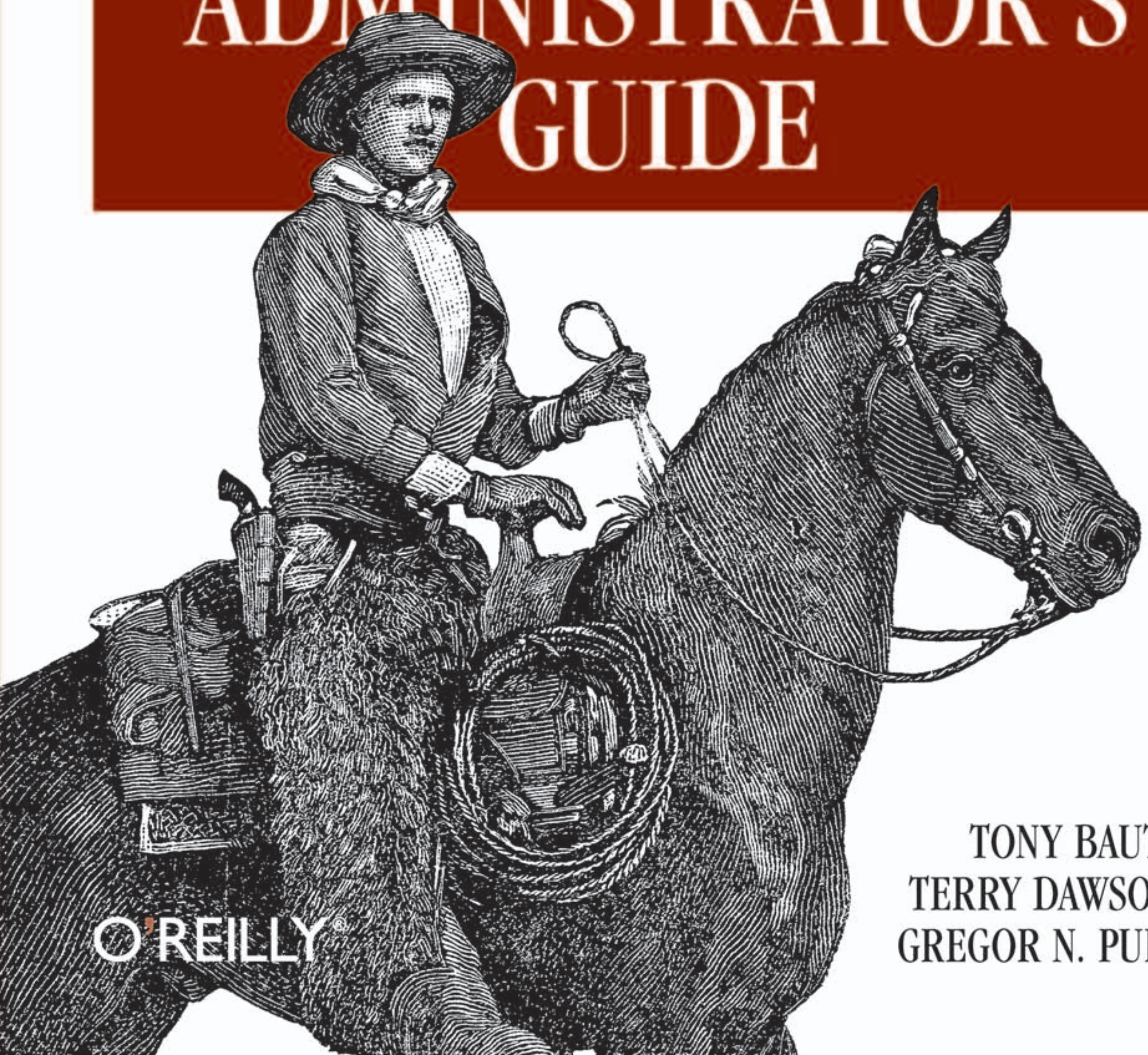
INFRASTRUCTURE, SERVICES, AND SECURITY

3rd Edition

LINUX NETWORK ADMINISTRATOR'S GUIDE



TONY BAUTTS,
TERRY DAWSON &
GREGOR N. PURDY

O'REILLY®

Networking

INFRASTRUCTURE, SERVICES, AND SECURITY

LINUX NETWORK ADMINISTRATOR'S GUIDE

For 10 years, this book has given Linux administrators all the basics they need to add a system to a network and administer a network. This highly acclaimed guide takes an in-depth look at all of the essential networking software and utilities that come with the operating system, including basic infrastructure (TCP/IP, wireless networking, firewalling) and the other popular services on Linux systems. This third edition adds new coverage of useful services such as Apache, Samba, and OpenLDAP and provides the latest information on cutting-edge services such as wireless, IMAP, and IPv6.

