communicate and to share. You communicate by either sending and receiving electronic mail, or by establishing a connection to someone else's computer and typing messages back and forth. You share by participating in discussion groups and by using the many programs and information sources that are available for free.

Does this mean that we are saying that the Internet resources will become as important to you as your telephone and your post office? Yes, that is exactly what we are saying.

In learning how to use the Internet, you are embarking upon a great adventure. You are about to enter a world in which well-mannered people from many different countries and cultures cooperate willingly and share generously. They share their time, their efforts, and their products. (And you will, too.)

For one more moment, take another look at the catalog. Each of those items is there because some person or some group volunteered their time. They had an idea, developed it, created something worthwhile, and then made it available to anyone in the world.

Thus, the Internet is much more than a computer network or an information service. The Internet is living proof that human beings who are able to communicate freely and conveniently will choose to be social and selfless.

The computers are important because they do the grunt work of moving all the data from place to place, and executing the programs that let us access the information. The information itself is important because it offers utility, recreation, and amusement.

But, overall, what is most important is the people. The Internet is the first global forum and the first global library. Anyone can participate, at any time: the Internet never closes. Moreover, no matter who you are, you are always welcome. You will never be excluded for wearing the wrong clothes, having the wrong colored skin, being the wrong religion, or not having enough money.

A cynic might say that the reason the Internet works so well is that there are no leaders. Actually, there is some truth to this. As unbelievable as it sounds, nobody actually "runs" the Internet. Nobody is "in charge" and no single organization pays the cost. The Internet has no laws, no police, and no army. There are no real ways to hurt another person, but there are many ways to be kind. Perhaps, under the circumstances, it is only natural for people to learn how to get along. (Although this does not stop people from arguing.)

What we choose to believe is that, for the first time in history, unlimited numbers of people are able to communicate with ease, and we are finding it is in our nature to be communicative, helpful, curious, and considerate.

That is the Internet.

Using the Internet

Using the Internet means sitting at your computer screen and accessing information. You might be at work, at school, or at home, using virtually any type of computer (including a PC or a Macintosh).

A typical session might begin with you checking your electronic mail. You can read your messages, reply to those that require a response and, perhaps, send a message of your own to a friend in another city.

You might then read a few articles in some of the worldwide discussion groups: Jokes from one of the humor groups, or perhaps recipes for a dinner that you are planning for the weekend. Maybe you are following a discussion about Star Trek or philosophy or literature or aviation.

After leaving the discussion groups, you might play a game, or read an electronic magazine, or search for some information on a computer in another country.

This is what using the Internet means and this is what we will show you how to do in this book.

How to Use This Book

The title of this book contains the word "Complete", so let's take a minute to discuss exactly what we mean by that.

We promise to teach you everything you need to know to be able to use the Internet and its basic resources. With the catalog that forms the latter part of the book, we offer a large, representative list of all that is available. Our intention is that for most people, most of the time, this book will be all that you need.

Since the purpose of this book is to show you how to understand and use the Internet, we need to start by providing the proper background and by teaching you some technical details. This is what we will do in Chapters 2, 3, and 4. These chapters explain the basics: how the Internet is organized, how you connect to the Internet and, most important, how to understand Internet addressing.

As you know, when you use a telephone you need a number to dial, and when you mail a letter you need a postal address. Similarly, the Internet has its own official "addresses". Each person and each computer has its own address, and Chapter 4 explains how the system works. Almost everything is built around these addresses, so it is crucial that you read this chapter at the beginning of your Internet career.

Once you have read Chapters 2, 3, and 4, you can go wherever your interests take you. If you want to send electronic mail, start with Chapters 5 and 6. If you want to participate in a Usenet discussion group, read Chapters 9, 10, and 11, and then whichever of chapters 12, 13, 14, or 15 is appropriate. Once you understand the basics, there is no special order for learning.

If you are not sure where to start, take a look at the catalog. Read through until you find something that interests you, and then turn to the chapter that shows you how to access that service.

If you are using this book to teach a course, you can follow the chapters in order. We have carefully designed the material to cover the most important topics first. You can start from the beginning and teach your students (or yourself) as much as you want, leaving the rest for another day.

Do You Need to Know Unix?

The answer is no, but read on.

Unix is a family of operating systems (master control programs) that are used to control computers. Virtually all types of computers can run Unix. Conversely, there are many variations of Unix that run on all sizes of computers.

When we use the word "Unix" as a general term, we mean more than an operating system. We are actually referring to an entire culture with its own language, technical terms, conventions, traditions, and a wide variety of computer-oriented facilities. To many people, the Unix culture is intimately connected to the Internet. Some people consider the Internet to be part of the Unix culture. Other people consider Unix to be part of the Internet culture.

The truth of it is that the Internet very much has a life of its own. Most of the Internet computers use Unix, but the details are hidden from you. Thus, you do not need to learn Unix *per se* in order to use the Internet.

However, you do need to understand the rudiments of using your own computer system. You must know how to start work, enter commands, use the keyboard (and mouse if you have one), and stop work when you are finished. It is also helpful—almost indispensable—to be able to manipulate data files, so you can save and retrieve information, and so you can create and edit your own information. For example, when you send someone a message using electronic mail, it is convenient to be able to use a text editing program to compose the message ahead of time.

When you use a Unix system to access the Internet, it is often difficult to say where Unix ends and the Internet begins. The programs that you will be using will all be Unix programs, perhaps not built into the operating system itself, but very much a part of the Unix culture. For example, many people read their electronic mail using the Unix mail program.

However, you are perfectly justified in keeping your knowledge of Unix to a practical minimum and spending your time, instead, exploring and using the Internet. There is no need for you to become a Unix expert.

Should You Learn Unix?

Having just told you that you do not need to become a Unix expert in order to use the Internet, we are now going to tell you that, yes, we do recommend that you learn Unix. You do not need to, but you ought to, and here is why.

As we mentioned earlier, Unix is a lot more than an operating system. It is actually a large, worldwide culture that is intimately connected to the Internet. On its own, Unix has a lot to offer and, hence, is worth learning.

Moreover, if you are using Unix to access the Internet, there are a great many advantages to having some technical knowledge of the underlying system. For example, you will probably want to send messages, either to another person or to a discussion group. To compose such messages, you should be able to use one of the Unix text editing programs.

As you work with the Internet, you may also want to collect information that you find on computers around the world. This means that you will need to know how to create and manipulate data files.

There are many such examples and they all illustrate the fact of life that, regardless of what anyone tells you, it is always a good idea to learn something about the technical details of your computer system. People have gone to a great deal of trouble to make computer systems easy to use and to make certain parts of the Internet easy to use, but—and this is an important but—you will be far better off if you learn more than the bare minimum.

Wait a minute, we hear some of you saying, that may be fine for the people who are using Unix-based systems, but I use a PC (or a Macintosh or some other fine non-Unix system) to access the Internet, and the people who set everything up promised it would be user friendly. They told me that I could select anything I wanted from a menu and that I wouldn't have to spend any time learning anything as evil and horrible as Unix. Besides, computers scare me.

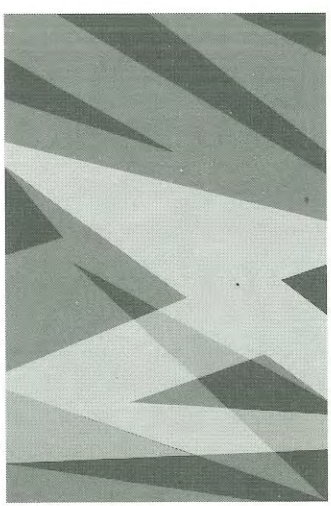
Well, far be it from us to deny you the comfort and security of your own fears. Moreover, the last thing that we would do in a family-oriented book like this would be to disagree contentiously with those misguided souls who firmly believe that the future of computing lies in the populace-at-large selecting items from menus.

Suffice it to say that, as you use the Internet, you will from time to time connect to remote computers which will be running Unix. At such times, a basic knowledge of the Unix culture and its tools can provide practical benefits.

In order to help you, we recommend the book *A Student's Guide to Unix* by Harley Hahn (McGraw-Hill College Division, 1993). That book and this Internet book complement one another nicely.

Hint

Learning the fundamentals of Unix will make your work with the Internet more comfortable and will expand your possibilities.



CHAPTER 8

Understanding the Internet

In order to use the Internet well, you need to understand something of what it is and how it works. In this chapter, we will start with the idea of computer networks, introduce a few basic ideas and terms, and then move quickly to the Internet itself.

After explaining the basic services upon which the Internet depends, we will survey the large variety of resources you will be able to use. If you have been wondering, "What can the Internet do for me?", this is the chapter that will explain it all.

If you hang around the Internet at all, you will from time to time hear people talk about something mysterious called "TCP/IP". At the end of this chapter we will discuss TCP/IP and show you what it really is. (What? You have never heard of TCP/IP? Obviously, you have been hanging around the wrong sorts of people. Well, we can fix that.)

Our Friend the Network

The term *network* refers to two or more computers connected together. There are a number of reasons to connect computers into networks but the two most important (from our point of view) are:

- to allow human beings to communicate
- to share resources

Once you start using the Internet, you can send messages to anyone else on the Internet. You can even send messages to people who use other networks that are connected to the Internet.

As for sharing, computer managers arrange networks so resources that are expensive or difficult to maintain can be used by anyone on the network. For instance, a manager might attach a costly printer to a network so that everybody who needs it can use the same printer. On the Internet, we share information resources rather than pieces of hardware. For example, the items in the catalog (later in this book) are some of the many resources that can be used by anyone on the Internet.

A *local area network*, or *LAN*, is a network in which the computers are connected directly, usually by some type of cable. When we connect LANs together, we call it a *wide area network* or *WAN*. Most wide area networks are connected via leased telephone lines, although a variety of other technologies, such as satellite links, are used as well. The wide area connections for most of the Internet travel over some telephone system or another. Indeed, the bottleneck in establishing Internet service within developing countries is usually due to the lack of a reliable telephone system.

Here is a typical example of a network. Imagine yourself sitting in a room full of computers in the Social Sciences Computing Facility at a major university. (The weather outside happens to be cold and rainy, but since it has nothing to do with the example, we won't mention it.) Your computer is connected in a LAN to all the other computers in the room and to the computers within people's private offices throughout the building. This arrangement is shown in Figure 2-1.

There are a number of other LANs on the campus. For example, the Psychology department has its own network of computers, as does the Math

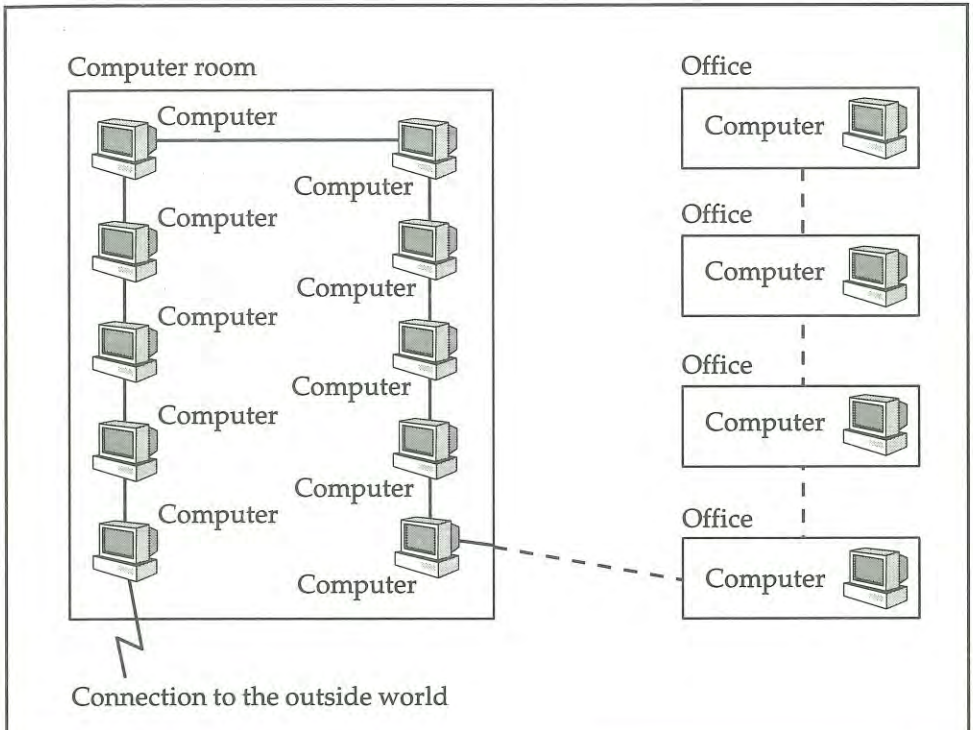


Figure 2-1. *A local area network*

department, the Computer Science department, and so on. Each of these LANs is connected to a high-speed link, called a *backbone*, to form a campus-wide WAN. This is shown in Figure 2-2.

Although we have used a university as an example, many types of organizations use similar arrangements: companies, governments, research facilities, other types of schools, and so on. If an organization is small, it may have only a single LAN. Large organizations may have multiple LANs connected into one or more complex WANs. Such organizations usually have a full-time staff to care for and feed the networks.

How are the LANs connected? By special-purpose computers called *routers*. The job of a router is to provide a link from one network to another. We use routers to connect LANs (to form WANs) and to connect WANs (to form even larger WANs). In other words, you can consider the computers within the Internet to be connected into LANs and WANs by a large number of routers. However, there is more to the picture that we need to explain. But first, let's take a moment to talk about hosts and terminals.

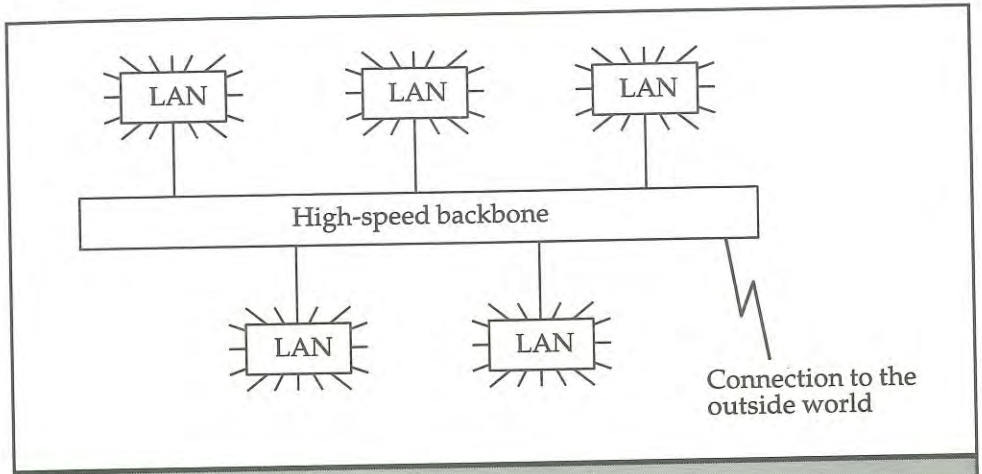


Figure 2-2. *A wide area network*

Hosts and Terminals

There are two meanings for the word *host* that you should know about.

First, within the Internet, each separate computer is called a host. For example, you might tell someone that he can find the information he wants by connecting to a host in Switzerland. If your computer is connected to the Internet, then it too is a host even though you may not share any resources with the rest of the Internet.

You may also see a computer referred to as a *node*. Here is why. If you draw a diagram of points and lines to represent the connections within a network, each computer will be a point and each connection will be a line. In the part of mathematics that deals with such diagrams, each such point is called a "node". Network specialists have borrowed this term to refer to any computer that is connected to a network. Thus, "node" is a more technical synonym for "host".

Hint

If you are trying to see if someone is a nerd, ask him "What is the technical term for a computer that is attached to the Internet?" If he says "node", chances are, he is a nerd.

The second meaning of the word "host" has to do with how certain computer systems are set up.

In general, there are two ways you might use a computer. You might have it all to yourself, or you might share it. For example, when you use a PC or a Macintosh, you are the only person using the computer. Single-user computers, especially the more powerful ones, are often called *workstations*.

Some computers, however, are made to support more than one user at the same time. These multi-user systems are often referred to as host computers.

A large mainframe computer, for example, can act as a host for hundreds of users at the same time. More commonly, a smaller computer, perhaps one that looks no larger than a PC, will act as a host for a small group of users. The Unix operating system, which we discussed in Chapter 1, is a multi-user system. Although some people use a Unix computer as a personal workstation, many Unix computers are used as hosts to support multiple users.

When you have your own computer, you interact by using the keyboard, screen, and (possibly) a mouse. These devices are part of the computer. With a multi-user computer, each person has his or her own *terminal* to use. A terminal has a keyboard, screen, perhaps a mouse, and not much more. All the terminals are connected to the host, which provides the computing power for everybody. This arrangement, called a *time-sharing system*, is shown in Figure 2-3.

Thus, there are two meanings for the word "host". Within the Internet, each computer is called a host. Within a time-sharing system, the main computer that supports each user on a separate terminal is also called a host. Of course, if such a computer were connected to the Internet, it would be both a time-sharing host and an Internet host.

Client/Server Systems

As you know, one of the principle uses of a network is to allow the sharing of resources. Much of the time, this sharing is implemented by two separate programs, each running on different computers. One program, called the *server*, provides a particular resource. The other program, called the *client*, makes use of that resource.

For instance, say that you are working with a word processing program that is running on your own PC. You tell the program that you want to edit a particular file that is stored on another computer on your network. Your program will pass a message to that computer asking it to send the file. In this case, your word processing program is the client while the program that accepts the request and sends the file is the server. More precisely, it is a file server.

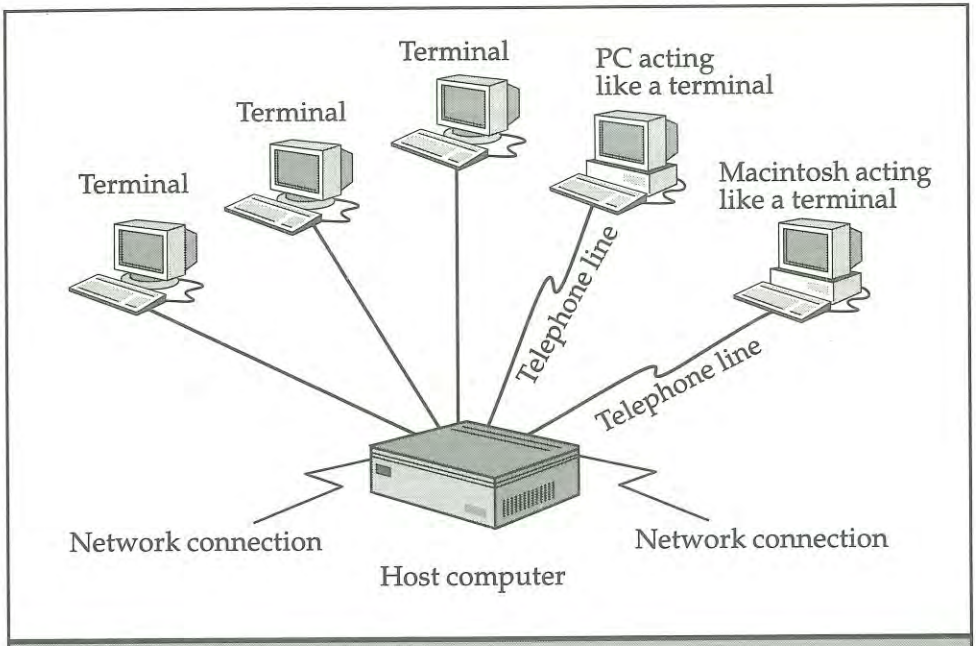


Figure 2-3. *A time-sharing system*

On local area networks, where the hardware is close by and visible, it is common for people to use the word "server" to refer to the actual computer that runs the server program. For example, one can imagine a network manager showing around the President of the United States and, while pointing to an otherwise nondescript beige machine, proudly announcing, "This is our file server." (To which proper etiquette requires that the President respond by nodding his head and saying, "How nice for you.")

On the Internet, hardware is normally not visible, and the terms "client" and "server" usually refer to the programs that ask for and provide services.

Here is an important example. Many Internet sites provide a service called a "Gopher" (which we will discuss in more detail later). Briefly, a Gopher allows you to select items from menus. Each time you select an item, the Gopher performs the required task. For instance, if the item describes a particular piece of information (such as "News of the Day"), the Gopher will retrieve this information and display it for you.

When you use a Gopher, two different programs are involved. First, there is the program that provides your interface. This is the program that interprets your keystrokes, displays the menus, and generally makes sure your requests are carried out. This program is called the Gopher client.

The other program is the one that supplies whatever it is the Gopher client has requested on your behalf. This program is called the Gopher server.

The beauty of this system is that the client and server programs do not necessarily run on the same computer. Indeed, more often than not, the client and server programs reside on different computers. For instance, you may be sitting in front of a PC in Rio Linda, California, using a Gopher to read the "News of the Day" at the National Security Agency in Virginia, three thousand miles away.

In this case, the Gopher client is a program running on your PC, while the Gopher server is a program running on a supercomputer on the other side of the country.

All of the Internet services make use of this *client/server relationship*. Learning how to use the Internet actually means learning how to use each of the client programs. Thus, in order to use an Internet service, you must understand:

1. How to start the client program for that service
2. How to tell the client program which server to use
3. Which commands you must use with that type of client

Your job is to start the client and tell it what to do. The client's job is to connect to the appropriate server and to make sure that your commands are carried out correctly.

Each type of Internet client has its own commands and conventions. For example, the commands you use with a Gopher client are different from the commands you use with, say, an Archie client (another Internet service). Fortunately, there are only a handful of basic services you need to learn.

As you learn about Internet services, you will find that the client programs—such as Gopher or Archie—have many different commands. However, you will also find that you actually need to learn only a few of the most important commands.

The Internet client programs all come with their own built-in help functions. Thus, if a situation arises in which you require an esoteric command that you have not yet learned, you can use the built-in help to find what you need.

And, of course, you always have this book.

X Window and X Clients

There is one special type of client/server system, called X Window, that you may need to know about. If your computer uses X Window, this section will explain

the basic concepts you should understand. As you will see, X Window gives you certain advantages when you use the Internet.

If you are not an X Window user, feel free to move on to the next topic. Still, X Window is used widely, and you may want to skim this section to see what all the fuss is about.

X Window is used with Unix systems that support graphical user interfaces. A *graphical user interface*, or *GUI*, allows you to operate a computer by using not only your keyboard, but also a mouse or some other type of pointing device. With the help of your mouse, you select items from menus and manipulate objects on the screen. You can run more than one program at the same time, each of which can reside in its own rectangular area called a *window*.

Of course, the idea of a GUI (pronounced "goo-ee") is nothing new. If you have ever used a Macintosh, or a PC running Microsoft Windows or OS/2, then you have used a GUI.

As we mentioned in Chapter 1, there are many different types of Unix. X Window was developed in order to provide a standard set of tools for programmers who develop graphical applications and a standard interface for users to interact with those applications. For convenience, X Window is usually referred to as X. For example, a friend tells you that he has a program you might want to use. You might ask him, "Does it run under X?"

Although there is nothing in the design of X that says it must be used with Unix, in practice, this is almost always the case.

Hint

At one time, an operating system named V (the letter *V*) was developed at Stanford University. To work with V, the programmers created a windowing system which they named W. Later, the W system was sent to someone at MIT who used it for a new windowing system which he named X. The modern X Window system, first developed in 1984, grew out of this initial effort. Today, the current version of X Window is version 11 release 5, usually written as "X11 R5". Note that the official name of this system is the singular "X Window", not the plural "X Windows".

In X terminology, the three devices you use to interact with your computer—the keyboard, the screen, and the mouse—are referred to as a *display*. X lets you run more than one program at the same time on a single display. As part of the graphical user interface, each program resides in its own window on the screen. When you want to switch from one program to another, you use the mouse to move from one window to another.

When you use X, the details of maintaining the GUI for all the programs that are running are handled by a single program called a *display server* or an *X server*.

For example, say that you have four programs running at the same time, each of which resides in its own window. As you work, you can move the windows or even change their size. Now, suppose one of the programs needs to draw a circle on the screen. Rather than doing the work itself, it sends a message to the X server—the program that is controlling the screen—telling it to draw a circle of a particular size at a particular location. The X server actually does the work.

This division of labor has several important advantages. First, it means that the entire GUI is controlled by a single program that will ensure that everything works as it is supposed to. For example, the window in which a program is running may be partially obscured by another window. The program itself does not need to know this, nor does it care. The X server will handle the details.

Second, when a programmer designs a new program, he or she does not have to worry about the user interface. All that is necessary is for the program to call on X in the standard way whenever such work needs to be done. This makes for smaller, more reliable programs that are portable from one X system to the next.

Since all X servers provide the same functions, a program that is written to depend on an X server will run under any X system. You can, for instance, find an X program anywhere on the Internet, copy it to your computer, and run it under your own graphical user interface. Once you learn how to use the Internet file transfer service, you will be able to acquire many such graphical programs for free.

The third advantage of X is that the graphical user interfaces are more or less standard.

The part of the system that provides the look and feel of your interface is called the *window manager*. (Technically, the window manager is itself a program that runs on top of X.) There are two widely used window managers, named *Motif* and *Open Look*. There are differences between the two, but they are not profound.

If you saw identical programs running on two computer screens, one using Motif and one using Open Look, the main thing you would notice is that the appearance of the windows and other graphical elements would be somewhat different. In addition, you would use the mouse and the menu system slightly differently.

However, the basic concepts are the same no matter which window manager you use. Certainly, the programs inside the windows would not change. Basically, if you know how to use one X system, you know how to use them all.

Figure 2-4 shows the screen of an X display in which you are running five different programs at the same time, each one in its own window. Notice that, as these programs execute, they depend on a single X server program to maintain the user interface.

By now, this should all remind you of the client/server relationship that we discussed in the previous section. Indeed, this is the case. The programs you run are clients and, as such, are referred to as *X clients*. They request

services—managing the user interface—from the X server program you run on your own computer. In other words, the term "X client" is a synonym for any program that runs under the auspices of X Window.

The most powerful feature of X Window is that the X clients do not have to run on the same computer as the X server. When you use X, the X server program will always run on your computer, the one in front of you. However, the X clients can run on any computer that is connected to your computer via the network. In Figure 2-4, you can see that of the five X clients, three are running on remote computers.

If you are an X user, the Internet is important to you in three ways. First, as we mentioned above, there are many X clients available for you to copy and use.

Second, you can use the Internet to connect to another computer and run an X program that will be displayed on your screen. (However, if the remote computer is far away, the response may be too slow for your liking.)

Finally, as an X user, you may have an advantage when you use Internet services.

As we explained in the previous section, you access Internet services by using a client program. For example, to utilize a Gopher server, you use a Gopher client.

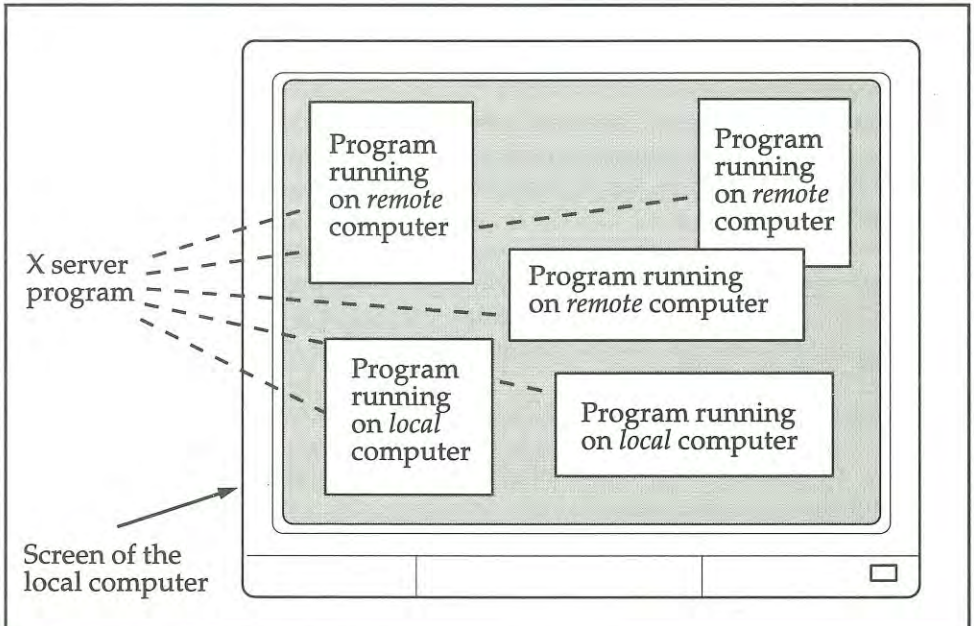


Figure 2-4. *An X Window system*

If you are an X user, you may be able to find more powerful X-based clients to use instead of the regular character-oriented clients. For instance, many installations have a graphical version of the Gopher client that runs under X. If this is the case where you work, you will be able to use a graphical Gopher client with a spiffy user interface while the peasants down the hall (whose computers do not run X) will have to make do with the regular client program.

Hint

Some people will tell you that the X Window system is confusing because the server program runs on your computer, while the client programs may run on remote computers. Such people maintain that this is the opposite of what you might expect.

As you now know, the idea behind X is simple, so don't let silly people confuse you. When you meet such a person, be graceful and remind yourself, "There but for the grace of God, go I."

Explain to this person that you already understand X just fine, thank you. If he looks disappointed at not being able to mix you up with a lot of unnecessary technical details, you might gently suggest that he use his time to perform some socially useful function like writing a letter to the editor or learning a new Unix command.

Four Important Services Provided by the Internet

The software that supports the Internet provides a large number of technical services upon which everything else is built. Most of these services operate behind the scenes, and you do not need to understand them. (However, we should take a moment to recognize the many overworked network administrators who do take the time to understand the details.)

Nevertheless, there are four important Internet services we do need to talk about. You don't need to know the details, but you need to know they exist. In this section, we will describe these services. In the following section, we will discuss the many Internet resources—available to you—that depend on these services.

First, the mail service reliably transmits and receives messages. Each message is sent from one computer to another on its way to a final destination. Behind the scenes, the mail service ensures that the message arrives intact at the correct address.

The next service, called *Telnet*, allows you to establish a terminal session with a remote computer. For example, you can use Telnet to connect to a host on the other side of the world. Once the connection is made, you can log in to that computer in the regular manner. (Of course, you will need a valid user account and password.) Telnet also allows two programs to work cooperatively by exchanging data over the Internet.

The word "telnet" is often used as a verb. For example, you might tell someone, "If you telnet to this computer, you will be able to use a computerized reference book."

The third service is called *FTP* (file transfer protocol). FTP allows you to transfer files from one computer to another. Most of the time, you will use FTP to copy a file from a remote host to your computer. This process is called *downloading*. However, you can also transfer files from your computer to a remote host. This is called *uploading*. In addition, should you find it necessary, FTP will allow you to copy files from one remote host to another.

Hint

To remember the difference between "uploading" and "downloading", think of the remote host as floating above you in the sky. You send files up, and you ask for files to be sent down.

The last Internet service you should understand is the general client/server facility we discussed earlier in the chapter. A client program can connect to another computer and ask for the help of a server program.

For example, the Gopher system we mentioned earlier works in just this way. Your Gopher client displays a menu for you. When you make a selection from the menu, the client connects to the appropriate server—no matter where it is on the Internet—and procures the service you requested.

A Quick Tour of the Internet Resources

Having covered the basic framework of the Internet, let's take a look at all it has to offer. There are many Internet resources, and new ones are added whenever some clever person can figure out a new way to use the network.

In this section, we will examine all the important Internet resources. In later chapters, we will discuss each resource in detail, showing you how it works and the best way to use it. Indeed, many of these resources demand an entire chapter to themselves.

The purpose of this section is to give you an idea of exactly what the Internet has to offer you. If you take a look at the entries in the catalog, later in the book, you will see that each one depends on one or more of these resources. Table 2-1 at the end of this section contains a summary of each of the Internet resources and tells you what chapter to read for more details.

Electronic Mail

As an Internet user, you can send and receive messages from anyone else on the Internet. Moreover, you can do the same for other mail systems—such as CompuServe or MCI Mail—that have connections with the Internet.

However, mail does not mean simply personal messages. Anything that can be stored in a text file can be mailed: computer (source) programs, announcements, electronic magazines, and so on.

When you need to send a binary file that cannot be represented as regular text—such as compiled computer programs or graphical pictures—there are facilities for encoding the data into text. At the other end, the recipient simply decodes the textual data into its original format.

Thus, you can mail virtually any type of data to anybody. The Internet mail system is the backbone (and original motivation) for the network itself.

Remote Login

As we described in the previous section, you can telnet to a remote computer anywhere on the Internet. Once you have established a connection, you can log in (as long as you have a valid account on that computer). Since most Internet computers use Unix, we borrow the terminology for logging in from Unix itself.

The name by which an account is known is called a *userid* (pronounced "user-eye-dee"). The secret code you must enter, to prove that it is really you, is called your *password*. As long as you have a valid userid and password, you can log in to any computer on the Internet.

As a public service, many Internet systems are set up to allow anybody to log in using a special *guest account*. For example, in the United States, there is a system that will display weather reports from around the country. Anyone can log in to this system and check out the weather.



FUN TIP: If you are working in a room without a window, it is often faster to check the weather by connecting to the weather server (which is in Michigan) than by going outside.

Finger Service

Most Internet computers offer a facility that allows you to ask for information about a particular user. This service is known by the descriptive name of *finger*. As you will see, people on the Internet are often known by their userid. You can use finger to find out the name of the person behind the userid. For example, you might find out that the userid **harley** is registered to Harley Hahn.

We use the word "finger" not only as a noun, but also as a verb. For example, you might overhear someone ask, "Who is userid **tln**?", to which a second person answers, "I don't know. Why don't we finger him and find out?"

Depending on how the finger service has been set up on the computer you contact, you may be able to find out other information about the person: phone number, office address, and so on. In addition, some finger systems will tell you when the person last logged in and if they have unread mail. This can come in handy when you need to check if someone has received an important message.

There is also a way for you to customize part of what people see when they finger your userid. You can specify certain information that you want displayed. For example, a professor might specify his office hours. Somebody giving a party might give directions to her house. You can display this information whenever you want just by fingering that person's userid.

You can also finger a computer rather than a userid. In this case, the computer will respond by showing you a summary of all the userids currently logged in.

Finally, some systems use finger to support a public request for certain specific information. For example, there is a particular userid and computer at the University of Washington at Seattle that you can finger to display information about recent earthquakes.



FUN TIP: In a more esoteric vein, you can finger a computer at Carnegie-Mellon University in Pennsylvania to find out if the Coca Cola machine is empty or not. By fingering a different userid, you can check if the candy machine is empty.

Usenet

Usenet—the name is a contraction of "User's Network"—is one of the main reasons people use the Internet. Usenet itself is not an actual network. It is a

system of discussion groups in which individual articles are distributed throughout the world.

Usenet has literally thousands of discussion groups, so there is definitely something for everyone. (See the list of groups in Appendix G.)

At each Internet site, the system administrator decides whether or not to carry the Usenet discussion groups. Thus, Usenet is not available everywhere. Moreover, even sites that do provide this resource will not carry every conceivable discussion group.



FUN TIP: Reading Usenet articles—especially the jokes—is a good way to spend time that would otherwise be wasted on work.

Anonymous FTP

As we mentioned in the previous section, the FTP service allows you to copy files from one computer to another. Anonymous FTP is a system in which an organization makes certain files available to the general public. You can access such a computer by using a userid of **anonymous**. No special password is required.

Anonymous FTP is one of the most important Internet services. Virtually every possible type of data is stored somewhere, on some computer, and it is all available to you for free. For example, many of the programs used on the Internet are created and maintained by individuals who then distribute the programs worldwide via Anonymous FTP. You can also find electronic magazines, archives of Usenet discussion groups, technical documentation, and much, much more. As a full-fledged Internet user, you will come to depend heavily on Anonymous FTP.

Archie Servers

There are thousands of Anonymous FTP servers around the world offering a vast number of files. The role of the Archie servers is to make the whole system manageable by helping you find what you need.

Suppose you want a particular file, for instance, a program you have heard about. You can use an Archie server to tell you what Anonymous FTP sites store that file. Once you know the names of the sites, it is a simple matter to use FTP and download the file.

If you consider the world of Anonymous FTP as an enormous, worldwide library that is constantly changing, you can think of the Archie servers as the catalog. Indeed, without the Archie servers, most of the Anonymous FTP resource would be unreachable.

The name "Archie" was coined from the idea of an "archive server". In deference to the well-known propensity of computer people to engage in whimsical personification, we usually refer to "Archie" as if it were human. For example, you might ask someone, "Do you know which Anonymous FTP sites carry the electronic magazine called *The Unplastic News*?" "No, I don't," replies your friend, "Why don't you ask Archie?"

Talk Facility

The Talk facility establishes a connection between your computer and someone else's. You can then use this connection to type messages back and forth (until you get bored).

The great thing about the Internet Talk facility is that it makes it possible to hold a conversation with someone no matter how far away they are. The other person sees what you type as you type it, and you can both type at the same time without your messages getting mixed up.

Internet Relay Chat

Internet Relay Chat (IRC) is like a Talk facility for more than one person at the same time. As you might imagine, IRC is used heavily and offers a lot more than simple conversation.

You can take part in public conversations with a large number of people. These conversations are loosely organized around various topics or ideas. Alternatively, you can use IRC to arrange a private conversation with people of your own choosing, much like a telephone conference call.

Hint

If you find yourself spending large amounts of time using a Chat facility, remind yourself that you are talking to the type of people who will spend a large amount of time using a Chat facility.

Gopher

Gophers provide a series of menus from which you can access virtually any type of textual information, including that provided by other Internet resources. There are many Gopher systems around the Internet, each one administered locally. Each Gopher contains whatever information the local Gopher people have decided to share.

While some Gophers are standalone systems, most Gophers are set up to connect to other Gophers. For example, say you are using a Gopher in California. With a simple menu selection, you can connect to another Gopher in Africa or in South America. What makes Gophers so powerful is that, no matter what Gopher you are using and no matter what information you are using, the interface is always the same simple menu system.

Veronica and Jughead

Nobody really knows how many Gophers there are in the world. Suffice it to say there are a great many, all of which have their own series of menu items offering information and services.

Veronica is a tool that keeps track of many Gopher menus from around the world. You can use Veronica to perform a search and look for all the menu items that contain certain keywords (whatever you specify). Jughead does the same thing for a specific group of Gopher menus.

The result of a Veronica or Jughead search is a custom menu, containing whatever items were found. Selecting any item from this menu automatically connects you to the appropriate Gopher, wherever it may happen to be. In fact, unless you specifically ask, you will not even know what computer you are using or what country it is in. Amazing.

Wais Servers

Wais servers provide another method of finding information that is spread around the Internet. Wais is able to access any of a large number of databases. To start, you tell Wais which databases you want to search. Next, you specify a list of one or more keywords to search for. Wais will search every word in every article in all the databases that you specified.

The result of a Wais search is a list of the articles, culled from the various databases, that are likely to be of interest to you. Wais presents them as a menu,

with the most relevant items first. From this list, you can ask Wais to display whichever articles take your fancy.

The name "Wais" stands for "Wide Area Information Service" and is pronounced "Wayz".

World-Wide Web

The World-Wide Web—often called "the Web"—is a hypertext-based tool that allows you to retrieve and display data based on keyword searches. What makes the Web so powerful is the idea of hypertext: data that contains links to other data.

For instance, as you are reading some information, you will notice that certain words and phrases are marked in a special way. You can tell the Web to jump to one of those words. It will follow the link, find the relevant information, and display it. In this way, you can jump from place to place, following logical links in the data.

White Pages Directories

Within the often overwhelming world of the Internet, nothing is more important than a person's electronic address. Once you know someone's address, you can send mail, have a Talk conversation, or even find out more about the person by using Finger.

What do you do when you want to contact someone, but you don't know his or her address? You use one of the White Pages Directories. The name, of course, reminds us of the standard telephone book.

However, the electronic counterpart is actually much different, principally because there is no single Internet directory. After all, as we explained in Chapter 1, nobody actually "runs" the Internet. With no single person or organization in charge, it is not surprising that there is no central source of names and addresses.

Rather, there are a number of different White Pages Directories—special-purpose servers—that you can search for a name. When you lose that bar napkin on which you carefully wrote someone's name during your last vacation abroad, there is a chance that a White Pages Directory may help you track down your quarry's electronic address, as long as he or she is an Internet user. (And why would you be associating with anyone who is not an Internet user?)

Electronic Magazines

The Internet is host to a variety of magazines that are published electronically. That is, the articles are stored as text files that are accessible to one and all. Some electronic magazines are scholarly journals of interest mainly to specialists. Other magazines are of general interest.



FUN TIP: If you want to read a strange collection of weirdness and esoterica, subscribe to The Unplastic News.

There are two ways in which electronic magazines are distributed. Some maintain a mailing list. When a new issue comes out, it is sent to you as a mail message. Other magazines are stored in well-known Anonymous FTP sites. You can download copies, including back issues, whenever you want.

Aside from the ease of distribution, electronic magazines have two important advantages over their conventional paper and ink counterparts. First, electronic magazines do not contain advertisements showing people who are richer and better looking than you, having more fun than you will ever have.

Second, electronic magazines do not contain irritating subscription cards that fall out when you turn the page.

Mailing Lists

A mailing list is an organized system in which a group of people are sent messages pertaining to a particular topic. The messages can be articles, comments, or whatever is appropriate to that topic.

All mailing lists—and there are thousands of them—have someone in charge. You can subscribe or unsubscribe to a list by sending a message to the appropriate address. Many mailing lists are "moderated", which means that someone decides which material will be accepted. Other lists will accept and send out messages from anybody.

If you feel lonely and ignored, subscribing to just a handful of mailing lists is guaranteed to keep your electronic mailbox filled to the Plimsoll line.

Internet BBSs

A BBS, or Bulletin Board System, is a repository for messages and files, often devoted to a particular topic. To use a BBS, you connect to it and select items from a series of menus.

Typically, a BBS will be maintained by a single person or by an organization. There are countless BBS systems in the world, most of them reachable by telephone. The Internet has many BBSs that you can reach by the more refined method of Telnet.

Games

Where would a computer system be without games? Of course, there are many computer games you can download via Anonymous FTP and run on your own computer. However, there are also special network games that take advantage of the Internet facilities.

For example, you can play the games of Chess and Go with other people from around the Internet. Or you can play the game of Diplomacy by electronic mail. Perhaps you would enjoy a simulated spaceship dogfight based on Star Trek. Maybe you would like to use your skills as a programmer to play Core War, a game in which players devise assembly language programs (for a virtual computer) that attempt to dominate and control the computer's memory.

Whatever your inclination, you can find it on the Internet. Just don't tell your boss or the powers that be. Certainly, the vast amounts of money and time that are spent maintaining the Internet are not expended merely so we can amuse ourselves with games. (Or are they?)

MUD

A MUD, or Multiple User Dimension, is a computer program that provides a virtual reality. To participate in a MUD, you telnet to a MUD server, take on a role, and explore. As you do, you interact with other users who are playing their own roles. In other words, a MUD allows you to exercise your fantasies and pretend, just like playing Dungeons and Dragons or visiting a singles' bar.

Rest assured, MUDs are at once complex, fascinating, and addictive. As you walk around the virtual reality of a particular MUD, you can talk with people, solve puzzles, explore strange places (like caves), and even create some reality of your own.



FUN TIP: Join a MUD and hide from reality. Forget to graduate. Lose your job. Ignore your friends, family, and loved ones. In other words, lots of good, clean fun.

Chapter	Resource	Description
5, 6	Electronic Mail	Send and receive messages
7	Remote Login	Connect to and use a remote host
8	Finger Service	Show information about a user
9–15	Usenet	Vast system of discussion groups
16	Anonymous FTP	Public access to data archives
17	Archie Servers	Search Anonymous FTP archives
19	Talk Facility	Converse with one person
20	Internet Relay Chat	Converse with a group of people
21	Gopher	Menu-based information
21	Veronica, Jughead	Search for Gopher menu items
22	White Pages Directories	Search for a user's address
23	Wais Servers	Search indexed databases
24	World-Wide Web	Access hypertext information
25	Mailing Lists	Information distributed by mail
*	Electronic Magazines	Magazines, journals, newsletters
*	Internet BBSs	Share information and messages
*	Games	Fun and diversions
*	MUD	Multi-person virtual reality

Table 2-1. *Summary of the Internet Resources*

**see the catalog later in this book*

What is TCP/IP?

In order to complete this chapter and our overview of the Internet, we need to spend a few moments talking about *TCP/IP*. As you know, the Internet is built on a collection of networks that cover the world. These networks contain many different types of computers, and somehow, something must hold the whole thing together. That something is TCP/IP.

The details of TCP/IP are highly technical and are well beyond the interest of almost everybody, but there are a few basic ideas you will want to understand.

To ensure that different types of computers can work together, programmers write their programs using standard *protocols*. A protocol is a set of rules that describes, in technical terms, how something should be done. For example, there is a protocol that describes exactly what format should be used for a mail message. All Internet mail programs follow this protocol when they prepare a message for delivery.

TCP/IP is the common name for a collection of over 100 protocols that are used to connect computers and networks. We have already mentioned two of the TCP/IP protocols, Telnet and FTP (file transfer protocol).

The actual name "TCP/IP" comes from the two most important protocols: *TCP* (Transmission Control Protocol) and *IP* (Internet Protocol). Although you don't need to know the details, it is useful to have an appreciation for what these protocols are and how they hold the Internet together.

Within the Internet, information is not transmitted as a constant stream from host to host. Rather, data is broken into small packages called *packets*.

For example, say that you send a long mail message to a friend on the other side of the country. TCP will divide the message into packets. Each packet is marked with a sequence number and with the address of the recipient. In addition, TCP inserts some error control information.

The packets are then sent over the network, where it is the job of IP to transport them to the remote host. At the other end, TCP receives the packets and checks for errors. If an error has occurred, TCP can ask for that particular packet to be resent. Once all the packets are received correctly, TCP will use the sequence numbers to reconstruct the original message.

In other words, the job of IP is to get the raw data—the packets—from one place to another. The job of TCP is to manage the flow and ensure that the data is correct.

Breaking data into packets has several important benefits. First, it allows the Internet to use the same communication lines for many different users at the same time. Since the packets do not have to travel together, a communication line can carry all types of packets as they make their way from place to place. Think of a highway in which separate cars all travel on a common road even though they are headed for different places.

As packets travel, they are routed from host to host until they reach their ultimate destination. This means the Internet has a lot of flexibility. If a particular connection is disrupted, the computers that control the flow of data can usually find an alternate route. In fact, it is possible that, within a single data transfer, various packets might follow different routes.

This also means that, as conditions change, the network can use the best connection available at the time. For example, when a particular part of the network becomes overloaded, packets can be routed over other, less busy, lines.

Another advantage of using packets is that, when something goes wrong, only a single packet may need to be retransmitted, rather than the entire message. This greatly increases the overall speed of the Internet.

All of this flexibility makes for high reliability. One way or another, TCP/IP makes sure the data gets through. In fact, the Internet runs so well that it may take only a few seconds to send a file from one host to another, even though they are thousands of miles apart and all the packets must pass through multiple computers.

Thus, there are several answers to the question, "What is TCP/IP?" The technical answer is that TCP/IP is a large family of protocols that are used to organize computers and communication devices into a network. The two most important protocols are TCP and IP. IP (Internet protocol) transmits the data from place to place, while TCP (transmission control protocol) makes sure that it all works correctly.

The best answer, though, is that the Internet depends on thousands of networks and millions of computers, and TCP/IP is the glue that holds it all together.

Hint

If you wonder what the route looks like from your computer to another, you can use the **traceroute** command. This command is not available on all systems. If your computer does have **traceroute**, just enter the command name followed by the address of an Internet host. (We will discuss Internet addresses in Chapter 4.) For example, to see the current route from your computer to the host whose address is **rtfm.mit.edu**, enter:

```
traceroute rtfm.mit.edu
```

The output will show you each step in the path between the two computers. If you try this command at various times, you may see different routes as conditions change.

CHAPTER 3

How to Connect to the Internet

It's stimulating to read about the wonderful things you can do once you have access to the Internet. But how do you get that access? Lying in bed at night, with visions of computerized sugarplums dancing in your head, it's natural to feel like Moses, perched on Mount Pisgah, staring down wistfully at the Promised Land. Shakespeare, of course, had the same problem when he was unable to get an Internet account. "Why," he asked plaintively, "should a man whose blood is warm within, sit like his grandsire cut in alabaster?"

Exactly. We couldn't have put it better ourselves.

So, in this chapter, we will show you how to connect to the Internet.

The Internet Connection

To start, let us consider the question, what does it mean to have access to the Internet?

This means you are using a computer that is a part of a network connected to the Internet. In practical terms, this means you can use the Internet resources we described in Chapter 2.

When you are using such a computer, we say that you are *on* the Internet. In common usage, the Internet is often referred to *the Net*.

For example, say that you are walking in the park one day and you happen to meet the woman (or man) of your dreams. Well, one thing leads to another and, after an enjoyable hour together, you part in a mutual flourish of good will and telephone numbers. However, you realize that the incipient relationship might well do better if you could correspond by electronic mail, so you ask, "Are you on the Net?"

> **Hint**
>
> Determining up front if a potential mate has access to the Internet can help you decide quickly and easily whether or not the relationship will be worthwhile.

It is important to realize that a great many people who do not have full Internet access are able to communicate with Internet users. For example, many people use some type of non-Internet mail service that can exchange messages with the Internet. It is also common to find people who are not on the Internet who have access to the Usenet discussion groups.

What you should understand is that while mail and Usenet are important, they are not everything. We want you to have access to all the Internet resources, including Telnet and FTP. In other words, we want you to be able to use a computer that has a full TCP/IP connection with the Internet.

Types of Internet Connections

Before we discuss the various ways in which you might obtain access to the Internet, we need to talk about the two different types of Internet connections.

First, you might use a computer that is directly connected to the Internet. For example, you might be using a PC or Macintosh or workstation that is part of a network connected to the Internet. In such a case, your computer will be a full-fledged Internet host, with its own electronic address.

The other way to connect to the Internet is by using a terminal that is connected to an Internet host. In this case, the terminal itself—not being a computer—is not on the Internet. You simply use the terminal to access a computer that is on the Internet.

Here is an example. Imagine that you are being taken on a tour of a building used by many Internet users. First, you are taken to the Computer Room, in which you see 40 different PCs, connected in a network. You are told that this network is connected to the Internet, so that all the PC users can access the Internet directly.

Next, you are taken to another room, the Terminal Room. Here you are shown 40 terminals, all of which are connected to a time-sharing computer in a closet at the back of the room. This computer is also connected to the Internet.

Thus, each PC user has his or her own Internet host. Each PC has its own Internet address and is a self-sufficient, standalone system.

In the Terminal Room, life is different. Each person is actually using the same computer, the one in the closet. These users access the Internet by logging in to the time-sharing system that provides Internet access. Thus, they all share one computer that has a single Internet address.

Telephone Connections

In the last section, we explained that you can use either a computer or a terminal to access the Internet. The example we used described two rooms. One contained 40 PCs connected into a network. The other contained 40 terminals connected to a time-sharing host. What they had in common was that all the devices—computers and terminals alike—were connected directly using some type of cable. This type of arrangement is called a *hard-wired* connection.

The main advantage of a hard-wired connection is its permanence. All you have to do is turn on your PC or terminal, and your connection is ready to use. The disadvantage, of course, is the lack of flexibility. If you want to move the computer or terminal to another location, you must deal with the cables.

A more flexible system is one in which the computer or terminal uses a *dial-up* connection over a telephone line. In such cases, you can work anywhere you want, as long as you have access to a phone line.

To use a dial-up connection, you need a hardware device to convert computer signals to telephone signals and back again. In technical terms, the

computer signal is "digital" and the telephone signal is "analog". A device that converts from digital to analog is called a *modulator*. A device that converts from analog to digital is called a *demodulator*. When we connect computers over a phone line, we must be able to send data in both directions. For this, we use a *modem*, a "modulator-demodulator".

There are various types of modems. A modem can be a separate box that attaches to your computer (usually via a "serial" cable). Or a modem might be an adaptor board that is installed in the computer. Figure 3-1 compares a hard-wired connection to a dial-up connection. Notice that a dial-up connection requires a modem at each end of the telephone line.

Hint

When you buy a modem for your own computer, you will encounter a bewildering number of technical terms. However, all you have to do is remember the following four guidelines:

- The speed of a modem is measured in *bits/second* or *bps*. Fast modems transmit at speeds up to 14,400 bps (often abbreviated as 14.4K bps). Slow modems transmit at 9,600 bps.
- Buy only a 14,400 bps modem. If you buy anything slower, you will be disappointed. No matter what anyone tells you, do not buy one of the very slow 2,400 bps modems (which were the standard about four or five centuries ago).
- You will see a number of strange terms that describe various functions: V.32bis, V.42, MNP 5, and so on. Don't worry about them. All modern 14.4K bps modems come with the right stuff. Just buy a 14.4K bps modem and you'll be okay.
- Modem manufacturers lie about the speeds of their modems, but they all lie in the same way, so it's okay. For example, it is commonly claimed that a 14,400 bps modem, using what is called "data compression", can achieve an overall performance of 57,600 bps (four times the nominal speed). Similarly, it is said that a 9,600 bps modem with data compression will perform at 38,400 bps. These numbers are highly inflated theoretical maximums that are completely divorced from reality. Most people do not know this, but now you do.

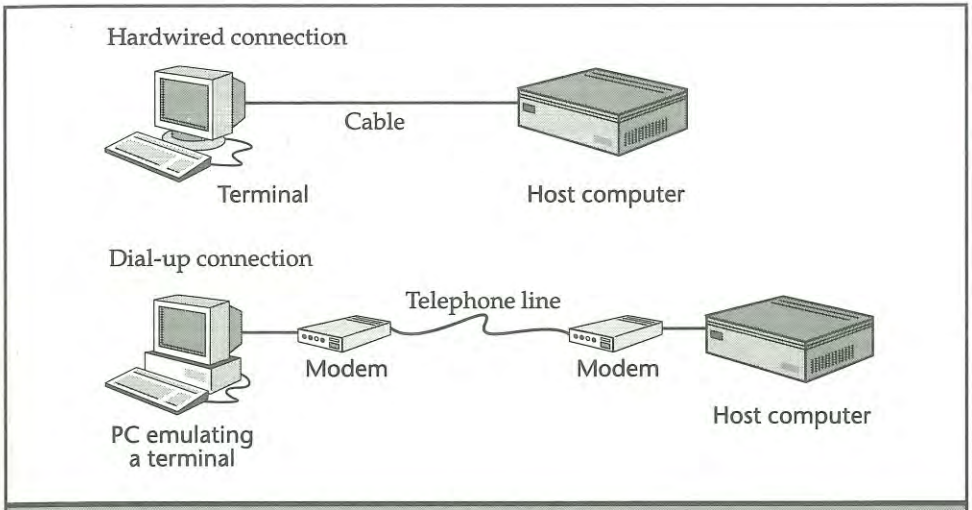


Figure 3-1. *Hardwired and dial-up connections*

Connecting a Terminal Over a Telephone Line

A common need is to be able to use a PC or Macintosh to access a remote computer over a telephone line. For instance, you may have an Internet host at work or at school that you want to use from your home.

Such hosts are usually configured to accept connections to a terminal. That is, you can connect a terminal to the telephone line (using a modem) and dial the remote host. Once you make the connection, you can work on your terminal in the usual manner.

The thing is, your PC or Macintosh is a full-fledged computer and not merely a terminal. But the remote host is set up to communicate only with terminals.

The solution is to run a program on your computer that *emulates* a terminal. In other words, you run a program that makes your computer act just like a terminal. As far as the remote host is concerned, your computer really is a terminal.

The standard way to do this is to use a *communications program* on your computer. Such a program will handle all facets of communicating over a telephone line. The program will help you keep a list of phone numbers, set up the proper communication options, dial the number, and generally run the show.

Most important, once a connection is established with the remote host, the communications program will emulate a terminal.

Now, there are many different types of terminals, and most programs will give you a choice as to which one you want to emulate. The standard choice is the *VT-100*. This is a terminal that used to be made by the Digital Equipment Corporation. It has been a long time since DEC actually made a VT-100, but nevertheless, it has become a standard. Indeed, many host computers, especially Unix systems, will assume that, by default, all dial-up connections use some variation of a VT-100, such as VT-102.

Hint

Just after you log in to an Internet host, you may see a short message that looks like this:

```
TERM = (vt100)
```

What is happening is that the system is asking you what type of terminal you are using. The name in parentheses, **vt100**, is the default. If you are using (or emulating) a VT-100 terminal, simply press the RETURN key. Otherwise, type the name of a different terminal and press RETURN. It is important to initialize your terminal type properly. Otherwise, the output on your screen may end up scrambled.

To summarize, if you want to use the Internet from your home, you need four things: a computer, a modem, a communications program, and the telephone number of a remote host to which you can connect. (We will give you guidelines on how to find such a system later in the chapter.)

Hint

There are two things that will make your work with a remote host a lot easier. First, choose a communications program capable of storing the information that is displayed on your screen. As new data is displayed, the previous data will scroll off the top of the screen. It is a great convenience to be able to ask your communications program to recall previous lines of output that otherwise would be gone for good. Sometimes such a facility is called a *scroll buffer*.

Second, use a computer system that will run more than one program at the same time. For example, with a PC, you should run your communications program under Microsoft Windows or, better yet, OS/2. This allows you to keep an Internet session in its own window while you do other work. Moreover, such systems enable you to cut and paste from one part of the screen to another, which can be a great advantage.

For example, let's say that you are reading a Usenet discussion group in which someone has mentioned the address of a computer that has a great new service. You would like to enter a command to connect to that computer to try out the service.

Normally, you would type the command, including the computer's address. However, Internet addresses can be long and must be typed exactly right. Instead, you can "cut" the address from one part of your screen and "paste" it back as part of the command. That way, you do not have to worry about making a spelling mistake. Once you become a veteran Internet user, you will find that such situations arise frequently.

Connecting a Computer over a Telephone Line

Most of the time, using a computer that emulates a terminal is the best way to connect to an Internet host over a telephone line. But remember, your computer is merely acting like a terminal. It is not on the Internet itself.

In certain situations, your computer should be an actual Internet host. For example, you might have a small company that cannot afford a network with a dedicated full-time Internet connection. However, you may still want your own Internet host. In such cases, there is a way to establish a full Internet connection over a telephone line.

To do so, you must first arrange for some other Internet host to act as your connection point. You then install a set of programs on your computer called *PPP* (Point to Point Protocol). Once a telephone line connection is made between the two computers (using modems, of course), PPP will endow your computer with TCP/IP capabilities. This enables your computer to be a real Internet host with its own official electronic address.

You will also hear about an older system called *SLIP* (Serial Line Internet Protocol). Both PPP and SLIP will work, but if you have a choice, use PPP.

When you use these systems, there are two ways to set up the telephone line. First, you might use a regular dial-up line with a standard modem, just as we described in the previous section. Although your computer will be considered an official Internet host, it will not be connected at all times. When such a system is set up, you must arrange for the computer that acts as your connecting point to save mail messages that arrive when you are not connected.

The alternative is to use a *dedicated phone line*. As the name implies, a dedicated phone line is always connected. This, of course, costs a lot more than a regular line. However, for a small company, a dedicated phone line using PPP can be a relatively economical way to establish Internet access. This connection can then provide Internet access for the rest of the computers in the company. Nevertheless, if there is to be a fair bit of Internet traffic, you are probably better off with a faster connection.

Later in the chapter, we will explain how certain companies are in the business of providing commercial access to the Internet. Many such companies offer, not only a host computer to which you can connect with a PC, but PPP or SLIP service for customers who require a full Internet connection.

Hint

If you want to access the Internet from your home, use a communications program and emulate a terminal. Establishing a PPP or SLIP connection sounds nice, but it is expensive and requires a fair bit of technical knowledge. When you use a terminal emulator to connect to a remote host, someone else is taking care of all the details for you. When you use PPP or SLIP, you will have to manage the system yourself.

Are You Already On the Internet?

Now that we have described the various ways in which a computer or terminal can connect to the Internet, we will spend the rest of the chapter explaining how you might go about getting Internet access for yourself.

To start, we will discuss the easiest solution of all. It may be that you already have access to the Internet and do not know it.

How can this be? Well, if you are using a computer that is part of a network in a university, you are probably already on the Internet. Virtually all universities have Internet facilities. Alternatively, if you work at a company or organization whose computers are networked together, there may be an Internet connection already set up. This is especially true for computer companies.

To find out if you are on the Internet, ask your system manager or a friendly computer nerd. The rule of thumb is that the type of person who would use a computer on a Saturday night is the type of person who would know if your installation is on the Internet. (Don't laugh. Once you learn how to use the Internet, you will find yourself using the computer on a Saturday night.)

Another way to check your Internet status is to enter a command that tries to connect to a remote computer and see what happens. You might try the **telnet** command (see Chapter 7). When you do, be sure to specify the name of a computer outside of your local area.

The **telnet** command is basic to the Internet. If you enter a **telnet** command and you see a message like this,

```
telnet: Command not found.
```

it means that the command does not exist on your system and you are probably not on the Internet.

If you find that you are already on the Internet, you can bask in the glow of your good fortune and move on straightaway to Chapter 4. Otherwise, read on for some ideas to help you get Internet access.

Using the Internet for Free

If you do not already have Internet access, there are several ways to get it. Let's start with those that are free.

As we mentioned, virtually all universities—and a lot of companies—have access to the Internet. To get free access, all you have to do is convince somebody

to give you an account on one of the computers. If you belong to such an organization, find out who is in charge of computer accounts. Then ask around discreetly and find out what criteria one has to meet to get an account. In some universities, for example, any student who asks will be given a computer account, but you have to ask. In other schools, you need to be taking a particular class or working on a particular project. Sometimes if you work for a professor you can get an account. Similarly, some companies give Internet access to any employee. Other companies give it only to qualified employees (such as researchers).

Hint

If anyone asks you why you want Internet access, the correct response is, "I want to be able to access free information that will be of great benefit to our organization." The incorrect response is, "I want to be able to read the jokes on Usenet."

If you are not a student, but there is a university nearby, you may be able to get someone to give you free access to one of the university computers. This may involve some finesse and will certainly demand discretion. But, as one of our readers, you are certainly not lacking in either of these qualities.

Community Computing and the Freenets

An alternative to sponging off a local university is to look for a free community computing service. Such services offer computer-mediated communication facilities to anyone for no charge. All you need is a computer with a modem. Using your modem, you connect to a host computer over a telephone line. The best known community computing services are the *Freenets* (sometimes spelled "Free-Nets"). There are a number of Freenets in the United States and Canada that offer all sorts of information services as a general service to the community. The original system is the Cleveland Freenet. Look in the catalog later in this book for more details. (Note: Not every such system has the word "Freenet" in its name.)

The Freenet system has given rise to an organization called the *National Public Telecomputing Network* or *NPTN*. The NPTN is a non-profit organization dedicated to providing free public access to a large variety of information.

Community computing services usually have an electronic mail system that can communicate with the Internet. Most of them also carry the Usenet discussion groups. Moreover, they have a number of unique information offerings, including special programs for schools. Perhaps their best feature is the price: they are free.

However, there are a number of important disadvantages. First, these systems do not have full Internet access. For example, a Freenet user cannot usually connect to a remote Internet host or use a Gopher.

Second, unlike the Internet, community computing services have Someone-In-Charge, just like the commercial services such as CompuServe or Prodigy. This means that there are rules and there is censorship (although it is usually not heavy-handed).

The Freenets are typically connected to a local university computer system. This is how they exchange mail messages with the Internet and how they access Usenet. Although their users cannot connect to remote Internet systems, Internet users can use a Freenet by using Telnet. See the catalog for details.

Public Internet Access

When all else fails, you can always get Internet access by paying for it. There are many companies or organizations that will provide such access. Typically, you must pay a certain amount per month, along with a fee for *connect time*, that is, the actual time you are using the service. Here are some suggestions as how to find such a service and how to make a good choice.

First, try a local university. Some universities offer Internet access to the general public. Even if you have to pay, the service will probably cost a lot less than from a commercial provider.

If this is not available, check the list in Appendix A. You will notice that some companies offer services to a large area, for example, the entire United States. Other companies are regional in scope.

Start by looking for a company that offers a phone number that is a local call for you. You can expect to spend many hours on the Internet, so it is best not to have to pay long distance charges.

Hint

You will find that some companies have toll-free numbers that you can use from anywhere in the country. However, if you compare closely, you will find there is a surcharge for using such a number. This surcharge may amount to more than paying for the long distance yourself.

After you have found a few likely Internet providers, call them and ask:

- What are your rates?
- Do you offer full Internet service?

Make sure that you do not have to settle for only electronic mail and Usenet. You want the whole set of resources that we described in Chapter 2. The question to ask is, "Do you offer Telnet and FTP?"

You should plan on spending many hours a month glued to the Net. Thus, look for a company that offers as many hours of connect time as you need for a flat fee. If you are not yet an experienced Internet user, take our word for it: the Internet will consume a lot of your time, and paying an hourly rate can get expensive. You may find that paying a long distance charge to call a flat-fee service is cheaper than using a local service that charges by the hour. This is especially true if you call at night or on the weekend when the rates are lower.

Hint

When choosing an Internet provider, look for:

- The full range of Internet services
- A local phone number to dial
- A flat monthly fee for unlimited connect time

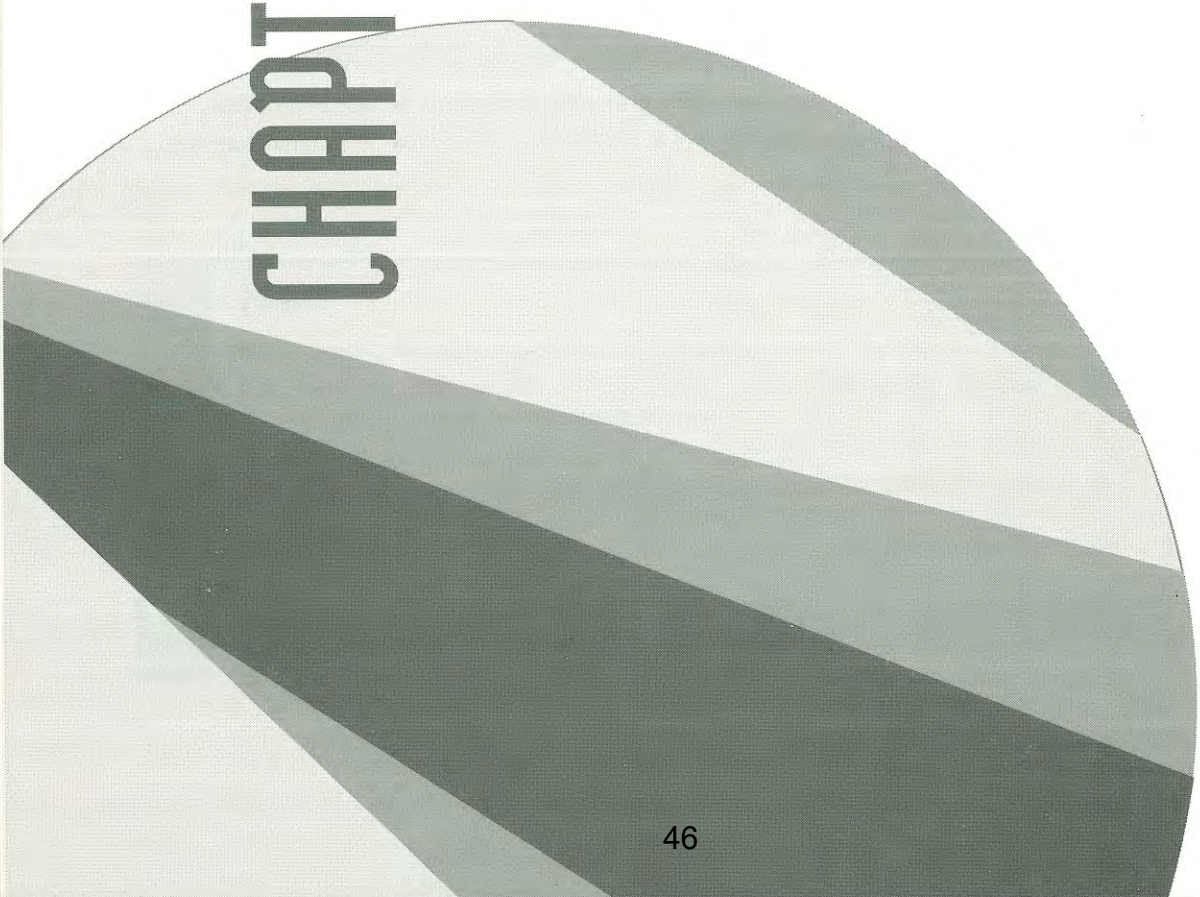
If you must call long distance (within the U.S. or Canada), check with the various phone companies to see which will offer you the most economical calling plan.

If you have access to Usenet and to electronic mail, there is a discussion group specifically to help people find Internet access in their area. The name of the group is **alt.internet.access.wanted**. This is the place to send a query like "Can someone please tell me where to find public Internet access in the Fuzzballville area?" If you do not have access to Usenet, perhaps you can find a friend to ask the question for you.

Finally, there is a large, up-to-date list of companies that provide public dial-up Internet access. This list is posted regularly to several Usenet groups and is also available by Anonymous FTP and by mail. See Appendix A for the details. If you are just starting, you might prevail upon an experienced Internet person to help you download this list.

One service to watch for is *Public Access Unix*. As we discussed in Chapter 1, most Internet computers run Unix. In some cities, there are companies that provide commercial access to a Unix system. These systems will typically offer some type of Internet access.

CHAPTER 4



Internet Addressing

Whatever you do, make sure you read this chapter. Here is why: Every computer that is on the Internet has its own unique address. Likewise, every person who uses the Internet has his or her own address. You must learn how to understand such addresses.

In the non-computer part of our life, we need to remember different types of information to be able to communicate with someone: a postal address, perhaps a separate street address, a home telephone number, a business telephone number, a fax number, and so on.

47

On the Internet, there is only one type of electronic address. Once we know someone's Internet address, we can send mail, transfer files, have a conversation, and even find out information about that person. Conversely, once you start to use the Internet, you need give other people only one simple address in order for them to be able to communicate with you.

Moreover, when someone wants to tell you where on the Internet certain information or a particular resource is to be found, he or she does so by giving you the address of a computer. For examples, look at the various entries in the catalog.

For these reasons, understanding the Internet addressing system is crucial to using the Internet. In this chapter, we will show you everything you need to know.

Standard Internet Addresses

On the Internet, the word *address* always refers to an electronic address, not a postal address. If a computer person asks for your "address", he or she wants your Internet address.

Internet addresses all follow the same form: the person's userid, followed by an @ character (the "at" sign), followed by the name of a computer. (Every computer on the Internet has a unique name.) Here is a typical example:

```
harley@fuzzball.ucsb.edu
```

In this case, the userid is **harley**, and the name of the computer is **fuzzball.ucsb.edu**. As this example shows, there are never any spaces within an address.

You will remember that in Chapter 2 we explained that each person has a user name called a userid (pronounced "user-eye-dee"). It is this userid that we use as the first part of someone's address. If you have a Unix system, your userid will be the name that you use to log in.

The part of the address after the @ character is called the *domain*. In this case, the domain is **fuzzball.ucsb.edu**.

Thus, the general form of all Internet addresses is:

```
userid@domain
```

As you might imagine, a userid by itself is not necessarily unique. For instance, within the entire Internet, there are probably a number of people lucky enough to have a spiffy userid like **harley**.

What must be unique is the combination of userid and domain. So, although there may be more than one **harley** on the Internet, there can be only one such userid on the computer named **fuzzball.ucsb.edu**.

If you read an Internet address out loud, you will see that using the @ character is appropriate. For example, let's say that you wanted to send mail to the person at the address we just mentioned. The command to do so is:

```
mail harley@fuzzball.ucsb.edu
```

As you enter this command, you can say to yourself, "I am sending mail to Harley, who is at the computer named **fuzzball.ucsb.edu**".

Sometimes an address in this form is called a *fully-qualified domain name* or *FQDN*.

Understanding a Domain Name: Sub-Domains

In the last section, we used **harley@fuzzball.ucsb.edu** as an example of an Internet address. We said that **harley** is the userid and **fuzzball.ucsb.edu** is the domain. Each part of a domain is called a *sub-domain*. As you can see, sub-domains are separated by periods. In our example, there are three sub-domains: **fuzzball**, **ucsb** and **edu**.

The way to understand a domain name is to look at the sub-domains from right to left. The name is constructed so that each sub-domain tells you something about the computer. The rightmost sub-domain, called the *top-level domain*, is the most general. As you read to the left, the sub-domains become more specific.

In our example, the top-level domain **edu** tells us that the computer is at an educational institution. (We will explain the meanings of the various top-level domains in a moment.) The next sub-domain, **ucsb**, tells us the name of this institution (the University of California at Santa Barbara). Finally, the leftmost sub-domain is the name of a specific computer named **fuzzball**. Thus, as you enter the command:

```
mail harley@fuzzball.ucsb.edu
```

you can say to yourself, "I am sending mail to Harley, at a computer named **fuzzball**, at the University of California at Santa Barbara, which is an educational institution."

When you type an address, you can mix upper- and lowercase letters. (With computers, *uppercase* refers to capital letters; *lowercase* means small letters.) For example, the following two addresses are equivalent:

```
mail harley@fuzzball.ucsb.edu
mail harley@FUZZBALL.UCSB.EDU
```

There are two common variations that you will see. First, some people type only the top-level domain in uppercase:

```
mail harley@fuzzball.ucsb.EDU
```

Other people like to emphasize the site of the computer:

```
mail harley@fuzzball.Ucsb.Edu
```

In all cases, the uppercase letters are optional.

Hint

As a general rule, use all lowercase for Internet addresses. There is really no need to mix in uppercase letters.

If you see an address in which some of the letters are uppercase, it is always safe to change them to lowercase.

If you do decide to use some uppercase letters, it is best not to change the userid. On some systems, it may make a difference (although it's not supposed to).

Variations on the Standard Internet Address Format

All Internet addresses follow the standard format:

```
userid@domain
```

However, there are several variations you may encounter. The example we have been using has three sub-domains:

```
harley@fuzzball.ucsb.edu
```

You will often see addresses that have more sub-domains in order to be more specific. Here is an example:

```
scott@emmenthaler.cs.wisc.edu
```

In this case, the userid is **scott**. The domain refers to a computer **emmenthaler** that is part of the Computer Science department at **wisc**, the University of Wisconsin (presumably, an educational institution).

Many Internet sites use patterns to name their computers. For example, you might see computers named after cartoon characters, mythical heroes, local landmarks, or whatever. At the University of Wisconsin, many computers are named after types of cheese.



FUN TIP: As all students of world geography know, it is the custom within the United States for each state to have its own nickname. For example, California is the Golden State and New York is the Empire State. The reason that there are so many "cheese" computers at the University of Wisconsin is that Wisconsin is known as the Dairy State.

(In other words, by the time someone got around to Wisconsin, all the good nicknames were already taken.)

Some Internet addresses have only two sub-domains (the minimum). Here are two examples:

```
rick@tsi.com
randy@ucsd.edu
```

When you see an address with only two sub-domains, it can mean two things. First, it might mean that the organization is so small that it only has one computer on the Internet. In the first example, this is the case. The top-level domain **com** tells us that this is a commercial organization. (More about top-level domains in a moment.) The other sub-domain, **tsi**, is the name of the computer. It happens that this is a company named Technology Systems Integrators—talk about snazzy names—that has only one computer on the Internet.

The second example is from an organization that has a great many computers. In such organizations, one computer is usually used to send and receive all the outside mail. In our example, the name of this computer is **ucsd.edu**. The system administrators for this organization have simplified the mailing addresses by arranging for everyone to be able to receive mail addressed to **ucsd.edu**. Here is how it works.

In general, the term *gateway* refers to a a link between two different systems. In this case, we have a *mail gateway*. The computer **ucsd.edu** acts as the mail link between the internal network and the outside world. The mail gateway has a list of userids and local addresses. When a message arrives, the gateway can check the list and forward the message to the appropriate local computer.

For example, say that a person has a userid of **melissa** on a computer named **misty**. Normally, her address would be:

```
melissa@misty.ucsd.edu
```

However, in order to make her mail address simpler, she registers with the mail gateway. From then on, she can receive mail at **melissa@ucsd.edu**. When mail arrives, the gateway will automatically forward it to the **misty** computer.

So, as you can see, an address with only two sub-domains usually means that the organization is very small (like **tsi.com**) or very large (like **ucsd.edu**).

We explain this in detail because there may be times when you need a person's exact address, for example, to contact them using the **talk** command (which we discuss in Chapter 19). In such cases, the simplified mail address may not work and you will have to ask for a longer, more specific one.

To conclude this section, we will discuss one final form of Internet addressing that you will sometimes see in a mail address. In this form, a % (percent) character is used as part of the address. In such addresses, the % character will be to the left of the @ character. For example:

```
melissa%misty@ucsd.edu.
```

The idea is that the computer that receives the message (in this case, **ucsd.edu**) will look at everything to the left of the @ character (in this case, **melissa%misty**) and try to make sense out of it.

Usually, the % character will separate a userid from the name of a local computer. In this example, userid **melissa** uses a local computer named **misty**. Within the local network, there may be several different connections from the mail gateway to that computer. At the time the message is received, the mail gateway will choose the best path to use to deliver the mail.

Hint

Many people know their Internet address even though they don't understand it. Some organizations have more than one way to address mail, and the system managers will usually tell their users which address works best.

So don't worry too much about the variations. When you send mail to someone, just use whatever address the person gave you.

Top-Level Domains

As we mentioned earlier, the way to understand an address is to read it from right to left. The top-level domain will be the most general specification. In the example we looked at earlier:

```
mail harley@fuzzball.ucsb.edu
```

the top-level domain of **edu** told us that the computer was at an educational institution. We also looked at another address:

```
rick@tsi.com
```

in which a top-level domain of **com** indicated a commercial organization.

In general, there are two types of top-level domains: the old-style *organizational domains* (as in these two examples) and the newer *geographical domains*.

The organizational domains are based on an addressing scheme that was developed before the days of international networks. It was intended to be used mainly within the United States.

The idea was that the top-level domain would show the type of organization that was responsible for the computer. Table 4-1 shows the various categories. All of these categories have been around since the beginning of the Internet, except **int**, which is a relatively recent addition for certain organizations that span national boundaries (such as NATO).

Once the Internet expanded internationally, new, more specific top-level domains became necessary. To meet this need, a new system of geographical domains was developed in which a two-letter abbreviation represents an entire country. There are many such top-level domains—one for every country on the Internet—and they are all listed in Appendix F. For quick reference, Table 4-2 shows a representative sample.

As an example, take a look at the following address:

```
michael@music.tuwien.ac.at
```

Here we have a computer at the Technical University of Vienna in Austria (top-level domain of **at**).

Some countries use a sub-domain, just to the left of the top-level domain, to divide it into categories. For example, you might see **ac**, referring to an academic organization or **co**, referring to a commercial company. In our example, we see that the Austrian address uses **ac**.

Domain	Meaning
com	commercial organization
edu	educational institution
gov	government
int	international organization
mil	military
net	networking organization
org	non-profit organization

Table 4-1. *Organizational Top-Level Domains*

Domain	Meaning
at	Austria
au	Australia
ca	Canada
ch	Switzerland ("Cantons of Helvetia")
de	Germany ("Deutschland")
dk	Denmark
es	Spain ("España")
fr	France
gr	Greece
ie	Republic of Ireland
jp	Japan
nz	New Zealand
uk	United Kingdom (England, Scotland, Wales, Northern Ireland)
us	United States

Table 4-2. *Examples of Geographical Top-Level Domains*

For the most part, the geographical domains are simply the standard two-letter international country abbreviations (the ones that everybody in the world knows about except the Americans). The exception to this scheme is Great Britain. Its international abbreviation is **gb**, but it also uses a domain name of **uk** for "United Kingdom".

This makes sense when you remember that Great Britain includes England, Scotland, and Wales, while the United Kingdom also includes Northern Ireland. (Evidently, the English use some alternative meaning of the word "united" that nobody else understands.)

As you can see from Table 4-2, the United States does have a geographical domain name (**us**), although it is not used much. Outside the U.S., though, geographical names are used almost exclusively. However, regardless of what type of top-level domain your organization uses, you can communicate with any address on the Internet. Both types of top-level domains are recognized everywhere.

Hint

Within Britain and New Zealand, the order of domains is often reversed. For example, you may see an address like:

```
peirce@uk.ac.oxford.compsci
```

The mailing system is supposed to reverse the domains in order to communicate with the outside world. However, occasionally, a reversed address will escape from one of these countries, like a bad table wine exported for foreign consumption. If you need to use such an address outside its native environment, be sure to reverse the sub-domains:

```
peirce@compsci.oxford.ac.uk
```

so that they form a standard address.

Pseudo-Internet Addresses

There are many organizations that would like to be on the Internet, but do not have the time or the money to maintain a permanent Internet connection. As an alternative, they make an arrangement with a nearby Internet site that agrees to act as a mail gateway for them. For example, a small company might make such an arrangement with an Internet access provider (see Chapter 3) or a university.

With such an arrangement, the organization can be given a mailing address that looks just like a standard Internet address. However, such organizations are not really on the Internet.

For example, say that the Marlinspike Consulting Company would like an Internet address. It contracts with SnowyNET, a local Internet access provider, to handle its mail. As part of the arrangement, the company is given the name **marlin.com**, which is registered with the Internet addressing system.

Any mail that is sent to **marlin.com** is automatically routed to the SnowyNET gateway computer. At certain times, the **marlin.com** computer connects over a telephone line with the SnowyNET computer and picks up its mail. At the same time, any mail from **marlin.com** that is being sent outside the company is passed to the SnowyNET computer.

A user at a company with such an arrangement will use an address that looks just like a standard Internet address. For example, someone might tell you that his address is:

```
tintin@marlin.com
```

When you see such an address, there is no way of knowing whether or not it represents a real Internet address. In other words, there is often no way to tell, from the address alone, if a computer is actually on the Internet. If you see a name like:

```
small-company-name.com
```

you might be suspicious. However, even some large companies use a gateway rather than a real Internet connection.

If it is important that you know for sure if a computer is on the Internet, you can use the **host** command, described in the next section.

IP Addresses: The host Command

So far, we have talked about Internet addresses in which each sub-domain is a name. However, underneath the socially acceptable veneer of this system lies a typical computer trick: the real Internet addresses are actually numbers, not names. For example, the computer **ucsd.edu**, which we mentioned earlier, is actually called **128.54.16.1**.

Names, of course, are easier for people to use, but every time you use a domain address, your system has to turn it into a number, although the details are hidden from you. You will recall that in Chapter 2 we explained that the part of the Internet that moves data packets from one place to another is called IP (Internet Protocol). For this reason, the numeric version of an address is called an *IP address*. For example, the computer **ucsd.edu** has an IP address of **128.54.16.1**.

As you can see, an IP address looks something like the domain addresses that we have already discussed in that there are several parts separated by periods. However, the parts of an IP address do not correspond directly to sub-domain names, so don't read too much into the pattern.