Anyone with a basic understanding of Linux can become a competent network administrator and run a server or firewall after reading this book. Each chapter combines detailed configuration and administration information with the background needed to help administrators get their Linux network up and running.

Included in the book are detailed explanations of essentials such as:

* Configuring the hardware and Ethernet interfaces

* Setting up a nameserver (either BIND or djbdns)

* Connecting over a serial line with PPP

* Setting up a firewall, along with masquerading and accounting

* Running inetd or related superservers

* Logging in remotely through ssh

The book also details how to provide critical services, such as mail (through sendmail and Cyrus IMAP), the Samba file and print server, the Apache web server, and the OpenLDAP directory service. Chapters on IPv6 administration and wireless networking establish this book at the forefront of networking developments.

The Linux Network Administrator's Guide was originally a volunteer effort at the Linux Documentation Project. It remains one of the most highly regarded books on Linux networking.

O'REILLY® www.oreilly.com

US $34.95 CAN $50.95
ISBN-10: 0-596-00548-2
ISBN-13: 978-0-596-00548-1

53495

9 780596 005481

LINUX
Network Administrator's Guide

LINUX
Network Administrator's Guide

THIRD EDITION

Tony Bautts, Terry Dawson,
and Gregor N. Purdy

O'REILLY®

Beijing · Cambridge · Farnham · Köln · Paris · Sebastopol · Taipei · Tokyo

Linux Network Administrator's Guide, Third Edition
by Tony Bautts, Terry Dawson, and Gregor N. Purdy

Printed in the United States of America.

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

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*safari.oreilly.com*). For more information, contact our corporate/institutional sales department: (800) 998-9938 or *corporate@oreilly.com*.

Editor: Andy Oram

Production Editor: Adam Witwer

Cover Designer: Edie Freedman

Interior Designer: David Futato

Printing History:

January 1995: First Edition.

June 2000: Second Edition.

February 2005: Third Edition.

RepKover™ This book uses RepKover™, a durable and flexible lay-flat binding.

ISBN: 0-596-00548-2
[M] [5/05]

Table of Contents

Preface . . . . **ix**

1. Introduction to Networking **1**
History 1
TCP/IP Networks 2
Linux Networking 11
Maintaining Your System 13

2. Issues of TCP/IP Networking **16**
Networking Interfaces 16
IP Addresses 17
The Internet Control Message Protocol 26

3. Configuring the Serial Hardware **29**
Communications Software for Modem Links 29
Accessing Serial Devices 30
Using the Configuration Utilities 34
Serial Devices and the login: Prompt 38

4. Configuring TCP/IP Networking **42**
Understanding the /proc Filesystem 43

5. Name Service and Configuration **66**
The Resolver Library 67
How DNS Works 71
Alternatives to BIND 92

6. **The Point-to-Point Protocol** **96**
PPP on Linux 97
Running pppd 98
Using Options Files 99
Using chat to Automate Dialing 100
IP Configuration Options 102
Link Control Options 105
General Security Considerations 107
Authentication with PPP 108
Debugging Your PPP Setup 112
More Advanced PPP Configurations 112
PPPoE Options in Linux 116

7. **TCP/IP Firewall** **119**
Methods of Attack 120
What Is a Firewall? 122
What Is IP Filtering? 124
Netfilter and iptables 125
iptables Concepts 127
Setting Up Linux for Firewalling 133
Using iptables 134
The iptables Subcommands 136
Basic iptables Matches 137
A Sample Firewall Configuration 141
References 144

8. **IP Accounting** **146**
Configuring the Kernel for IP Accounting 146
Configuring IP Accounting 146
Using IP Accounting Results 151
Resetting the Counters 151
Flushing the Rule Set 152
Passive Collection of Accounting Data 152

9. **IP Masquerade and Network Address Translation** **154**
Side Effects and Fringe Benefits 156
Configuring the Kernel for IP Masquerade 157
Configuring IP Masquerade 157
Handling Nameserver Lookups 158
More About Network Address Translation 159