You can use an IP address anywhere you would use a regular address. For example, the following two mail commands are equivalent:

```
mail randy@ucsd.edu
mail randy@128.54.16.1
```

The part of the Internet that keeps track of addresses is called the *Domain Name System* or *DNS*. DNS is a TCP/IP service that is called upon to translate domain names to and from IP addresses. Fortunately, it is all done behind your back, and there is no reason to bother with the details.

In rare cases, it may be that your system has trouble understanding a domain address. If this happens, the IP address may work better (as long as it is correct).

If you would like to test the DNS, you can use the **host** command. There are two formats:

```
host standard-address
host IP-address
```

If you specify a standard address:

```
host ucsd.edu
```

DNS will display the IP address. If you specify an IP address:

```
host 128.54.16.1
```

DNS will display the standard address.

Hint

Only real Internet hosts have IP addresses. Thus, you can use the **host** command to check if a computer is on the Internet. If **host** displays an IP address, the computer you specified is on the Internet.

If you specify an address that is not on the Internet, you will see:

```
Host not found.
```

If this happens, be sure to check your spelling.

Hint

The **host** command is not available on all systems. If your system does not have **host**, try **nslookup**.

UUCP

If you have read the previous sections of this chapter, you have learned just about everything you need to know about standard Internet addresses. However, we do want to spend a little time discussing addressing schemes used by a few other networks. As an Internet user, you can exchange mail with these networks, and it will help to understand what type of addresses they use.

To start, we will discuss the Unix-based UUCP network. We will then conclude the chapter by explaining what addresses to use to send mail to other popular networks: CompuServe, MCI Mail, FidoNet, and Bitnet.

All Unix systems come with a built-in networking system called *UUCP*. UUCP is a family of programs. The name UUCP comes from one of these programs, **uucp**. The **uucp** program copies files from one Unix system to another. Thus, the name means "Unix to Unix copy".

(You may see a similarity here to the name TCP/IP. As we mentioned in Chapter 2, TCP/IP is a large family that is named after its two most important members, TCP and IP.)

UUCP is not as powerful as TCP/IP. For example, UUCP does not provide remote login. Moreover, as we will see in a moment, the UUCP mail facility is slower and more awkward than the Internet system. However, UUCP does have an important advantage. It is a standard part of Unix (free with most systems), and it runs cheaply and reliably over dial-up or hardwired connections.

UUCP works by allowing Unix systems to connect together to form a chain. For example, say that you are using a computer named **alpha**. Your computer is connected to another computer named **beta**. This computer is connected to **gamma**, which is, in turn, connected to **delta**.

You decide to send mail to a person with a userid of **murray**, who uses **delta**. You send out the message from your computer, **alpha**. UUCP will pass the message from **alpha** to **beta** to **gamma** to **delta**, where it will be delivered to userid **murray**.

Our example involves four computers and three different connections. It might be that these are hardwired connections, in which the computers are joined by a cable. Typically, though, the connections are made over a telephone line. At certain intervals, each computer calls (or is called by) its neighbor. When

they connect, they swap whatever mail they have waiting for one another. Some of the mail will be for local users. Other mail will need to be forwarded to another computer.

The system works well in that it provides an economical way to send mail from computer to computer over large distances. However, there is an important limitation: since many UUCP connections are made over a telephone line at certain predefined times, mail delivery can take hours or even several days.

Compare this to the Internet, in which connections are permanent and messages are transmitted quickly, often within seconds, usually within minutes.

UUCP Addresses and Bang Paths

Many sites which at one time depended on UUCP for mail and file transfer now use the faster, more dependable Internet. However, there are still a lot of UUCP installations, so it is a good idea to know something about their addresses.

To send mail to a UUCP address, you must specify the route that you want the message to take. For example, you must say, "I want this message to go to computer **beta**, and from there to computer **gamma**, and from there to computer **delta**. At that point, I want the message delivered to userid **murray**."

To do so, you construct an address consisting of each of these names in turn, separated by **!** (exclamation mark) characters. For example, here is a **mail** command that will send a message to the userid we just described.

```
mail beta!gamma!delta!murray
```

When you create such a message, your system will store it until contact is made with computer **beta**, at which time the message will be sent on its way.

In Unix terminology, one of the slang names for the **!** character is *bang*. Thus, a UUCP address that specifies multiple names is sometimes called a *bang path*. When a Unix person reads such a path out loud, he will pronounce the **!** character as "bang". For example, you might hear someone say, "I tried to send you mail at beta bang gamma bang delta bang murray."

Hint

In Unix systems, the program that reads and interprets the commands you enter is called a *shell*. Some shells—especially the C-Shell—recognize a ! character as part of a facility called *history substitution*. This facility allows you to recall and edit previously entered commands. As you might imagine, history substitution can be a real time-saver.

However, this means that the ! character has a special meaning, and when you type this character as part of a UUCP address, it will cause an error. For example, if you enter:

```
mail beta!gamma!delta!murray
```

the C-Shell will try to interpret the command as a history substitution request. We won't go into the details here, except to say that you will see an error message like this:

```
gamma!delta!murray: Event not found.
```

(The term "event" refers to a previously entered command.)

Thus, if you are using a shell like the C-Shell, you need to tell it that the ! characters in a UUCP address are to be taken literally. To do so, you preface each ! with a \(backslash) character:

```
mail beta \!gamma\!delta\!murray
```

The \ characters are not really part of the address. They are there only to tell the shell not to misinterpret the ! characters.

Simplified UUCP Addressing

As we explained in the previous section, UUCP is inexpensive and accessible to anyone with a Unix system, a modem, and another computer with which to connect. Indeed, before the Internet became so popular, many people used to

send mail over a large, worldwide UUCP network. Today, many of these people have migrated to the Internet, but there are still a large number of computers out there that are reachable only by UUCP.

One of the problems with UUCP addressing is that the addresses can be long. Moreover, you must specify an exact path from one computer to the next. In the last section, we looked at a sample **mail** command that specified a path via three different computers.

```
mail beta!gamma!delta!murray
```

This is okay, albeit inconvenient, as long as you know which path to use. But many UUCP paths are much longer. Moreover, how do you know how to construct the path?

Let us say, for example, that you have a friend with a userid of **albert** who uses a computer called **gendeau.com**. If he were connected to the Internet, you would send mail to him by using the command:

```
mail albert@gendeau.com
```

But suppose he is on the UUCP network, not the Internet. How do you have any idea what path to use from your computer to his?

In general, this is a big problem with UUCP because the path to a computer depends very much on where you start. Thus, if your friend wants to correspond with people at different locations, he might have to give each of them a different address. The nice thing about the Internet and the Domain Name Service is that all you have to do is specify the address of the destination. The system automatically figures out the best route to take.

To bring the same sort of convenience to UUCP, an undertaking was started called the *UUCP mapping project*. This project regularly publishes "maps" of data that are sent to many key UUCP computers. When UUCP mail reaches these computers, they can look at the maps and decide the best route to use. In essence, this allows you to use a UUCP address that is similar to an Internet address, and let the system do the work.

Thus, on occasion, you may see an address that uses a top-level domain of **uucp**. For example, a friend might say that you can send mail to him at:

```
albert@gendeau.uucp
```

When you use such an address, it is a signal to the mail routing software to find the name in the UUCP mapping data and figure out the best path to use.

Your computer may be able to do this itself, or it may send it to another computer to do the job.

In practice, all you have to do is get the address right and it should work. But if you ever have a choice between a UUCP address and an Internet address, choose the Internet one.

Sending Mail to Other Networks

The Internet has mail gateways to a large number of other networks. As long as you know the right way to address mail, you can send messages to people on these networks via their gateway. Some of these gateways serve commercial networks that charge for their services. Nevertheless, as an Internet user, you can use the gateway for free.



FUN TIP: No matter how much a user of a commercial network might pay for the privilege of sending and receiving mail from you, as an Internet user you will never have to pay anything.

To conclude this chapter, we will discuss a few of the most widely used mail gateways, and we will show you how to use an Internet-style address to send mail to users on these networks.

Let's start with two popular commercial systems, CompuServe and MCI Mail. A user from one of these networks has his or her own account number. Here is how to convert this account number into a proper Internet address.

A CompuServe account number consists of a sequence of digits, with a comma somewhere within the sequence. For example:

```
12345,678
```

To send mail to a CompuServe account, all you have to do is replace the comma with a period, and use a domain of **compuserve.com**. Thus, to send mail to the user with the account we just mentioned, use the command:

```
mail 12345.678@compuserve.com
```

MCI Mail addresses are similar. Customers can use either an identification number or a user name. For instance, you might have a friend whose MCI Mail

number is **12345** and whose user name is **hhahn**. To send him mail, use either the number or the name with a domain of **mcimail.com**. For example:

```
mail 12345@mcimail.com
mail hhahn@mcimail.com
```

Another address that you might encounter is one from FidoNet. This is a worldwide network of personal computers that connect via telephone lines. (In principle, FidoNet is not unlike UUCP.) To reach FidoNet, you use a domain name that ends in **fidonet.org**. The name of the actual FidoNet computer is specified as a series of sub-domains. It works like this.

In the terminology of FidoNet, a computer name consists of three parts: a zone number, a net number, and a node number. The zone number is followed by a : (colon) character, and the net number is followed by a / (slash) character. For example, someone might tell you that his FidoNet computer is:

```
1:234/567
```

In this case, the zone number is **1**, the net number is **234**, and the node number is **567**.

To specify a FidoNet computer name from the Internet, you use these same three numbers, in reverse order, according to the following pattern:

```
fnode.nnet.zzone.fidonet.org
```

Within FidoNet, users are known by their full names. You separate each part of the name with a period. Thus, a person named Rick Shaw will have a user name of **Rick.Shaw**.

For example, to send mail to Rick Shaw at the FidoNet computer **1:234/567**, use:

```
mail Rick.Shaw@f567.n234.z1.fidonet.org
```

The last type of mail address that we will mention is for Bitnet users. Bitnet is a collection of different networks based in the United States, Canada, Mexico, and Europe. To send mail to a Bitnet user, you need to know his or her user name and host computer. From the Internet, the address is simple. Use a top-level domain of **bitnet**. To the left, put the name of the Bitnet host.

For example, say that you want to send mail to a friend whose Bitnet user name is **lunaea**. The name of her computer is **psuvm**. Use the command:

```
mail lunaea@psuvm.bitnet
```

If you hang around with Bitnet people, you will notice that a lot of the computer names end in "vm". This is because they are IBM mainframe computers using the Virtual Machine operating system.

The name **bitnet** is not an official Internet domain and is an example of what is called a *pseudo domain*. (The name **uucp** that we saw in the previous section is also a pseudo domain.) When you use an address with a pseudo domain, the mailing software on your system must recognize the domain, rewrite the address, and send the message to a computer that knows how to send mail to that particular network. In this case, your mailing program would have to send the message to a *Bitnet/Internet gateway*.

On some systems, this type of address will not work because the local mail software is not set up to recognize a **bitnet** pseudo domain. If this is the case on your system, you can send the message directly to any one of the Bitnet/Internet gateways. Here are several of them:

cornellc.cit.cornell.edu
cunyvm.cuny.edu
mitvma.mit.edu
pucc.princeton.edu
vm1.nodak.edu

There are two addressing formats that you can use. The preferable one uses the UUCP bang path notation:

```
gateway!computer.bitnet!userid
```

For example, to send a message to **lunaea** on the Bitnet computer named **psuvm**, you might use the address:

```
cornellc.cit.cornell.edu!psuvm.bitnet!lunaea
```

The second format uses the % notation that we discussed earlier:

```
userid%computer.bitnet@gateway
```

For example:

```
lunaea%psuvm.bitnet@cornellc.cit.cornell.edu
```

Technically speaking, this form of the address is not officially supported by the Internet, although such addresses will usually work.

CHAPTER 5

The Internet Mail System

By far, electronic mail is used more than any other Internet resource. Indeed, for many people, mail *is* the Internet.

In this chapter, we examine the Internet mail system and discuss everything you need to understand to get started. We begin with the basic ideas and concepts: What are the programs that provide the Internet mail service? What technical words and terms will you encounter?

From there, we complete the orientation by giving you some guidelines for using the mail system, and by explaining the technical information that is a part of each message. In Chapter 6, we will show

you how to use the standard Unix mail program. At that time, we will give you a hands-on description of how to send and receive messages.

The Basis for the Internet Mail System: The Transport Agent

The Internet mail system is the most important Internet resource. Every day, countless messages are sent from one part of the Internet to another. As you would expect, many of these messages are personal notes from one user to another. However, the mail system is a general service that can transport all types of information: documents, publications, computer programs, and so on. The only requirement is that the data be stored as ASCII characters. (That is, the data uses the regular set of characters that are on our keyboard.)

In some circumstances, it is possible to mail non-textual data, such as pictures or sound recordings. We will discuss this later in the chapter.

There are millions of people who use the Internet, and the mail system transports just about every type of information you can think of. One of the most significant uses of this system is to allow people who are not near one another to work together on a project. There are many projects in which people from all over the world participate. Indeed, it is common for someone to collaborate with a person who is thousands of miles away, someone they will never actually meet in person.

For example, as this book was being written (in California), each chapter was sent to a person in Austria and a person in Ireland, both of whom would review the work and make comments. The text of the chapter was sent as a mail message. Each reviewer would insert his comments right into the text and mail it back. Because the Internet is so fast and reliable, it was not uncommon for us to be able to review an entire chapter within a day. If we had to depend on standard post office mail, the whole process would take weeks.

The same process is carried out, on a much larger scale, when decisions have to be made that affect the Internet itself. A new idea or a new standard is proposed, after which anyone can participate in a worldwide debate via the mail system. (The Usenet discussion groups are also used for this purpose.)

Thinking about all of this, it is natural to wonder what holds it all together. After all, the Internet connects thousands of different networks, each of which has its own mix of computers and software. How can all these different systems work together to exchange mail?

The answer is that the delivery of mail is standardized by a system called *SMTP*. SMTP, which stands for *Simple Mail Transfer Protocol*, is part of the TCP/IP

family of protocols. It describes the format of a mail message and how messages are handled as they are delivered. Every Internet computer runs a mail program that works behind the scenes to ensure that messages are addressed and transported in an orderly fashion. This program, called a *transport agent*, follows the SMTP protocol and, in doing so, provides your mail link to the outside world.

On most systems, the transport agent runs by itself in the "background", always ready to respond to whatever requests it may receive. In Unix terminology, such a program is called a *daemon* (yes, that is the correct spelling).

Every Unix system has various daemons that are lurking in the background, quietly providing services for you. In theory, it doesn't matter what transport agent your system uses as long as it knows how to send and receive mail using SMTP. Most Unix systems use a daemon named **sendmail**.

Now, are we saying that you should understand the technical details of how all of this works? Of course not. What we do want you to appreciate is that the mail system works because everybody's network has at least one computer running a transport agent that sends and receives mail according to the SMTP standard. We also want you to know that on many systems the transport agent is a daemon named **sendmail**. You will never work with **sendmail** directly, but at least you know its name and what it does.

Your Interface to the Mail System: The User Agent

As a user, you do not interact with your system's transport agent; it operates behind the scenes sending and receiving mail via SMTP. The mail program that you use is called a *user agent*. It is the user agent that acts as your interface into the Internet mail system, allowing you to read your messages, compose new messages, delete messages that you have already read, and so on.

There are many different user agents used on the Internet. In this section, we will survey the more common ones. We will start with mail programs that you might find on a Unix system. We will then discuss the user agents that you might use on a PC or Macintosh. Which user agent you use to compose and read mail depends on what is available on your system. On some systems, there is only one mail program. On other systems, you have a choice.

The most widely available user agent is the Unix **mail** program. This program comes with all Unix systems and, hence, is available to most Internet users. Your system may have a better program—we will discuss a few of these in a moment—that you prefer to use. This is fine. Still, **mail** is a standard tool and is

the only user agent that you can expect to see on every Unix system. For this reason, we will discuss **mail** in more detail, in Chapter 6.

Hint

The first version of **mail** was primitive, offering only the most rudimentary functions to compose and read messages. Some years ago, newer versions of **mail** were developed that are far more powerful. At first, there were two versions of this program, one for each of the two main types of Unix: Berkeley Unix (known as BSD, for Berkeley Software Distribution) and System V Unix (which came from AT&T's Bell Labs).

The BSD user agent was called **mailx**. The System V user agent was called **Mail**. (As you may know, Unix distinguishes between upper– and lowercase letters, so **Mail** and **mail** are two different names.)

At first, users had to remember to use the right name. If they used **mailx** (or **Mail**), they would get the new version of the program. If they used **mail**—the more logical name—they would get the old version.

These days, the old program is rarely used. When you use the **mail** command, you will automatically get either **mailx** or **Mail**, whichever is on your system. There is a way to use the old program, but you will never need it.

Common Unix–Based Mail Programs

We have mentioned that **mail** is the standard Unix user agent, the program that you use to send and receive mail. However, other systems have other user agents and, if one is available, you may choose to use it instead of **mail**. Moreover, some Unix systems have more than the standard **mail** program. There are many such programs, and we won't discuss all of them. However, let's take a quick look at the most widely used ones.

One of the most popular user agents is named *Elm*. Elm is a full–screen program that is easy to use and simple to learn. Even if you have never used electronic mail before, you may be able to start using Elm without any documentation. As you become more experienced, you will find that Elm offers a lot more than simple mail handling. There are all kinds of advanced features that you can customize.

Another popular program that is even easier to use is *Pine*. Pine uses a simple menu-driven interface that is especially suited to beginners or casual users. Pine

is great for inexperienced people who have to sit down at a computer and start using mail immediately. The menu items are clear and easy to understand, and you can display help information at any time. However, if you find yourself sending and receiving a lot of mail, you will be better off using a different user agent. Although Pine is easy to learn, the other programs are more powerful and streamlined, and better suited to experienced users.

Another user agent that you may encounter is called *MH* (*Message Handler*). MH is actually a set of relatively simple, single–purpose programs. Instead of using one main program for everything, you use different programs for each task.

For example, when you use **mail** or Elm, you enter one command to start the program. Everything that you want to perform is done under the auspices of this program.

With MH, you use separate commands for each task. For instance, to check what messages are waiting for you, you use the **scan** command; to display messages, you use the **show** command; to reply to a message, you use the **repl** command; to compose a new message, you use the **comp** command; and so on.

The disadvantage of such a system is obvious: you have to memorize a lot of commands. However, the advantages of MH are important. You don't have to stop what you are doing just to do work with your mail. You can enter an MH command at any time, to perform one simple task for you.

For example, say that you have just read a message from a friend in which he asks if you have the latest version of a particular document. To check this, you need to enter a file command to examine the document.

If you are using **mail** or Elm, you would have to stop the program—or at least put it on hold—in order to enter the file command. With MH, you are not working within a mail program. You can simply enter the file command, look at the information, and then enter the MH command to send a reply.

Another user agent that you may find on your system is *Mush*. The name stands for *Mail User's Shell*. Mush is flexible in that it can be used in two different ways: either like **mail** (which uses a line–oriented interface), or like Elm (using the entire screen at once). There is also a commercial product, called Zmail, that is based on Mush.

So far, all the user agents we have mentioned will work on any Unix system. If you are an X Window user (see Chapter 2), there will probably be one or more X user agent programs that you can use to take advantage of the graphical user interface. For example, if you are using a Sun workstation, you can use a graphical mail program called Mailtool.

The last user agent that we want to mention in this section is *Rmail*. This is a mail facility that is built into some versions of Emacs. *Emacs* is a complete working environment that is based on a powerful text editing program. Within the Emacs environment, you can not only edit text, you can develop programs, read Usenet articles, and, using Rmail, send and receive mail. We won't go into the details of Emacs in this book. However, we do want you to be aware of

Rmail. If you are an Emacs user, you will find that Rmail, being integrated into the Emacs environment, is particularly convenient.

PC– and Macintosh–Based Mail Programs

In Chapter 2, we explained how you might access the Internet in two ways. First, you can use a terminal that is connected to a host computer. Second, you can use a computer of your own—such as a PC or a Macintosh—that is connected to a local area network. With such systems, there are a separate class of user agents that can be used to access the mail system. In this section, we will discuss two such user agents: Pegasus and POP-based programs.

Pegasus, also called *P-Mail*, is a user agent for local area networks. Pegasus works just like Novell's Message Handling Service (MHS). However, Pegasus, which was developed by David Harris of Dunedin, New Zealand, is distributed free over the Internet and does not require Novell's MHS software. Pegasus runs on both PCs and Macintoshes and is used widely throughout the Internet.

The other PC and Macintosh mail programs that enjoy wide use are based on a different system. As we discussed earlier, Internet hosts follow a protocol called SMTP when they send and receive mail. SMTP is part of the TCP/IP family and describes how messages are to be handled. For networks that have PCs and Macintoshes, there is another protocol that is sometimes used named *POP*, or *Post Office Protocol*. These personal computers can run a special POP–based user agent that takes advantage of this protocol.

Within each Internet site, there are one or more computers that act as mail repositories. Messages that are waiting to be read are kept on these computers. With POP–based user agents, the messages are not kept in a central computer; they are sent to your personal computer where they are stored until you read them.

To see how this works, imagine a message is being sent to you from the other side of the world. At each step along the way, the message is sent from computer to computer over the Internet. Eventually, the message arrives at the computer in your network that acts as a mail repository. At this point, your POP–based user agent can ask this computer to send the message directly to your PC or Macintosh, where you can read it.

As the message travels through the Internet, the computers use SMTP, the standard TCP/IP mail protocol. But on the last leg of its journey, the message is sent to your PC or Macintosh using the POP protocol.

POP–based user agents have certain important advantages. First, they give you complete control over your mail, because messages are stored on your

personal computer. This means, for example, that you can keep a lot of messages around without having a system manager complain that your messages are taking up too much disk space. (This becomes especially important when you are paying for disk space on the central computer. Storage space on your own computer will be free.)

Second, POP–based user agents are written to make use of the special features of the computer. For example, a Macintosh mail program will take advantage of the Macintosh mouse–driven graphical user interface, with its easy-to-use pull–down menus. Similarly, a DOS program written for a PC can make use of the special features of the computer, such as the function keys. Many users find it easier to use a program that was written specifically for their type of system than to learn how to use, say, the standard Unix **mail** program.

Some of the common POP–based user agents that you may encounter are Eudora, NuPop, and Popmail/PC. Such programs offer the advantage of being native to your environment. However, there are some important disadvantages.

The first problem is one of security. In some organizations, PCs and Macintoshes must be shared. For example, there may be a computer room that anyone can use. With a regular mail program, no one can read your mail unless they are able to log in under your account. In other words, unless someone knows your password—or unless you walk away from your terminal leaving it logged in—your messages are safe from prying eyes.

With a POP–based user agent, all your messages are moved from the central repository to your computer and stored in regular files. If someone else has access to the same computer, he or she will be able to look at your messages. (Remember, PCs and Macintoshes do not use passwords.) Thus, a POP system works best when each computer is used only by a single individual.

Aside from the security problem, there are also logistical considerations. Since all your mail is moved to your personal computer, what happens if you want to access the messages from another computer? With the regular mail system, it is possible to log in to any computer on the network and access your mail. With a POP–based system, messages that have been moved to a particular computer are accessible only on that computer.

Say, for example, that you routinely use two different computers, a standard desktop PC at work and a small portable computer at home. With the home computer, you use a modem to connect to the network over a phone line. You use the portable computer to access the network and the Internet from home. However, once you move messages to a POP–based system on your office PC, you will not be able to read these messages unless you are at the office.

Some versions of POP–based programs solve these problems by allowing you to read mail from the central repository without having to first move it to your computer.

Hint

POP–based user agents are best for people who use only one PC or Macintosh that they do not have to share. In such cases, a POP–based program can provide a powerful, easy-to-use mail service.

Finally, there is one last consideration we would like to mention. The most common user agent programs used on the Internet are the ones we mentioned in the previous section, **mail** (which is actually part of Unix) and Elm. It is always a good idea to be able to use standard tools. For instance, if you know how to use **mail**, you will know how to send and receive mail on any Unix machine.

If you only know how to use a PC or Macintosh POP–based program, your options are limited. Although this may seem fine today, both you and the Internet will be around for a long time, and you never know what systems you will be called upon to use. For example, if you are a student, you might use Eudora (a Macintosh POP–based user agent) at your university. But after graduation, you could easily find yourself working for a company that doesn't have Macintoshes.

Basic Terminology

In this section, we will discuss the technical terms that you will encounter as you use the Internet mail system.

To start, we will remind you that when you see the word "mail", it always means electronic mail, and the word "address" always refers to an Internet address. On those rare occasions when it is necessary to talk about regular post office mail, the reference will be explicit. Thus, if someone on the Internet asks "What is your address?", give him or her your electronic address.

Sometimes, you will see post office mail referred to facetiously as *snail mail*. For example, it is common for people who send in articles to Usenet discussion groups to put their addresses at the end of the articles. Sometimes you will also see "Snail Mail:" followed by a postal address. The name, of course, refers to the fact that post office mail is much slower than electronic mail.

After you read a message, there are several ways to dispose of it. First, you can get rid of the message by *deleting* it. While you are still within the mail program, it is usually possible to get back a deleted message should you change your mind. However, once you quit the program, all the deleted messages are gone for good.

If the message requires an answer, you can *reply* to it. Your mail program will make it easy for you to send a response back to the same userid that sent the original message. If you want, you can even include parts of the original message in your response.

Another option is to *save* a message. This means that the contents of the message are copied to a file that you can keep as long as you want. You may also choose to *forward* a message. The mail program will make a copy of the message and send it to whomever you specify.

Hint

Be careful what you write to other users. Most mail programs make it easy to forward mail, and some people love to do so. Don't be surprised if a private note to a colleague ends up in someone else's mailbox.

When you send a message, you specify the userid of the recipient. At the same time, you can specify more than one recipient and the message will be sent to each one.

You can also designate one or more userids to receive copies of the message. There are two types of copies, regular ones and *blind copies*. Normally, each recipient will see the names of all the other userids who were sent a copy of the message. A blind copy is a secret copy that no one else knows about.

Here is an example. Let's say you work with four people whose userids are **curly**, **larry**, **moe**, and **harley**. (Sounds like an interesting place to work, doesn't it?) You send a message to **curly**, with a copy to **larry** and **moe**, and a blind copy to **harley**. The result will be that all four people will get the same message. Everybody will know that a message was sent to **curly**, and that both **larry** and **moe** were sent copies. However, **curly**, **larry**, and **moe** will not know that **harley** also received a copy.

Hint

When you receive a message, there is no way for you to tell if a blind copy was sent to someone else.

You might ask, is there any difference between specifying several recipients and specifying a single recipient with several copies? For instance, does it make any difference if you send a message to **curly**, **larry**, and **moe**, or if you send a message to **curly** with copies to **larry** and **moe**?

The answer is, unless you send blind copies, there really is no difference. However, you might want to be aware of the political implications of how you choose to address mail. If **curly** were the boss, for example, it might be better to name him as the main recipient and specify the other two as receiving copies.

From time to time, you may find yourself sending mail to the same group of people repeatedly. In such cases, you can establish what is called an *alias*, a name to represent the group. Each time you send a message to the alias, it will automatically be sent to the members of the group.

For example, say that you define an alias of **executives** to represent the three names, **curly, larry, and moe**. Sending mail to **executives** is the same as sending mail to each of the three userids. Of course, as the need arises, you can change or even delete the alias. How you define and manipulate aliases depends on which mail program you use. You will have to check the documentation for your particular mail program for details.

A similar, but more formal arrangement is the *mailing list*. As the name implies, this is a list of userids who are all to receive mail. Mailing lists can be quite large and, as a rule, will have some person who acts as an administrator.

Within the Internet, there are literally thousands of different mailing lists devoted to different subject areas. For example, there is a mailing list that is used to discuss visual arts. To join a list, you send mail to the administrator. From time to time, messages—usually articles of some type—will be mailed out to everybody on the list. We will discuss mailing lists in detail in Chapter 25.

Automatically Forwarding Your Mail

If you are using the Unix system, there is a way to have all your mail forwarded automatically to another address. For example, if you have several accounts on different computers, you may find it more convenient to have all your mail go to just one computer. Similarly, if you are visiting a place where you have been given a guest computer account, you may decide to temporarily forward your mail to that account.

To forward mail automatically, create a file named **.forward** in your home directory (the . character is part of the name). Within this file, put a single line that contains the address to which you want to forward mail. For example, you might use:

```
harley@fuzzball.ucsb.edu
```

From now on, all mail that comes to you on the computer with the **.forward** file will be redirected to the specified address. At any time, you can stop the forwarding by removing the **.forward** file.

Mail Tips and Guidelines

Before we get down to the nitty gritty of reading and sending mail, let's cover a few important tips that might save your life.

If you have never used electronic mail, it will take awhile for you to appreciate how different it is from regular mail or from talking on the telephone.

Because it is so easy to send a quick note anywhere over the Internet, people often forget how permanent such a note can be. As a general rule, don't send messages that you would not want to see a year from now. It is easy for someone to save your messages in a file and then, when you least expect it, dredge them up again.

Hint

When someone does something to make you angry—such as send you a stupid message—resist the temptation to lash out by mailing back something abusive or sarcastic. You will be far better off to save the message and wait a day before replying. Once you mail a message, there is no way to get it back, even if it has not yet been delivered.

(Of course, we realize that nobody will follow this advice, but don't say we didn't warn you.)

Even more important, do not assume that mail is private. Although you may send what you think is a private message, it is easy for the recipient to send a copy to someone else, and you will have no way of knowing what has happened. Similarly, do not assume that any message you have received is private. Someone else may have received a blind copy. When it comes to private matters, such as love letters, you will find that it is more romantic and much safer to follow traditional customs (such as using your employer's Federal Express number).

Another variation that you may encounter is the form letter. One time, we had received a lot of mail overnight and our mailbox (the system file that holds unread mail) had grown quite large. When we read the mail, we noticed that one of the messages was from **root**, which, on Unix systems, is the userid of the system manager. The message informed us that our mail was taking up a lot of space and would we please dispose of it.

Being the perfect users that we are, we immediately sent a polite reply saying that we would take care of the problem right away. (It is always a good idea to respect the wishes of the system manager.) Later, we found out the original letter

was automatically generated by a program that, each night, looked for users with large mailboxes.

Imagine what the system manager thought when he received a strange message in which we promised to "take care of the problem" right away. He had not asked us to do anything; the form letter had simply been sent off under the auspices of his userid. To this day, we are still embarrassed at our obvious naivete (so please don't tell anybody).

Another situation in which you might encounter form letters occurs when you are working with a program and the computer goes down unexpectedly. Later, when the computer is restarted, you may be sent a message telling you that some of your data was saved automatically.

A common example is the Unix **vi** text editor. If you are using **vi** to edit a file and the computer goes down, you will later receive a form letter. This message will tell you that your file was saved and will give you instructions for recovering your data.

The last point that we would like you to appreciate is that electronic mail has a lot in common with talking on the phone or in person: it is quick, simple, and usually informal. For this reason, mail messages often have a sense of immediacy missing in conventional correspondence. What is missing, of course, are the inflections and personality of your voice and your body language.

Because of this, you will find that it is easy to get carried away with the informality of the system and insult someone accidentally. What you mean to be humorous might be taken literally and cause offense. For this reason, there is a convention that whenever you write something in jest that might be misunderstood, you should include a *smiley* at the end of the remark.

A smiley is a tiny picture of a smiling face, drawn with punctuation characters, for example:

```
:-)
```

(To see the face, turn your head sideways to the left.)

For example, say that a friend has just sent you a message telling, in great detail, how he met the woman of his dreams the night before at a Unix singles club. If he were to tell you this story in person, you could make a funny remark without risking an insult (wink, wink, nudge, nudge). By mail, however, it is easy to be insulting where no offense was meant.

Thus, if you want to reply in a humorous vein, it is best to include a smiley:

```
Oh yes, I know who you mean...
She'll go home with anyone who has a real IBM computer :-)
```

This ensures that your friend will be able to appreciate the subtle irony of your comment.

We will talk about smileys more in Chapter 10.

Understanding Mail Headers

Mail messages have a standard format consisting of two parts: the *header* and the *body*. The header consists of a number of lines of information at the beginning of the message. The body is the actual text of the message.

Figure 5–1 contains a sample message that we will examine. The format of the header may vary on your system, but the general idea will be the same.

```
From rick@tsi.com Wed Mar 31 14:47:02 1993
Received: from hub.ucsb.edu by engineering.ucsb.edu
        id AA15594 to harley; Wed, 31 Mar 93 13:19:25 PST
Received: from fuzzball (fuzzball.ucsb.edu) by hub.ucsb.edu;
        id AA11868
        sendmail 4.1/UCSB-2.0-sun
        Wed, 31 Mar 93 13:23:58 PST for harley@cs.ucsb.edu
Received: by fuzzball (5.57/UCSB-v2)
        id AA07200; Wed, 31 Mar 93 13:17:40 PST
Received: from tsi.com by ucsd.edu; id AA03169
        sendmail 5.67/UCSD-2.2-sun via SMTP
        Wed, 31 Mar 93 13:23:52 -0800 for harley@fuzzball.ucsb.edu
Received: by sdcc12.UCSD.EDU (4.1/UCSDGENERIC.3)
        id AA25582 to harley@fuzzball.ucsb.edu;
        Wed, 31 Mar 93 13:23:51 PST
Date: Wed, 31 Mar 93 13:23:51 PST
From: rick@tsi.com (Rick Stout)
Message-Id: <9303311840.AA06711@tsi>
X-Mailer: Mail User's Shell (7.1.2 7/11/90)
To: harley@fuzzball.ucsb.edu
Subject: this is the subject
Cc: addie@nipper.com kim@nipper.com
Status: RO

Harley: I have set up an appointment for next Thursday afternoon.
Please let me know if you can attend.

-- Rick
```

Figure 5-1. *A sample mail message*

The first line of a header will always start with the word **From**. This line shows the address of the userid that sent the message. In our case, the message was sent by **rick@tsi.com**.

The rest of the lines do not necessarily come in the same order. Nor will they always be present. It depends on how the sending and receiving mail systems are configured and what options are set with your particular mail program. At the bare minimum, you will always see an initial **From** line, a **Date** line, and a **Subject** line.

In our example, the initial **From** line is followed by fourteen lines of technical information containing five **Received** statements. These lines show us the path the message took, as well as the times, dates, and what programs were being used at each step of the way. You can usually ignore these lines. However, if you take a moment to look closely, you can see some interesting information. For example, along the way, the message was handled by two different versions of **sendmail** (the daemon that we mentioned earlier that acts as the transport agent).

Now take a look at the fourth **Received** statement. It contains the line:

```
Wed, 31 Mar 93 13:23:52 -0800 for harley@fuzzball.ucsb.edu
```

This is the time and date that the message was received at this particular location. Notice that the time is given as **13:23:52**. This is because the Internet uses a 24–hour time system. The time here represents 1:23 PM (and 52 seconds).

Notice also, the notation **-0800** after the time. This is an important convention that you should understand.

Since the Internet spans the globe, its users are in many different time zones. It is often important to know what time a message was sent, but local time might be confusing. Suppose, for example, that you are in Vienna, Austria, and you receive a message from someone in California. The time on the message is **13:23:52**. How do you know if this is California time or Austrian time?

This same problem arises in many situations. So, as a solution, the Internet has adopted Greenwich Mean Time as a standard. Sometimes you will see this written as GMT. You may also see it referred to as Universal Time or UT (which is the newer, more official name). Whatever you see, just remember that whenever a standard reference time is needed, GMT is used.

The line that we listed above indicates that a local time is being used but that the local time is 8 hours less than GMT. In other words, the message was sent at 1:23 PM local time, which is 9:23 PM GMT.

Aside from GMT or UT, there are other common acronyms that you will see in North America. These names refer to local North American time zones and are shown in Table 5–1.

Now, to return to the header of our sample message, we see that the next line starts with the word **Date**. This shows the time and date that the message was sent.

Acronym	Time Zone
UT	Universal Time (same as GMT)
GMT	Greenwich Mean Time
EST	Eastern Standard Time
EDT	Eastern Daylight Time
CST	Central Standard Time
CDT	Central Daylight Time
MST	Mountain Standard Time
MDT	Mountain Daylight Time
PST	Pacific Standard Time
PDT	Pacific Daylight Time

Table 5-1. *Summary of Time Zone Names*

The following line is another **From** line. This shows extra information about the userid that sent the message. In this case, we can see that this userid is registered to Rick Stout. This means that either Rick or someone using his account sent the message.

Following this is a **Message–id** line that shows the unique identification tag that was assigned to this message. You can ignore this line.

Next, there is an **X–Mailer** line. This shows us the program that the sender used as his user agent. We see here that he is using the Mail User's Shell (Mush). Again, this is information you can ignore.

The **To** line which follows shows us the address to which the message was sent. Of course, this will be your address. If the message was also sent to other people, their addresses will appear on this line as well.

Following this is the **Subject** line. As you will see in Chapter 6, you specify a subject—a short description—each time you compose a message. Whatever you specify is included in the header as the **Subject** line. When you read mail, your mail program will show a summary containing the subject of each message. Thus, you can see at a glance what messages are waiting to be read.

Hint

If you are sending mail to someone who is too busy to pay attention to everything he or she receives, it behooves you to make your subject interesting. Many busy people will ignore mail that looks unimportant.

For example, a subject of "Your office is being demolished tomorrow" will get a faster response than "Upcoming infrastructural modification".

After the **Subject** line, we see the **Cc** line. This shows us the userids that have received copies of the message. In this case, there are two such userids, **addie@nipper.com** and **kim@nipper.com**. The **Cc** designation is the traditional abbreviation for "copy sent to". (Originally—in the days of typewriters—"cc" stood for "carbon copy".) If no copies were sent, this line will be omitted.

Although copies are indicated, blind copies are not. After all, they are secret. So remember, somebody you do not know about may have also received a copy of the message.

The last line in our example is the **Status** line. This is inserted by the mail reading program. Not all programs include such a line. There are two different statuses that you are likely to see. A status of **R** indicates that you are currently reading the message for the first time. A status of **RO** (as in this example) means that the message has already been read and is considered to be old mail.



FUN TIP: It is common for mail programs to display a **Status** *line. However, it is rare to find someone who knows what the different designations mean, because they are not documented anywhere.*

Here, then, is a great way to make some easy money. Go into a bar where a lot of Internet people hang out. Offer to bet anyone in the place that you can name a mail header line that they don't understand. Lose the first few bets by offering easy lines, like **From**, **Subject**, *and* **Date**.

Then, ask for long odds for one last bet. You can now clean up by asking, "What does a status of **RO** *mean?"*

The Difference Between Text and Binary Data

In the next section, we will discuss how you might be able to use the Internet mail system to send and receive all types of data—pictures, sound recordings, and so on—as opposed to simple messages. Before we approach this topic, we

need to take a moment to talk about how computer people classify different types of data.

The term *data* refers to any type of information that might be stored or processed by a computer. It is helpful to consider data as being of one of two basic types: text or binary data.

Text consists of ordinary characters: letters, numbers, punctuation, and so on. Special characters, such as the space and tab, are also considered to be text. An example of text is a message that you might type at your keyboard and mail to a friend. Within this chapter, all the data we have looked at is text.

A file that contains such data is called a *text file*. Another name for such a file is an *ASCII file*. (The name comes from the *ASCII code*, a specification that defines how all the various characters are represented as computer data. ASCII stands for "American Standard Code for Information Interchange". We will not go into the details here.) Any data that is not simple text is referred to as *binary* data. An example of binary data is a file that contains a picture. Such a file does not contain characters. Rather, it contains information that represents the many small dots that make up the image. A file that contains binary data is called a *binary file*.

Where does the name "binary" come from? Imagine an image that you might display on your computer screen. The image consists of many tiny dots. For the sake of this example, let us say that each dot is either black or white.

You might wonder, how are such images stored in files? And, is it possible to mail such a file to a friend so he can display the same image on his computer?

A binary file that stores a picture does not contain the actual image in the way that a photograph album might hold a picture. Rather, the binary file stores the data necessary to recreate the image. Here is how it works.

Each individual dot is encoded as one of two numbers, either a 0 or a 1. In our example, a 0 might represent a white dot; a 1 might represent a black dot. In other words, a file that contains an image actually consists of a long string of 0's and 1's. The program that displays the image must be able to read and understand this type of data in order to recreate the picture on your screen.

In computer science terms, an element that can contain either one of two values—such as 0 or 1—is called a *bit*, which is an abbreviation for "binary digit". In technical terms, we can say that our file contains a large number of bits, all of which have a value of either 0 or 1. Each of these bits represents either a black dot or a white dot.

In this context, the word "binary" indicates that only two different values are used. In computer terms, any data that does not consist of characters and must be represented by sequences of bits is called "binary data".

Our explanation is, of course, simplified to the level of a normal human being. Actual computer scientists (and other related fauna) recognize many different types of bit patterns and, hence, many different types of binary data. For instance, to store color pictures, you would need to use the 0's and 1's

differently. To store sound recordings, you need to use yet another method of encoding 0's and 1's.

The important idea is that it is relatively simple for a program to work with text data. With binary data, each individual bit is important, and manipulating the data becomes more complex.

All the Internet user agents (mail programs) can send and receive text messages: messages you can type at your keyboard and display on your screen in individual characters. However, many mail programs cannot handle binary data. Those that do use a special protocol that we will discuss in the next section.

Hint

When you transfer data from one computer to another, there will be times when it is necessary for you to know whether you are working with text or binary data. As a general rule, data consisting of ordinary characters is text; anything else is binary data. However, there are times when you may get fooled.

It is obvious that a file that holds a picture or a sound recording is a binary file, since such information cannot be represented by ordinary characters. What you may not know is that many common computer tools—such as word processors and spreadsheet programs—also store data as binary files. Although the information may look like characters when it is on your screen, the program puts in special non-character codes when it stores the data in a file. For example, a word processor will use such codes to indicate italics or boldface words. Thus, files such as word processing documents or spreadsheets are stored in a special format and are properly classified as binary files.

Using MIME to Mail Binary Data

In the last section, we explained that binary files contain data that does not consist of simple text. There is a way to mail such files using the Internet mail system. However, it will only work if the mail programs at each end are set up to handle binary data.

In Chapter 2, we explained that the Internet uses a large family of protocols to ensure that all the different types of computers and programs can work together. Each protocol is a set of rules and specifications that describes how something should be done. For example, SMTP (Simple Mail Transfer Protocol)

describes how mail is transported. Using SMTP, Internet hosts can send and receive messages that consist of text.

To enable people to send binary data, another protocol, named *MIME*, was developed. The name stands for *Multipurpose Internet Mail Extensions*. A system using MIME can include binary data along with a regular message. The whole thing is then transported (using SMTP) to the destination computer. At the other end, another MIME system will make the binary data available to the recipient. Of course, everything is automatic, so you don't need to care about any of the details.

In order to send or receive binary data, all you need is a user agent (mail program) that supports MIME. We say this rather glibly, but the fact is, many mail programs cannot use MIME. This is the case for the standard Unix **mail** program that we will discuss in detail in Chapter 6. However, MIME is important, and you will find that modern versions of the newer mail programs do support it.

The way to send binary data is to store it in a file that is combined with a regular text message. We say that you *attach* the file to your message. If your mail program supports MIME, it will have a facility for creating such attachments when you compose a message. Typically, all you need to do is specify the name of the binary file you want to attach. In this context, the data you include is often referred to as *richtext*, the idea being that the data is "richer" than ordinary text.

As you would expect, when you receive a message that contains richtext, you cannot process it properly unless your mail program supports MIME. If so, the mail program will tell you that the message contains an attachment. When you read the message, your program will separate the attachment and store it in a file. If your mail program does not support MIME, you may see the binary data as part of the message, but it will be gibberish.

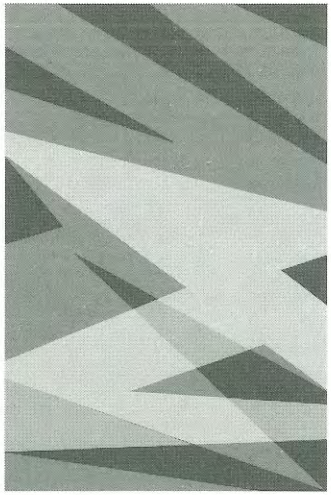
It is important to understand that the sender and recipient do not have to use the same mail program. All that is necessary is that both programs support MIME.

So, for example, say that you are using a computer system that will let you record a voice note and store it in a file. When you play back the file, you hear your voice. You have a friend elsewhere on the Internet who has the same type of computer. Both of you use mail programs that support the MIME protocol.

To send a voice note to your friend, you start by recording the message and saving it in a binary file. Next, you send a message to your friend, attaching the binary file to the message. When your friend receives the message, he saves the attachment in a file of his own. He can now use his computer to play back that file and hear your voice.

Hint

If you do not have access to MIME, there is another way to send a binary file via the mail system. You can use a program named **uuencode** to encode the binary data as text. You can then mail the text as part of a regular message. At the other end, the recipient can take the text and, using a program named **uudecode**, convert it back to its original binary format. Both **uuencode** and **uudecode** are available on many Internet hosts. These programs are discussed in Chapter 18.



CHAPTER 5

The Standard Unix Mail System Program

In Chapter 5, we looked at the Internet mail system. We saw that Internet computers send and receive messages using a system called SMTP (Simple Mail Transport Protocol).

At each Internet computer, there are two important programs that provide the mail service. The first program, the transport agent, works behind the scenes to provide your mail link to the outside world. The second program, the user agent, is your personal mail program. This is the program that you use to send and receive messages.

Of all the user agents that we mentioned in Chapter 5, the most important is the standard Unix mail program,

named **mail**. This program is the only mail program that is available on all Unix systems. Even if you use a different program, you should at least skim the rest of the chapter. The hints and techniques are generally applicable to all mail programs. In particular, you should know about the **from** and **biff** commands.

Hint

No matter what mail program you use, it is a good idea to have some familiarity with the standard Unix **mail** program. It is the only program that you can count on finding on all Unix systems.

Orientation to the Unix mail Program

To start, here is a quick overview of how the **mail** program works. In the following sections, we will discuss sending and receiving mail in more detail.

The **mail** program is used in two ways. To send mail, you type **mail**, followed by one or more addresses to which the message should go. For example, if you want to send a message to **harley@fuzzball.ucsb.edu**, you would enter:

```
mail harley@fuzzball.ucsb.edu
```

Most systems are set up to ask you for the subject and possibly some other information. You can then enter your message, one line at a time.

When you are finished, start a new line and press CTRL–D. (That is, hold down the "CTRL" key and press D.) In Unix, this key is called the **eof** (end of file) key. You use CTRL–D to tell a program that there is no more data to be processed. Once you press CTRL–D, the message will be sent on its way and the **mail** program will end.

To read your mail, enter the simple command:

```
mail
```

If you have messages waiting, **mail** will show you a list and wait for you to enter a command. By using the various commands, you can read and respond to messages as you see fit. When you are finished, you enter the command **q** (for quit) and the **mail** program will end.

If there are no messages, you will see a message telling you that there is nothing for your userid. For example:

```
No mail for harley
```

Specifying Mail Addresses

To send mail, you enter the **mail** command followed by the names of one or more addresses. For example:

```
mail harley@fuzzball.ucsb.edu
mail addie@nipper.com  mschuster  mpeirce
```

The first example sends mail to a single address. The second example sends mail to three addresses.

Let's talk for a minute about the addresses in the second example. The first address (**addie@nipper.com**) is a full address, with a userid and a domain. The second and third addresses only have userids. This is because they are used on the same computer as the person sending the message, so a full address is not necessary. Here is how it works.

Say that your full address is **harley@fuzzball.ucsb.edu**. As a general rule, when you send mail to someone, you can omit the sub–domain names that you have in common. For example, say that you want to send a message to **kim@furface.ucsb.edu**. Since the last two sub–domains (**ucsb** and **edu**) are the same, you can leave them out and send mail to **kim@furface**. Another way to say this is that, because **fuzzball** and **furface** are on the same local network, you need only put in the computer name.

Now, if you are on the exact same computer as someone, the entire domain will be the same and you only have to use the userid. Thus, if you logged in to **fuzzball** and you want to send a message to someone else at **fuzzball**, you need specify their userid only. For example, if your full address is **harley@fuzzball.ucsb.edu** and you want to send a message to someone whose full address is **mschuster@fuzzball.ucsb.edu**, all of the following commands will work.

```
mail mschuster@fuzzball.ucsb.edu
mail mschuster@fuzzball
mail mschuster
```

For more information about Internet addresses, see Chapter 4.

Hint

If you send mail to an invalid address, the message will be sent back to you with an explanatory note. This may take awhile, as the message may have to make its way to the destination computer before it is discovered that the address is wrong.

In such cases, we say that the message has *bounced*. For example, you might call a friend and say, "Can you please tell me your address again. I sent mail to the address you gave me yesterday, but the message bounced."

Composing a Message

Once you enter the **mail** command, you will see the following:

```
Subject:
```

You are being asked to enter the subject of the message. Type anything you want and press RETURN. Whatever you type will become part of the **Subject** line in the header.

Hint

Keep your subject descriptions short. As a rule of thumb, do not use more than 40 characters.

Depending on how your program is set up, you may now see the line:

```
Cc:
```

You are being asked if you want to send copies of the message to anyone. If so, type the address (or addresses) and press RETURN. Otherwise, just press RETURN.

You can now enter your message, one line at a time. When you are finished, press RETURN (to move to a new line) and then press CTRL-D. As we explained earlier, CTRL-D is the **eof** (end of file) key. It indicates to the program that there is no more data.

Figure 6-1 shows a sample message being sent. The words that the user typed are shown in boldface. (The % character in the first line is the Unix shell prompt: the signal that the shell is waiting for a command.) Notice that, at the end of the message, after we press CTRL-D, **mail** responds by displaying **EOT**. This stands for "End of Transmission". It is confirmation that the message is complete and is ready to send on its way.

If you are composing a message and you change your mind, you can cancel it by pressing CTRL-C twice in a row. On Unix systems, CTRL-C is called the **intr** (interrupt) key and is used to abort a program that is running. When you want to abort a message, **mail** makes you press CTRL-C twice. The first time you press it, you will see:

```
(Interrupt--one more to kill letter)
```

This is a safeguard for your own protection. If you pressed CTRL-C by mistake, you can ignore the message and continue typing. However, if you really did want to abort the message, press CTRL-C again to confirm.

```
% mail kim@furface.ucsb.edu
Subject: An important visit
Cc:

Kim:
   The great scholar T.L.Nipper will be visiting next week
and giving a guest lecture. Do you want me to get tickets?

-- Harley
CTRL-D
EOT
```

Figure 6-1. *Sending a message*

The Tilde Escape Commands

As you are typing a message, there are certain commands that you can send to the **mail** program. In order to make sure that these commands are not mixed up with what you are typing, they all start with a ~ (tilde) character. When a single character is used in this way—to signal that what follows is to be treated in a special manner—it is called an *escape character*. For this reason, these commands are called the *tilde escapes*. When you enter a tilde escape, it must always be at the beginning of a line.

Table 6-1 summarizes the tilde escapes. In this table, we use the standard Unix command syntax. That is, we describe the technical format of the commands by using the standard Unix conventions. The words in italics represent names you must fill in. When you see "...", it means you can fill in more than one name.

Command	Description
`~?`	help: display a summary of the tilde escapes
`~b` *address...*	add addresses to "Blind copy" line
`~c` *address...*	add addresses to "Copy" line
`~d`	read in the contents of `dead.letter` file
`~e`	invoke the text editor
`~f` *messages*	(follow–up) read in messages
`~h`	edit all the header lines
`~m` *messages*	read in messages, shift right one tab
`~p`	display (print) current message
`~q`	quit (same as pressing CTRL-C twice)
`~r` *file*	read in the contents of a file
`~s` *subject*	change the "Subject" line
`~t`*address...*	add new addresses to the "To" line
`~v`	invoke an alternative editor (usually same as `~e`)
`~w` *file*	write current message to a file
`~!`*command*	execute shell command, then return to message
`~|`*command*	pipe current message through a filter

Table 6-1. *Summary of the* **mail** *Tilde Escapes*

We won't go over each of the tilde escapes. You should take some time and experiment for yourself. However, we do want to give some examples of the most important ones.

First, the ~? command will display a summary of all the tilde escapes, much like the one in Table 6-1.

The most important command is **~e**. This will start a text editor to allow you to make changes before you send off the message. As we explained in the previous section, you compose a message one line at a time. While you are typing a line, you can backspace and make corrections. However, you cannot backspace to change something on a previous line.

The solution is to use **~e**. This will start your default text editor. Normally, this will be either **vi**—the standard Unix text editor—or **emacs**. Now you have full control to modify any line and add as much text as you want. (Of course, you have to know how to use the editor.) When you are finished, simply quit the editor in the usual fashion. You will now be back in **mail** where you can press CTRL-D to send your message.

Another useful tilde escape is **~h**. This tells **mail** that you want to modify the header lines of your message. You will be prompted for several items in turn. You can change them as you wish. If you don't want to make any changes to a particular item, just press RETURN.

First, you will see:



```
To:
```

You can now change the address to where the message should be sent. Next, you will see:



```
Subject:
```

You can now change the subject of the message. After this, you will see:



```
Cc:
```

You can specify the addresses of any people to whom you want to send copies of the message. Finally you will see:



```
Bcc:
```

Here you can specify if you want to send blind copies to anyone. (Remember, if you don't want to change anything, just press RETURN.)

After you have finished with each item, you will be returned to where you were and you can continue typing your message.

If you would like to change one of the header lines directly, instead of looking at each one in turn, you can use one of the following commands: **~t** (To:), **~s** (Subject:), **~c** (Cc:), or **~b** (Bcc:). Simply specify the new information as part of the command. For example, to change the subject of your message to "Important new item", use:

```
~s Important new item
```

If you decide to send a blind copy of the message to a friend whose address is **kim@furface.ucsb.edu**, enter:

```
~b kim@furface.ucsb.edu
```

The next two tilde escapes come in handy when you are responding to mail that you have just read. We will discuss the details later in the chapter, but for now we will explain that after you read a message, **mail** makes it easy for you to send a reply to the userid that sent the message. When you do, it is often helpful to include a copy of the original message as part of your reply.

To do so, use the **~f** (follow–up) tilde escape. If you use **~f** by itself, **mail** will insert the message you have just read into the text of your reply. If you want to include a different message, just specify its number. (Each message that you read will have a number.) For example, to include message #4, enter:

```
~f 4
```

The **~m** command will do the same thing, except that each line of the included message will be prefaced with a tab character. This way, when the person reads your reply, the included text will be indented.

A nice way to reply to a message is to include all or part of the message in your reply, and insert your remarks within the old message. For example, say that you receive the following message:

```
Do you want to go to a movie later today?

If so, would you like to see the film version of
the book 'A Student's Guide to Unix'?
```

You can use ~f to insert this message into your reply. Next, use ~e to start your text editor. Edit the message and add whatever comments you want. The reply might look like this:

```
> Do you want to go to a movie later today?

Yes, that would be wonderful

> If so, would you like to see the film version of
> the new book 'A Student's Guide to Unix'?

I have already seen it. It is great.
Actually, I wouldn't mind seeing it again.
```

Hint

Here is how to make your replies look nice. First, use ~f to insert the original message into your reply. Next, use ~e to bring the whole thing into your text editor. Now delete all the header lines from the message along with anything extraneous that you do not wish to quote in your reply.

Now, for whatever lines remain, place the two characters "> " (greater-than, followed by a space) at the beginning of each line. (Some mail programs will do this for you, but **mail** won't, so you have to do it yourself.) If you are using the **vi** editor, the following command will insert these characters in front of every line in the editing buffer:

```
:%s/^/> /
```

Now move to each place where you want to reply to a specific point and insert your reply. When you are finished, quit the editor and press CTRL-D to send the message on its way.

Another tilde escape that will include text as part of a new message is ~r (read). This will insert the contents of an existing file into your message. Just type ~r followed by the name of the file. For example, to read in a file named **memo**, enter:

```
~r memo
```

The last two commands we want to mention are ~! and ~|. These are used to access Unix commands while you are composing a message. The ~! tilde escape will pause the **mail** program, execute the Unix command that you specify, and then return to **mail**. For example, to pause briefly in order to display the time and date, enter:

```
~! date
```

With ~|, you also specify the name of a Unix command. In this case, **mail** will send the contents of your message to this command to be processed. Your message will be replaced by the output of the command. For example, say that you have typed in a message that consists of a long list of items. Now that the list is typed, you want to use the Unix **sort** command to sort the list before you send the message. Enter:

```
~| sort
```

Your message will be replaced by the output of the **sort** command, that is, by all the lines in sorted order. If, at this point, you want to make a few small changes, you can use **~e** to start the text editor.

Hint

Unix has a command named **fmt** that was developed specifically to format mail messages. The **fmt** command will read data and, line by line, format it to be as close to 72 characters per line as possible. The command will not change spacing at the beginning of a line or in between words. It will also leave blank lines untouched. In other words, **fmt** will format a message nicely as long as you use a blank line to separate paragraphs.

To format an entire message before you send it, enter the tilde escape:

```
~| fmt
```

If you want to check the results, you can use **~p** (print) to display the message, or **~e** to start the text editor.

A Shortcut When Sending Mail

We have described how to send mail: you enter the **mail** command, along with one or more addresses. You then specify a subject, after which you compose your message. When you are finished, you press CTRL-D to send the message.

To speed things up, there are several shortcuts you can take by using variations of the **mail** command. First, you can specify the subject as part of the command. To do so, type **mail**, followed by **-s**, followed by the subject. For example:

```
mail -s Meeting harley@fuzzball.ucsb.edu
```

This command specifies that the subject line in the header should be "Meeting". If you want a more complex subject line, with spaces or punctuation, enclose it in single quotes. For example:

```
mail -s 'Meeting next week' harley@fuzzball.ucsb.edu
```

When you specify a message in this way, **mail** will not prompt you for a subject. If you want to change the subject, you can use the **~s** or **~h** tilde escape.

Hint

In Unix terminology, the **-s** is called an *option* or a *switch*. According to Unix rules, all options must come at the beginning of the command. Thus, make sure you specify the addresses after the **-s**. For example, it will cause a mistake if you enter:

```
mail harley@fuzzball.ucsb.edu -s 'Meeting for next week'
```

The **mail** program will think that you are trying to send a message to three different addresses—**harley@fuzzball.ucsb.edu**, **-s**, and **'Meeting for next week'**—and everything will go awry.

Another shortcut you can take is to prepare the message ahead of time and store it in a file. You can then tell **mail** to take the message directly from the file.

To do so, type a < (less-than) character at the end of the command, followed by the name of the file.

For example, say that you have a message stored in a file named **memo**. You want to send the message to **harley@fuzzball.ucsb.edu** with a subject of "Meeting next week". Enter the following command:

```
mail -s 'Meeting next week' harley@fuzzball.ucsb.edu < memo
```

What we are doing is telling Unix that the input for the **mail** program should come from a file rather than from the keyboard. In Unix terminology, we are "redirecting the standard input" from a file named **memo**. In this case, the entire message will be read from a file, so you do not need to press CTRL-D to indicate the end of the message.

How Can You Tell If a Message Has Been Read?

In general, there is no way to tell if a message has been read by the recipient. However, there is a command that will work some of the time.

This command is called **finger**, and we will discuss it in detail in Chapter 8. Basically, **finger** will display public information about any userid on the Internet. (There are sites, however, that prohibit **finger** access to their systems. This may be for security reasons, or simply to lower the number of network requests that come to that system.)

Some versions of **finger** will tell you if there is mail waiting for that userid that has not been read. For example, let's say that you send mail to a friend whose address is **harley@fuzzball.ucsb.edu**. You can display information about him by using:

```
finger harley@fuzzball.ucsb.edu
```

If the **finger** system on his computer displays mail information, the output will contain something like this:

```
New mail received Thu Apr  1 12:39:22 1993;
unread since Thu Apr  1 08:16:11 1993
```

(We will look at the full output of the **finger** command in Chapter 8.)

Hint

Most systems have a command that will check to see what mail is waiting for you. For example, Unix has the **from** command (which we will meet in the next section). Such commands display a list of messages along with their subjects.

Once a person has used the **from** command, it looks to **finger** as if he or she has already read the mail, even though this is not the case. More precisely, **finger** looks at the last time the file that holds the mail was accessed. Since **from** must access this file, it may erroneously look as if the mail has been read, when all that has happened is that the person has displayed the list of unread messages.

Keep this in mind before you fire off an angry message saying, "I know that you have already read my last message. Why haven't you replied?"

Checking to See If Mail Has Arrived: from, biff

Aside from using **mail**, Unix systems have two other commands that will tell you if mail is waiting. The first command is **from**. Simply enter:

```
from
```

You will see a summary of all the messages that are waiting. Here is an example:

```
From addie@nipper.com Wed Mar 31  23:49:50 1993
From kim@fuzzball Thu Apr 1 06:42:25 1993
From mschuster@music.tuwien.ac.at Thu Apr 1 06:50:48 1993
From addie@nipper.com Thu Apr 1 08:22:28 1993
From mpeirce@shamrock.tcd.ie Thu Apr 1 08:23:45 1993
```

As you can see, **from** displays the first line of the header for each message. If there are no messages waiting, **from** will not display anything.

The second command, named **biff**, works in the background, monitoring the file that holds your mail. Whenever a new message arrives while you are logged

Hint

When messages arrive for you, they are stored in a file until you can read them. Each user has his own such file. The **from** command works by looking in this file to see if there are any messages.

On some systems, the mail file is actually removed when there are no more messages in it. When a new message arrives, the file is re-created. Thus, when you have no mail, the **from** command will find not an empty file, but a missing file. You will see a warning similar to:

```
Can't open /usr/spool/mail/harley
```

Literally, this means that **from** cannot find your mail file. What it really means is that you have no mail.

in, **biff** will let you know automatically. That way, you do not have to keep typing **from** to check for new mail.

The **biff** command sets an on/off switch. When the switch is on, **biff** will keep track of your mail. When the switch is off, **biff** will not check for new mail. Most of the time, it is handy to keep **biff** on. However, if you are working on something special, and you do not want to be interrupted whenever mail arrives, you may want **biff** off.

To turn **biff** on, enter the command followed by **y** (for "yes"):

```
biff y
```

To turn off **biff**, use **n** (for "no"):

```
biff n
```

If you want to check the current status, enter the command name by itself:

```
biff
```

You will see either:

```
is y
```

or:

```
is n
```

If you are using an X Window system (see Chapter 2), there is an X client

Hint

When you log in to a Unix system, you may see a message telling you that you have mail. This message is coming from your shell (the command processor), not from **biff**.

named **xbiff** that you may want to use. The **xbiff** program draws a picture of a mailbox for you. When mail arrives, the flag on the mailbox will go up. To put the flag back down, move the mouse pointer to the box and click.

*FUN TIP: The command **biff** was named after a dog named Biff who belongs to a programmer named Heidi Stettner. In the summer of 1980, Heidi was working for a professor of Computer Science at the University of California at Berkeley as she waited to enter graduate school. At the time, Heidi used to bring Biff to her office, not far from where the Unix programmers were working on the new version of BSD (Berkeley Unix).*



*There is an apocryphal story that the **biff** command was named after Biff the Dog because he used to bark whenever the mailman came. That is not true. The real story is that Biff was a universal favorite among the professors and students. Biff had his picture on the bulletin board that had photos of all the Computer Science graduate students. (In fact, Biff even got a B in a compiler class, but that is another story.)*

*One day, one of the graduate students who was working on Unix—John K. Foderero—decided to create a command and name it after Biff. He came up with the idea to write a program to check for mail. Ever since, Unix people around the world have been using **biff** to check their mail.*

(By the way, as we write this book, Biff the Dog is retired and still living in Berkeley, California, with Heidi. Like many of the elderly Unix pioneers, Biff spends most of his time sleeping.)

Reading Your Mail: An Overview

To read your messages, enter the **mail** command by itself:

```
mail
```

If there is mail waiting for you, **mail** will display information about each message, in the form of short one–line descriptions called header summaries. Here is an example:

```
*Mail version 2.18 5/19/83.  Type ? for help.
"/usr/spool/mail/harley": 5 messages 5 unread
>U 1 addie@nipper.com Wed Mar 31  23:49  35/1204 "Re: seeing a movie"
 U 2 kim@fuzzball Thu Apr ? 06:42  138/5518 "Re: having dinner"
 U 3 mschuster@music.tuwien.ac.at Thu Apr ? 06:50  46/1592 "Students Guide To Unix"
 U 4 addie@nipper.com Thu Apr ? 08:22  48/1595 "Something interesting about TLN"
 U 5 mpeirce@shamrock.tcd.ie Thu Apr ? 08:23  343/16810 "Schedule for Chapter 5"
```

Each line in the summary represents one message that is waiting for you. If a header summary is very long, it may be broken onto more than one line on your screen. As you can see, the messages are numbered and, in this example, we have five messages.

After the number, you see the name of the userid that sent the message. For example, message #1 was sent by **addie@nipper.com**. Next is the date and time. After this are two numbers, separated by a / (slash) character. This tells you the size of the message: the number of lines and the number of characters. In our example, message #1 consists of 35 lines and has a total of 1204 characters. Finally, you see the subject of the message.

Once the header summaries are displayed, you will see an & (ampersand) character displayed on a line by itself:

```
&
```

This is a *prompt*, a signal from **mail** that it is waiting for you to enter a command. You can now enter one command after another to read and process your mail. When you are finished, enter the **q** (quit) command to stop the program.

When you see the & prompt, there are many commands that you can enter. These are summarized in Tables 6-2 and 6-3. Table 6-2 shows all the important commands, listed in alphabetical order. Table 6-3 shows the same commands organized by function.

Abbreviation	Full Name	Description
?	—	display command summary
!	—	execute a single shell command
	—	display the next message
-	—	display the previous message
RETURN	—	display the next message
number	—	display message #*number*
d	delete	delete message
dp	—	delete current msg, display next msg
e	edit	use text editor on message
h	headers	display header summaries
l	list	list names of all available commands
m	mail	send new message to specified userid
n	next	display the next message
p	print	display (print) messages
pre	preserve	keep messages in system mailbox
q	quit	quit mail
r	reply	reply to sender & all other recipients
R	Reply	reply to sender only
s	save	save messages to specified file
sh	shell	pause mail, start a new shell
to	top	display top few lines of message
u	undelete	undelete previously–deleted messages
w	write	same as s, only do not save header
x	exit	quit mail, neglect any changes
z	—	show next set of header summaries
z-	—	show previous set of header summaries

Note: On some systems, the meaning of **r** *and* **R** *is reversed.*

Table 6-2. *Summary of Important* **mail** *Commands: Alphabetical Order*

Abbreviation	Full Name	Description
Stopping `mail`		
`q`	`quit`	quit `mail`
`x`	`exit`	quit `mail`, neglect changes
Help		
`?`	—	display summary of commands
`l`	`list`	list names of commands
Header		
`sh`	`headers`	display header summaries
`z`	—	show next header summaries
`z-`	—	show previous summaries
Displaying Messages		
`+`	—	display the next message
`-`	—	display the previous message
RETURN	—	display the next message
`number`	—	display message #*`number`*
`n`	`next`	display the next message
`p`	`print`	display (print) messages
`to`	`top`	display top lines of messages
Replying and Mailing		
`m`	`mail`	compose a new message
`r`	`reply`	reply to sender & recipients
`R`	`Reply`	reply to sender only
Processing a Message		
`d`	`delete`	delete messages
`dp`	—	delete current msg, display next msg
`e`	`edit`	use text editor on messages
`pre`	`preserve`	keep messages in system mailbox
`s`	`save`	save messages to specified file
`u`	`undelete`	undelete previously–deleted msg
`w`	`write`	same as `s`, do not save header

Table 6-3. *Summary of Important* **mail** *Commands: Organized by Function*

Abbreviation	Full Name	Description
Shell Commands		
!	—	execute a single shell command
sh	shell	pause `mail`, start a new shell

Note: On some systems, the meaning of **r** *and* **R** *is reversed.*

Table 6-3. *Summary of Important* **mail** *Commands: Organized by Function (continued)*

The Current Message and Message Lists

In a moment, we will cover the most important **mail** commands. These are the commands that you will use to read and process your messages. Before we talk about these commands, we must discuss the idea of "message lists".

Take another look at the header summary that we showed you in the last section.

```
Mail version 2.18 5/19/83.  Type ? for help.
"/usr/spool/mail/harley": 5 messages 5 unread
>U 1 addie@nipper.com Wed Mar 31  23:49  35/1204 "Re: seeing a movie"
 U 2 kim@fuzzball Thu Apr ? 06:42  138/5518 "Re: having dinner"
 U 3 mschuster@music.tuwien.ac.at Thu Apr ? 06:50  46/1592 "Students Guide To Unix"
 U 4 addie@nipper.com Thu Apr ? 08:22  48/1595 "Something interesting about TLN"
 U 5 mpeirce@shamrock.tcd.ie Thu Apr ? 08:23  343/16810 "Schedule for Chapter 5"
```

Notice that message #1 is marked with a > (greater-than) character at the far left. The > character indicates the **current message**. The current message is the default when you enter any command that does not specify a message explicitly.

For example, as we will see in a moment, the **d** command deletes messages. The command:

```
d 3-5
```

will delete messages #3 through #5. If you do not specify a message number, the command will act on the current message. For instance, if you enter:

```
d
```

it will delete the current message (in our case, #1).

When you start reading your mail, the current message starts at message #1. As you read one message after another, the current message will change to the last one that was read. Thus, at any time, any command that does not specify a message explicitly will act on the last message that you read.

Sometimes, though, you will want to specify a message. For most commands, you can use one or more message numbers. For instance, in the second last example, we specified that we wanted to delete messages #3 through #5. The way that you specify messages is by using a *message list*.

A message list can include a combination of one or more numbers, a range of numbers, or characters that have special meanings. Table 6-4 shows the various ways in which you can specify a message list.

Most of the time, you will use commands that specify only one number or no message list at all. However, being able to specify a message list can come in handy, and you should be familiar with the idea and how it works. Table 6-5 shows a number of message list examples using the **d** (delete) command.

Specification	Meaning
. [a period]	the current message
n	message number *n*
n-*m*	all messages from *n* to *m* inclusive
^ [a circumflex]	the first message
$ [a dollar sign]	the last message
* [an asterisk]	all messages
userid	all messages from specified userid
/*pattern*	all messages containing pattern in subject
:n	all new messages
:o	all old messages
:r	all messages that have been read
:u	all messages that are still unread

Table 6-4. *Summary of the Various Ways to Specify a Message List*

Command	Meaning
`d`	delete the current message
`d .`	delete the current message
`d 3`	delete message
`d 3-5`	delete messages 3 though 5 inclusive
`d ^`	delete the first message
`d $`	delete the last message
`d *`	delete all messages
`d harley`	delete all the messages from userid harley
`d /hello`	delete all the messages with "hello" in subject
`d :n`	delete all the new messages
`d :o`	delete all the old messages
`d :r`	delete all the messages that have been read
`d :u`	delete all the messages that are still unread

Table 6-5. *Examples of Specifying a Message List*

Hint

Within a message list, **:r** refers to the messages that you have already read. The **:u** specification refers to unread messages. How do you keep track of which are which?

Take another look at the header summary at the beginning of this section. Notice that there is a **U** character at the left–hand side of each header line. This indicates that the message is as yet unread. Once you read a message, the **U** will disappear.

Displaying a Message

The **mail** program is designed to make it easy to display one message after another. Here is how it works. When you start reading, the current message is set to #1. To read the first message, just press RETURN. To read the next message, press RETURN again. In other words, you can read all your messages, one after another, just by pressing RETURN.

If you want to jump right to a particular message, you can use the **p** (print) command. For example, to display message #4, enter:

```
p 4
```

For convenience, if you enter a number by itself, **mail** assumes that you want to display that message. So an easier way to display message #4 is to enter:

```
4
```

In addition, you can enter + (the plus character) to display the next message and – (the minus character) to display the previous message.

When a message is too long to fit on one screen, **mail** will call upon a *paging program* to display the message for you. A paging program displays data one screenful at a time. At the bottom of each screen, the program displays a note showing you there is more to come. Which paging program is used depends on how your system is set up. The three most common paging programs are called **more**, **pg**, and **less**.

Hint

When you are using **more** or **less**, you advance to the next screenful of data by pressing the SPACEBAR. With **pg**, you must press the RETURN key.

Saving and Deleting Messages

Once you have read a message, you must dispose of it in some way. You have two choices. You can delete it, or you can save it to a file. To delete a message, use the **d** command. To save a message, use the **s** command. If you enter the command without a message list, it will act upon the current message, the one you have just read.

For example, say that you have just read a message. To delete it, simply enter:

```
d
```

Of course, you can use the **d** command with a message list. For example, to delete message #2 and messages #5 through #7, enter:

```
d 2 5-7
```

If you change your mind about deleting a message, you can undelete it by using the **u** command. However, you must do so before you quit the program. For example, to undelete message #6, enter:

```
u 6
```

When you save a message, you must specify the name of the file. For example, to save the message you have just read to a file named **important**, enter:

```
s important
```

If you save a message to a file that already exists, **mail** will append the new data to the end of the file. If the file does not exist, **mail** will create it for you automatically.

Hint

If you want to save messages from a number of people, use files named after the people. For example, every time you read a message from someone named Harley, you can save it by entering:

```
s harley
```

This keeps all your correspondence nicely cataloged.

Each time you send a message to Harley, send yourself a copy. You can then save the copy to the **harley** file. In this way, you will be able to save all of your messages—sent and received—pertaining to a particular person.

When you quit the **mail** program (by entering the **q** command), **mail** will remove all the deleted and saved messages from your mailbox. In addition, **mail** will save any messages that you have read, but not deleted or saved, to a file named **mbox**. Messages that you have not yet read are left alone.

Why is **mail** so anxious to dispose of your messages? When messages arrive, they are kept in a file, called your *mailbox*, that is part of the system area of the disk. On many systems, the disk storage used by your mailbox is not counted against your personal quota. However, when **mail** dumps messages into a file, it is stored in one of your directories. (In Unix, files are kept in directories.) The storage space is now yours.

Hint

On some systems, especially those for which you pay real money, the storage space used by unread messages in your mailbox is billed to you.

If you have accidentally made too many deletions, or have mucked up your messages in some other way, you can quit **mail** by using the **x** (exit) command instead of **q**. This tells **mail** to ignore any changes that were made during the current session.

If you do not have time to dispose of some of the messages that you have read, you can use the **pre** (preserve) command before you quit. This tells **mail** not to move certain messages to the **mbox** file. For example, to keep messages #2 through #6 in your mailbox, enter:

```
pre 2-6
```

To keep all of the messages in your mailbox, use:

```
pre *
```

Hint

Normally, you read the messages that are being kept in your system mailbox. However, there will be times when you may want to read messages that are stored in one of your own personal files. For example, you may have saved a number of messages from a friend named Harley in a file named **harley**. Or the **mail** program may have saved some of your messages to a file named **mbox** (as we described above).

In such cases, use the **–f** (file) option when you enter the mail command. After the **–f**, type the name of the file. The **mail** program will read messages from this file. For example, to read messages that are stored in a file named **harley**, enter:

```
mail -f harley
```

To read messages stored in a file named **mbox**, use:

```
mail -f mbox
```

Replying to a Message

To reply to a message, use the **r** command. The **mail** program will automatically use the address of the person who sent the message. You can now type your message, one line at a time, just as we described earlier. And, since you are composing a message, you can use all the tilde escapes. In particular, you can use the **~m** command to include the original message within your response. (See the discussion earlier in the chapter.)

Here is an example. You have just read a message that says:

```
Do you want to go to a movie tonight?--Kim
```

To reply, you enter:

```
r
```

You can now start typing your message:

```
Yes, I would love to go to a movie,
as soon as I finish writing Chapter 6.
--Harley
```

After typing the last line of the message, press CTRL-D to send it on its way.

If copies of the original message were sent to other userids, using the **r** command will automatically send your reply to everyone. In some cases, this may be what you want. However, there are times when you may want to reply only to the person who sent the message. For example, if you are replying to a memo that was sent to thirty different people, you will probably not want your reply to be sent to all thirty people. In such cases, use the **R** command, instead of **r**. This tells **mail** to reply only to the sender of the message.

Hint

Once you start receiving electronic mail, it is easy to end up with a lot of messages in your mailbox (or in your **mbox** file). In order to keep things from getting out of hand, dispose of a message as soon as you have read it. Follow these three simple guidelines:

- As soon as you have read a message, decide if you want to reply and do so right away.
- Before you move on to the next message, either delete or save the one you just read.
- When it doubt, throw it out.

Composing New Messages and Forwarding Mail

As you are reading messages, you may decide to send a new message to someone. You do not have to stop the **mail** program and enter a new command. Rather, you can use the **m** command to start a new message.

For example, say that you are reading your messages and you realize that you need to send a quick note to a friend whose address is **harley@fuzzball.ucsb.edu**. Simply enter:



```
m harley@fuzzball.ucsb.edu
```

You will be asked to enter the subject of the message. After doing so, you can type your message in the regular manner. Once you press CTRL-D to send the new message, you will be back where you left off, reading your messages.

Some mail programs have a command to forward a message to another person. The **mail** program does not, but it is not too hard to do so yourself.

First, use the **m** command to start a new message to the person. Second, use the ~f tilde escape to include the message you have just read in the new message.

For example, say that you have just read message #3 and you want to forward it to a friend whose address is **kim@nipper.com**. Start by entering the command to send a new message:



```
m kim@nipper.com
```

You will see:



```
Subject:
```

Type the subject and press RETURN. Now enter the command to copy the current message into the new one:



```
~f
```

You will see:



```
Interpolating: 3(continue)
```

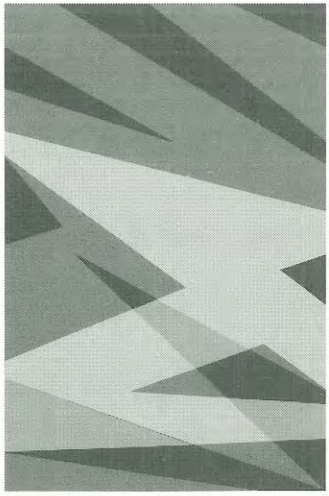
This tells you that **mail** has copied message #3 to your new message. Now press CTRL-D. You will see:

```
EOT
```

The new message is on its way, and you can go back to reading your mail.

Hint

When you forward a message in this way, you may want to use the **~e** tilde escape to start the text editor after you use **~f**. This will allow you to remove the old header lines before you send the message.



CHAPTER 7



Using Telnet to Connect to Remote Hosts

One of the wonderful things about the Internet is that it is as easy to use a computer on the other side of the world as it is to use a computer across the hall. In this chapter, we will explain how to connect to a remote computer by using the Telnet service.

We will discuss how there are two principal ways to use a remote computer. First, you can log in to any Internet host on which you have an account. For a Unix computer, for instance, this means that you must have a userid and password. Once you log in, you can use the computer in the regular manner.

119

Second, there are many Internet computers that offer some type of public Telnet access. Such systems are available to anyone and do not usually require a password. Many of the resources in our catalog are accessed in just this way.

As you will see, Telnet is so transparent that it is easy to forget that you are separated from the remote computer. The main limitation you will find is that, when Internet network traffic is high, the response from a faraway computer may slow down slightly. Still, it is common for experienced users to work with several different Internet computers, moving from one to another smoothly and easily.

An Overview of Telnet Connections

In Chapter 2, we explained that the Internet offers a large number of services that are based on standard protocols. The service that allows you to connect to a remote Internet host is called *Telnet*.

To use Telnet, you run a special program, called **telnet**, on your computer. This program uses the Internet to connect to the computer you specify. Once the connection is made, **telnet** acts as an intermediary between you and the other computer. Everything you type on your keyboard is passed on to the other computer. Everything that the other computer displays is sent to your computer, where it appears on your screen. The end result is that your keyboard and screen seem to be connected directly to the other computer.

In Telnet terminology, your computer is called the *local* computer. The other computer, the one to which the **telnet** program connects, is called the *remote* computer. We use these terms no matter how far away the other computer actually is: whether it is across the world or in the same room.

As we mentioned in Chapter 2, we often refer to Internet computers as hosts. So, using Telnet terminology, we can say that the job of the **telnet** program is to connect your local computer to a remote Internet host.

One last comment on terminology: We often use the word *telnet* as a verb. For example, say that you are visiting a friend in a distant city. You might ask him, "Can I use your computer for a minute? I want to telnet to my computer to read my mail."

Starting the telnet Program

To make a remote connection, you use the **telnet** program. There are two ways to start this program. In this section, we will show you how it is done most of the time. In the next section, we will discuss an alternate method.

To start **telnet**, enter the name of the command followed by the address of the remote host to which you want to connect. For example, say that you want to connect to a computer named **fuzzball**, whose full address is **fuzzball.ucsb.edu**. Enter:

```
telnet fuzzball.ucsb.edu
```

If you are connecting to a computer on your local network, you can usually get by with just the name of the computer, instead of the full address. For example:

```
telnet fuzzball
```

Hint

As we explained in Chapter 4, all Internet hosts have an official address known as an IP address. This address consists of several numbers separated by periods. For example, **128.54.16.1** is the official IP address of the computer whose standard address is **ucsd.edu**.

Some systems have trouble dealing with certain standard addresses. If you encounter such a problem with **telnet**, try using the IP address. For example, either of the following commands will connect to the same host:

```
telnet ucsd.edu
telnet 128.54.16.1
```

For more information about IP addresses and Internet addresses in general, see Chapter 4.

When the **telnet** program starts, it will initiate a connection to the remote host that you specified. As **telnet** is waiting for a response, you will see:

```
Trying...
```

or a similar message. Once the connection is made—which might take a few moments if the host is far away—you will see a message like the following:

```
Connected to fuzzball.ucsb.edu.
Escape character is '^]'.
```

(We will explain the reference to an "escape character" later.)

If, for some reason, **telnet** is unable to make the connection, you will see a message telling you that the host is unknown. For example, say that you want to connect to the remote host **nipper.com**. However, you mistakenly enter:

```
telnet nippet.com
```

You will see:

```
nippet.com: unknown host
telnet>
```

At this point you can either specify another host name or quit the program. We will explain how to do this in the next section.

Hint

The **telnet** message "unknown host" can be misleading. There are many reasons why **telnet** might not be able to make a remote connection. The three most common are:

- You spelled the address of the computer wrong.
- The remote computer is temporarily unavailable.
- You specified the name of a computer that is not on the Internet.

Another problem that may arise is that your local network may, for some reason, not be able to make connections to certain parts of the Internet. One reason is that a particular host may be off-limits for security reasons. Another reason is that some locations just do not have a way to connect to other locations.

For example, one of our friends in Ireland complains that he is unable to connect to computers in Australia. In such cases, **telnet** will display a message like:

```
Host is unreachable
```

If this happens to you, double-check that you are entering your **telnet** command correctly. You can also ask your system manager if there is some trick that you don't understand about making such a connection. However, if the remote host is really unreachable from where you are, there is not much you can do about it. (The best advice we could give to our Irish friend is that many people never connect to Australian computers and still live full, rich lives.)

Once **telnet** makes a connection, you will be interacting with the remote host. At this point, most hosts display some type of informative message, usually identifying the computer. If you are expected to log in, you will see the standard prompt. For example, if you have connected to a remote Unix computer, you will see:

```
login:
```

You can now log in in the regular manner. Type your userid and press RETURN. You will then see:

```
Password:
```

Now enter your password and press RETURN again. (Note: When you type your password, it will not be displayed. This prevents someone else from finding out your password by watching you log in.)