10. Important Network Features . . . 160
The inetd Super Server 160
The tcpd Access Control Facility 163
The xinetd Alternative 164
The Services and Protocols Files 167
Remote Procedure Call 169
Configuring Remote Login and Execution 170

11. Administration Issues with Electronic Mail . . . 179
What Is a Mail Message? 180
How Is Mail Delivered? 182
Email Addresses 183
How Does Mail Routing Work? 184
Mail Routing on the Internet 184

12. sendmail . . . 186
Installing the sendmail Distribution 186
sendmail Configuration Files 192
sendmail.cf Configuration Language 198
Creating a sendmail Configuration 203
sendmail Databases 210
Testing Your Configuration 222
Running sendmail 227
Tips and Tricks 228
More Information 231

13. Configuring IPv6 Networks . . . 233
The IPv4 Problem and Patchwork Solutions 234
IPv6 as a Solution 235

14. Configuring the Apache Web Server . . . 244
Apache HTTPD Server—An Introduction 244
Configuring and Building Apache 244
Configuration File Options 247
VirtualHost Configuration Options 250
Apache and OpenSSL 252
Troubleshooting 256

15. IMAP **258**
IMAP—An Introduction 258
Cyrus IMAP 263

16. Samba **266**
Samba—An Introduction 266

17. OpenLDAP **278**
Understanding LDAP 278
Obtaining OpenLDAP 280

18. Wireless Networking **294**
History 294
The Standards 295
802.11b Security Concerns 296

Appendix: Example Network: The Virtual Brewery **309**

Index **311**

Preface

The Internet is now a household term in many countries and has become a part of life for most of the business world. With millions of people connecting to the World Wide Web, computer networking has moved to the status of TV sets and microwave ovens. You can purchase and install a wireless hub with just about an equal amount of effort. The Internet has unusually high media coverage, with weblogs often "scooping" traditional media outlets for news stories, while virtual reality environments such as online games and the rest have developed into the "Internet culture."

Of course, networking has been around for a long time. Connecting computers to form local area networks has been common practice, even at small installations, and so have long-haul links using transmission lines provided by telecommunications companies. A rapidly growing conglomerate of worldwide networks has, however, made joining the global village a perfectly reasonable option for nearly everyone with access to a computer. Setting up a broadband Internet host with fast mail and web access is becoming more and more affordable.

Talking about computer networks often means talking about Unix. Of course, Unix is not the only operating system with network capabilities, nor will it remain a frontrunner forever, but it has been in the networking business for a long time and will surely continue to be for some time to come. What makes Unix particularly interesting to private users is that there has been much activity to bring free Unix-like operating systems to the PC, such as NetBSD, FreeBSD, and Linux.

Linux is a freely distributable Unix clone for personal computers that currently runs on a variety of machines that includes the Intel family of processors, but also PowerPC architectures such as the Apple Macintosh; it can also run on Sun SPARC and Ultra-SPARC machines; Compaq Alphas; MIPS; and even a number of video game consoles, such as the Sony PlayStation 2, the Nintendo Gamecube, and the Microsoft Xbox. Linux has also been ported to some relatively obscure platforms, such as the Fujitsu AP-1000 and the IBM System 3/90. Ports to other interesting architectures are currently in progress in developers' labs, and the quest to move Linux into the embedded controller space promises success.

Linux was developed by a large team of volunteers across the Internet. The project was started in 1990 by Linus Torvalds, a Finnish college student, as an operating systems course project. Since that time, Linux has snowballed into a full-featured Unix clone capable of running applications as diverse as simulation and modeling programs, word processors, speech-recognition systems, World Wide Web browsers, and a horde of other software, including a variety of excellent games. A great deal of hardware is supported, and Linux contains a complete implementation of TCP/IP networking, including PPP, firewalls, and many features and protocols not found in any other operating system. Linux is powerful, fast, and free, and its popularity in the world beyond the Internet is growing rapidly.

The Linux operating system itself is covered by the GNU General Public License, the same copyright license used by software developed by the Free Software Foundation. This license allows anyone to redistribute or modify the software (free of charge or for a profit) as long as all modifications and distributions are freely distributable as well. The term "free software" refers to freedom of application, not freedom of cost.

Purpose and Audience for This Book

This book was written to provide a single reference for network administration in a Linux environment. Beginners and experienced users alike should find the information they need to cover nearly all important administration activities required to manage a Linux network configuration. The possible range of topics to cover is nearly limitless, so of course it has been impossible to include everything there is to say on all subjects. We've tried to cover the most important and common ones. Beginners to Linux networking, even those with no prior exposure to Unix-like operating systems, have found earlier editions of this book good enough to help them successfully get their Linux network configurations up and running and get them ready to learn more.

There are many books and other sources of information from which you can learn any of the topics covered in this book in greater depth. We've provided a bibliography when you are ready to explore more.

Sources of Information

If you are new to the world of Linux, there are a number of resources to explore and become familiar with. Having access to the Internet is helpful, but not essential.

Linux Documentation Project Guides

The Linux Documentation Project is a group of volunteers who have worked to produce books (guides), HOWTO documents, and manpages on topics ranging from installation to kernel programming.

Books

Linux Installation and Getting Started
: By Matt Welsh, et al. This book describes how to obtain, install, and use Linux. It includes an introductory Unix tutorial and information on systems administration, the X Window System, and networking.

Linux System Administrators Guide
: By Lars Wirzenius and Joanna Oja. This book is a guide to general Linux system administration and covers topics such as creating and configuring users, performing system backups, configuring of major software packages, and installing and upgrading software.

Linux System Adminstration Made Easy
: By Steve Frampton. This book describes day-to-day administration and maintenance issues of relevance to Linux users.

Linux Programmers Guide
: By B. Scott Burkett, Sven Goldt, John D. Harper, Sven van der Meer, and Matt Welsh. This book covers topics of interest to people who wish to develop application software for Linux.

The Linux Kernel
: By David A. Rusling. This book provides an introduction to the Linux kernel, how it is constructed, and how it works. Take a tour of your kernel.

The Linux Kernel Module Programming Guide
: By Ori Pomerantz. This guide explains how to write Linux kernel modules. This book also originated in the LDP. The text of the current version is released under the Creative Commons Attribution-Share Alike License, so it can be freely altered and distributed.

More manuals are in development. For more information about the LDP, consult their server at *http://www.linuxdoc.org/* or one of its many mirrors.

HOWTO documents

The Linux HOWTOs are a comprehensive series of papers detailing various aspects of the system—such as how to install and configure the X Window System software, or write in assembly language programming under Linux. These are available online at one of the many Linux Documentation Project mirror sites (see next section). See the file *HOWTO-INDEX* for a list of what's available.

You might want to obtain the *Installation HOWTO*, which describes how to install Linux on your system; the *Hardware Compatibility HOWTO*, which contains a list of hardware known to work with Linux; and the *Distribution HOWTO*, which lists software vendors selling Linux on diskette and CD-ROM.

Linux Frequently Asked Questions

The *Linux Frequently Asked Questions with Answers* (FAQ) contains a wide assortment of questions and answers about the system. It is a must-read for all newcomers.

Documentation Available via WWW

There are many Linux-based WWW sites available. The home site for the Linux Documentation Project can be accessed at *http://www.tldp.org/*.

Any additional information can probably be found with a quick Google search. It seems that almost everything has been tried and likely written up by someone in the Linux community.

Documentation Available Commercially

A number of publishing companies and software vendors publish the works of the Linux Documentation Project. Two such vendors are Specialized Systems Consultants, Inc. (SSC) (*http://www.ssc.com*) and Linux Systems Labs (*http://www.lsl.com*). Both companies sell compendiums of Linux HOWTO documents and other Linux documentation in printed and bound form.

O'Reilly Media publishes a series of Linux books. This one is a work of the Linux Documentation Project, but most have been authored independently:

Running Linux
: An installation and user guide to the system describing how to get the most out of personal computing with Linux.

Linux Server Security
: An excellent guide to configuring airtight Linux servers. Administrators who are building web servers or other bastion hosts should consider this book a great source of information.

Linux in a Nutshell
: Another in the successful "in a Nutshell" series, this book focuses on providing a broad reference text for Linux.

Linux iptables Pocket Reference
: A brief but complete compendium of features in the Linux firewall system.

Linux Journal and Linux Magazine

Linux Journal and *Linux Magazine* are monthly magazines for the Linux community, written and published by a number of Linux activists. They contain articles ranging from novice questions and answers to kernel programming internals. Even if you have Usenet access, these magazines are a good way to stay in touch with the Linux community.