We mentioned earlier that some remote hosts are set up to offer a public service. In such cases, you will not have to use a password when you log in. For example, in Chapter 17, we will discuss an important Internet resource called Archie servers. When you log in to a public Archie server, you use a userid of **archie**. Once you enter this userid, the Archie program will start automatically.

Some public hosts are even more automatic. As soon as you use **telnet** to connect, the remote program starts by itself; you do not even need to log in.

When you are finished working with the remote computer, all you need to do is log out in the regular manner. The connection will break and **telnet** will stop automatically.

A Second Way to Start telnet

In the previous section, we mentioned that there are two ways to start the **telnet** program. The first way is to enter the **telnet** command along with the address of the remote host. For example:

```
telnet fuzzball.ucsb.edu
```

The second way is to start **telnet** without specifying a host. Simply enter:

```
telnet
```

The program will start, but will not make a connection. You will see:

```
telnet>
```

This is the **telnet** prompt. It means that the program has started and is waiting for you to enter a command. To make a connection to a remote host, type **open**, followed by the address of the host. For example:

```
open fuzzball.ucsb.edu
```

The connection will be made just as if you had specified it when you entered the **telnet** command.

In the previous section, we gave an example in which a **telnet** command had a bad address. In the example, the remote host was named **nipper.com**, but we mistakenly entered:

```
telnet nippet.com
```

What happens in such a case is that **telnet** tries to make the connection. When it can't, it gives up and displays its prompt, waiting for you to enter a command. In this case, you would see:

```
nippet.com: unknown host
telnet>
```

You can now enter an **open** command with the correct address:

```
open nipper.com
```

If this address doesn't work, you can try another one. If you decide to give up, enter:

```
quit
```

This will stop the **telnet** program.

Summary of Starting and Stopping telnet

There are two ways to start **telnet**. Either enter the command with the address of a remote host:

```
telnet fuzzball.ucsb.edu
```

or enter the command by itself:

```
telnet
```

and then, at the **telnet>** prompt, enter an **open** command:

```
open fuzzball.ucsb.edu
```

There are two ways to stop **telnet**. If you are connected to a remote host, log out in the regular manner and **telnet** will stop automatically. Otherwise, at the **telnet>** prompt, enter the quit command:

```
quit
```

Connecting to a Specific Port Number

Within the Internet, there are numerous hosts that offer Telnet access to public services. Indeed, if you look in the catalog, you will see that a large number of the items are accessed via Telnet.

Many of these hosts require that you specify a particular *port number* when you make the connection. The port number identifies the type of service that you are requesting. Here is an example.

There is a host at the University of Michigan that provides American and Canadian weather reports. The name of this computer is **downwind.sprl.umich.edu**. When you connect to this computer, you must specify a port number of 3000. This tells the system that you want to use the weather service.

All you have to do is enter the port number at the end of the **telnet** command:

```
telnet downwind.sprl.umich.edu 3000
```

(Be sure to leave a space between the address and the port number.) If you connect by using an **open** command at a **telnet>** prompt, type the port number in the same way:

```
open downwind.sprl.umich.edu 3000
```

When you connect to a remote host that uses a port number, the program that you want will usually start automatically, and you will not have to log in. When you end the program, the connection will be broken and **telnet** will stop by itself.

Hint

In this chapter, we describe the **telnet** command as it is used on most Unix systems. On certain other types of systems, the format is slightly different. For example, on some VAX computers running the VMS operating system, you need to put **/port=** in front of the port number:

```
telnet downwind.sprl.umich.edu /port=3000
```

If you have any problems on your computer, the best idea is to check the local documentation for **telnet**.

More About Port Numbers

In computer terminology, the term *port* refers to a connection between two devices or systems. For example, you might attach a printer to a port on the back of your computer. In a Unix system, we say that each terminal is connected to its own port on the host.

This same idea is used in Internet terminology. In Chapter 2, we explained that the Internet uses a protocol called TCP (Transmission Control Protocol) to transfer data from one host to another. Whenever TCP connects one Internet host to another, a port number is used to identify the type of connection. In fact, there is an Internet organization—called the Internet Assigned Number Authority—that maintains the official list of port numbers (of which there are many), and makes sure that unique numbers are allotted when necessary.

By default, ordinary Telnet connections are made using port number 23. In other words, when you do not specify a port number, the **telnet** program automatically connects using port 23. Thus, the following two commands will make the same type of connection:

```
telnet fuzzball.ucsb.edu
telnet fuzzball.ucsb.edu 23
```

In order for you to connect to a remote host, it must be running a program that is prepared to communicate over the port that you use. When such a program is waiting for a connection, we say that it is *listening* on that port.

Thus, we can say that for a host to support regular Telnet connections, it must have a program that is listening on port 23. Another way to put it is that when you use **telnet** to make a regular connection to a remote host, it contacts that host and checks to see if there is a program listening on port 23.

It is only when you want to use a Telnet connection over a different port that you need to specify an actual port number. Many Internet systems use different port numbers to offer some type of special service.

In the previous section, we connected to a remote host named **downwind.sprl.umich.edu** using port number 3000. We did this in order to use a special-purpose program that displays weather reports. In order for this connection to work, the weather report program must be running on this computer and it must be listening on port number 3000. In most cases, a host can support more than one connection to a specific port at the same time. This particular weather server, for example, will support up to 100 simultaneous users.

The telnet Escape Character: CTRL-]

As you interact with a remote host, there is a way to put your work on hold and enter commands directly to **telnet**. For example, if you are having problems with the remote host, you can pause your work session, return to **telnet**, and enter the **quit** command.

The way to do this is to press a special key combination CTRL-]. That is, hold down the CTRL key and press] (the right square bracket). When you press this key, it sends a signal to **telnet** to pause the remote connection and display the prompt:

```
telnet>
```

You can now enter any **telnet** command that you want. (We will discuss the most important ones in a moment.) With some commands, **telnet** will automatically resume the connection after carrying out the command. Otherwise, you can resume the connection at any time by pressing RETURN at the **telnet>** prompt.

The technical term for a key like CTRL-] is an *escape character*. Many programs allow you to use an escape character to request a special service or to indicate that what follows is to be interpreted differently.

In the world of Unix, there is a convention that CTRL keys are often indicated by using the ^ (circumflex) character. For example, CTRL-C would be written as ^C.

Now we can make sense out of the message that appears whenever **telnet** makes a remote connection:

```
Escape character is '^]'.
```

You are being reminded that the Telnet escape character is CTRL-]. It is possible to change this to another character, but there is usually no reason to do so.

Using telnet Commands

Any time you are at the **telnet>** prompt, there are a number of different commands that you can use. In this section, we will go over the most important

ones. Before we do, we will remind you that if instead of entering a command you just press RETURN, **telnet** will resume the remote connection.

To display a summary of the various **telnet** commands, you can enter the **?** character:

```
?
```

Here is a typical summary:

```
Commands may be abbreviated. Commands are:

close     close current connection
display   display operating parameters
mode      try to enter line-by-line or character-at-a-time mode
open      connect to a site
quit      exit telnet
send      transmit special characters ('send ?' for more)
set       set operating parameters ('set ?' for more)
status    print status information
toggle    toggle operating parameters ('toggle ?' for more)
z         suspend telnet
?         print help information
```

Out of all these commands, the most important ones are **?**, **open**, **close**, **quit**, and **z**.

The **open** command tells **telnet** to make a connection to a remote computer. Enter **open** followed by the address of the computer. For example:

```
open fuzzball.ucsb.edu
```

The **close** command terminates a remote connection without stopping the **telnet** program. Here is how this might come in handy.

Let's say that you are working with a remote host and something goes wrong. For some reason, the host seems to be ignoring your commands. No matter what you type, nothing happens, and you can't even log out. One solution is to press CTRL-], wait for the **telnet>** prompt, and then enter the **close** command. You can now reestablish a connection to the same host. You can, of course, also connect to a different host.

The **quit** command stops **telnet**. If a remote connection is active, **telnet** will terminate it.

Hint

Before you close a connection or quit **telnet**, remember to log out from the remote host. Most hosts will log you out automatically when the connection drops, but it is better to do so yourself. This way, you can ensure that whatever program you were using is terminated properly and all your data was saved.

All the other commands (except **z**, which we will discuss in the next section) are less important, and you will probably never need to use them. For the most part, they control various technical aspects of the communication session that you almost always ignore. If you want to display a summary of the **send**, **set**, or **toggle** command, enter the command followed by a **?** character. For example:

```
send ?
```

Hint

If you want more technical information about **telnet**, see the documentation for your particular system. If you are using a Unix computer, you can display the **telnet** entry in the online manual by using the command:

```
man telnet
```

Job Control

Unix systems support a facility called *job control* that allows you to pause a program, work with another program, and then return to the first one. We won't go into all the Unix details here, but we will explain how, if your system has job control, **telnet** will cooperate.

At the **telnet** prompt, you can enter the **z** command. (We will explain the name in a moment.) This tells **telnet** to pause itself and return you to the shell (the program that reads and processes your commands). This allows you to enter regular commands in the middle of a remote session.

The **z** command will only work if your shell supports job control. If you are using a modern shell, such as the C-Shell or the Korn shell, this will be the case. If you are using the older Bourne shell, there will be no job control and the **z** command will not work. In fact, **telnet** itself may freeze.

The program with which you are currently working is said to be in the *foreground*. When you put a program on hold, we say that it is in the *background*.

When you enter the **z** command, **telnet** will put itself in the background and return you to your local shell. You can now enter as many regular Unix commands as you want. For example, you can check your mail, display the time and date, or whatever. When you want to resume your remote connection, enter the command:

```
fg
```

This tells Unix to reactivate (move to the foreground) the last program that was put on hold. You can now resume your remote connection.

You might wonder, why is the **telnet** job control command named **z**? The answer is that, on Unix systems that support job control, you can move your current program into the background by pressing the **susp** (suspend) key. Usually, this key is CTRL-Z. Thus, the **telnet** command is named after the Unix **susp** key.

Hint

Many hosts will automatically log you out if nothing happens for a specified period of time. For example, a system might log you out if you have not typed anything for 15 minutes. Remember this when you use the **z** command to put a **telnet** session on hold. If you do not resume the session before it times out, you may be disconnected automatically.

Figure 7-1 shows an example of how this all works. The commands that we entered are printed in boldface.

At the beginning of the example, we are logged in to a computer named **nipper**. You can see the shell prompt:

```
nipper%
```

At this prompt, we enter a **telnet** command to connect to the remote computer whose address is **fuzzball.ucsb.edu:**

```
nipper% telnet fuzzball.ucsb.edu

Trying...
Connected to fuzzball.ucsb.edu
Escape character is '^]'.

ULTRIX V4.2A (Rev. 47) (fuzzball)
Welcome to the Fuzzball System.

login: harley
Password:

Last login: Sun Apr 18 00:09:58 from nipper.com
fuzzball% date
Sun Apr 18 00:24:54 CDT 1993
fuzzball% CTRL-]
telnet> z
Stopped

nipper% mail
No mail for harley
nipper% fg
telnet fuzzball.ucsb.edu

fuzzball% logout
Connection closed by foreign host.

nipper%
```

Figure 7-1. *Using Job Control with* ***telnet***

```
telnet fuzzball.ucsb.edu
```

Once the connection is made, **fuzzball** displays the standard Unix login prompt and we log in with a userid of **harley**. Notice that, for security reasons, the password is not displayed as we type it. After the login is complete, the remote host displays some information followed by a shell prompt:

```
fuzzball%
```

We now enter the **date** command to display the time and date.

At this point, we decide to return to **nipper** temporarily and check our mail. First, we press CTRL-]. This puts the remote connection on hold and returns us to **telnet**. You can see the prompt:

```
telnet>
```

Next, we enter the **z** command. This places the **telnet** program in the background. We now see the **nipper** shell prompt. We use the **mail** command (see Chapter 6) and find out that there are no messages.

We now enter the **fg** command. The shell responds by displaying the last command that was placed in the background (in this case, it was the **telnet** command). The shell then brings this program back into the foreground, which automatically resumes the remote connection. Once again, we can see the **fuzzball** shell prompt.

Finally, we enter the **logout** command to log out from **fuzzball**. The remote connection is closed automatically, and **telnet** stops. We are left where we started, at the **nipper** shell prompt.

Hint

On Unix systems, the shell is the program that reads and processes your commands. There are several Unix shells and you may be able to choose the one you want.

When a shell is ready to accept a command, it displays a prompt. If you have an account on more than one computer, it is a good idea to customize your prompts so that they contain the name of the computer. (We won't go into the details here.) In this way, your shell prompt is always a handy reminder of which system you are using.

Traditionally, the last character of a prompt is used to show the type of shell you are using. The % character (as in our example) indicates the C-Shell. The $ character indicates the Korn shell or Bourne shell.

CHAPTER 08

Fingering the World

With so many millions of people using the Internet, it is important to be able to find out something about a particular person. The *Finger* service allows us to do just that. With a simple command, we can find out certain public information about another Internet user.

In this chapter, we will explain how the Finger service works. We will show how to ask for information and how to control the public information about yourself. We will also explain how this same service is used to offer various other types of information.

What Is the Finger Service?

In Chapter 2, we explained that a client/server system consists of two programs: a client that requests a particular resource and a server that provides that resource. In many cases, the client and server programs run on different computers, allowing you to access a remote host. The Finger service is a client/server system that provides three main types of information.

First, you can display certain public information about any user on an Internet host. What you see will vary from host to host. Indeed, some hosts will not display any user information at all, for security reasons. In general, though, most hosts will share basic information about their users.

All you need to know is which host a person uses, and either their userid, first name, or last name. With this information, you can use the Finger service to display some or all of the following:

- the person's userid
- the person's full name
- if the userid is currently logged in
- the last time someone logged in using that userid
- whether or not mail has been read
- a phone number
- an office location
- information that the person has specifically prepared for the general public (for example, a list of office hours)

Hint

If you know the name of a person, but you do not know which host he or she uses, you may be able to find them by using the techniques described in Chapter 22.

A second way to use the Finger service is to check to see who is currently using an Internet host. You can display a summary that shows some or all of the following information for each userid that is logged in:

- userid
- full name
- when the userid logged in
- how long it's been since there was activity on that terminal
- phone number and office information
- name of computer or terminal server from which the person logged in

The third use for Finger is to communicate with certain Internet hosts that have been set up to offer other, more esoteric types of information. For example, one host will return current information about earthquake activity.

Most of the time, the basic reason to use Finger is to display public information about a particular person. To use this service, you run the **finger** program. For example, to display information about the person whose userid is **harley** and who uses a computer whose address is **fuzzball.ucsb.edu**, you can enter:

```
finger harley@fuzzball.ucsb.edu
```

Notice that we are using a standard Internet address. As we explained in Chapter 4, the Internet has one main addressing system and we use it for everything. Thus, you use the same address to finger someone as you would to send mail.

Here is a typical example of output from such a request. For now, don't worry about the details. Just take a quick look to get an idea of how the service works.

```
[fuzzball.ucsb.edu]
Login name: harley              In real life: Harley Hahn
Phone: 202-456-1414
Directory: /usr/harley          Shell: /bin/csh
Last login Wed Apr 21 21:20 on ttyp4
No unread mail
Project: Writing books about the Internet and Unix.
Plan:
You can find me in my office most days.
If you have any questions about the Internet catalog,
please send them to Rick Stout.
```

The **finger** program is the client. It acts on your behalf and sends a request for information to the appropriate host, in this case, **fuzzball.ucsb.edu**. On the remote host, a Finger server is waiting for such requests. In our example, the Finger server on **fuzzball.ucsb.edu** honored the request and sent back the information displayed above.

On most Internet systems, some type of Finger server is always running, ready to handle any request for information that may arrive. On Unix systems, the server is usually named **fingerd**, "finger daemon" (although you don't really need to know the name). As we explained in Chapter 5, a daemon is a program that runs in the background, usually to provide a service of general interest.

When we talk about the Finger service, it is common to use the word *finger* as a verb. For example, suppose you are at a party and you meet the woman (or man) of your dreams. As you leave, you might say, "By the way, if you need my phone number, just finger me." (Yes, Internet people really do talk this way.)

Displaying Information About a Person

To display information about a person, you need to know the address of his or her computer. You also need either a userid, a first name, or a last name. Enter the **finger** command using the following format:

```
finger name@address
```

For example, say that you want to display information about a friend of yours named Ben Dover who works at the Hong Fatt Noodle Company. His userid is **bdover**, and the address of his computer is **noodle.com**. You can use any of the following commands:

```
finger bdover@noodle.com
finger ben@noodle.com
finger dover@noodle.com
```

The first command specifies the userid. This is the best thing to use if you know it. If not, you can try the first name or last name as we did in the second and third commands. However, if you do so, for example, by using **ben**, there will often be more than one person with the same name. In such cases, the Finger server will display information about all the users with that name.

When you specify a first or last name, it does not matter if you mix upper- and lowercase letters. For example, the last two commands are the same as:

```
finger Ben@noodle.com
finger DOVER@noodle.com
```

Hint

When specifying a name in a **finger** command, the rule is that first and last names can mix upper- and lowercase letters, but userids must be exact. Since virtually all userids are lowercase, the best idea is to always use lowercase for everything.

If you are fingering someone who uses a computer on your local network, you usually need to specify only the name of the computer. For example, say that you are logged into a computer named **fuzzball.ucsb.edu**. You want to finger a friend named Michael Schuster, who has a userid of **mschuster** on a local computer named **furface.ucsb.edu**. You can use any of the following commands:

```
finger mschuster@furface
finger michael@furface
finger schuster@furface
```

If you want to finger someone on the same computer that you are using, you may omit the address entirely. For example, say that you want to finger someone named Artie Choke who has a userid of **achoke** on your computer. You can enter any of the following commands:

```
finger achoke
finger artie
finger choke
```

If a Finger server cannot find the person you want, it will not say so directly. Rather, it will respond with a message that contains no personal information. For example, say that on the **noodle.com** host, there is no person that can be identified by the name of "addie". However, you enter the command:

```
finger addie@noodle.com
```

The remote Finger server will return a message that looks like this:

```
[noodle.com]
Login name: addie              In real life: ???
```

Don't be fooled. This message does not mean that there is an "addie", but that no information is available. The message really means that, as far as this Finger server is concerned, "addie" does not exist.

Understanding Finger Information

In this section, we will take a look at the type of information Finger will show you about a person. From system to system, there is some variation as to what you will see. What we have here is a typical example.

To start, you enter the following command:

```
finger harley@fuzzball.ucsb.edu
```

As soon as it confirms that the address is valid, **finger** will display the name of the remote host:

```
[fuzzball.ucsb.edu]
```

A moment later, you will see the output from the remote Finger server:

```
Login name: harley                  In real life: Harley Hahn
Phone: 202-456-1414
Directory: /usr/harley              Shell: /bin/csh
Last login Wed Apr 21 21:20 on ttyp4
No unread mail
Project: Writing books about the Internet and Unix.
Plan:
You can find me in my office most days.
If you have any questions about the Internet catalog,
please send them to Rick Stout.
```

Let's go over this output line by line.

The first line shows basic information about the user. In this case, we see the userid **harley** (euphemistically referred to as **Login name**) and the full name **Harley Hahn**.

There is an important variation of this line that you might encounter. It looks like this:

```
Login name: harley  (messages off)  In real life: Harley Hahn
```

The notation **(messages off)** indicates that the person has chosen to not allow other people to send messages to his screen. Unix users can set this preference by using the **mesg** command.

In Chapter 19, we will discuss the **talk** program, which enables you to have a conversation with another Internet user. When you enter a **talk** command, it sends a message to the other person telling him that you want to start a conversation. If you have trouble getting through, try fingering the person. It may be that he or she has set "messages off" for privacy. If that is the case, the person will not get your **talk** message.

The second line of the Finger output is straightforward: it shows the person's telephone number.

The next line shows technical Unix information about the person's account. The directory is the name of the person's home directory, the place that has been allotted to that user to store his files. The next piece of information is the name of the user's shell. In this case, we see that this person uses the C-Shell (if we recognize that **csh** is the name of the C-Shell program).

Hint

You can ignore the technical Unix information that is part of a Finger report. There is probably some set of circumstances in which it is important to know the name of someone's home directory and shell. However, in over 250 years of using the Internet, we have yet to come across such a situation.

Following the technical Unix information, we see the last time that the person logged in. In this case:

```
Last login Wed Apr 21 21:20 on ttyp4
```

This tells us that the last login was on Wednesday, April 21 at 9:20 PM. (Remember, Unix and the Internet use a 24-hour clock.) The last part of the line tells us that the person's terminal was connected to the port named **ttyp4** on his computer. Most of the time, this is another piece of information you can ignore.

If the person logged in from another computer (using Telnet, for instance) or from a terminal server, you may see the name of that machine. For example:

```
Last login Tue Apr 20 20:33 on ttyp0 from unix1.tcd.ie
```

If you finger someone who is currently logged in, you will see a message similar to the following:

```
On since Apr 26 09:21:12 on ttyp1 from engrserv
```

In this example, the person logged in on April 26 at 9:21 AM and is still logged in. He or she is using a terminal that is connected to port **ttyp1**. The connection was made via **engrserv** (which happens to be a terminal server).

Hint

Before you use the **talk** command to have a conversation, finger the person. This way, you check if he or she is logged in and is accepting messages.

After the time of the last login, we see some information about mail. In this case, we see:

```
No unread mail
```

This tells us that the person has read and disposed of all his mail. Such information can come in handy when you send a message to someone and you want to know if he or she has read it yet. However, there are some limitations you should know about. We will discuss this whole topic in more detail later.

After the mail status, we see the final two items, called **Project** and **Plan**. These two pieces of information are important because you can change them whenever you want to display public information. We will discuss this in a moment.

Is There a Way to Find Out If Someone Has Fingered You?

No.

The .plan and .project Files

In the previous section, we looked at an example of output from a **finger** command. At the end of the output, we saw the following:

```
Project: Writing books about the Internet and Unix.
Plan:
You can find me in my office most days.
If you have any questions about the Internet catalog,
please send them to Rick Stout.
```

Here is where this information comes from and why it is important.

The original **finger** command was developed at the University of California at Berkeley as part of their Unix project. (The author, by the way, was Earl T. Cohen.) At the time, the command was designed to display information that was appropriate to Berkeley's academic environment.

One of the built-in facilities allowed professors and students to tell everyone what projects they were working on. For example, a professor might describe his research interests. A graduate student might describe her dissertation topic.

Another facility allowed people to show their current plans. For instance, a professor might show his office hours and when he planned to be out of town at conferences.

The way it works is that the Finger server looks for two special files named **.project** and **.plan** in your home directory. (In Unix, each user has a home directory for storing his or her files.) If the **.project** file is present, Finger will display the first line. If the **.plan** file is present, Finger will display the entire file.

In Unix, a file that begins with a . character is called a *dotfile*. When we talk about such files, we pronounce the . character as "dot". For example, we might ask someone, "Have you changed your dot-plan file to reflect your new office hours?"

The significance of dotfiles has to do with the Unix command that lists the names of your files. You use this command—named **ls**—to keep track of your files. Now, certain files, like the ones that hold your project and plan information,

are needed by most people. It is a bother to see these same names every time you list your files.

To solve this problem, the **ls** command will not display file names that begin with a . character unless you use a special option. Thus, you can have files with names such as **.project** and **.plan** and not have to look at them every time you list your files. Unix uses many different dotfiles, although we won't go into the details here.

All you have to do is create these files in your home directory and put in whatever information you want. Remember, Finger will display the entire **.plan** file, but only the first line of the **.project** file. If Finger cannot find a **.plan** file, it will display the message:

```
No Plan.
```

(However, there is no such message for **.project**.)

This message can be puzzling if you don't know exactly what it means. Many people do not create a **.plan** file. When you finger such people, you will see the standard public information followed by **No Plan.**, as if to imply that this person is wandering aimlessly through life (although this may well be the case).

How do you create your own **.project** and **.plan** files? You need to be able to use a text editor program. The standard Unix text editor is named **vi**. Another common editor is Emacs. In addition, there are a number of other such programs that you will find. Unfortunately, the details of using a text editor are well beyond the scope of this book.

We strongly recommend that you do learn to use a text editor. For example, it is helpful to be able to compose mail or to create articles to send to a Usenet discussion group.

Hint

When you start learning Unix, you will encounter people who will tell you that the **vi** editor (or Emacs) is difficult to learn. Some people will even tell you to stay away from these programs.

Don't listen to such people; they are just being ignorant. They are also being shortsighted. It is impossible to use the Internet fully without being able to edit your own files. Take it from us, you do need to learn to use a text editor. Yes, it may take a little practice, but you will be glad you put in the effort.

In the meantime, ask someone who does know how to use a text editor to help you prepare your **.project** and **.plan** files.



FUN TIP: You can put whatever information you want in your **.project** *and* **.plan** *files. For example, if you are having a party, you can put directions for getting to your home in your* **.plan** *file. You can then mail an invitation to your guests, telling them, "If you need directions, just finger me." You will find that people use their* **.plan** *files to hold all types of imaginative items: jokes, poems, comments on life, drawings made up of computer characters, and so on.*

Interpreting Information About Mail

In a previous section, we mentioned that a Finger server may tell you something about a person's mail. In this section, we will discuss the type of information that Finger might show you and how to interpret what you see.

First off, you should understand that the output of a **finger** command depends on the Finger server at the remote host. Thus, you will get different types of output depending on which host you finger. Some Finger servers are happy to display mail information. Other systems won't even mention it.

As we explained in Chapter 6, each userid's mail is kept in its own file called a mailbox. When you finger a user, the Finger server can examine the user's mailbox. The server can check the last time the mailbox was accessed and compare it to the last time that new mail arrived. Here are some of the messages that you might see from a Finger server.

If the mailbox is empty, you will see:

```
No unread mail
```

If the mailbox is not empty, there are three different messages you might see. If you see something like:

```
Unread mail since Wed Apr 21 20:46:13 1993
```

it means that the mailbox has not been accessed since the mail arrived. If you see:

```
Mail last read Wed Apr 21 20:46:13 1993
```

it means that the mailbox has been accessed since the mail arrived. Finally, if you see:

```
New mail received Wed Apr 21 20:46:13 1993;
unread since Wed Apr 21 20:32:32 1993
```

it means that the mailbox has been accessed, but new mail has arrived since then.

The problem is that it is possible to access your mailbox without reading any mail, which leads to misleading Finger messages.

How can this be? In Chapter 6, we explained that you can use the **from** program to check if mail is waiting for you. The **from** program does not show you your mail: it merely displays a one-line summary of each message. However, to do so, it must access your mailbox. Thus, whenever you check your mail with **from** (or a similar program), the access time on your mailbox is updated, and the Finger server will jump to the conclusion that you have read your mail.

On some systems, it is possible to specify a list of commands that are executed automatically whenever you log in. Some people use this facility to have a program like **from** display a mail summary each time they log in. So be careful: what a Finger server tells you about someone's mail may be misleading.

Here is a real-life example of how you might get into trouble.

You are conducting delicate diplomatic negotiations between the Syldavians and the Bordurians regarding joint military exercises to be held on St. Vladimir's Day (which is tomorrow). These two countries have a long history of conflict, so the utmost tact is necessary.

At 18:00 (6:00 PM), you send a confidential message to the Bordurian ambassador. You wait impatiently for an answer. Finally, at 22:00 (10:00 PM), you decide to finger the ambassador to see what is happening. You enter:

```
finger musstler@kurvi-tasch.bo
```

As part of the output, you see:

```
Mail last read Wed Apr 21 20:46:13 1993
```

Aha, you say, the ambassador read his mail at 20:46 (8:46 PM), which was over an hour ago. Obviously, he has decided not to respond to my message.

You inform the Syldavians who quickly conclude that the Bordurians must be planning a preemptive strike. Hoping to beat the Bordurians at their own game, they immediately declare war.

Later, you find out that the Bordurian ambassador had not actually seen your message. When he logged in, a **from** command had automatically accessed his mailbox. However, at the time you fingered him, he was busy looking at the jokes in a Usenet discussion group and had not yet got around to reading his mail.

Imagine your embarrassment.

Fingering a Remote Host

So far, all our examples have specified the name and address of a particular user. Finger responds by displaying information about that user. Another way to use Finger is to specify only the address. The Finger server at that remote host will display a summary of all the users who are logged in.

The format of such a command is:

```
finger @address
```

For example:

```
finger @fuzzball.ucsb.edu
```

Hint

When you finger a remote host, remember to put the @ character before the address. If you leave out this character, Finger will think that you are referring to a user on your own computer. For example, if you enter:

```
finger fuzzball.ucsb.edu
```

it looks as if you are fingering a user named **fuzzball.ucsb.edu** on your own computer. You will see a message like this:

```
Login name: fuzzball.ucsb.edu          In real life: ???
```

If you want to finger a computer that is on the same local network as your computer, you can usually abbreviate the address. For example, say that you are logged in to **fuzzball.ucsb.edu** and you want to finger the computer named **furface.ucsb.edu**. You can probably use:

```
finger @furface
```

If you want to finger your own computer, just omit the address:

```
finger
```

When you finger a remote host, the format of the output may vary, depending on how the host's Finger service is implemented. Let's take a look at two typical examples of what you might see. Here is the first one:

```
Login       Name            TTY  Idle    When         Office
pmason   Perry Mason        *p0        Fri 08:32  Law Building 260
dstreet  Della Street        p1     2d Wed 08:55  Law Building 261
hburger  Hamilton Burger     p2     10 Fri 09:16  City Hall 3rd Fl
atragg   Arthur Tragg        p5   3:05 Fri 09:05  City Hall 5th Fl
```

The first two columns show each person's userid (**Login**) and name. The third column shows the port to which the user's terminal is connected. (In Unix, "TTY" is often used as an abbreviation for "terminal".) The important thing about this column is that if the person has chosen to disallow messages, you will see an * (asterisk) character. This is the case with the userid **pmason**.

The next two columns give you vague information about the person's activity. The **When** column shows the day and time that the person logged in. The **Idle** column shows how long it has been since there was some activity on that terminal. If the entry is blank, it means that the person is actively using his or her terminal. Otherwise, you will see the amount of time that the terminal has been idle. You may see a number of minutes (such as **10**), a number of hours and minutes (**3:05**), or even a number of days (**2d**).

Finally, the last column shows general information under the heading of **Office**. Such information varies because, on most systems, people can change this whenever they want.

Here is a second example of output from fingering a remote host. This example shows another common format that you may see:

```
Login       Name            TTY  Idle    When         Where
freddy   Frederick Bean      p3        Mon 16:09  pigpen.bean.com
jinx     Jinx T. Cat       *p1   2:08 Mon 12:47
wiggins  Mrs. Wiggins        p4      5 Mon 10:47  cowbarn.bean.com
```

The main difference between the two examples is in the rightmost column. In this case, the **Where** column shows how the person has logged in. If the entry is empty, the person is using a terminal that is connected directly to the host. Otherwise, the person connected via the computer or terminal server whose name is given. For a discussion of such connections, see Chapter 3.

Finally, we will mention that if you finger a host that no one is using, you will see a message like:

```
No one logged in.
```

Changing Your Personal Finger Information

We have already explained how you can use your **.project** and **.plan** files to display whatever you want. On some systems, you are also allowed to change some of the other information that Finger gives out on your behalf.

If you have a Unix system, try using the **chfn** (change finger) command:

```
chfn
```

If this command is enabled on your system, it will prompt you for information.

Here is an example. Say that your userid is **harley**. You enter the **chfn** command, and you see the following:

```
Changing finger information for harley.
Default values are printed inside of '[]'.
To accept the default, type <return>.
To have a blank entry, type the word 'none'.

Name [Harley Hahn]:
```

The name in square brackets is the current value. If you want to change it, simply type a new name and press RETURN. Otherwise, just press RETURN.

On some systems, all you can change is your name. On other systems, you will be prompted to enter further information, such as a room number, an office phone number, and a home phone number. If you do not want to specify one of these items, simply enter the word **none**. Out of courtesy, you should at least allow people to know your name.

Hint

Some systems are set up so that you cannot modify your own Finger information. If this is the case on your computer, you will have to ask your system manager to make changes.

A Guide to Safe Fingering

So far, we have described how the Finger service will provide you with all kinds of information about Internet users. However, there are a few limitations and warnings you should understand.

First, not all Internet hosts provide a Finger service. For the most part, this service is offered only by Unix hosts, and there are many non-Unix computers on the Internet. Although there are a few non-Unix hosts that have a Finger service, it is unusual. If you finger a host and nothing seems to happen, it is likely that you have reached a non-Unix computer, such as a PC or Macintosh, or a VAX running VMS.

Second, many Unix hosts will not respond to Finger requests from outside their local network. This may be for security reasons or simply to cut down on network traffic. For example, let's say you try to finger someone who uses the host named **microsoft.com**:

```
finger billg@microsoft.com
```

You will see:

```
[microsoft.com]
connect: Connection refused
```

This means that the Finger server on the remote host is refusing to connect to an outside host. Alternatively, some Finger servers will accept such connections, but will not return any information.

The second limitation is that not all remote hosts that support Finger will display all possible information. For example, some hosts will not show the contents of **.project** and **.plan** files to outsiders.

Perhaps most important is that you should not always believe what you see. As we explained earlier, it is possible for users on some systems to change their public Finger information, even their full names.

Hint

If you are unsuccessful fingering someone by using a real name, it may be because he or she has changed this information. This is a fairly common phenomenon among students whose sense of humor has not yet fully matured.

For example, if you finger a university computer to ask about a userid named **ugrad-56**, and you see that the real name is "James T. Kirk", you might be a tad suspicious.

The final limitation we want to mention is one that we discussed earlier in the chapter. It is possible that Finger will tell you that someone has read his or her mail, when all he or she has done is display a summary. Don't jump to conclusions.

Special-Purpose Finger Services

So far, we have looked at the ways in which you can use Finger to display information about people. A number of Internet hosts use this same service to offer other kinds of specialized information.

If you look through the catalog, you will see that some of the entries tell you to finger a special userid and address to get certain information.

In case you are interested in experimenting, here are a few of the special-purpose Finger services that were available at the time we wrote this chapter.

To display recent baseball scores:

```
finger jtchern@sandstorm.berkeley.edu
```

To display a summary of recent auroral activity (the Northern Lights):

```
finger aurora@xi.uleth.ca
```

To display chart listings of the top songs as ranked by *Billboard* magazine:

```
finger buckmr@rpi.edu
```

To display information about recent earthquake activity around the world:

```
finger quake@geophys.washington.edu
```

To display daily press releases from NASA (National Aeronautics and Space Administration):

```
finger nasanews@space.mit.edu
```

To display a seasonal hurricane forecast for the Atlantic region:

```
finger forecast@typhoon.atmos.colostate.edu
```

Coke Servers

Perhaps the most unexpected use of Finger is called a *coke server.* This is a Finger service that is used to display information about a non-traditional device, such as a vending machine. The first coke server was set up in the mid-1970s for the Computer Science department at Carnegie-Mellon University.

At the time, the main terminal room housed a Coca-Cola machine that was frequently used by programmers to replenish their natural stores of caffeine. A recent expansion of the department had placed some of the programmers' offices on a different floor from this room. To their consternation, they would often make the long trip to the machine only to find that there was no Coke. Or, almost as bad, they would find Coke that had been in the machine only a short time and was still warm.

Their solution was to install switches in the machine to sense how many bottles of soda were available and to keep track of how long each bottle had been cooling.

To allow people to check the machine from a distance, a special Finger server was set up. When you fingered a particular userid (**coke**), it would display the status of the contents of the machine.

In the early 1980s, this system was discontinued. However, in recent years, it has been resurrected and the term "coke server" is now used to denote any similar system, whether or not it provides soft drink information. Carnegie-Mellon inhabitants can now use a coke server to check not only a soft drink machine, but a candy machine as well.

If you would like to try this for yourself, there are three ways to finger the CMU coke server. First, to display basic information about the state of the soft drink machine, enter:

```
finger @coke.elab.cs.cmu.edu
```

The output will look like this:

```
Coke Server Ver 0.99 2-26-93
Coke:           Cold:   7   Warm:   0          Buttons
Diet coke:      Cold:  14   Warm:   0       C: COLD
Sprite:         Cold:   1   Warm:   0   C: EMPTY   D: COLD
                                       C: COLD    D: COLD
                                       C: COLD    D: COLD
                                                  C: COLD
                                                  S: COLD
```

For basic information about the candy (M&M) machine, you can finger the **mnm** userid:

```
finger mnm@coke.elab.cs.cmu.edu
```

The output looks like:

```
Coke Server Ver 0.99 2-26-93
Information may not be correct, use at your own risk.
The M&M machine is 46% full.
```

Finally, to see a pictorial representation of both machines, you can enter:

```
finger bargraph@coke.elab.cs.cmu.edu
```

Here is some sample output:

```
Coke Server Ver 0.99 2-26-93
M&M information may not be correct, use at your own risk.
    M & M                          Buttons
   /-----\           C: CC.......................
   |     |         C: ............  D: CCCCC........
   |     |         C: C...........  D: CCCC.........
   |*****|         C: CC..........  D: CCCCC........
   |*****|                          C: CC...........
   \-----/                       S: C............
      |          Key:
      |           0 = warm; 9 = 90% cold; C = cold; . = empty
      |           Leftmost soda/pop will be dispensed next
   ---^---
```

If you want to look at a couple of other Coke servers, you can try the following commands to access Finger servers at the Rochester Institute of Technology.

```
finger drink@csh.rit.edu
finger graph@drink.csh.rit.edu
```

For more on Coke servers, look in the catalog.

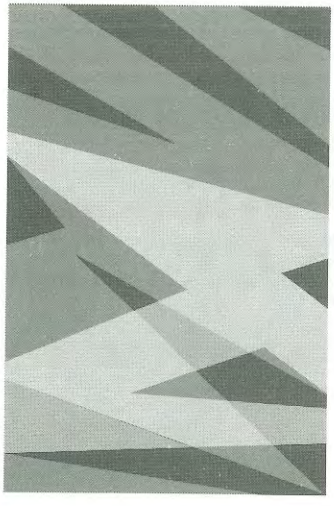
The University of Wisconsin has a computerized Coke machine. To buy a drink, you log in to the terminal next to the machine and select the appropriate command. Of course, you need to pay money in advance to have credit in your account.

If you want to see the instructions for this system, finger the following server:

```
finger coke@cs.wisc.edu
```



If you are a researcher looking for some interesting work, apply for a grant to implement an Internet coke server at your location.



CHAPTER 9

Introduction to Usenet

Like most people, you will probably soon find that Usenet is your favorite part of the Internet. You will be able to partake in discussions with people from all over the world on virtually every conceivable topic.

In this chapter, we will discuss the basics and lay the foundation for using Usenet. We will start by answering the question: What is Usenet? We will then cover the basic terminology that you will encounter.

After this orientation, we will discuss how Usenet data is transported from place to place and how it is organized. We will show you how Usenet works and what you can expect to find.

In the following chapters, we will build on this foundation and show you the details of accessing Usenet.

What Is Usenet?

Usenet is a large collection of discussion groups involving millions of people from all over the world. Each discussion group is centered around a particular topic. Jokes, recipes, mathematics, philosophy, computers, biology, science fiction—just about any subject you can think of has its own group.

In total, Usenet has over 5,000 different discussion groups. Many of these are of regional or local interest. For example, there is a discussion group for discussing restaurants in the San Francisco Bay Area. Nevertheless, over 2,500 groups are of general interest and are read by people throughout the world.

One of the first questions that people ask is: What does it cost to use Usenet?

The answer is that Usenet is free. You may have to pay something for Internet access (as we discussed in Chapter 3), but there is no charge for Usenet per se. In fact, if you have access to an Internet computer at no cost, then everything, including Usenet, is free.

Just about every topic of interest to human beings has a place in some Usenet discussion group or another. When the need arises for a new group, there are well established procedures to form one. Unlike the commercial services (such as Compuserve), there is no central authority that controls Usenet. Thus, when the users decide that there should be a new discussion group, one is formed.

This system has two important results: First, new groups can be created in a timely manner whenever the need arises. Second, there are a great many groups devoted to esoteric subjects. To sample this variety, take a quick look at the list of groups in Appendix G.



FUN TIP: No matter what your interests, there will be Usenet discussion groups for you.

Basic Terminology

The original Usenet network was conceived in order to display notices and news items. The idea was to create a computerized version of a bulletin board. Usenet soon outgrew its original blueprint, but the legacy of an electronic news network

remains. Although Usenet is used primarily for discussion groups, we still talk about it using news-oriented terms.

For example, Usenet itself is often referred to as *the news*, or *netnews*, even though there is little real news in the sense of a newspaper. For example, you might hear someone say, "I picked up a recipe for groat cakes while reading the news yesterday." What he means is that he found the recipe in one of the Usenet discussion groups.

Similarly, the Usenet discussion groups are usually referred to as *newsgroups* or, more simply, *groups*. Within each newsgroup, the individual contributions are called *articles* or *postings*. When you submit an article to a newsgroup, we say that you *post* the article. Thus, you can imagine the following conversation between two Usenet people in Fargo, North Dakota, both of whom happen to be named Mike:

Mike: These are great groat cakes. New recipe?
Mike: Yes, I saw an article in the vegetarian cooking
group about how nutritious groats are,
so I posted a request for recipes. The next day, someone
from France had posted a reply with this wonderful recipe
for groat cakes with truffles.
Mike: Well, they certainly taste good, Mike.
Mike: Thank you, Mike.

What Is It Like When You Use Usenet?

To read Usenet articles, you use a program called a *newsreader*. The newsreader acts as your interface: you tell it which newsgroups you want to read, and it presents the articles for you, one at a time. (Remember, although we call them newsgroups, they are really discussion groups.) You can read any newsgroup that you want, as long as it is carried by your local site. Most sites do not offer all the newsgroups, as it would take up too much disk space.

When you start to learn how to use your system, you may find that you have a choice of newsreader programs. If so, you can pick the one that suits you best. Newsreaders are usually complex programs with many commands, so it is a good idea to settle on one that you like and take some time to learn it well.