Linux Journal is the oldest magazine and is published by SSC, for which details were listed in the previous section. You can also find the magazine at *http://www.linuxjournal.com/*.

LinuxMagazine is a newer, independent publication. The home web site for the magazine is *http://www.linuxmagazine.com/*.

Linux Usenet Newsgroups

If you have access to Usenet news, the following Linux-related newsgroups are available:

comp.os.linux.announce
: A moderated newsgroup containing announcements of new software, distributions, bug reports, and goings-on in the Linux community. All Linux users should read this group.

comp.os.linux.help
: General questions and answers about installing or using Linux.

comp.os.linux.admin
: Discussions relating to systems administration under Linux.

comp.os.linux.networking
: Discussions relating to networking with Linux.

comp.os.linux.development
: Discussions about developing the Linux kernel and system itself.

comp.os.linux.misc
: A catch-all newsgroup for miscellaneous discussions that don't fall under the previous categories.

There are also several newsgroups devoted to Linux in languages other than English, such as *fr.comp.os.linux* in French and *de.comp.os.linux* in German.

Linux Mailing Lists

There are a large number of specialist Linux mailing lists on which you will find many people willing to help with your questions.

The best-known of these is the Linux Kernel Mailing List. It's a very busy and dense mailing list, with an enormous volume of information posted daily. For more information, visit *http://www.tux.org/lkml*.

Linux User Groups

Many Linux User Groups around the world offer direct support to users, engaging in activities such as installation days, talks and seminars, demonstration nights, and other social events. Linux User Groups are a great way to meet other Linux users in your area. There are a number of published lists of Linux User Groups. One of the most comprehensive is Linux Users Groups Worldwide (*http://lugww.counter.li.org/index.cms*).

Obtaining Linux

There is no single distribution of the Linux software; instead, there are many distributions, such as Debian, Fedora, Red Hat, SUSE, Gentoo, and Slackware. Each distribution contains everything you need to run a complete Linux system: the kernel, basic utilities, libraries, support files, and applications software.

Linux distributions may be obtained via a number of online sources, such as the Internet. Each of the major distributions has its own FTP and web site. Some of these sites are as follows:

Debian
: *http://www.debian.org/*

Gentoo
: *http://www.gentoo.org/*

Red Hat
: *http://www.redhat.com/*

Fedora
: *http://fedora.redhat.com/*

Slackware
: *http://www.slackware.com/*

SUSE
: *http://www.suse.com/*

Many of the popular general WWW archive sites also mirror various Linux distributions. The best-known of these sites is *http://www.linuxiso.org*.

Every major distribution can be downloaded directly from the Internet, but Linux may be purchased on CD-ROM from an increasing number of software vendors. If your local computer store doesn't have it, perhaps you should ask them to stock it! Most of the popular distributions can be obtained on CD-ROM. Some vendors produce products containing multiple CD-ROMs, each of which provides a different Linux distribution. This is an ideal way to try a number of different distributions before settling on your favorite.

Filesystem Standards

In the past, one of the problems that afflicted Linux distributions, as well as the packages of software running on Linux, was the lack of a single accepted filesystem layout. This resulted in incompatibilities between different packages, and confronted users and administrators with the task of locating various files and programs.

To improve this situation, in August 1993, several people formed the Linux File System Standard Group (FSSTND). After six months of discussion, the group created a draft that presents a coherent filesystem structure and defines the location of the most essential programs and configuration files.

This standard was supposed to have been implemented by most major Linux distributions and packages. It is a little unfortunate that, while most distributions have made some attempt to work toward the FSSTND, there is a very small number of distributions that has actually adopted it fully. Throughout this book, we will assume that any files discussed reside in the location specified by the standard; alternative locations will be mentioned only when there is a long tradition that conflicts with this specification.

The Linux FSSTND continued to develop, but was replaced by the Linux File Hierarchy Standard (FHS) in 1997. The FHS addresses the multi-architecture issues that the FSSTND did not. The FHS can be obtained from *http://www.freestandards.org*.

Standard Linux Base

The vast number of different Linux distributions, while providing lots of healthy choices for Linux users, has created a problem for software developers—particularly developers of non-free software.

Each distribution packages and supplies certain base libraries, configuration tools, system applications, and configuration files. Unfortunately, differences in their versions, names, and locations make it very difficult to know what will exist on any distribution. This makes it hard to develop binary applications that will work reliably on all Linux distribution bases.

To help overcome this problem, a new project sprang up called the Linux Standard Base. It aims to describe a standard base distribution that complying distributions will use. If a developer designs an application to work with the standard base platform, the application will work with, and be portable to, any complying Linux distribution.

You can find information on the status of the Linux Standard Base project at its home web site at *http://www.linuxbase.org/*.

If you're concerned about interoperability, particularly of software from commercial vendors, you should ensure that your Linux distribution is making an effort to participate in the standardization project.

About This Book

When Olaf Kirche joined the LDP in 1992, he wrote two small chapters on UUCP and smail, which he meant to contribute to the System Administrator's Guide. Development of TCP/IP networking was just beginning, and when those "small chapters" started to grow, he wondered aloud whether it would be nice to have a Networking Guide. "Great!" everyone said. "Go for it!" So he went for it and wrote the first version of the Networking Guide, which was released in September 1993.

Olaf continued work on the Networking Guide and eventually produced a much enhanced version of the guide. Vince Skahan contributed the original sendmail mail chapter, which was completely replaced in that edition because of a new interface to the sendmail configuration.

In March of 2000, Terry Dawson updated Olaf's original, adding several new chapters and bringing it into the new millennium.

The version of the guide that you are reading now is a fairly large revision and update prompted by O'Reilly Media and undertaken by Tony Bautts. Tony has been enthusiastic Linux user and information security consultant for longer than he would care to admit. He is coauthor of several other computer security-related books and likes to give talks on the subject as well. Tony is a big proponent of Linux in the commercial environment and routinely attempts to convert people to Gentoo Linux. For this edition he has added a few new chapters describing features of Linux networking that have been developed since the second edition, plus a bunch of changes to bring the rest of the book up to date.

The three iptables chapters (Chapters 7, 8, and 9) were updated by Gregor Purdy for this edition.

The book is organized roughly along the sequence of steps that you have to take to configure your system for networking. It starts by discussing basic concepts of networks, and TCP/IP-based networks in particular. It then slowly works its way up from configuring TCP/IP at the device level to firewall, accounting, and masquerade configuration, to the setup of common applications such as SSH, Apache, and Samba. The email part features an introduction to the more intimate parts of mail transport and routing and the myriad of addressing schemes that you may be confronted with. It describes the configuration and management of sendmail, the most common mail transport agent, and IMAP, used for delivery to individual mail users.

Chapters on LDAP and wireless networking round out the infrastructure for modern network administration.

Of course, a book can never exhaustively answer all questions you might have. So if you follow the instructions in this book and something still does not work, please be patient. Some of your problems may be due to mistakes on our part (see "How to Contact Us," later in this Preface), but they also may be caused by changes in the networking software. Therefore, you should check the listed information resources first. There's a good chance that you are not alone with your problems, so a fix or at least a proposed workaround is likely to be known—this is where search engines are particularly handy! If you have the opportunity, you should also try to get the latest kernel and network release from *http://www.kernel.org*. Many problems are caused by software from different stages of development, which fail to work together properly. After all, Linux is a "work in progress."

The Official Printed Version

In Autumn 1993, Andy Oram, who had been around the LDP mailing list from almost the very beginning, asked Olaf about publishing this book at O'Reilly & Associates. He was excited about this book, but never imagined that it would become as successful as it has. He and Andy finally agreed that O'Reilly would produce an enhanced Official Printed Version of the Networking Guide, while Olaf retained the original copyright so that the source of the book could be freely distributed. This means that you can choose freely: you can get the various free forms of the document from your nearest LDP mirror site and print it out, or you can purchase the official printed version from O'Reilly.

Why, then, would you want to pay money for something you can get for free? Is Tim O'Reilly out of his mind for publishing something everyone can print and even sell themselves?* Is there any difference between these versions?

The answers are "It depends," "No, definitely not," and "Yes and no." O'Reilly Media does take a risk in publishing the Network Administrator's Guide, but it seems to have paid off for them (since they've asked us to do it two more times). We believe this project serves as a fine example of how the free software world and companies can cooperate to produce something both can benefit from. In our view, the great service O'Reilly provides the Linux community (apart from the book becoming readily available in your local bookstore) is that it has helped Linux become recognized as something to be taken seriously: a viable and useful alternative to other commercial operating systems. It's a sad technical bookstore that doesn't have at least one shelf stacked with O'Reilly Linux books.