There are a number of different newsreaders, and each one works in its own way. On Unix systems, four of the most popular newsreaders are named **nn**, **rn**, **tin**, and **trn**. If you are using a Unix system, at least one of these newsreaders should be available on your system. In Chapter 11, we will talk about reading the news. In chapters 12 through 15 we will explain how to use the newsreader programs.

One of the functions of a newsreader is to keep track of which newsgroups you want to read. At any time, you can add or delete newsgroups from your list. When you add a group to your personal list, we say that you *subscribe* to that group. Similarly, if you indicate that you do not want to read a newsgroup any longer, we say that you *unsubscribe*.

Don't misinterpret these terms. There is no formal subscription process nor is there any fee. Subscribing simply means that you tell your newsreader that you wish to follow a newsgroup. Moreover, the system is private: no one keeps track of which groups you read.

As you subscribe and unsubscribe to groups, and as you read articles, your newsreader maintains a file on your behalf. This allows the newsreader to keep track of which newsgroups are subscribed to and which articles you have read. On Unix systems, this file is named **.newsrc** and is stored in your home directory.

As you read an article, the newsreader allows you to perform many different actions. For example, you can move from one article to another, save an article to a file, mail a reply to a person who posted an article, compose an article of your own, and so on. If you were to watch someone reading the news, you would see him reading text on the screen and, from time to time, typing a command.

In Chapter 2, we explained that many services are built upon a client/server relationship. The client is a program that requests a service; the server is a program that supplies this service.

When you read the news, you are using a client/server system. The actual articles are stored and managed by a program called a *news server*. Your newsreader acts as a client. Each time you enter a command to read an article, your newsreader requests that article from the news server.

A typical setup on a local area network is to have one computer run the news server program. This computer acts as the central news repository for everybody on the network.

Imagine many people, each at his or her own computer or terminal, reading the news. Each person is interacting with his or her own newsreader program. Each of these programs acts as a client, requesting services from the central news server.

Normally, the whole thing works automatically. You tell your newsreader that you want to read an article and it just appears. However, if the news server should go down (stop working) for some reason, no one will be able to read the news until the person in charge fixes the problem.

In some large networks, such as a university, many people may want to read the news at the same time. If the news server is busy, you may experience a delay at times, when your newsreader is waiting to receive an article.

Hint

On some networks, you may not be able to read the news at peak periods. When you start your newsreader, it will display a message telling you that the work load on the news server is too high and suggesting that you try again later. For example:

```
Server hub.ucsb.edu responded with code 400, which
probably means the load on the server is too high.
Please try again later.
```

What Is the Difference Between Usenet and the Internet?

The names "Usenet" and "Internet" sound familiar and—at least at first—you may find it easy to confuse the two, so let's take a moment and get the differences straight in our minds.

The name "Usenet" was chosen as an abbreviation for "users network". However, the name is a misnomer. Usenet is not really a network in the sense that a network is a group of computers connected together. Usenet is a collection of discussion groups.

The Internet, conversely, really is a network. More precisely, the Internet is a collection of tens of thousands of networks worldwide.

In other words, the Internet is a general-purpose carrier of information, while Usenet is simply one type of service that makes use of this capability. (As you know, the Internet supports a number of other services, such as electronic mail, remote Telnet connection, file transfer, and so on.)

Now, you might ask, is every Internet computer part of Usenet? The answer is no. Many Internet computers are owned by people who, for some reason, choose to not participate in Usenet. To be a part of Usenet requires someone to act as an administrator. It also requires that a computer with a large amount of disk space be available to hold news articles and programs. Some people do not want to spend the time and money that it would take to maintain such a system.

You might also ask, are all Usenet sites part of the Internet? Again, the answer is no. The only requirements to being part of Usenet are:

1. You must furnish a computer to act as a local repository for articles
2. You must have someone to administer the system
3. You must find another Usenet site to which your computer can connect in order to swap articles back and forth.

There are many computers that are part of Usenet, but are not on the Internet; they use some other networking system. One such system, used by many Unix computers, is called UUCP. We won't go into the details, except to mention that the Internet is faster, better, and more expensive. Suffice it to say that many people participate in Usenet by using UUCP—or another networking arrangement—instead of the Internet.

Who Runs Usenet?

The fascinating thing about Usenet is that there is no central authority. Usenet is run by the people who use it.

Usenet was started in 1979 in North Carolina (USA) as an experiment. The idea was to create an electronic bulletin board to facilitate the posting and reading of news messages and notices. At first, there were only two sites: the University of North Carolina and Duke University.

Before long, other places joined and, with the explosion of networking in the 1980s and 1990s, Usenet expanded enormously. Today, there are tens of thousands of Usenet sites with more than two and a half million participants.

Each Usenet site is run by a person called the *news administrator* or *news admin*. In some places, the news administrator is the same person who manages the system. However, this need not be the case. In large installations, such as a university, the news administrator may be a staff member, or even an ambitious volunteer, who reports to the system manager.

Each news administrator is responsible for his or her own site alone and nothing else. The arrangement works because the news administrators stay in contact with one another and cooperate. Indeed, there are Usenet newsgroups specifically set up for news administrators to have discussions.

This means that, in the global scheme of things, nobody can tell anybody else what to do. This lack of central authority is what gives Usenet its charm, and is

what distinguishes it from other discussion group systems (such as CompuServe or Prodigy) where there are Rules and People-In-Charge.

This is not to say that Usenet is total anarchy. There are a good number of conventions that have been developed over the years. As a responsible member of Usenet, you are expected to learn and follow these conventions. Still, when people misbehave—for example, by repeatedly sending silly or offensive messages to a newsgroup—more right-thinking people usually have to confine themselves to public criticism or to sending letters of complaint to the person's electronic mailbox.

Actually, as you will see, when there are no rules, most people choose to cooperate.

Hint

When everyone else is cooperating, acting like a jerk gets boring real fast.

How Is the News Transported?

If there is no central authority to coordinate everything, how are news articles transported all over the world? The answer is that new articles are passed from one computer to another. Eventually, a copy of each article spreads throughout the entire Usenet system.

Let's discuss a typical example in which you read articles in one of the Usenet newsgroups devoted to mathematics. As you read, you decide to post an article of your own. We will take a look at some of the details and follow your article as it travels around the world. The details may vary from one system to another, but the general idea is the same.

To begin, you log in to an Internet host and enter the command to run your newsreader program. You start by telling the newsreader which newsgroup you want to read. As we mentioned earlier, your newsreader keeps a file in which it records the newsgroups you subscribe to and the articles you have read.

In this case, you specify a particular mathematics group, so the newsreader checks its records and then connects to the news server to request the next article from this particular newsgroup. The news server responds by sending this article, which your newsreader then displays on your screen. When you finish reading an article, your newsreader requests and displays the next article.

As it happens, someone on the other side of the world has posted an article in which he asks if anybody has a shorter solution to Fermat's Last Theorem. You

remember that you came up with just such a solution the other day. You enter a newsreader command to tell the newsreader that you want to compose an article in response to the one you are currently reading. (This is called a "followup" article.)

Your newsreader starts a text editor program so you can compose your article. You say that you have a wonderfully short solution to the problem, but, unfortunately, it is a bit too long to write down at this time.

After you finish writing, you enter the command to quit the text editor. The newsreader program regains control and sends a copy of your new article to the central news server. At this point, your article can be read by anyone in your local network. However, the article still needs to be transported around the world.

Now, when one news server supplies Usenet articles to another server, we say that it offers a *news feed* or, more simply, a *feed*. In order to participate in Usenet, your news server gets a news feed from another Usenet site. From time to time, your news server will connect to the server at this site in order to get its feed.

Each time this happens, your server passes on any new articles that have not yet been received at the other location. In particular, the new article that you just composed will be sent out. Later, when that news server connects to its news feed, your article will be sent to another site. This is how Usenet works: your article is passed automatically from news server to news server, one connection at a time.

What makes the system fast is that some news servers act as way stations, providing news feeds for many other servers. Once your article hits one of these way stations, it will be sent to many news servers within a short amount of time.

By the next day, your article is available on virtually every news server in the country. Within two or three days, your article is available around the globe. By the end of the week, your great mathematical accomplishment is recognized worldwide and you are famous.

You might ask, if new articles are constantly being transported from one news server to another, don't they just pile up indefinitely? This is the same question that you might ask as you ride down an escalator: Why doesn't the basement fill up with steps?

With an escalator, the answer is, of course, that the steps are recycled. With Usenet, the solution is to keep each article for a certain amount of time and then throw it away. Each news administrator decides how long articles should be kept. The news server regularly checks the articles and deletes those that are older than the specified time interval. When this happens, we say that the articles have *expired*.

The news administrator at your location can specify different expiration dates for the various newsgroups. For example, he or she may decide that the newsgroup in which people discuss general Unix questions should have a longer expiration period than the newsgroup in which people swap tasteless jokes. (Or,

perhaps, the other way around.) If you want to know the exact expiration policy at your location, you will have to ask your local news administrator.

Hint

As a general rule, most newsgroup articles are kept from two days to two weeks.

How Big Is Usenet?

Since Usenet has no central administration, it is difficult to know exactly how big the system is. However, various people have implemented programs to estimate Usenet statistics.

At DEC Network Systems Laboratory, Brian Reid runs the Measurement Project, which regularly publishes such numbers. Reid uses a program called **arbitron** that many Usenet news administrators run on their systems. This program compiles data on how many people read Usenet, which newsgroups they read, and so on. (It is all anonymous, so privacy is protected.) Reid then analyzes the results mathematically and estimates statistics for the entire Usenet system.

Here are the statistics that were current at the time we wrote this chapter. (Remember, these are estimates. The actual numbers are not important. All we want is for you to have an appreciation for the size of Usenet.)

There were 76,000 different sites that carried Usenet. (Remember, not all of these are on the Internet.) At these sites, there were 8,445,000 users of all types, of whom about 29 percent (2,417,000) participated in Usenet.

Another question to ask is: How much data is actually transported on Usenet? To answer this question, the Measurement Project looks at how much Usenet data is received at its own news server. This computer, named **decwrl**, receives most of the newsgroups and provides a news feed to other servers.

On the average, **decwrl** received 26,400 new messages a day, containing a total of 56.2 megabytes (56.2 million characters) of data.

Within Usenet, the most important news feed service is provided by a company called Uunet Technologies. They also compile statistics on Usenet data. At the time we wrote this chapter, the numbers for the previous two weeks were as follows.

During that time, Uunet received 400,805 articles totaling about 744 megabytes. As we will see in Chapter 10, each Usenet article has standard information called a "header". The header contains technical data, the address of

the person who posted the article, the time and date, the subject of the article, and so on. Including headers, the 400,805 articles totaled about 951 megabytes.

This all works out to about 28,700 articles a day, comprising 53 megabytes per day (68 megabytes counting headers).

You might wonder, why are the Uunet numbers higher than the Measurement Project numbers? The answer is in the way articles are counted. When you post an article, you can send it to more than one newsgroup at a time. (This is called "cross-posting".) Uunet counts an article once for every newsgroup in which the article appears. The Measurement Project counts each article only once.

To continue, Uunet reported that during this two week period, the articles came from 31,123 different Usenet sites and were submitted by 91,726 different users.

To put this all in perspective, in January 1989, Usenet had about 450 newsgroups. (There are now over 5,000.) At that time, there was an average of 1,400 articles a day, consisting of 3 megabytes of data.



FUN TIP: Usenet consists of about 2.5 million people, using 76,000 different computers, who post about 27,000 new articles every day. If all the people who participate in Usenet were laid end to end, they would stretch from Lausanne, Switzerland, to the confluence of the Tigris and Euphrates rivers.

The Mainstream and Alternative Hierarchies

As we have explained, there are a vast number of Usenet newsgroups, with new ones being added all the time. In order to make everything manageable, we use a system in which the newsgroups are collected into categories called *hierarchies*. Each hierarchy has its own name and is devoted to a particular area of interest. Table 9-1 shows the most important hierarchies.

The hierarchies in Table 9-1 are distributed all over the world (although, as we will see in a moment, not all hierarchies are carried at every Usenet site). Each newsgroup is given a name that consists of two or more parts, separated by . characters. The first part of the name is the hierarchy to which the newsgroup belongs.

For example, in the **news** hierarchy, there is a newsgroup for people who are learning how to use Usenet. In this newsgroup, you can ask any question you want about Usenet and some kind soul will answer you. The name of this newsgroup is **news.newusers.questions**.

Here is another example. In the **rec** hierarchy, there are several newsgroups devoted to various aspects of Star Trek. For discussions about Star Trek conventions and memorabilia, there is **rec.arts.startrek.fandom**. For reviews of Star Trek episodes, movies, and books, you can read **rec.arts.startrek.reviews**.

Name	Topic
alt	alternative newsgroups, many different topics
bionet	biology
bit	many topics: from Bitnet mailing lists
biz	business, marketing, advertisements
comp	computers
ddn	Defense Data Network
gnu	Free Software Foundation and its GNU project
ieee	Institute of Electrical and Electronics Engineers
info	many topics: from University of Illinois mailing lists
k12	kindergarten through high school
misc	anything that doesn't fit into another category
news	about Usenet itself
rec	recreation, hobbies, the arts
sci	science of all types
soc	social issues
talk	debate on controversial topics
u3b	AT&T 3B computers
vmsnet	DEC VAX/VMS and DECNET computer systems

Table 9-1. *The Most Important Usenet Newsgroup Hierarchies*

The two Star Trek examples each have a four-part name, with the first three parts the same. This is the system that is used to name newsgroups. The first part is the hierarchy; the other parts show categories and sub-categories. For example, the newsgroup devoted to science fiction movies is **rec.arts.sf.movies**. The group for discussing boating is **rec.boats**.

The Usenet hierarchies are divided into two categories, *mainstream* and *alternative*. The mainstream hierarchies are carried on all Usenet news servers. The alternative hierarchies are considered optional. Many Usenet sites carry them, but many choose not to.

At each Usenet site, the news administrator decides which hierarchies and newsgroups to carry. At most places, you can expect to see all the mainstream

hierarchies and at least some of the alternative hierarchies. However, bear in mind that even if your site carries a particular hierarchy, it may not have all the newsgroups in that hierarchy.

The principal difference between the two types of hierarchies has to do with the way new groups are formed. As we will explain later, people must follow well-defined procedures to start a mainstream newsgroup. There must be a discussion and a vote, and enough people must express interest. The alternative hierarchy is more lax: any person who knows how to do so can start a newsgroup. Historically, the mainstream hierarchies were the original Usenet categories and were strictly controlled. The alternative hierarchies came later and were created as a less-restrictive facility.

As a whole, mainstream newsgroups tend to be more stable and are more readily accepted by news administrators. For example, an administrator who is short of disk space might decide that **news.newusers.questions** is more important than **alt.sex.bondage**. Even so, many of the alternative newsgroups are popular and are widely circulated. However, as you might imagine, you will often find ridiculous or spurious alternative newsgroups, especially within the **alt** hierarchy.

Hint

The distinction between mainstream and alternative newsgroups is made only for purposes of organization. When you read the articles, there is no real difference. The main consideration is that your news server will probably not carry all of the alternative newsgroups.

There are seven mainstream hierarchies and eleven alternative hierarchies. They are shown in Tables 9-2 and 9-3.

The Cultural, Organizational, and Regional Hierarchies

In Tables 9-2 and 9-3, we listed the mainstream and alternative hierarchies along with the number of newsgroups. In total, we had 972 mainstream groups and

Hierarchy	Number of Newsgroups
comp	459
misc	40
news	22
rec	273
sci	71
soc	87
talk	20
TOTAL	972

Table 9-2. *The Mainstream Usenet Newsgroup Hierarchies*

Hierarchy	Number of Newsgroups
alt	586
bionet	41
bit	192
biz	32
ddn	2
gnu	28
ieee	12
info	39
k12	36
u3b	5
vmsnet	32
TOTAL	1005

Table 9-3. *The Alternative Usenet Newsgroup Hierarchies*

1,005 alternative groups. We said earlier that there are over 5,000 different newsgroups. What about the others?

As you remember, the mainstream and alternative hierarchies are distributed worldwide (although most Usenet sites do not carry every single newsgroup). In addition, there are many other hierarchies that are more cultural, organizational, or local in nature. These hierarchies are usually circulated only in their areas of interest.

Cultural hierarchies contain newsgroups devoted to a particular people. For example, four of the more well-known cultural hierarchies are **de** (German [Deutsch] newsgroups), **fj** (Japanese), **aus** (Australian), and **relcom** (Russian).

Although most Usenet articles are in English, the cultural hierarchies often have articles in their own language. For such articles, you may need special software to display the non-English characters. For example, some of the Japanese articles in the **fj** hierarchy use the Kanji alphabet.

The organizational hierarchies contain newsgroups devoted to a university, company, or other organization. There are many such hierarchies. For instance, the University of California at Santa Barbara has its own hierarchy, named **ucsb**. Trinity College in Dublin, Ireland, has a **tcd** hierarchy. If you are part of a large organization, chances are it has its own hierarchy and its own newsgroups.

Within the **ucsb** hierarchy, for example, are newsgroups of interest to the university community. For instance, there is a group named **ucsb.rides** for people who need local transportation. There is also a newsgroup, **ucsb.english.eng109c**, just for one particular English class. As you might imagine, such newsgroups are created and dissolved as the need arises.

Aside from the cultural and organizational hierarchies, there are a large number of regional hierarchies. These contain newsgroups of interest to people who live in a particular area. For example, the **ba** hierarchy contains many newsgroups pertaining to the San Francisco Bay area, such as **ba.market.housing**.

Many of the cultural, organizational, and regional hierarchies are distributed throughout the world. For example, many Usenet sites carry the **fj** hierarchy. After all, there are Japanese people everywhere and not just in Japan. Similarly, someone who used to live in San Francisco may want to read a regional newsgroup to keep in touch. Moreover, if you are planning to move to a new area, you may want to post an article of your own to a regional newsgroup to ask for information, say, about housing.

Clarinet: Real News

As you know, we refer to Usenet as the "news" and to discussion groups as "newsgroups". However, Usenet itself does not carry real news in any organized

fashion. There is a real news service, however. It is named Clarinet, and it has its own hierarchy, **clari**.

The Clarinet service is provided by a private company. Unlike Usenet, if your organization wants to carry Clarinet, it must pay for it. However, the costs are reasonable, and many organizations choose to offer this service.

The **clari** hierarchy contains many different newsgroups. (At the time we wrote this chapter, there were 246 such groups.) Some are of global interest, others are regional or local. Figure 9-1 shows a few of these groups. If your news server carries Clarinet, you can read any of these groups using your newsreader program. You will notice no difference from reading a Usenet newsgroup, except that virtually all the groups are for reading only; you cannot post articles to them.



FUN TIP: Although your organization must pay a subscription fee to offer the Clarinet newsgroups, there is no cost to you to read them.

Clarinet news comes from a number of different sources, including a live UPI newswire. The various newsgroups contain a wide variety of topics. They cover just about anything that is news, and the articles are continually being updated. Many of the newsgroups are devoted to specialized areas or to features, such as well-known columnists.

If your news server does not carry the **clari** hierarchy, there is still a Clarinet newsgroup that you can read. It is named **biz.clarinet.sample**. Clarinet regularly sends interesting articles to this newsgroup as free samples.

```
clari.biz.economy.world
clari.biz.market.report
clari.canada.politics
clari.feature.dave_barry
clari.feature.movies
clari.local.new_york
clari.nb.ibmclari.net.newusers
clari.news.books
clari.news.sex
clari.sports.tennis
clari.tw.science
```

Figure 9-1. *Examples of Clarinet newsgroups*

How Many Different Newsgroups Are There?

Now that we have covered all the hierarchies, we are in a good position to answer the question: How many Usenet newsgroups are there?

The total number of newsgroups changes frequently, especially in the alternative hierarchies. At the time we wrote this chapter, there were 972 mainstream groups and 1,005 alternative groups. This makes a total of 1,977 newsgroups that are widely distributed throughout the world. (For reference, there is a list of these groups in Appendix G.)

There were also an estimated 3,325 cultural, organizational, and local groups. (We say "estimated" because many such groups never make it outside their local news server.) To this, we can add the 246 Clarinet groups.

This gives us a grand total of 5,548 different newsgroups.



FUN TIP: Even before we could finish writing this chapter, the number 5,548 was out of date. To give you a feeling for how Usenet grows, we looked at figures from six months ago. At that time, there were 4,410 newsgroups. Thus, in the last six months, the number of newsgroups had grown by 1,139 (about 26 percent).

Newsgroup Naming Conventions

Newsgroup names are easy to understand. The first part of the name shows the hierarchy. For example, a group whose name begins with **comp**, such as **comp.unix.questions**, is in the computer hierarchy; the groups whose names begin with **talk**, such as **talk.environment**, are debate oriented; and so on. You can expect to find the weirdest newsgroups in the **alt** hierarchy. For example, that is where you will find **alt.sex.bestiality**.

Thus, when you are checking out a list of newsgroups (such as the one in Appendix G), trying to decide what to read, start by looking at the first part of the name. To show you some examples, Figure 9-2 contains a list of the

```
alt.tasteless.jokes
rec.humor
rec.humor.funny (moderated)
```

Figure 9-2. *Newsgroups devoted to jokes*

newsgroups devoted to jokes, and Figure 9-3 shows all the newsgroups that have something to do with sex. (As you might imagine, these newsgroups have some of the largest audiences on Usenet.)

```
alt.binaries.pictures.erotica
alt.binaries.pictures.erotica.blondes
alt.binaries.pictures.erotica.d
alt.binaries.pictures.erotica.female
alt.binaries.pictures.erotica.male
alt.homosexual
alt.politics.homosexuality
alt.politics.sex
alt.sex
alt.sex.bestiality
alt.sex.bondage
alt.sex.fetish.feet
alt.sex.masturbation
alt.sex.motss
alt.sex.movies
alt.sex.pictures
alt.sex.pictures.d
alt.sex.pictures.female
alt.sex.pictures.male
alt.sex.sounds
alt.sex.stories
alt.sex.stories.d
alt.sex.wanted
alt.sex.wizards
alt.sexual.abuse.recovery
clari.news.group.gays (moderated)
clari.news.law.crime.sex (moderated)
clari.news.sex (moderated)
rec.arts.erotica (moderated)
soc.bi
soc.motss
```

Figure 9-3. *Newsgroups related to some aspect of sex*

After the hierarchy name, you will see categories and (possibly) sub-categories. When two newsgroups are related, they will have similar names, differing only in the last part of the name. For example:

```
alt.binaries.pictures.erotica.blondes
alt.binaries.pictures.erotica.female
alt.binaries.pictures.erotica.male
```

Another point to notice is that some of the newsgroup names end in **.d**. This means that the group is for discussion of the contents of another group. For example, we have:

```
alt.sex.stories
alt.sex.stories.d
```

The first group contains stories about sex (check it out sometime) and only stories. The second group is for people who want to discuss the stories. If you post an article to **alt.sex.stories** that is not a story, someone may remind you that all discussion should take place in the **.d** group. This is so people who want to read stories only do not have to bother with anything else.

There is another important example of this convention that you should know about. The newsgroup **rec.humor** is for people who want to post and read jokes. If you want to discuss jokes, you should post to **rec.humor.d**. This is also the place to send a request like, "Does anybody have the list of all the light bulb jokes?" (That is why we did not list **rec.humor.d** in Table 9-2. It does not contain jokes. Rather, it is a group in which people talk *about* jokes.)

Hint

In the non-moderated joke newsgroups (**rec.humor** and **alt.tasteless.jokes**), the unofficial rule is that all postings must contain at least one joke. You will often see postings in which a person has given in to temptation and made some sort of non-joke comment. However, to be polite and to adhere to the convention, he or she has also included a joke. This is called an *obligatory joke* or an *objoke*.

You will sometimes see the same prefix, "ob", used in other groups to indicate the same convention: that a person is respecting the purpose of the group by making sure to include a relevant item within an otherwise questionable posting.

Moderated Newsgroups

You will notice that some of the newsgroups listed in Figures 9-2 and 9-3 (jokes and sex) are *moderated*. This means that you cannot post articles directly to the group. You do submit articles in the regular way, but they are automatically rerouted and sent to one person, called the *moderator*, who decides what will go in the group. A moderator—who is an unpaid volunteer—will not only pass judgment on what goes into a group, he or she will often edit and organize the articles.

The reason for moderators is to minimize the number of low-quality articles in a newsgroup. Perhaps the best examples are the humor groups. The groups **rec.humor** and **alt.tasteless.jokes** are not moderated. This means that anyone can post to these groups, with the result that there is usually a lot of silliness and repetition (not to mention old jokes).

The group **rec.humor.funny**—started by Brad Templeton, the creator of Clarinet—is moderated. Only those jokes that are deemed funny by the moderator, Maddi Hausmann, are posted. People all over the world submit jokes to Maddi, and she picks the ones she thinks are best. As a result, **rec.humor.funny** is read by more people than **rec.humor** (although they are both popular).

Some moderated newsgroups offer a special type of posting called a *digest*. The moderator creates a digest by collecting submissions, questions, answers, and bits of information. He or she will edit the information into a series of interesting items and post it as one large article, the digest. Two examples of such newsgroups are **comp.sys.ibm.pc.digest** and **sci.psychology.digest**.

A digest is much like an issue of an electronic magazine, with a volume and issue number, and a table of contents. Most newsreader programs have a special command to let you jump from one item to another as you read a digest.

Overall, the idea of moderated newsgroups provides a nice balance to Usenet. Many newsgroups carry a lot of low-quality articles, and it can be nice to read a group where everything is interesting (at least by one person's standards).

Of course, moderated newsgroups are a form of censorship because one person controls everything that is posted to the group. However, Usenet has many groups, almost all un-moderated, so you need never feel deprived.

What Are the Most Popular Newsgroups?

One of the most interesting questions about Usenet is: What are the most popular newsgroups? There is more than one way to answer this question depending on what you mean by "popular".

At regular intervals, the Measurement Project (which we mentioned earlier in the chapter) sends estimates of readership statistics to several newsgroups. (If you want to read such postings, check out the group **news.lists**.)

One of the postings identifies the 40 most popular newsgroups based on the estimated number of total readers. The thing is, not all groups are carried at all Usenet sites. For example, many news administrators will not carry groups like **alt.sex** even though, as you can imagine, they are extremely popular.

Let's take a look at some of these statistics. One of the perennial favorites is **rec.humor.funny**. At the time we wrote this chapter, this newsgroup had an estimated audience of 160,000 people. The group **alt.sex.stories** had only 130,000 readers. By this standard of measurement, the humor group is a lot more popular.

However, **rec.humor.funny** is carried at 82% of all Usenet sites. The group **alt.sex.stories** is carried at only 53% of the sites. The real question is: How well would a particular newsgroup do if it were carried at all sites? In other words, instead of asking how many people read the group, we ask, how much do people like the group?

To come up with such numbers, we start with two of the statistics estimated by the Measurement Project: the total number of readers and the percentage of sites that carry each newsgroup. To compare readership equally, we divide the total readers by the percentage at that site. This gives us an estimate of how many total readers a newsgroup would have if it were carried at all sites.

For example, **rec.humor.funny** would have 160,000 divided by 0.82, or 195,122 readers. The **alt.sex.stories** group would have 130,000 divided by 0.53, or 245,283 readers. Thus, the sex group is more popular than the humor group.

To make these hypothetical readership numbers easier to understand, we normalize them, setting the most popular one to the value 100. To do so, we divide each such number by the hypothetical readership of the most popular group and multiply by 100. The most popular newsgroup is **news.announce.newusers**, which contains valuable information for new Usenet users. It has a hypothetical readership of 280,000 divided by 0.91, or 307,692.

The result is a number between 0 and 100, which provides a picture of true popularity. We call this value the *Hahn Popularity Index* or *HPI*. For example, **rec.humor.funny** has an HPI of 195,122 divided by 307,692 multiplied by 100, or 63.

Table 9-4 contains the relevant numbers with all the arithmetic done for the 25 most popular Usenet newsgroups at the time we wrote this chapter. Draw your own conclusions.



FUN TIP: According to our research, the most popular newsgroup that could exist would be **alt.sex.jobs.offered.newusers**.

Ranking	Newsgroup	HPI	Readers	Sites
1	news.announce.newusers	100	280,000	91%
2	misc.forsale	98	250,000	83%
3	misc.jobs.offered	92	240,000	85%
4	alt.sex	87	180,000	67%
5	news.answers	82	220,000	87%
6	alt.sex.stories	80	130,000	53%
7	alt.binaries.pictures.erotica	71	120,000	55%
8	rec.arts.erotica	67	150,000	73%
9	rec.humor.funny	63	160,000	82%
10	alt.sex.bondage	57	110,000	63%
11	alt.activism	53	110,000	68%
12	rec.humor	52	130,000	81%
13	alt.binaries.pictures.misc	49	88,000	58%
14	news.groups	47	130,000	90%
15	news.announce.newgroups	47	130,000	90%
16	soc.culture.indian	45	100,000	73%
17	news.newusers.questions	43	120,000	90%
18	comp.graphics	42	110,000	85%
19	comp.lang.c	41	110,000	88%
20	misc.jobs.misc	39	100,000	83%
21	alt.bbs	37	83,000	72%
22	misc.wanted	37	93,000	81%
23	comp.binaries.ibm.pc	37	94,000	82%
24	alt.sources	37	89,000	79%
25	talk.bizarre	36	80,000	72%

Table 9-4. *The 25 Most Popular Usenet Newsgroups*

How Are New Newsgroups Started?

A new newsgroup is created by sending a special message, called a *control message*, throughout Usenet. Various types of control messages are used by news administrators to govern the operation of Usenet. There is one particular type of control message that is used to start a new group. (There is also another type of control message to remove an obsolete or spurious group.)

When a news administrator sends out such a message, it propagates from news server to news server, just like a regular article. Eventually, the message reaches all the news administrators. Each administrator decides whether or not to create the new group on his or her system.

News administrators almost always honor an authorized request to start a new mainstream group. This is because such a message will not be issued until certain well-defined criteria are met (described below).

Conversely, a control message to start an alternative newsgroup can by issued be anyone who knows how to do it. As you might imagine, there are all sorts of requests to start bizarre alternative groups. For this reason, many news administrators will pick and choose which new alternative newsgroups they are willing to create on their systems.

Here is how a new mainstream group gets formed.

To start, somebody must have the idea for a new group. Perhaps an existing group should be split into two. Perhaps a whole new area of discussion has arisen. Most ideas are discussed in existing newsgroups or by mail for some time, as ideas are proposed, examined, and modified.

Once the idea for the new group is set, someone sends a message to the group **news.announce.newgroups** (which is moderated). At the same time, the message is cross-posted to any other relevant groups.

The moderator of **news.announce.newgroups** will post an article explaining the name and purpose of the proposed group. There is now a 30-day discussion. This discussion takes place in the newsgroup, in other, related newsgroups, and by private mail correspondence.

At the end of 30 days, if there is clear agreement as to the name and purpose of the group, the moderator of **news.announce.newgroups** will post a general request for people to vote. Anyone who is interested can vote (once) by sending an appropriate mail message to a specified address. The period of voting is set in advance and is usually between 21 and 31 days.

At the end of the voting period, the totals are posted along with a full list of everyone who voted and how they voted. (No secrets on Usenet.) There is now a five-day waiting period in which anyone can request a correction to a particular vote or to the voting procedure.

At the end of five days, the vote is finalized. The vote is successful if the new group was approved by at least two thirds of the voters, and if the yes votes were

at least 100 more than the no votes. After a successful vote, the moderator of **news.announce.newgroups** sends out a control message to start the new group.

If the vote was unsuccessful, the same newsgroup cannot be brought up for discussion for at least six months.

Hint

If you want to start a new alternative group, do the following:

1. Make sure that you have at least a few months of Usenet experience, so you understand how the system works.
2. Suggest your idea by posting an article in one or more existing groups (choose appropriately), and see what other people think. The newsgroup **alt.config** is used for discussing proposals for new alternate groups. Remember, alternative groups can only be created successfully if other people cooperate.
3. Read **news.announce.newgroups** for a while to see what types of issues arise during the formation of new groups.
4. After a reasonable amount of time—and as a reader of this book we know you are reasonable—decide if you still want to start the group. If so, ask your local news administrator for help in sending out the control message.

Frequently Asked Question (FAQ) Lists

As you start to use Usenet, you will find that you have many questions. Moreover, as you read the various newsgroups, you will have questions about those topics. Many of these questions will be the same ones that everybody else asks when they first start to read a particular newsgroup.

For example, the group **misc.consumers** is often used to discuss consumer credit. A common question is: "What is the difference between Visa and Mastercard?" Here is another example. The newsgroup **rec.arts.disney** discusses anything related to the world of Disney. A question often asked by newcomers is: "Is it true that Walt Disney was frozen in cryogenic suspension after he died?"

You will find that the participants of many newsgroups have asked and answered questions like these many times. Although they are interesting to a

beginner, you can imagine that experienced readers grow tired of seeing the same questions posted repeatedly.

The solution is the *frequently asked question list* or *FAQ*. An FAQ list is a document, maintained by a volunteer, that identifies and answers all the frequently asked questions for a particular newsgroup. Many newsgroups have FAQ lists which the keeper of the FAQs posts to the group regularly. It is considered good manners to refrain from asking questions in a newsgroup until you have read the FAQ list. (Although not all newsgroups have such lists.)

There are four ways to get a FAQ list. First, you can read a newsgroup regularly and, if there is a FAQ list, it will be posted eventually. (Typical intervals are every week, every two weeks, or every month, depending on the group.)

Second, you can read the newsgroup named **news.answers**, which consists of nothing but FAQ lists and related material. This is an interesting group to follow as you will see lists of the best questions and answers on topics you would normally not read about.

The third way to get a FAQ list is to use Anonymous FTP to download the FAQ list from a Usenet archive. (We will explain how to do this in Chapter 16.)

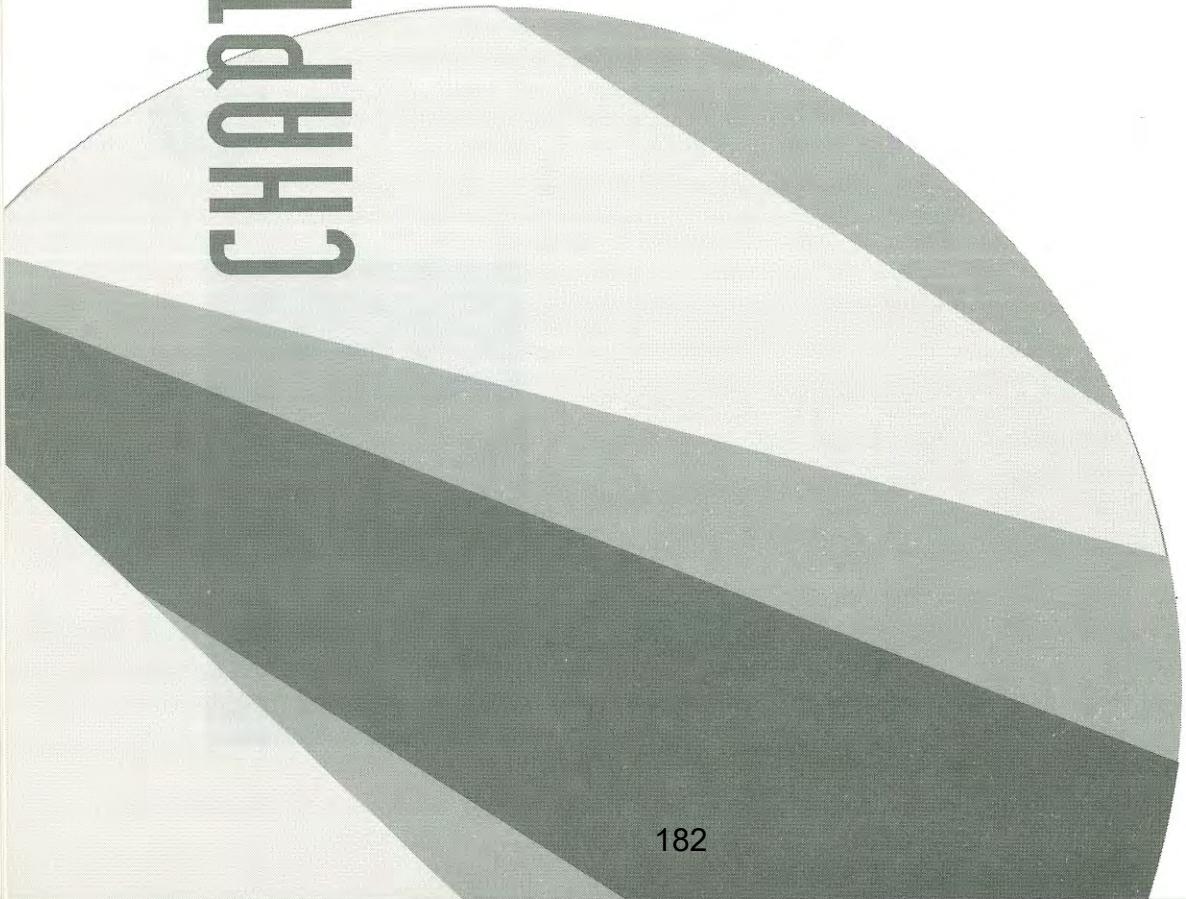
Finally, after trying the other methods, you can post a short article to the group asking if a FAQ list exists and, if so, can someone mail you a copy or tell you how to get it.



FUN TIP: Learn how to use Anonymous FTP and take a look at the FAQ lists in the Usenet archives. There are all kinds of interesting questions and answers.



CHAPTER 10



Understanding Usenet Articles

In Chapter 9, we introduced Usenet, the worldwide collection of over 5,000 different discussion groups. We saw that people participate by posting articles to the various groups and by using a program, called a newsreader, to read the articles.

In this chapter, you will find out what you can expect to see when you participate in Usenet. We will show you what a typical article looks like and how to make sense out of all the technical information. We will then discuss the conventions of Usenet: what guidelines you will want to follow and what new terminology you can expect to encounter.

After reading this chapter, you will be prepared to start reading the news and posting your own articles. In Chapter 11, we will discuss the general considerations. In chapters 12 through 16, we will explain how to use the various newsreader programs.

The Format of a News Article

A news article consists of three parts: a *header*, followed by the *body*, followed by an optional *signature*.

The header contains technical information about the article. There are twenty different types of lines that you might commonly see in the header, each of which contains a different type of information. We will discuss headers in detail in the next section.

The body of the article is actual text: the main part of the article.

Finally, the signature consists of a few lines that come at the end of the article. These lines are composed by the person who sent the article and are automatically appended to the end of every article that he or she posts. We will talk more about signatures later in the chapter.

Figure 10-1 contains a typical Usenet article. In this article, the header consists of the first 13 lines. The body of the article consists of the next 7 lines. In this case it happens to be a joke. Most articles are longer than our example, and, in most cases, the body is larger than the header. Finally, the last 4 lines of our sample contain a signature. In our example, the signature identifies the person who posted the article, along with an address, phone number, and a short quotation.

The Header

All Usenet articles must have a header. A header consists of special lines that are at the beginning of the article. As we mentioned in the last section, there are twenty different types of lines that you might see in a header. These are listed in Table 10-1.

Not all articles will contain all twenty lines. For example, the article in Figure 10-1 contains only 13 header lines. These are the ones that you will see most often. For reference, though, we will briefly describe all twenty types of header lines, so you will be able to understand them when you see them.

```
Path: ucsbcsl!mustang.mst6.lanl.gov!nntp-server.caltech.edu!
news.claremont.edu!uunet!news.univie.ac.at!email!mich
From: mich@music.tuwien.ac.at (Michael Schuster)
Newsgroups: rec.humor
Subject: The Secret of Life
Summary: Advice for understanding life.
Keywords: life, philosophy
Message-ID: <1993May14.073130.15261@email.tuwien.ac.at>
Date: 14 May 93 07:31:30 GMT
Distribution: world
Sender: news@email.tuwien.ac.at
Organization: Tech Univ Vienna, Dept of Realtime Systems, AUSTRIA
Lines: 12
Nntp-Posting-Host: idefix.music.tuwien.ac.at

Here is some great advice I just found:

"When you get serious about foolishness,
you are getting into serious foolishness."

By the way, does anyone know who first said it?

-
Michael Schuster            |"I love you for your beauty; love me
TU Vienna, Austria          | though I am ugly"
mich@music.tuwien.ac.at     | - Miguel Cervantes, Don Quixote
+43/1/12345
```

Figure 10-1. *A typical Usenet article*

Before we do, you should understand that whether or not you even see the header at all depends on which newsreader program you are using and how it is configured. Some newsreaders show you all of the header lines by default. Other newsreaders will not show you any header lines unless you ask for them; you will see only the body and the signature. However, all newsreaders allow you to customize how articles should be displayed. Moreover, as you are reading an article, it is always possible to ask your newsreader to redisplay the current article along with the full header.

We will now discuss the various types of header lines that you might see. Few articles contain all twenty of the standard header lines, but it is a good idea to know what they mean. The header lines in Figure 10-1 are the most common ones.

Please do not think that you have to memorize what all these header lines mean. Just read through this section once. Then, whenever you see an article with a header line that puzzles you, you can look it up.

(From time to time, you will encounter other types of header lines that we do not mention here. Almost always, these will be non-standard lines that you can safely ignore.)