* Note that while you are allowed to print out the online version, you may not run the O'Reilly book through a photocopier, much less sell any of its (hypothetical) copies.

Why are they publishing it? They see it as their kind of book. It's what they would hope to produce if they contracted with an author to write about Linux. The pace, level of detail, and style fit in well with their other offerings.

The point of the LDP license is to make sure no one gets shut out. Other people can print out copies of this book, and no one will blame you if you get one of these copies. But if you haven't gotten a chance to see the O'Reilly version, try to get to a bookstore or look at a friend's copy. We think you'll like what you see and will want to buy it for yourself.

So what about the differences between the printed and online versions? Andy Oram has made great efforts at transforming our ramblings into something actually worth printing. (He has also reviewed a few other books produced by the LDP, contributing whatever professional skills he can to the Linux community.)

Since Andy started reviewing the Networking Guide and editing the copies sent to him, the book has improved vastly from its original form, and with every round of submission and feedback, it improves again. The opportunity to take advantage of a professional editor's skill is not to be wasted. In many ways, Andy's contribution has been as important as that of the authors. The same is also true of the production staff, who got the book into the shape that you see now. All these edits have been fed back into the online version, so there is no difference in content.

Still, the O'Reilly version *will* be different. It will be professionally bound, and while you may go to the trouble to print the free version, it is unlikely that you will get the same quality result. Secondly, our amateurish attempts at illustration will have been replaced with nicely redone figures by O'Reilly's professional artists. Indexers have generated an improved index, which makes locating information in the book a much simpler process. If this book is something you intend to read from start to finish, you should consider reading the official printed version.

Overview

Chapter 1, *Introduction to Networking*, discusses the history of Linux and covers basic networking information on UUCP, TCP/IP, various protocols, hardware, and security. The next few chapters deal with configuring Linux for TCP/IP networking and running some major applications.

Chapter 2, *Issues of TCP/IP Networking*, examines IP a little more closely before we get our hands dirty with file editing and the like. If you already know how IP routing works and how address resolution is performed, you can skip this chapter.

Chapter 3, *Configuring the Serial Hardware*, deals with the configuration of your serial ports.

Chapter 4, *Configuring TCP/IP Networking*, helps you set up your machine for TCP/IP networking. It contains installation hints for standalone hosts and those connected to a network. It also introduces you to a few useful tools you can use to test and debug your setup.

Chapter 5, *Name Service and Configuration*, discusses how to configure hostname resolution and explains how to set up a name server.

Chapter 6, *The Point-to-Point Protocol*, covers PPP and *pppd*, the PPP daemon.

Chapter 7, *TCP/IP Firewall*, extends our discussion on network security and describes the Linux TCP/IP firewall iptables. IP firewalling provides a means of very precisely controlling who can access your network and hosts.

Chapter 8, *IP Accounting*, explains how to configure IP Accounting in Linux so that you can keep track of how much traffic is going where and who is generating it.

Chapter 9, *IP Masquerade and Network Address Translation*, covers a feature of the Linux networking software called IP masquerade, or NAT, which allows whole IP networks to connect to and use the Internet through a single IP address, hiding internal systems from outsiders in the process.

Chapter 10, *Important Network Features*, gives a short introduction to setting up some of the most important network infrastructure and applications, such as SSH. This chapter also covers how services are managed by the inetd superuser and how you may restrict certain security-relevant services to a set of trusted hosts.

Chapter 11, *Administration Issues with Electronic Mail*, introduces you to the central concepts of electronic mail, such as what a mail address looks like and how the mail handling system manages to get your message to the recipient.

Chapter 12, *sendmail*, covers the configuration of *sendmail*, a mail transport agent that you can use for Linux.

Chapter 13, *Configuring IPv6 Networks*, covers new ground by explaining how to configure IPv6 and connect to the IPv6 backbone.

Chapter 14, *Configuring the Apache Web Server*, describes the steps necessary to build an Apache web server and host basic web services.

Chapter 15, *IMAP*, explains the steps necessary to configure an IMAP mail server, and discusses its advantages over the traditional POP mail solution.