Header Line	Description
`Approved:`	identifies moderator who posted article
`Control:`	contains special administrative commands
`Date:`	time and date that article was posted
`Distribution:`	recommendation for where to send article
`Expires:`	recommendation for when to remove article
`Followup-To:`	shows where followup articles will be sent
`From:`	userid and address that posted the article
`Keywords:`	one or more words to categorize the article
`Lines:`	size of the body + signature
`Message-ID:`	unique identifier for the article
`Newsgroups:`	newsgroups to which the article was posted
`NNTP-Posting Host:`	name of Internet host that posted article
`Organization:`	describes the person's organization
`Path:`	shows transit route of the article
`References:`	identifies article to which followup refers
`Reply-To:`	address to send personal replies
`Sender:`	address of computer that sent out article
`Subject:`	short description of contents of article
`Summary:`	one-line summary of article
`Xref:`	local cross-posting information

Table 10-1. *Header Lines for Usenet Articles*

Approved: In Chapter 9, we explained that some newsgroups are moderated. This means that when you post an article to such a group it is not sent to the group directly. Rather, it is sent to a person, called the moderator, who decides which articles should be sent to the group. Within moderated groups, the **Approved:** header contains the mail address of the moderator. This header line is also used with some types of **Control:** messages.

Control: This type of header line contains special commands that are used by news administrators to control the Usenet system. For example, there is a special type of control line that is used to start a new newsgroup. It is possible that you will never see a **Control:** line, as it is not used in regular articles.

Date: This line shows the time and date that the article was posted. The time will often be in Greenwich Mean Time [GMT], which is the standard Internet time. (Sometimes GMT is called Universal Time.) We discuss the Internet time conventions in Chapter 5.

Distribution: When you post an article, you will be given a chance to specify where you would like the article to be sent. Some news posting programs will show you several choices and ask you to pick one. Typical choices include: your organization, your region, your country, and worldwide. On some systems, you are not shown the choices, you just have to know what is available in your area. As an example, here are the choices you will see when you post an article at the University of California at San Diego.

```
local:   local to the site
 ucsd:   local to the UCSD campus
   uc:   all the University of California campuses
sdnet:   local to San Diego County
   ca:   everywhere in California
  usa:   everywhere in the USA
   na:   everywhere in North America
world:   everywhere in the world
```

Here is another example, from one of the computers at the Technical University in Vienna, Austria.

```
inst182:   local to department #182 (Institut #182)
 tuwien:   local to university (Technische Universitaet Wien)
     at:   everywhere in Austria
 europe:   everywhere in Europe
  world:   everywhere in the world
```

(Note: The international domain name for Austria is **at**.) An important point to understand is that a distribution does not guarantee where the article will be propagated. It is only a suggestion. Each news administrator decides which distributions he or she will accept. Some administrators will accept every newsgroup they can get. Thus, for example, we have had the experience of logging in to a computer in Palo Alto, California, and reading local news from Edmonton, Canada.

Expires: As you know, each news administrator sets up the time intervals that articles will be retained on his or her news server. The **Expires:** header is used when you want to recommend a different expiration date.

For example, if an article is announcing an upcoming seminar, it makes sense to have the article expire the day after the seminar. Most articles do not use this header line. They expire according to whatever the local policy happens to be.

Remember though, it is the news administrator that has the last word about when articles expire. If you see an article with a particular expiration date, it's completely possible that the article may disappear before then.

Followup-To: From time to time, you may wish to submit an article in which you respond to a previous article. In such cases, your posting is called a *followup* article. All newsreader programs make it easy to compose and post followup articles.

Normally, a followup article will be sent to the newsgroup (or newsgroups) in which the original article appeared. However, in certain cases, a person may want to control where the followup articles should go. In such cases, a **Followup-To:** header line specifies to which newsgroup such articles should be sent.

For example, say that you post a joke to the **rec.humor** newsgroup. It happens to be the type of joke that you know people will want to comment on. However, the **rec.humor** newsgroup is supposed to be for jokes only. Discussion of jokes should be in the group **rec.humor.d**. So, when you post the joke to **rec.humor**, you can use a **Followup-To:** header line that specifies **rec.humor.d**. This technique is also handy when you post an article to more than one newsgroup and you want to direct all subsequent discussion to a single group.

There is one special followup designation that you should know about. If this header line specifies **poster**, it means that followup articles cannot be sent to the newsgroup. Rather, you should continue the discussion with the person who posted the article by sending a message to his or her personal mail address. (Your newsreader program will help you do this easily.)

From: This header line is an important one, as it tells you who posted the article. You will always see the person's mail address. In most cases, you will also see their real name.

Although knowing who posted an article is mildly interesting, the **From:** header line comes in handy in another way. There is a computer named **rtfm** at MIT that provides a Usenet archive service. Many important Usenet articles—such as the FAQ (frequently asked question) lists—are stored on this computer, and you can access them by using Anonymous FTP (see Chapter 16). This computer also offers another service called the Usenet address server. A program on **rtfm** scans all the Usenet articles and stores the **From:** header lines in a database. You can send a mail request to this server, asking it to search for a particular person. You will get a response, also by mail. If that person has posted to Usenet, there is a good chance that the address server can find him. The details are explained in Chapter 22.

(By the way, the old name of the **rtfm** computer was **pit-manager**, in case you ever see the name somewhere. The significance of the new name, **rtfm**, will become clear later in the chapter.)

Hint

It is possible for people to change the real name that is shown on the **From:** header line. So, if something looks suspicious, be suspicious.

Keywords: This line contains one or more words or terms that categorize the content of the article. Some people look at this line to decide if they want to read the article (although not as often as the **Subject:** line).

Lines: This header line is straightforward. It shows the total number of lines in the article. The number of lines includes the body and the signature, but not the header.

Message-ID: On this line is a unique identifier that is generated automatically by the program that sent out the article. The last part of this name is the address of the computer the article was posted from. This information is used only by news programs. You can ignore it.

Newsgroups: This is an important line. It shows to which newsgroups the article has been posted. When you are using a newsreader program to look at an article, it has obviously been posted to the newsgroup you are reading. However, if you are looking at an old article—that you have previously saved, that someone has mailed to you, or that you have found in the Usenet archive mentioned above—it is handy to know what newsgroup the article originally appeared in.

When you post an article, you will be asked to specify what group you want to send the article to. If you want, you can specify more than one group. This is called *cross-posting*. When an article is cross-posted, you will see more than one group name in the **Newsgroups:** header line.

It is considered good etiquette to post to only one or, at most, a small number of newsgroups. It is not appreciated when people send an article to many newsgroups, most of which have only a tenuous relationship to the topic under discussion. For example, if you have a Unix question, send it to the most appropriate of the many different Unix-oriented newsgroups. Do not send it to every Unix group.

Hint

When you cross-post an article, make sure you put a **Followup-To:** line in the header to direct subsequent discussion to one specific group. If appropriate, consider using **Followup-To: poster** to direct all subsequent discussion to your personal mailbox.

NNTP-Posting Host: In Chapter 2, we explained that TCP/IP is a large family of protocols. (A protocol is a set of technical rules.) We have already discussed TCP (Transmission Control Protocol) and IP (Internet Protocol), which are used to transport data, and SMTP (Simple Mail Transfer Protocol), which is used for sending mail messages. In Chapter 16, we will talk about FTP (File Transfer Protocol), which is used to copy files from one Internet host to another.

The TCP/IP protocol that is used to transport Usenet articles is called NNTP (Network News Transfer Protocol). The **NNTP-Posting Host:** header line shows the name of the Internet computer from which the article was posted.

In other words, this line will show you what computer the person was using when he or she posted the article. The **Sender:** line, if present, shows the name of the computer—often the news server—that actually sent out the article over Usenet.



FUN TIP: If you want to hunt for forgeries (which do turn up from time to time), look for articles in which the name of the computer in the **NNTP-Posting Host:** *header line does not agree with the name of the computer in the* **From:** *line. It is easy to change the* **From:** *line, but not so easy to modify the* **NNTP-Posting Host:** *line.*

Organization: This header line contains a short phrase describing either the organization to which the person who posted the article belongs, or the organization that owns the computer. The purpose of this header line is to help identify the person who posted the article. The **From:** line does contain an address, but sometimes such addresses can be difficult to understand.

Path: The information on this line consists of a number of computer names, separated by ! (exclamation mark) characters. The names show the path that the article took, from one computer to another, to reach your news server. (To understand the path, read it from right to left.)

Although we consider this information to be a single header line, it is often so long as to be broken onto more than one line on your screen. This is the case in Figure 10-1 where you can see the names of the computers that the article passed though on its way from Vienna, Austria, to Santa Barbara, California.

You can safely ignore the information in this header line.

References: You will see this header line only in followup articles. It will contain the identifier from the **Message-ID:** of the original article. The example in Figure 10-2 is a followup article and, as such, contains a **References:** header line.

This is another header line that you can ignore. It is used by newsreader programs to group related articles together. In this way, after you read an article, your newsreader can show one followup article after another. A connected series of articles is called a *thread*. As you read the news, you will often want to *follow* a thread, reading one followup article after another.

Reply-To: This header line has the same format as the **From:** line. If this line is present, replies that are mailed to the person who posted the article will be sent to this address. This header line is handy when you want private mail replies to go to a different address than the one from which the article was posted.

Sender: The purpose of this header line is to show the name of the computer from which the article was posted. A **Sender:** line will be generated automatically whenever the information in the **From:** line might be misleading: for example, if someone manually enters his own **From:** line rather than having the news posting program do it automatically.

For example, let's say that you are visiting a friend and using her account to post an article. Normally, the news posting program will put in a **From:** header line with her name and address. Instead, you delete this line and put in your own **From:** line, showing your name and address.

The news program will automatically generate a **Sender:** header line containing the name of the computer that posted your article. Sometimes you will also see the name of the system userid that sends out news articles.

```
Path: ucsbcsl!nipper.ucsb.edu!harley
From: harley@nipper.ucsb.edu (Harley Hahn)
Newsgroups: rec.humor
Subject: Re: The Secret of Life
Summary: Advice for understanding life.
Keywords: life, philosophy
Message-ID: <1993May16.060436.13827@nipper.ucsb.edu>
Date: 16 May 93 10:13:12 GMT
References: <1993May14.073130.15261@email.tuwien.ac.at>
Distribution: world
Lines: 22

In article <1993May14.073130.15261@email.tuwien.ac.at>
mich@music.tuwien.ac.at (Michael Schuster) writes:

> Here is some great advice I just found:
>
> "When you get serious about foolishness,
> you are getting into serious foolishness."
>
> By the way, does anyone know who first said it?

Yes, this advice was first offered by the
Canadian philosopher, Tim Rutledge.

Obligatory Joke...

Hahn's Maxim: If something is worth doing,
  it's worth doing to excess.

-
Harley Hahn
writer of Unix and Internet books
```

Figure 10-2. *A typical Usenet followup article*

Yes, as you suspect, this feature makes it a little more difficult for bored undergraduates to get away with forging news articles.

Subject: This line is, by far, the most important header line. It contains a short description of what the article is about. You create this description when you compose the article.

Why is this header line so important? Of course, it helps you understand what the article is about. But there is another reason. Some newsreader programs ask users to choose which articles they want to read based on the contents of **Subject:** header lines. This means that the contents of this header line are often the only thing a person sees before choosing whether or not to read the article.

Hint

When you post an article, it behooves you to take a moment to make your **Subject:** line interesting and accurate. Most people will not even look at an article unless the subject description takes their fancy.

Summary: This heading provides a brief, one-line summary of the article. This information is useful in followup articles. The **Summary:** header line is not used all that often, but you can put one in if you want.

Xref: When an article is cross-posted to more than one newsgroup, the **Xref:** header line (which stands for "cross-reference") will show which newsgroups contain the article. It will also show the local article numbers that identify the article in each group. You can ignore this line. It is only there to be read by your local newsreader program.

The Signature

We said earlier that a Usenet article has three parts: the header, the body, and the signature. We have already talked about the header (the technical information) and the body (the actual text of the article). Let's take a moment now to discuss the signature.

The signature is an optional addendum that shows information about the person who sent the article. For example, here is the signature from our sample article:

```
Michael Schuster                |"I love you for your beauty; love me
TU Vienna, Austria              | though I am ugly"
mich@music.tuwien.ac.at         | - Miguel Cervantes, Don Quixote
+43/1/12345
```

If you want to use a signature, you have to create it for yourself and store it in a file. Your news posting program will automatically append the signature to the end of each article you post before it is sent out. Here is how it works on Unix systems. (If you are using a non-Unix computer, you will have to read the documentation for your system.)

You use a text editor to create a file called **.signature** in your home directory. (The . character is part of the name.) Whenever you post an article, the news program looks to see if such a file exists. If so, the program appends the contents of the file to the end of your article.

You might ask, What if I don't know how to use a text editor program? Well, this is why, in Chapter 1, we suggested that you learn some Unix. At the very least, you should learn how to use a text editor so you can create things like a **.signature** file. You could, we suppose, ask someone else to help you create your **.signature** file. However, it wouldn't do you much good because if you can't use a text editor, you won't be able to compose an article in the first place.

(However, we don't want to be too dogmatic. You do not need to use a text editor if you just want to read articles, only if you want to post articles of your own, or if you want to reply to someone via a mail message.)

The standard information to put in a signature is the name of the person who posted the article and a mail address. You may also see the name of the organization to which the person belongs, a postal address, and perhaps even a telephone number. The signature in our example shows that the article was posted by Michael Schuster, at the Technical University of Vienna. We see his mail address and his phone number. To the right, we see an interesting quotation.

The wonderful thing about signatures is that people use them in highly imaginative ways. Michael Schuster, as you can see, includes a quotation. You will see many quotations, as well as witty sayings, drawings, jokes, and so on. Indeed, there are people who collect interesting signatures and post them, from time to time, as a unique bit of Usenet art (usually in the **rec.humor** newsgroup).



FUN TIP: To see some bizarre signatures—as well as comments about bizarre signatures—try the **alt.fan.warlord** *newsgroup.*

Hint

It is not considered imaginative to use your signature to publicize your religion.

Once you start reading a lot of Usenet articles, you will find that it is annoying to have to read long signatures. For this reason, it is a well-established convention that signatures should be no more than four lines long. Some news posting programs will actually enforce this rule by removing any extra lines. Although this may seem restrictive, you will be amazed at how much a creative person can do with only four lines.



FUN TIP: When it comes to creating a signature, anything goes. Aside from the four-line limit, there are no rules, so use your imagination.

Followup Articles

As we mentioned earlier, a followup article is one that is posted in response to a previous article. Creating a followup article is easy. Say that, as you are reading the news, you come across an article to which you would like to follow up. All you have to do is enter a simple command to your newsreader program.

The newsreader will make everything as easy as possible by creating a skeleton for the new article (a header and part of the body), and will start your text editor so you can compose your response. Once you finish, you quit the text editor. The newsreader will take over again and post the article for you automatically.

Take another look at the sample followup article in Figure 10-2. This article is a response to the one in Figure 10-1. Notice that the body of the new article starts with two lines that identify the previous article and the person who posted it. These lines are followed by the original article with each line marked with a > character.

Most of this was generated automatically by the newsreader program. All we had to do to create the followup was compose our response.

When you read an article, there are three ways to tell if it is a followup. First, the subject will be the same as the original article, but will have the characters **Re:** in front of it. For example, in Figure 10-2, we see:

```
Subject: Re: The Secret of Life
```

This is automatically put in by the newsreader when it creates the header of a followup article. However, no more than one **Re:** will be put in. Thus, a followup to a followup has only a single **Re:**.

The second thing to look for (if your newsreader is displaying the header for you) is a **References:** line. This contains the identifier of the original article. Although you can ignore this header line, it does serve as official notice that you are reading a followup article. If you are reading a followup to a followup, you may see more than one identifier in the **References:** line.

Finally, a followup article will often have all or part of the previous article embedded. As we mentioned, the lines from the previous article will be prefaced with a special character. Most of the time, this will be > (a greater-than sign) as in our example. However, you will sometimes see other characters.

In such cases, we say that the followup article is *quoting* those parts of the original article. In our example, the followup article quoted six lines from the original article.

Finally, if you see more than one > character, it means that an article is quoting something that has already been quoted. For example, you might see:

```
In article <1993May17.002440.8495@nipper.ucsb.edu>
harley@nipper.ucsb.edu (Harley Hahn) writes:

> In article <1993May14.222614.9344@unix1.tcd.ie>
> mepeirce@unix1.tcd.ie (Michael Peirce) writes:

>> Does anyone know the name of the computer that
>> contains the Usenet archives?

> Yes, it is pit-manager.mit.edu.

Actually, the name has been changed to rtfm.mit.edu.

-
Jonathan Kamens      Geer Zolot Associates          jik@GZA.COM
```

In this example, a reader named Michael Peirce had originally posted an article asking for the name of a particular computer. Another reader, Harley Hahn, sent a followup article answering the question. A third reader, Jonathan Kamens, then created a second followup article in which he corrected Hahn's answer.

Hint

It can be tiresome to read followup articles in which someone has quoted a long article and added only a small comment to the end. This is especially true when you are following a thread in which person after person has quoted the same long passages.

For this reason, when you create a followup article, it is considered good manners to delete as much superfluous material as possible. For example, if the original article was 100 lines long, but you are really responding to only 3 lines, you should delete all but those 3 lines.

Actually, some newsreader programs will refuse to post a followup article if the quoted material is longer than the reply.



FUN TIP: From time to time, you will see people propagate a long series of silly followups, called a cascade. *Sometime when you have nothing to do and you are in a capricious mood, check out the* **alt.cascade** *newsgroup (which is devoted to such creative endeavors).*

Usenet Acronyms

What would a computer system be without acronyms? Those marvelous little abbreviations that make you feel so important when you know what they mean and like a piece of cheese when you don't.

Not to fear. Table 10-2 contains all of the common Usenet acronyms. Not every possible acronym that you will ever encounter, but enough to get you through the night.

Study this table well. These acronyms are used in all manner of Internet discourse, not only within Usenet articles, but in mail messages and when using the **talk** command (see Chapter 19) to have a conversation with somebody.

(BTW, the MUD that we mention is a multi-user game. For more information, look under the "Games" category in the catalog.)

Usenet Slang

Aside from the acronyms and abbreviations in Table 10-2, you will also encounter a great number of slang terms and expressions. So, to finish this chapter, let's take a few minutes and look at the world of Usenet slang.

Acronym	Meaning
BRB	be right back
BTW	by the way
CU	see you (good-bye)
FAQ	frequently asked question
FAQL	frequently asked question list
FOAF	friend of a friend
FYI	for your information
IMHO	in my humble opinion
IMO	in my opinion
MOTAS	member of the appropriate sex
MOTOS	member of the opposite sex
MOTSS	member of the same sex
MUD	multiple user dimension
Ob-	[as a prefix] obligatory
Objoke	obligatory joke
OS	operating system
PD	public domain
SO	significant other (spouse, boy/girlfriend...)
ROTFL	rolling on the floor laughing
RTFM	read the [expletive]* manual
WRT	with respect to

Table 10-2. *Common Communication Acronyms*

**Expletive deleted by Editor as a public service.*

To begin, let us recall the new terms that we have already mentioned in this chapter and in Chapter 9.

Usenet is a vast system of discussion groups. We refer to Usenet as the *news*, or *netnews*, although there is little real news (in the sense of a newspaper). Thus, the discussion groups are often called *newsgroups*.

A computer that acts as the Usenet repository for an organization is called a *news server*. The person who manages the news server is the *news administrator*.

Each news server gets its information from another news server on a regular basis. This arrangement is called a *news feed*.

The items that you read in a Usenet discussion group are called *articles* or *postings*. A Usenet article is divided into three parts: the *header*, the *body*, and an optional *signature*. When you send an article to Usenet, we say that you *post* it. An article that is sent to more than one newsgroup is *cross-posted*. Usenet articles are kept for a predetermined time. When the time is up, the articles *expire* and are deleted from the news server.

An article that responds to another one is called a *followup article*. When a followup article contains parts of the original, we say that it *quotes* the original article. A series of followup articles is called a *thread*. When you read such articles, we say that you are *following* the thread.

There are over 5,000 different newsgroups, organized into *hierarchies*. *Mainstream hierarchies* are carried by all news servers. *Alternative hierarchies* are carried according to the wishes of the news administrator.

Moderated newsgroups are managed by a person called a *moderator*. The moderator decides which articles will be posted to the group. Such groups sometimes offer a collection of edited articles called a *digest*.

To read Usenet articles, you use a program called a *newsreader*. When you tell your newsreader that you want to start reading a particular group, you are *subscribing* to that group. Similarly, when you tell your newsreader that you do not want to read a group any longer, you are *unsubscribing*.

(We hope you have all of that, because there will be a short quiz at the end of the class.)

Now here are some new terms. We mentioned in Chapter 9 that anyone who knows how to do so can try to start a newsgroup in one of the alternative hierarchies. As you might imagine, such latitude often leads to newsgroups that don't really exist. Someone might send out a control message to start a new group, but for one reason or another, the group never gets started.

Nevertheless, the name of such a newsgroup may find its way into the master list on your news server. When your newsreader tries to find this group, it won't be there. Your newsreader will proudly inform you that it has found a *bogus* or empty newsgroup.

Another term you will encounter frequently is *flame*. This refers to a followup article (or a personal mail message) in which someone says something critical about someone else. The word "flame" is also used as a verb, as in "Scott posted an article without a joke to **rec.humor**, and he was flamed from all over the world."

All too frequently, a *flame war* will start, in which people send a large number of argumentative articles and mail messages excoriating one another. In such situations, it doesn't take long for other people to jump in and criticize the critics which, as you might imagine, only fans the flames. Although flame wars can have their interesting moments, most such postings are usually the written equivalent of each person sticking his or her tongue out at someone else. Eventually, the thread will die out, more from boredom than anything else.

As you are reading an article, you can tell your newsreader that you want to skip all the rest of the articles in the same thread. We say that you *kill* or *junk* the thread. Some newsreaders will let you specify that all articles with certain subject descriptions should be killed automatically. Your newsreader will save such requests in a special *kill file*.

Some newsgroups discuss works of art such as movies or novels in which it would be reasonable to discuss the plot. Of course, if you have not yet seen the movie or read the book, you may not want to find out the plot ahead of time. For this reason, it is customary that, when an article gives away a plot, the **Subject:** header line should say that the article contains a *spoiler*.

Well, that's it: about all the basic terminology that you need to know to understand Usenet. However, to complete your basic education, there are five more terms that you need to know. They are so important, in fact, that we will give them their own sections.

Foo, Bar, and Foobar

There are three marvelous words that you will see a lot: *foo*, *bar*, and *foobar*. These words are used as generic identifiers throughout Usenet as well as within the world of Unix.

The idea is that whenever you want to refer to something without a name, you can use "foo" (or less often, "foobar"). When you want to refer to two things without a name, you use "foo" and "bar". Nobody knows how this tradition got started, but it is used a lot.

For example, say you are reading an article in the **comp.unix.questions** newsgroup. (This is the group to which you can send Unix questions for experienced people to answer.) Someone is asking about editing files. You read:

```
Can anyone tell me how to move more than one file
at a time?  For example, say that I want to move two
files named foo and bar.  I tried using the command
'mv foo bar' but I got an error message...
```

Or you might read the following in the **rec.arts.movies**:

```
...Can anyone remember the musical in which Frank Sinatra
played an old-time Chicago gangster?  The name is something
like "Foobar and the Seven Hoods"...
```

So, where do these strange words come from? The word "foobar" derives from the World War II acronym FUBAR, which meant "fouled up beyond all recognition".

The word "foo", however, seems to have a more robust history. There is no doubt that foo owes much of its popularity to foobar. Nevertheless, it seems to have been used on its own even earlier. For example, in a 1938 cartoon, Daffy Duck holds up a sign that reads, "Silence is Foo" (which is absolutely correct). Some authorities speculate that foo might have roots in the Yiddish "feh" and the English "phoo".

RTFM

The term *RTFM* embodies the single, most important idea in the world of the Internet. RTFM means that before you ask anyone else for help, you should try to solve your problem by checking with a book or manual.

This idea is not mean-spirited; after all, the Internet in general (and Usenet in particular) is filled with people who will be glad to help you with anything. However, it is usually faster, and a whole lot more satisfying, if you can find an answer for yourself. Moreover, all new users tend to ask the same questions, and it is understandable that many people feel that, before you post an article asking for help, you should at least check all the standard references.

This raises the question: What are the standard references? Books, of course, like this one, as well as technical manuals. However, there is one standard reference for all Unix systems that is so important that you *must* know how to use it.

Every Unix system has an *online manual*. This is a computerized facility that will display help information about any Unix command. Using the online manual takes a while to learn, and we won't go into all the details here—for that you will have to read a Unix book. However, here is a brief summary.

To display information about a Unix command, you use the **man** (manual) command. Enter **man** followed by the name of the command you want to learn about. For example, to learn about the Unix **cp** (copy files) command, enter:

```
man cp
```

Unix will display a technical description of the specified command, one screenful at a time. As you are reading, you can move to the next screenful by pressing SPACE (on some systems) or the RETURN key (on other systems). (Try both and see which one works on your computer.) If you want to quit reading, press the **q** key.

Hint

To display help about the **man** command itself, enter:

```
man man
```

To display the official manual for a newsreader program, use **man** with the name of your newsreader. For example, to display the manual for the **nn** newsreader, enter:

```
man nn
```

Please don't forget: Before you post a question that has something to do with a Unix command, you should at least use **man** to check the online manual. Then, if you still don't have the answer you need, feel free to ask the Net. If you do ask a question that is answered in the manual, you may be gently (or not so gently) reminded to RTFM.

Now, where does this funny term come from?

Originally, RTFM was an acronym that stood for:

Read the [expletive]* manual

However, through the years, RTFM has taken on the more refined meaning of "try to find the answer yourself before you ask someone". (Actually, you will be surprised how often you *can* find the answer yourself.)

The term RTFM is also used as a verb, as in:

```
...Does anyone know how to save previously read articles to a file
using the foo newsreader?  I RTFM'ed, but I couldn't find the
answer anywhere.
```

**Expletive deleted by Editor as a public service.*

By now, it should make sense to you why the computer that contains the Usenet archives is named **rtfm.mit.edu**. This computer holds copies of all the frequently asked question (FAQ) lists. (And in Chapter 16, we will show you how to use Anonymous FTP to access such information.)

Smileys

The last Usenet term we will discuss in this chapter is both cute and useful. It is called a *smiley*, and it is used to indicate irony. Here is how it works:

When you talk with someone in person, you use your body language and voice inflections to project all kinds of non-verbal messages. For example, you can jokingly insult someone and get away with it (at least sometimes) as long as you get across the idea that you are just kidding.

With a Usenet article, this is not possible. Moreover, Usenet articles are read all over the world and not everybody will be able to appreciate the subtle nuances of another culture's humor. For example, it is altogether possible that someone outside the U.S. would not understand that the word "mother" is not always a term of endearment.

Enter the smiley.

A smiley is a small drawing, using only regular characters, that looks like a face. Here is the basic smiley:

```
:-)
```

To see the bright, smiling face, all you need to do is turn your head to the left.

We use smileys to make sure that someone does not accidentally misinterpret a potentially ambiguous remark. Putting a smiley at the end of a sentence is sort of like saying "just kidding".

For example, say that you are participating in a flame war in **rec.food.cooking**. You might post a followup article with the sentence:

```
How do you expect people to use your recipe for
Consomme aux Pommes d'Amour when you don't even
know how to cook groat cakes :-)
```

Hint

Smileys are used all over the place, not only within Usenet articles. For example, you can use them in private mail messages and when you are having a conversation by using the **talk** command (see Chapter 19).

Throughout the years, people have developed many different smileys. Indeed, there are vast collections of all kinds of strange smileys that are occasionally posted to **rec.humor**. Figure 10-3 shows a few such creations.

Smiley as Typed	Smiley Sideways	Meaning
:-)	:-)	smiling
:-D	:-D	laughing
;-)	;-)	winking
:-(	:-(	frowning
:-I	:-I	indifferent
:-#	:-#	smiley with braces
:-{)	:-{)	smiley with a mustache
{:-)	(:-}	smiley with a toupee
:-X	:-X	my lips are sealed
=:-)	(:-)=	punk rocker
=:-(	)-:=	real punk rockers don't smile

Figure 10-3. *A selection of smileys*

Now, to conclude this chapter, here is one last smiley:



```
%-)
```

Turn your head to the left. This is the face of a writer of Internet books who has stayed up all night to finish a chapter.

CHAPTER 11

Reading and Posting Usenet Articles

In Chapter 9, we talked about Usenet. We saw that there are over 5,000 different discussion groups with more than 2,500,000 participants. In Chapter 10, we took a look at what Usenet articles look like and what technical terms and conventions you can expect to see.

In this short chapter, we will explain what you will need to understand in order to read and post Usenet articles. We will start by showing you how your newsreader program is part of a client/server system. We will then talk about the various newsreaders. We will show you how to decide which newsreader will suit you best and how to go about learning it.

207

To conclude the chapter, we discuss a couple of technical topics that pertain to reading the news.

In the following chapters, we discuss the four most popular Unix newsreaders: **rn** (Chapter 12), **trn** (Chapter 13), **nn** (Chapter 14), and **tin** (Chapter 15). After you finish this chapter, turn to whichever newsreader chapter pertains to you, at which time you will (finally) start to read the news.

Understanding Your Newsreader as a Client

Like most Internet services, news reading makes use of a client/server arrangement (see Chapter 2). In this case, the news server resides on one computer in your network. Its job is to maintain a news feed to the outside world and to manage the large number of articles and newsgroups that are received every day.

To read the news, you use a client program, called a *newsreader*. The newsreader acts as your interface to Usenet, enabling you to choose the newsgroups to which you want to subscribe, select articles from these newsgroups, and display the articles one page at a time. As you read, you can preserve an article by saving it to a file, by mailing a copy to someone, or by printing it. You can also respond to the article, either by sending a private mail message to the author or by posting a followup article of your own.

As we explained in Chapter 10, news articles are transported throughout the Internet using a protocol called NNTP (Network News Transfer Protocol). As articles arrive at a news server, they are stored in a standard format. To read the news, all you need is a newsreader program that knows how to understand this format. There are many different newsreader programs, written to operate on a large variety of computers.

However, as we discussed in Chapter 1, many of the people who use the Internet use some type of Unix computer. In this chapter, we will discuss the general ideas involved in reading the news and posting your own articles. In the following chapters, we will discuss the details of the four most popular Unix-based newsreaders.

How Does a News Server Keep Track of All the Articles?

In order to keep track of the articles in each newsgroup, your news server assigns an identification number to each article. The numbers are assigned in the order in which the articles arrive.

The numbering starts with **1**. Whenever a new article comes in, it is assigned the next number in the sequence. For instance, say that a new article arrives for the **rec.humor** newsgroup and it happens that the last such article was number **1055**. So, the new article will be assigned the number **1056**. The next article in this newsgroup will be called **1057**, and so on. Some newsreaders show you this number each time you read an article. Other newsreaders don't bother.

Once an article expires, its number is removed from the system. Thus, on a particular day, the **rec.humor** newsgroup might have articles **1055** through **2110** available. Eventually, the numbers will reach some specific maximum value and the numbering will start again with **1**.

Each newsgroup has its own, separate set of numbers, so it is possible that more than one group may have articles with the same numbers. For example, both **rec.humor** and **comp.unix.questions** newsgroups may have articles numbered **1056**.

Your newsreader uses these numbers to keep track of which articles you have read for each newsgroup. To do so, it maintains a file named **.newsrc** in your home directory. This file contains the names of each newsgroup. For each name, there is information showing if you are subscribed to that newsgroup and which articles you have already read.

You don't have to worry about what's in your **.newsrc** file unless you want to. The first time you read the news, your newsreader will create a **.newsrc** file for you. From then on, your newsreader will maintain the file on your behalf, making changes whenever you read an article and whenever you subscribe or unsubscribe to a newsgroup. In addition, each time you start the newsreader, it will check if new newsgroups are available and update your **.newsrc** file appropriately.

Some people like to use a text editor to make changes to their **.newsrc** file directly. For this reason, there is a section at the end of this chapter that will show you the format of the **.newsrc** file along with some suggestions for making your own changes should you decide to do so.

Hint

In Unix, it is customary for files that contain initialization commands or special information to be given names that start with a . (period) character and end in **rc**. For example, we not only have **.newsrc** for newsreaders, but **.mailrc** for the mail program, **.cshrc** for the C-Shell, **.exrc** for the **ex** and **vi** text editors, **.bashrc** for the Bash shell, and so on.

The initial . means that these are hidden files: when you use the **ls** command to list your files, you do not see such names unless you specifically ask for them. When you pronounce names like this, the . character is called "dot". For example, you might hear someone ask, "What newsgroups are in your dot-news-r-c file?" For this reason, such files are sometimes called *dotfiles*.

The **rc** is an old convention that originally stood for "run commands", that is, commands to be run automatically each time a program started.

How to Choose a Newsreader

In this section, we will introduce you to the most popular Unix newsreaders—**rn**, **trn**, **nn**, and **tin**—and show you how to choose among them (if you have a choice).

Although these are the most widely used newsreaders, you may have other choices if you work within either X Window or Gnu Emacs. As we explained in Chapter 2, X Window provides a graphical user interface in which you can use more than one program at a time, each within its own window. There are a few newsreaders specifically designed to take advantage of X Window, the most notable being **xrn** and **xvnews**. Both of these newsreaders are modelled on **rn** (described below), but have extra features.

Another special working environment is Gnu Emacs. We won't go into the details here; suffice it to say that this system is built around a popular text editor named Emacs. There are two newsreader facilities designed specifically for Gnu Emacs, **gnus** and **gnews**.

For the most part, however, the most widely used newsreaders are the four that we mentioned above and those are the only ones that we will talk about in detail.

rn is the oldest of these newsreaders. It was designed by Larry Wall of the System Development Corporation in Santa Monica, California, and first released on April 8, 1983. (Wall is also the creator of Perl, a popular programming language.) Today, **rn** is maintained by Stan Barber, at the Baylor College of Medicine in Houston, Texas.

Wall developed **rn** to replace an older newsreader named **readnews**. He carefully designed **rn** to minimize the interaction between the program and the user in order to display information as quickly as possible. The result is a newsreader that displays as many articles as possible in the shortest amount of time.

trn was created by Wayne Davison of the Borland Corporation, in Scotts Valley, California. **trn** is a modern variation of **rn** that was first released on July 21, 1990. It has all the features and commands of **rn** along with some extra capabilities. Unlike **rn**, **trn** groups articles into threads and makes it easy to choose which ones you want to read. (As you may remember from Chapter 10, a thread is a connected series of articles: that is, the original article and all of the followups.)

nn was created by Kim Storm of Texas Instruments A/S in Denmark. It was first released in Denmark in 1984, in Europe in 1988, and worldwide in July 1989. Today, **nn** is maintained by Peter Wemm of Demaas Proprietary Limited in Australia.

Storm designed **nn** at a time when Usenet had grown large and there were far more articles than people could read. Even confining yourself to your favorite newsgroup often meant that you still had too many articles to wade through. Thus, Storm designed **nn** to make it easy to scan a large number of articles quickly. The trade-off is that this newsreader requires more user interaction than **rn** or **trn**.

tin is the newest of the four newsreaders. It was created by Iain Lea and first released on August 23, 1991. By that time, Usenet had become huge. Not only were there more articles than a single person could deal with, there were too many newsgroups. People who liked to follow a variety of discussions quickly found that the number of newsgroups was unmanageable. Lea designed **tin** to make it easy to scan a large list of newsgroups and choose which ones you want to investigate. Once you choose a group, **tin** presents you with a list of articles—somewhat like **nn**—and lets you choose which ones you want to read.

In other words, **rn** presents articles quickly and with a minimum of user interaction. **trn** works like **rn** except that it makes it easy to work with entire threads. **nn** requires more interaction, but lets you scan a large number of articles quickly. **tin** requires the most interaction, but it allows you to handle a large number of newsgroups comfortably and, within each group, to scan a large number of articles.

So, if you have a choice of newsreaders, here is how to decide: If you want to read most of the articles in a few newsgroups, use **rn** or better yet, **trn**, if it is available. If you want to pick and choose articles from a small number of newsgroups, use **nn**. And if you want to read a large number of newsgroups, use **tin**.

Hint

To choose which newsreader is best for you, consider how you like to read a newspaper.

If you read a newspaper thoroughly, one page after another, you are an **rn** person.

If you read each article as a whole, turning pages back and forth when necessary, you should use **trn**.

If you scan the newspaper quickly, reading only what strikes your fancy, choose **nn**.

And finally, if you like to read a whole pile of newspapers, zipping through one after the other, you will love **tin**.

As you can imagine, one of the eternal arguments among Usenet people is: Which newsreader is the best? As you can see, the best newsreader for you depends a great deal on how your eye and your mind assimilate information. When it comes to newsreaders, one size does not fit all.

If you would like to use a newsreader that is not on your system, show this discussion to your system manager, and ask him or her (nicely) to install the program. All of the newsreader programs that we mentioned in this section are available for free via Anonymous FTP (which is explained in Chapter 16).



FUN TIP: You might be wondering, which is our favorite newsreader? Well, we use all four newsreaders at various times. However, if we had to pick a single newsreader to take with us to a desert island, we would choose **tin**.

(Don't laugh about reading the Usenet news on a desert island. The Pagesat company of Palo Alto, California, offers an economical Usenet system that receives data via a small, personal satellite dish. For details, you can send a message to: **pagesat@pagesat.com**.*)*

What Do the Newsreader Names Mean?

Are you wondering what the names of the newsreaders mean?

Before **rn**, there was an older newsreader named **readnews** which was slow and displayed information line by line. Larry Wall designed **rn** to be a fast,

screen-oriented replacement for **readnews**. The name **rn**, then, indicates that this "read news" program is sleeker than the older one it replaced.

The name **trn** is straightforward: it stands for "threaded **rn**". (You will remember that **trn** is like **rn** except with the added capability of working with entire threads of articles.)

If you use **rn**, you will see that it shows you at least the first page of each article that you scan. Kim Storm designed **nn** so that the only articles you need to look at are the ones you choose to read. He felt that the less news you read, the better. Thus, the name **nn** stands for "No News (is good news)".

Finally, we have **tin**. This newsreader was developed by Iain Lea and based on an older newsreader named **tass**. Lea chose the name **tin** to stand for "Tass + Iain's Newsreader".

Learning How to Use a Newsreader

In Chapters 12, 13, 14, and 15, we will discuss the details of the four newsreaders that we mentioned earlier: **rn**, **trn**, **nn**, and **tin**. At that time, you will actually start reading the news. Of course, you only need to learn about the newsreader that you are using, so you can skip the other chapters.

Before you start learning about your particular newsreader, we want to spend a little time going over some general information that will help you no matter which newsreader you use.

You will find that all the newsreaders have many, many commands. Don't bother trying to learn them all straight off. And don't for a minute worry because there are so many commands that you don't understand right away. Begin by learning the basics: how to start the newsreader, how to select a newsgroup to read, and how to read articles. All the newsreaders are designed so that, most of the time, you need only a small handful of commands.

You will find that the better you are at controlling your newsreader, the more enjoyment you will get out of Usenet. Eventually, you will want to learn most of the commands, but the best way to do it is one command at a time.

At various times, you will have different commands available to you. For example, when you are selecting which newsgroup you want to read, you will have different commands at your disposal than when you are reading an article. In the newsreader chapters, we will provide you with summaries of the various commands. You can use these summaries for reference as you read the news.

In addition, each newsreader has a special help command that you can use at any time. With **rn**, **trn**, and **tin**, the command is **h**. With **nn**, the command is **?**. Whenever you want, all you have to do is press this key to display a summary of whatever commands are available.

Hint

The way to learn to use a newsreader well is the same way that you get to Carnegie Hall: practice, practice, practice. Fortunately, reading the news is a lot more fun (and a lot quieter) than learning to play the violin.

In the following chapters, we will introduce you to each newsreader in turn. However, we don't have the room (or the stamina) to discuss every detail. Once you start reading the news, you may want to check the official documentation for your particular newsreader. To do so, you can use the online Unix manual. Here is how it works.

All Unix systems come with a built-in reference manual. This manual contains a file of information about each command. To display this information, you use the **man** command. Simply enter **man**, followed by the name of the command you want to learn about.

For example, to learn about the **ls** command (which you can use to list the names of your files), enter:

```
man ls
```

To learn about using the **man** command itself, enter:

```
man man
```

Thus, to read the official reference manual for your newsreader, all you have to do is enter **man** followed by the name of the newsreader program. The choices are:

```
man rn
man trn
man nn
man tin
```

Bear in mind that what you will see is reference information, not introductory tutorials.

When you enter such a command, **man** uses a *paging program* to display the output. The job of a paging program is to display information, one screenful at a time. The custom is to speak of each screenful of information as a *page*. Thus,

we can say that the job of a paging program is to display information, one page at a time. After you read what is on the screen, you press a key to display the next page.

Like much of Unix, paging programs come with many options and a large number of commands. We won't go into the details here. Suffice it to say that the **man** command on your system will be using one of three common paging programs: **more**, **pg**, or **less**.

If you are using **more** or **less**, you can display the next page by pressing SPACE. If you are using **pg**, you press the RETURN key instead. In all three programs, you can press **h** for help and **q** to quit. If you are not sure which pager you are using, try pressing both SPACE and RETURN and see what happens.

Posting Your Own Articles

As a Usenet reader, you will, from time to time, desire to post your own articles. You may want to request some information, respond to a previous article, offer a new idea for discussion, or whatever. Each newsreader has its own commands to make posting articles simple and easy. We will go over the details in the appropriate chapters.

In this section, we would like to take a moment to discuss a few of the general considerations that apply to posting articles.

First, you must understand that it is not easy to post an article of any type unless you know how to use a text editor program. After all, your article does not even exist until you create it. Moreover, if you are responding to someone else's article, you will probably want to edit what they have said and intersperse your own comments. All of this requires a text editor.

The two most common Unix text editors are **vi** and **emacs**. Life being what it is, both of these editors are complex and take a while to learn. If you are using a PC or a Macintosh that attaches to a remote host by emulating a terminal (see Chapter 3), you may be able to compose an article on your own computer and send it to the host. Still, it is not the best way to compose articles.

The real truth is that, if you are using a Unix system, you must learn how to use a Unix text editor. You can get by without one, but your life on the Internet will not be as satisfying as it could be. (For example, it will not be easy to post your own articles to Usenet.)

If you are not sure which text editor to learn, use **vi**. It is a universal standard, available on every Unix system. Don't be intimidated by people who may tell you that **vi** is difficult to learn. All it takes is a little practice with a good book.

Hint

If you are using a Unix system, it is well worth your time to learn how to use the **vi** text editor.



FUN TIP: The best book to use to teach yourself **vi***—and Unix in general—is Harley Hahn's* A Student's Guide to Unix *(McGraw-Hill 1993).*

The next point to appreciate is that the best way to respond to an article may not be to send a followup. All the newsreaders make it easy to send a personal mail message to the author of an article. Ask yourself if your response will be interesting to everyone or just to the author. Unless what you plan to say is of general interest, send a response by mail.

Hint

In many cases, you are better off responding to an article by sending personal mail to the author than by posting a followup that will go all over the world.

When you do send a followup, your newsreader will make it easy for you to quote (include) all or part of the previous article within your followup. It is considered good manners to edit out all the parts of the original article that are not relevant to your followup. (See Chapter 10 for the details.) It is irritating to read a followup in which the writer has quoted a long article and merely added a few new lines to the end.

Hint

When you create a followup, make it easy for other people to read by deleting all the superfluous parts of the original article.

As a rule of thumb, the quoted part should not exceed your own comments. (Some newsreader programs enforce this rule.)

When you first start to post articles, you will probably want to send out a practice article or two. You should know that there are special newsgroups just for such tests. In Table 11-1, we have listed the practice newsgroups in the mainstream and alternate hierarchies. Notice that all the names end in **.test**. At any time, you can send a posting to one of these groups just for practice.

If your location has a local hierarchy (see Chapter 9), you may find a local test newsgroup. If so, this is the best place to send a practice posting. Otherwise, you should use **misc.test** (to practice posting to a mainstream newsgroup) or **alt.test** (to send to an alternate group).



REMEMBER: When you want to practice posting an article, use only the **.test** *newsgroups. It is considered bad manners to send a test posting to a non-test newsgroup.*

We mentioned in Chapter 10 that you can append a signature to the end of your postings by creating a file named **.signature** in your home directory. When you are testing out a new signature, you can send a practice posting to one of the newsgroups in Table 11-1 to see if the new signature looks okay.



FUN TIP: There are a number of computers around the Internet that run a special program that looks for and responds to postings in the **.test** *newsgroups. When you post such an article, you may get an automated response, by mail, telling you that your practice article has arrived at such-and-such computer.*

Newsgroup Name
alt.test
bit.listserv.test
biz.test
gnu.gnusenet.test
u3b.test
vmsnet.test
misc.test

Table 11-1. *Newsgroups for Practice Postings*

The last point that we would like to mention is that Usenet has a number of conventions and traditions that you should understand before you post an article. We discussed these ideas in Chapter 10. In particular, before you send a question to a newsgroup, make sure that you have read the FAQ (frequently asked question) list if that group has one.

If you are a new user, there is a group set up especially for you called **news.newusers.questions**. There are a number of regular postings sent to this newsgroup that explain all about Usenet. This is also the place for you to send a question that pertains to using Usenet. If you have a question about using a newsreader, you can send the question to the **news.software.readers** group.

The Format of the .newsrc File

As we explained earlier in the chapter, your newsreader maintains a file named **.newsrc** to keep track of which newsgroups you subscribe to and which articles you have read.

In this section, we will describe the format of this file in case you know how to use a text editor and you want to modify your own **.newsrc** file. If you don't care about such matters, you can skip this section with impunity.

There are two reasons why you would want to edit your own **.newsrc** file. First, you will probably want to change the order of the newsgroups. When you start your newsreader, it will look at the newsgroups in the order it finds them. By editing your **.newsrc** file, you can put the more interesting groups first. You can use your newsreader to make such changes, but it is usually a lot easier to edit the **.newsrc** file and do it yourself.

The second reason to edit your **.newsrc** file is to make changes to your newsgroup subscriptions—that is, to specify exactly which newsgroups you want to read. With some newsreaders, you will automatically start off by being subscribed to every newsgroup. Normally, you would want to unsubscribe to all but a few of these groups. Again, this is something that you can do with your newsreader, but if you want to make a lot of changes, editing the file by hand is faster.

Hint

Before you edit your **.newsrc** file, make a backup copy. That way, if you ruin the original, you can restore it from the copy. This is good advice for editing any important file.

The **.newsrc** file is kept in your home directory and maintained automatically by your newsreader. All of the newsreader programs use this same file and there is only one format.

There is one line for each newsgroup. This line contains:

- the name of the newsgroup
- a colon (:) or an exclamation mark (!)
- a space
- a list of numbers

If there is a colon after the newsgroup name, it means that you are currently subscribed to that group. An exclamation mark means that you are unsubscribed. The list of numbers indicates which articles you have already read. (You will remember that each article is given an identification number as it arrives at the news server.) The list of numbers contains single numbers or ranges of numbers, separated by commas.

Here are a few lines from a typical **.newsrc** file:

```
alt.fan.wodehouse: 1-819
rec.humor.funny: 1-8192
rec.humor: 1-41234,41236,41239
comp.unix.questions! 1-6571
misc.books.technical!
```

You can see that the first three newsgroups are subscribed and the last two are unsubscribed. Within each newsgroup, you can also see which articles have been read. For example, in the **rec.humor** group, articles 1 through 41,234, plus articles 41,236 and 41,239, have been read.

If there are no numbers on a line, it means that no articles have ever been read in that newsgroup. This is the case with **misc.books.technical**.

Finally, notice that although **comp.unix.questions** is unsubscribed, it was subscribed at one time. You can tell because articles 1 through 6,571 have been read.

Hint

Edit your **.newsrc** file to put all your favorite newsgroups first.

Rot-13

As we explained in Chapter 9, Usenet has no central authority and no enforceable rules. The content of the newsgroups is decided by the participants, and decorum is maintained by voluntary cooperation.

One of the problems that arises, from time to time, is what to do about offensive articles. This occurs most often in the humor newsgroup **rec.humor** (and to a lesser extent in **rec.humor.funny**), where you will see jokes that some people would consider to be offensive.

One side of the argument is that Usenet should be free and unfettered, and that anyone has the right to post anything he or she wants. The other side is that it is fundamentally wrong to post articles that are patently offensive. Like all such arguments, this one has never been resolved and, no doubt, never will be.

The solution is to encode a potentially offensive article so that, unless you use a special command, the text looks like gibberish. The person who posts the article describes in the subject line who the article may offend. For example, you might see an article that looks like gibberish with a subject line of **Offensive to Americans**, or **Offensive to Italians**, or whatever.

You can decide whether or not you want to read the article. If you do, you type the special key—the exact command depends on which newsreader you are using—and the article will be decoded for you. Since someone would have to use a special command to decode the article, no one can complain about being offended accidentally.

The scheme that is used to encode an article is called *rot-13*. Rot-13 uses a simple replacement cipher in which each letter is rotated 13 positions within the alphabet. The actual substitutions are shown in Figure 11-1. For reference, Table 11-2 shows the commands to use in order to read a rot-13 encoded article. Note that rot-13 only encodes letters of the alphabet, not numbers or punctuation.

The rot-13 convention is used to encode more than offensive jokes. It is used for any article that you would not want someone to read by accident, for example, articles that give away the ending to a book or movie, answers to puzzles, answers to riddles, and so on.

On a Unix system, you can encode or decode text by using the **tr** (translate) command. We won't go into the details, but if you want to experiment, here is the actual command that will encode and decode text using the rot-13 scheme. (In this command, *infile* is the name of your input file, and *outfile* is the name of your output file.)

```
tr '[a-m][n-z][A-M][N-Z]' '[n-z][a-m][N-Z][A-M]' <infile >outfile
```

a → n	n → a	A → N	N → A
b → o	o → b	B → O	O → B
c → p	p → c	C → P	P → C
d → q	q → d	D → Q	Q → D
e → r	r → e	E → R	R → E
f → s	s → f	F → S	S → F
g → t	t → g	G → T	T → G
h → u	u → h	H → U	U → H
i → v	v → i	I → V	V → I
j → w	w → j	J → W	W → J
k → x	x → k	K → X	X → K
l → y	y → l	L → Y	Y → L
m → z	z → m	M → Z	Z → M

Figure 11-1. *The substitutions used to implement the rot-13 code*

If you want to practice decoding, use this **tr** command on the following rot-13 encoded text:

```
"Jura lbh trg frevbhf nobhg ohyyfuvg,
lbh ner trggvat vagb frevbhf ohyyfuvg."
-- Pnanqvna cuvybfbcure, Gvz Ehgyrqtr
```

If you want to post a rot-13 encoded article, there is no built-in way to do it with most newsreader programs. You will have to create the article with a text editor, and then use a command like **tr** to encode the article before you post it.

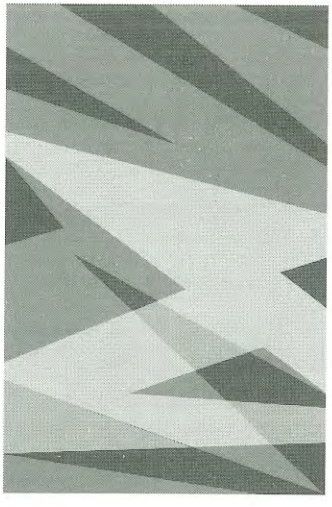
Newsreader	Command
rn	CTRL-X
trn	CTRL-X
nn	D
tin	d

Table 11-2. *Commands to Read a Rot-13 Encoded Article*



FUN TIP: Unlike **rec.humor** *and* **rec.humor.funny**, *the jokes in* **alt.tasteless.jokes** *are never encoded using rot-13. This is because, by definition, all the jokes in this newsgroup are supposed to be tasteless. Indeed, if you send a joke in good taste to* **alt.tasteless.jokes**, *you will be told that such jokes should be sent to* **rec.humor**.

If you complain about a joke in **alt.tasteless.jokes**, *no one will pay any attention. There is no rot-13 protection, so, if you are easily offended, don't even look at this newsgroup. (By now, you probably can't wait to read it, so move on to the next chapter.)*



CHAPTER 12

The rn Newsreader

We have talked a lot so far about Usenet, the worldwide system of over 5,000 discussion groups. In this chapter, we will explain how to read the news and post your own articles using the **rn** newsreader. Before you read this chapter, you should at least have skimmed the previous Usenet chapters. In particular, you should understand what Usenet is and how it works (Chapter 9), the conventions and terminology that we use (Chapter 10), and what is involved in using a newsreader (Chapter 11).

Overview of the rn Newsreader

The **rn** program was designed to allow you to read articles quickly and easily with a minimum amount of intervention on your part. The program has many options and commands, and offers far more functionality than you will ever need. The best strategy is to learn the basics and practice for a while. Once you have some experience, you can look at the command summaries at the end of this chapter and experiment with whatever looks interesting.

To introduce you to **rn**, we will start with an overview of how it works. In the following sections, we will go into the details.

You start the program by entering the **rn** command. **rn** checks with your **.newsrc** file (see Chapter 11) and sees if there are any new newsgroups that are not in your file. If so, **rn** will present them to you, one at a time, and ask if you want to subscribe.

Once this is done, you are now ready to start reading articles. **rn** shows you the name of a newsgroup. You can accept this newsgroup or choose a different one. Once you choose a newsgroup, **rn** displays the first page of the next unread article in the group. You can now decide whether or not you want to read the entire article. If so, **rn** will display the article, one page at a time. If not, **rn** will show you the first page of the next unread article.

When you finish an article, **rn** displays the next one automatically. When the last article in a newsgroup is read, **rn** moves to another group. At any time, you can skip to a new article or to a different group.

In other words, at all times you are in one of three situations: selecting a newsgroup, examining an article, or paging through an article. (If this seems a bit confusing, don't worry. It will all make sense when we explain the details and when you try **rn** for yourself.)

At each point, there are a large number of commands available. As a reference, we have summarized all of the important **rn** commands in three lists at the end of the chapter.

Figure 12-1 shows the commands you can use when selecting a newsgroup. Figure 12-2 shows the commands for when you are looking at the first page of an article and deciding whether or not you want to read the whole thing. And Figure 12-3 shows the commands that are available as you are paging through an article. You may want to take a minute now to glance at these summaries.

Within the summaries, we have followed the common Unix convention of indicating CTRL keys by using the ^ character. For example, when you see **^N**, it means CTRL-N. (That is, hold down the CTRL key and press **N**.)

In the following sections, we will discuss the details of using **rn**. As you read, and as you practice, there are four things that we would like you to remember.

First, at any time, you can press **h** (help) to display a summary of whatever commands are available.

Second, to stop a command that is in progress, press CTRL-C.

Third, for all single-letter commands, you do not have to press the RETURN or ENTER key after you type the command. All you have to do is press a single key and **rn** will react immediately. For example, to display help information, you press the **h** key. (When a Unix program reads input in this way, we say that it is operating in *cbreak mode*.)

Finally, whenever you have to choose a command, **rn** will present you with a short list of possibilities. For example, you might see:

```
[ynq]
```

In this example, you can press **y** (for yes), **n** (for no), or **q** (for quit). If you are not sure what the choices mean, or if you would like to see what other choices you have that are not listed (and there are always some), press **h** (for help).

Whenever you are called upon to type a command, you can press SPACE to select the first choice by default. In the example above, pressing SPACE would be the same as pressing **y**. The defaults are set up so that the first choice is usually the one you want.

Hint

You can read news all day just by pressing SPACE.

Starting and Stopping

If you have never read the news before, you will not have a **.newsrc** file (see Chapter 11). Before you use **rn** for the first time, you can enter the command:

```
newsetup
```

This will create a brand new **.newsrc** file for you containing every newsgroup that is carried by your news server. If you start **rn** for the first time without first using **newsetup**, **rn** will run **newsetup** for you automatically.

When your **.newsrc** file is created, every newsgroup will be marked as being subscribed. You will probably want to unsubscribe to most of the groups. There are two ways to do this. First, you can do it within **rn** by using the **u** command, although this will probably take a long time. The more practical method is to use

a text editor, such as **vi** or Emacs, and edit the **.newsrc** file for yourself (see Chapter 11).

To start **rn**, you enter the **rn** command. There are several forms of this command and a large number of options that you can use. In this section, we will cover the most important of these choices.

First, you can enter the command by itself:

```
rn
```

This starts **rn** in a straightforward manner.

To stop **rn**, press the **q** (quit) key. If you are reading or selecting an article, you may have to press **q** more than once to work your way back out to the top level. (This will make sense when you do it.)

Alternatively, if you are at the top level (selecting a newsgroup) you can quit **rn** by pressing the **x** (exit) key. This not only quits **rn**, but restores your **.newsrc** file to what it was when you started. This abandons all the changes you may have made, such as subscribing or unsubscribing to newsgroups, reading articles, and so on.

Most of the time, there is a better way to start **rn**. You will remember we said that the first thing **rn** does is check to see if there are any new newsgroups that are not listed in your **.newsrc** file. Although this is a good idea, it can be time consuming, perhaps taking several minutes each time.

Most people prefer to have **rn** skip this check. You can do so by using the command:

```
rn -q
```

The **-q** is called an **option** or a **switch**. In this case, the **-q** option tells **rn** to perform a "quick" start. Using **-q** will start **rn** faster, but will not, of course, pick up any new newsgroups. For this reason, we recommend starting **rn**, say, once every week or two, without the **-q** option, just to see what is new.

There may be times when you are interested in reading only a single newsgroup. In that case, you can specify that group directly. For example, say that you only want to read the **rec.humor** newsgroup. You can enter:

```
rn -q rec.humor
```

There is one special way to use the **rn** command that we want you to know about and that is by using the **-c** (check) option:

```
rn -c
```

This will not allow you to read news. Rather, **rn** will simply report to you if there is any unread news in the groups to which you are subscribed and then stop. You can use this command whenever you want to check quickly if there is any news.

Hint

You can place a **rn -c** command in your initialization file so that each time you log in you will be told if there is news that you have not yet read. The name of your initialization file depends on which shell you are using. For the C-Shell, the file is **.login**; for the Bourne or Korn shell, it is **.profile**; and for the Bash shell, it is **.bash_profile**.

Selecting a Newsgroup

When you start **rn**, it begins by showing you the first five subscribed newsgroups that have unread articles. The groups will be in the order that **rn** finds them in your **.newsrc** file. Here is a typical example:

```
Unread news in rec.humor.funny                    11 articles
Unread news in rec.humor                          83 articles
Unread news in alt.tasteless.jokes                67 articles
Unread news in comp.unix.questions               121 articles
Unread news in biz.clarinet.sample                 3 articles
etc.
```

Once **rn** has issued this initial report, it looks at the master list of active newsgroups on the news server to see if there are any new groups that are not in your **.newsrc** file (unless you have started **rn** with the **-q** option). If **rn** finds any such groups, it lists them one by one and asks you if you want to subscribe. For example:

```
Newsgroup alt.death-of-superman not in .newsrc--subscribe? [ynYN]
```

You must now make a choice. Press **y** (or SPACE) to subscribe or **n** to unsubscribe. If you want to subscribe or unsubscribe to all new groups without looking at all the names, press **Y** or **N** respectively. If you do press **y**, you will be asked to specify where you want the new group to be placed in your **.newsrc** file. Press **h** (help) to see your choices and pick the one you want.

Once **rn** has finished these preliminaries, you are ready to select the newsgroup that you want to read. **rn** will present the first newsgroup with unread articles and ask you if you want to read it. For example:



```
******** 11 unread articles in rec.humor.funny--read now? [ynq]
```

At this point, you have a large number of commands that you can use. We have summarized the most important commands for you in Figure 12-1.

The simplest thing to do is to press SPACE (or **y**) and read the newsgroup. Otherwise, you can press **n** to go on to the next group with unread articles, or **q** to quit **rn**.

Selecting and Reading Articles

As soon as you have selected a newsgroup to read, **rn** will look for the first unread article and display the first page.

As you are reading an article, there are many different commands that you can use. These are summarized in Figures 12-2 and 12-3. You can use the commands in Figure 12-2 whenever you are reading an article. Figure 12-3 shows extra commands that you can use as you page through a long article.

In this section, we will discuss the most important commands. If you want to learn how to learn **rn** well, take a look at the summaries from time to time and make a point of trying out new commands.

Whenever an article is finished, the bottom line of the screen will have a message like the following:



```
End of article 8278 (of 8284)--what next? [npq]
```

If you want to read the next article, simply press SPACE or the **n** key. If you want to jump to the next article in the thread—in other words, if you want to follow the discussion—press **^N** (CTRL-N). If you want to kill (junk) all the articles in this thread, press **k**. And, if you want to quit reading this newsgroup, press **q**.

If you press ^N, this command will become the default. The next time you finish an article you will see a message like:

```
End of article 8279 (of 8284)--what next? [^Nnpq]
```

The beauty of this is that it allows you to read through an entire thread, article by article, just by pressing SPACE.

In fact, once you have read all the articles in a newsgroup, pressing SPACE will go to the next group. If you ever get to the point where you have read the last article in the last newsgroup, pressing SPACE a couple more times will quit **rn**. Thus, as we said earlier, it is possible to read the news all day long just by pressing this one key.

When an article is long, **rn** will display the first page. On the bottom line, you will see a message like this:

```
--MORE--(45%)
```

This tells you that there is more to come and shows you how far you are into the article. (In this case, you are 45% of the way through.) At this point, you can press SPACE to read the next page, or use any of the commands that we discussed above.

Before we leave this section, there are three more points that we would like to mention. As you are reading, you can display a list of all the articles remaining in the newsgroup by pressing the = (equal sign) key. **rn** will show you a list of article numbers and subjects. For example:

```
8280 New light bulb joke
8281 Riddle about Unix
8282 Silly computer names
8283 Joke about acquisition editors
8284 Horrifyingly Funny...
```

If you would like to jump directly to one of these articles, just type the number and press RETURN (or ENTER). For example, to read article number **8283**, enter: **8283**.

If you want to save a copy of an article in a file, use either the **s** (save) or **w** (write) command followed by the name of a file. The only difference is that the **w** command will not save the header.

For example, to save the current article, without the header, to a file named **harley**, enter:

```
w harley
```

If this is a new file, you will be told that it does not already exist (in which case **rn** will be glad to create it for you) and you will be asked the following:

```
use mailbox format? [ynq]
```

In other words, would you like to save the file in a format that you can later read with your mail program? If you are not sure what this means, don't worry about it: just press **n** (no).

The final point we want to mention is that, from time to time, you may see an article that is encoded in a special format known as rot-13 (see Chapter 11). This format is used to encode articles that are potentially offensive. When you see a rot-13 article, look at the subject and decide if you want to read the article. If so, press **^X** (CTRL-X) and **rn** will do the decoding for you.

Replying to an Article

When you find an article to which you want to reply, ask yourself if it is better to send a personal mail message to the author of the article or a followup article to the newsgroup.

To mail a message to the author, you can use the **r** or **R** command. The difference is that the **R** command places the original article in your message so that you can include parts of it in your reply.

Once you type one of these commands, **rn** will ask you if you have a prepared file that you want to include in your message:

```
Prepared file to include [none]:
```

If not, simply press RETURN. If you have prepared a file in advance, type its name and press RETURN.

rn will now set up the message for you and start the text editor. You can now edit and compose whatever you want. When you are finished composing the message, quit the text editor in the usual manner. **rn** will regain control and ask you how to proceed:

```
Send, abort, edit, or list?
```

If you want to send the message, enter **s**. If you want to forget about the message completely, enter **a**. If you want to re-edit the message, enter **e**. Or if you want to display the message, enter **l**.

Posting an Article

If you would like to reply to an article by posting a followup, you can use the **f** or **F** command. The difference is that the **F** command places the original article in your followup article so that you can include parts of the original article in your reply.

If you use the **F** command, **rn** assumes that you want to send a followup article on the same topic. However, if you use **f**, **rn** will ask you if you want to specify a different topic:

```
Are you starting an unrelated topic? [yn]
```

If you press **y**, you will be given a chance to specify a new topic. Otherwise, press **n**.

You will now see a standard warning, designed to remind one and all that posting articles costs somebody money:

```
This program posts news to thousands of machines throughout
the
entire civilized world.  Your message will cost the net
hundreds
if not thousands of dollars to send everywhere. Please be sure
you know what you are doing.

Are you absolutely sure that you want to do this? [ny]
```

Enter **y**. (As a reader of this book, you *always* know what you are doing.)

From here on, the procedure is just as we explained it in the previous section. You will be asked if you have a prepared file to include. **rn** will then start the text editor for you. When you are finished, **rn** will ask you to confirm that you want to send the message.

That's all there is to it.

There may be times when you are not reading the news and you would like to post an article. You do not have to start **rn** at all. You can invoke the news

posting program directly by using the **Pnews** command. (Yes, the first letter is an uppercase P. This is to make it difficult for someone to post an article by accident.)

You can enter this command by itself or with the name of a newsgroup. For example:

```
Pnews
Pnews misc.test
```

If you do not specify a newsgroup, **Pnews** will ask you for one.

The **Pnews** program works just like we described above. (In fact, when you use an **f** or **F** command, **rn** calls on **Pnews** to do the work.) The only difference is that, in this case, **Pnews** begins by asking you what distribution you want. The distribution (described in Chapter 10) tells the news servers that handle your article where you want the article sent. If you are not sure what distribution to use, choose the smallest one that looks adequate.

Commands to Use While Selecting a Newsgroup

Basic Commands

h	display help information
q	quit rn
x	quit rn, abandon all changes to your .newsrc file
v	show what version of rn you are using

Start Reading Articles

SPACE	perform the default command
y	read the current newsgroup
=	same as y, but start by showing list of subjects

Controlling Newsgroups

n	go to next newsgroup with unread articles
p	go to previous newsgroup with unread articles
^	go to first newsgroup with unread articles
$	go to the end of the list of newsgroups
g *newsgroup*	go to specified newsgroup
/ *pattern*	scan forward for newsgroup name containing pattern
? *pattern*	scan backward for newsgroup name containing pattern
/	scan forward for previous pattern
?	scan backward for previous pattern
u	unsubscribe to the current newsgroup
l *pattern*	list unsubscribed newsgroups containing pattern
L	list current state of newsgroups in .newsrc file
c	mark all articles in newsgroup as read [catch up]

Entering a Unix Command

! *command*	execute the specified Unix command
!	pause rn and start a shell

Figure 12-1. *Commands to use while selecting a newsgroup*

Commands to Use While Selecting an Article

Basic Commands

h	display help information
q	quit this newsgroup
SPACE	perform the default command
n	go to next unread article
^N	go to next unread article on same subject
k	mark as read, all articles with same subject [kill]
u	unsubscribe to the newsgroup
c	mark all articles in newsgroup as read [catch up]

Re-displaying the Current Article

^R	re-display the current article
v	re-display the current article with header [verbose]
^X	decode the current article using rot-13
X	decode the current page using rot-13
b	go back one page
^L	re-display the current page

Moving to Another Article

-	re-display the last article that was displayed
N	forward: to next article, whether read or unread
$	forward: to end of the last article
p	backward: to previous article
^P	backward: to previous article with same subject
P	backward: to next article, read or unread
^	backward: to first unread article

Figure 12-2. *Commands to use while selecting an article* (continued next page)

Using Article Numbers

=	display list of all the unread articles
number	go to article with the specified number
#	display the number of the last article

Searching for an Article

/ *pattern*	forward: look for subject containing pattern
/	forward: repeat search using previous pattern
? *pattern*	backward: look for subject containing pattern
?	backward: repeat search using previous pattern

Responding to an Article

r	send a mail message to author of article [reply]
R	same as r, include original message
f	start Pnews program to create a followup article
F	same as f, include original message

Saving an Article

s *file*	save entire article to specified file
w *file*	same as s, but do not save the header [write]

Entering a Shell Command

! *command*	execute the specified shell command
!	pause rn and start a shell

Figure 12-2. *Commands to use while selecting an article* (continued)

Commands to Use While Paging Through an Article

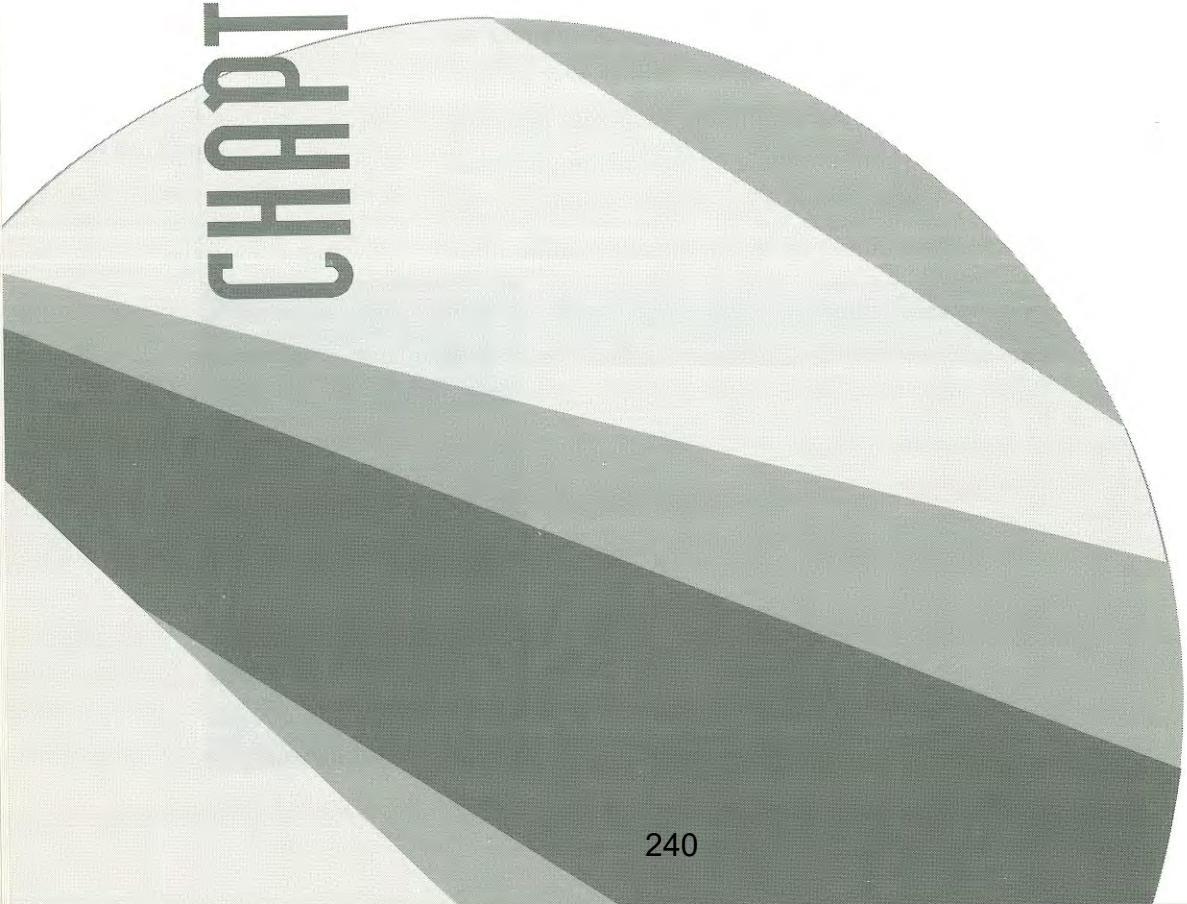
As you are paging through an article, you can use all the commands in Figure 12-2 as well as the extra commands in this list.

Paging Through an Article	
h	display help information
q	go to the end of the article
SPACE	display the next page
RETURN	display one more line
d	display half a page more
b	go back one page
x	decode the next page using rot-13
j	mark article as read and go to end [junk]
g *pattern*	forward: search for specified pattern
G	forward: repeat search using previous pattern
^G	while reading a digest: go to next subject
TAB	with followups: skip text from previous articles

Figure 12-3. *Commands to use while paging through an article*



CHAPTER 13



The trn Newsreader

In this chapter, we will explain how to use the **trn** newsreader to read Usenet articles. As we explained in Chapter 11, **trn** is actually an extension of **rn**. In this chapter, we discuss only those features of **trn** that are different from **rn**. Before you read this chapter, you should already know how to use **rn**. Even if you are an experienced **rn** user, you might want to take a moment to skim through Chapter 12.

241

Overview of the trn Newsreader

trn does everything that **rn** does, plus more. More precisely, while you are reading the news with **trn**, you can use all of the regular **rn** options and commands as well as the extra **trn** features.

The best way to learn how to use **trn** is to first learn how to use all the **rn** features. So, if you are a beginner, read Chapter 12 and practice with **rn** until you become comfortable. If you are already an experienced **rn** user, read on.

What **trn** adds to **rn** is the ability to work with threads as separate entities. When you start **trn**, it looks pretty much like **rn**. The only difference is that, when you choose a newsgroup, you can go into thread selection mode. **trn** will organize all the articles in threads and you can choose which threads you wish to read. After you make your selections, **trn** will show you one thread at a time. Within each thread, you will see each article in order of arrival time.

For the rest of this chapter, we will discuss those features of **trn** that are different than **rn**. We assume that you are acquainted with the basic **rn** commands. If not, you can refer to Chapter 12. As we mentioned, most of **trn** is identical to **rn**. In particular, paging through an article, saving an article, and posting your own articles is all the same in both newsreaders.

For reference, Figures 13-3, 13-4, 13-5, and 13-6 at the end of the chapter contain summaries of all the basic **trn** commands. Within these summaries, we have followed the common Unix convention of indicating CTRL keys by using the ^ character. For example, when you see **^N**, it means CTRL-N. (That is, hold down the CTRL key and press N.)

Starting and Stopping

To start **trn**, you use the **trn** command. You can use all the **rn** options as well as some extra ones. In particular, you can use the **-q** (quick start) option to suppress the initial check for new newsgroups. Thus, the two most common ways to start **trn** are:

```
trn -q
trn
```

As with **rn**, you can specify a specific newsgroup to read:

```
trn -q rec.humor
trn rec.humor
```

Again, like **rn**, you can also use the **-c** option to check for unread news:

```
trn -c
```

This is a good command to put in your **.login**, **.profile**, or **.bash_profile** initialization file.

Like all newsreaders, **trn** has a large number of options. The only ones we will talk about here are **-x** and **-X**. For more information, see the **trn** description in the online manual. To read this description, use the Unix command:

```
man trn
```

When a system manager installs **trn**, he or she must choose whether to have **trn** offer all its special features by default or to behave the same as **rn**. Most system managers choose to enable **trn** completely. However, your system manager might have chosen to have **trn** act exactly like **rn**. The reasoning is that novice users will not be bothered by any of the non-**rn** features.

If this is the case for your system, you will have to explicitly turn on the **trn**-specific features by using the **-x** option:

```
trn -x
trn -q -x
```

In such cases, you should also use the **-X** option, This tells **trn** to make the **+** command the default when you are selecting a newsgroup. (This is the command that invokes the thread selector. We will discuss it in the next section.)

Thus, two common ways to start **trn** are:

```
trn -x -X
trn -q -x -X
```

As with most such options, you can combine them using a single **-** character:

```
trn -xX
trn -qxX
```

These two commands are equivalent to their counterparts above.

If the **trn** features are turned on for your system and you want to force **trn** to act like **rn** (with none of the extended features), you can use the **+x** option:

```
trn +x
trn -q +x
```

You stop **trn** the same way you stop **rn**: press the **q** (quit) key. If you are reading or selecting an article or a thread, you may have to press **q** more than once to work your way back out to the top level.

Alternatively, if you are at the top level (selecting a newsgroup), you can quit **trn** by pressing the **x** (exit) key. This not only quits **rn**, but restores your **.newsrc** file to what it was when you started. This abandons all the changes you may have made, such as subscribing and unsubscribing to newsgroups, reading articles, and so on.

Selecting a Newsgroup

Selecting a newsgroup with **trn** works much the same as with **rn**. You will be presented with the name of a newsgroup and asked to make a decision:

```
******** 11 unread articles in rec.humor.funny--read now? [+ynq]
```

At this point, you press SPACE to invoke the default command. You can press **y** (yes) to read the current newsgroup, and use **n** or **p** to go to the next or previous group. You can also use the **g** command to go to a specific newsgroup, and the **/** and **?** commands to search for a newsgroup whose name contains a specific pattern. All of these commands are summarized in Figure 13-3.

However, as a **trn** user, you have two extra commands. First, you can press **+** to start the thread selector. This will be the default command, so, normally, you would just press SPACE.

The second extra command is **A**. This tells **trn** to abandon all the changes that you might have made to the current newsgroup. Use this command when you have made a mess of marking articles as read or unread, and you would rather just forget the whole thing.

Selecting a Thread

The biggest difference between **rn** and **trn** is that **trn** has a selection capability which allows you to choose which threads you want to read. You invoke this

feature by using the + command at the newsgroup selection level. When you do, **trn** will display all the threads in the newsgroup, one page at a time. Figure 13-1 shows a typical screen.

The top line of the screen shows the name of the newsgroup and the number of unread articles. In this case, we are looking at the **rec.humor** newsgroup, which has 83 unread articles.

The bottom line shows us that we are 16% of the way through the newsgroup. To the right, we are reminded that the two most likely commands we would want to use are > and **Z** (which we will get to in a moment). Since > is listed first, it is the default and we can use this command by pressing SPACE.

The main part of the screen displays as many threads as will fit. Each thread is given a *thread id*, a letter or number that identifies the thread. In this case, we see four threads, labelled **a**, **b**, **c**, and **d**. For each thread, we see the author and subject of the individual articles. For example, thread **b** has eight articles, all with the same subject. You may see threads that have more than one subject because people do modify it when they compose followup articles.

trn has three different display formats that it can use to show you the list of threads. You can cycle from one format to another by pressing the **L** key. Press **L** a few times and see which one you like best.

```
rec.humor       83 articles

a Bertie Wooster     1  >Joke about Wembley
b+Reggie Jeeves      8  >Riddle about Unix
  Bingo Little
  Rosie M. Banks
  Agatha Spenser
  Dahlia Travers
  Catsmeat Pirbright
  Roderick Glossop
  Florence Craye
c Oofy Prosser       2  >Money and computers
  Gwendolyn Moon
d+Tuppy Glossop      2  >Irish water-spaniel
  Miss Dalgleish

- Select threads - 16% [>Z] -
```

Figure 13-1. *A typical* **trn** *thread selection screen*

As you read the thread summary, the idea is to look at subjects and authors and select only those threads that you want to read. After you have made your selections, you can tell **trn** to go ahead and display the articles. From that point on, everything works pretty much the same as with **rn**.

There are a large number of commands that you can use to select threads. These commands are summarized in Figure 13-4. However, we will take a moment to discuss the most basic ones.

The easiest way to select a thread is to type the thread id. For example, to select thread **b**, press the **b** key. Once a thread is selected, you will see a **+** character to the right of the thread id. In our example, this is the case for threads **b** and **d**.

If you want to unselect a thread, just press the thread id key again. For example, to unselect thread **b**, press the **b** key again. The **+** character will disappear.

Once you have selected threads from the current page, you have two choices. You can move to the next page of threads and select some more. Or you can read what you have already selected.

In other words, there are two basic strategies for selecting and reading. Some people like to skim through all the thread selection pages, make all their selections, and then read the threads. Other people like to select and read, one page at a time. The choice is yours.

To move to the next page of threads, press **>**. Since this is the default command, you can simply press SPACE. When you are ready to read what you have selected, press **Z**. (If you find it more convenient, you can press TAB instead.)

As we mentioned, there are a lot more commands that you can use while selecting threads. We suggest that, from time to time, you take a look at the summary in Figure 13-4 and teach yourself a new command.

Selecting and Reading Articles

While you are selecting and reading articles, everything works almost exactly the same as with **rn**. In particular, you can use the usual commands to page through an article, respond to an article, and so on.

Nonetheless, there is one major difference: **trn** is thread-oriented. First, **trn** will, by default, display articles in the order that they appear in the thread. Second, there are special thread-oriented commands that you can use to move within the thread, from one article to another.

To help you see where you are, **trn** will draw a little diagram in the upper right-hand corner of the screen. Figure 13-2 shows a typical page of an article as it is displayed by **trn**.

The top line tells us that we are reading the **rec.humor** newsgroup. We are looking at article number 8281. There are 7 more unread articles in this thread and 74 other unread articles in this group. Below, we see the header, body, and signature as we described in Chapter 10. The bottom line shows us what article

```
rec.humor #8281 (7 + 74 more)                    (1)+-[1]-[1]
From: jeeves@ganymede.com (Reggie Jeeves)            |-[1]
Newsgroups: rec.humor                                |-[2]+-[2]
[1] Riddle about Unix                                |    \-[2]
Date: Thu May 20 00:28:07 PDT 1993                   \-[1]
Organization: The Junior Ganymede Club
Lines: 20
Distribution: world

Somebody at the club here was asking the following riddle, but
nobody could come up with the answer. Does anyone on the Net
know?

-> How many Unix does it take to screw in a light bulb?

Obligatory Joke:
   "I'm a Rotationist, you know."
   "Ah, yes. Elks, Shriners and all that. I've seen pictures
of them, in funny hats."
   "No, no. You're thinking of Rotarians. I'm a Rotationist,
which is quite different. We believe that we are reborn as
one of our ancestors every ninth generation."

-
R. Jeeves                                   Rem acu tetigisti
jeeves@ganymede.com
                    Junior Ganymede Club
              Curzon Street, W, London, England

End of article 8281 (of 8362)-what next? [npq]
```

Figure 13-2. *A typical* **trn** *article screen*

we are reading and suggests that we might want to use the **n** (go to next article in the thread), **p** (previous article), or **q** (quit) command.

There are, of course, many commands that we can use at this point, and they are summarized in Figure 13-5. Since the default is **n**, we can move to the next article by pressing SPACE.

The most intriguing item on this screen is the small diagram in the upper right-hand corner. This is a tree-representation of the current thread. The shape of the tree shows the form of the thread. Each number indicates a different subject. In this example, we see the relationship of the eight articles in the thread.

The original article has four followups. The first followup has one followup of its own. The third followup has two followups of its own. Moreover, when the third followup was composed, the author changed the subject. Articles with the original subject are marked as **1**, while articles with the new subject are marked as **2**, although they are all considered to be part of the same thread.

Articles that you have already read are marked with **( )**. Unread articles, including the current article, are marked with **[ ]**. The current article will be indicated by having the brackets and the number highlighted (for example, **[1]**). The last article you read will be indicated by having only the number highlighted (for example, **[1]**). This will all make sense when you see it on your screen.

Hint

If you have trouble understanding the tree structure, you are not alone. However, there is an easy way to learn how to read these diagrams.

As you move within the tree, the diagram will change. Choose a newsgroup that has a lot of followups (**rec.humor** will do just fine) and use some of the commands described below. As you display one article after another—that is, as you move from one part of the tree to another—watch how the diagram changes. It won't take long before you understand how it all works.

With large threads, the full tree will not fit into the corner of the page, so only part of the tree will be displayed. With such threads, you can display a diagram of the entire thread, including a list of all the different subjects, by pressing **t**.



FUN TIP: Select an extra-large thread and then use the **t** *command. You will see an awesome tree diagram.*

As we mentioned above, there are special **trn** commands that allow you to move throughout the thread. Press] to go to the article that is one position to the right in the tree. (This is called the *child*.) If you are already as far to the right as you can go in this branch,] will move down and to the right.

Press [to go the article that is one position to the left in the tree. (This is called the *parent*.) If you are already as far to the left as you can go in this branch, [will move up and to the left.

The { command moves to the top left of the tree (the *root*). The } command moves all the way to the right (the *leaf*).

Finally, if you get tired of what you are reading, it is easy to jump to a completely different thread. Press < to go to the previous thread; press > to go to the next thread.