Chapter 16, *Samba*, helps you understand how to configure your Linux server to play nicely in the Windows networking world—so nicely, in fact, that your Windows users might not be able to tell the difference.*

Chapter 17, *OpenLDAP*, introduces OpenLDAP and discusses the configuration and potential uses of this service

Chapter 18, *Wireless Networking*, finally, details the steps required to configure wireless networking and build a Wireless Access Point on a Linux server.

* The obvious joke here is left to the reader.

Conventions Used in This Book

All examples presented in this book assume that you are using an sh-compatible shell. The bash shell is sh compatible and is the standard shell of all Linux distributions. If you happen to be a csh user, you will have to make appropriate adjustments.

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

Italic
: Used for file and directory names, program and command names, email addresses and pathnames, URLs, and for emphasizing new terms.

Boldface
: Used for machine names, hostnames, site names, and for occasional emphasis.

`Constant Width`
: Used in examples to show the contents of code files or the output from commands and to indicate environment variables and keywords that appear in code.

`Constant Width Italic`
: Used to indicate variable options, keywords, or text that the user is to replace with an actual value.

`Constant Width Bold`
: Used in examples to show commands or other text that should be typed literally by the user.

Indicates a tip, suggestion, or general note.

Text appearing in this manner offers a warning. You can make a mistake here that hurts your system or is hard to recover from.

Safari Enabled

When you see a Safari® Enabled icon on the cover of your favorite technology book, that means the book is available online through the O'Reilly Network Safari Bookshelf.

Safari offers a solution that's better than e-books. It's a virtual library that lets you easily search thousands of top tech books, cut and paste code samples, download chapters, and find quick answers when you need the most accurate, current information. Try it for free at *http://safari.oreilly.com*.

How to Contact Us

We have tested and verified the information in this book to the best of our ability, but you may find that features have changed (or even that we have made mistakes!). Please let us know about any errors you find, as well as your suggestions for future editions, by writing to:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
(800) 998-9938 (in the United States or Canada)
(707) 829-0515 (international or local)
(707) 829-0104 (fax)

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

info@oreilly.com

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

bookquestions@oreilly.com

We have a web site for the book, where we'll list examples, errata, and any plans for future editions. You can access this page at:

http://www.oreilly.com/catalog/linag3

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

http://www.oreilly.com

Acknowledgments

This edition of the Networking Guide owes much to the outstanding work of Olaf, Vince, and Terry. It is difficult to appreciate the effort that goes into researching and writing a book of this nature until you've had a chance to work on one yourself. Updating the book was a challenging task, but with an excellent base to work from, it was an enjoyable one.

This book owes very much to the numerous people who took the time to proofread it and help iron out many mistakes. Phil Hughes, John Macdonald, and Kenneth Geisshirt all provided very helpful (and on the whole, quite consistent) feedback on the content of the third edition of this book. Andres Sepúlveda, Wolfgang Michaelis, and Michael K. Johnson offered invaluable help on the second edition. Finally, the book would not have been possible without the support of Holger Grothe, who provided Olaf with the Internet connectivity he needed to make the original version happen.

Terry thanks his wife, Maggie, who patiently supported him throughout his participation in the project despite the challenges presented by the birth of their first child, Jack. Additionally, he thanks the *many* people of the Linux community who either nurtured or suffered him to the point at which he could actually take part and actively contribute. “I’ll help you if you promise to help someone else in return.”

Tony would like to thank Linux gurus Dan Ginsberg and Nicolas Lidzborski for their support and technical expertise in proofreading the new chapters. Additionally, he thanks Katherine for her input with each chapter, when all she really wanted to do was check her email. Thanks to Mick Bauer for getting me involved with this project and supporting me along the way. Finally, many thanks to the countless Linux users who have very helpfully documented their perils in getting things to work, not to mention the countless others who respond on a daily basis to questions posted on the mailing lists. Without this kind of community support, Linux would be nowhere.

CHAPTER 1
Introduction to Networking

History

The idea of networking is probably as old as telecommunications itself. Consider people living in the Stone Age, when drums may have been used to transmit messages between individuals. Suppose caveman A wants to invite caveman B over for a game of hurling rocks at each other, but they live too far apart for B to hear A banging his drum. What are A's options? He could 1) walk over to B's place, 2) get a bigger drum, or 3) ask C, who lives halfway between them, to forward the message. The last option is called *networking*.

Of course, we have come a long way from the primitive pursuits and devices of our forebears. Nowadays, we have computers talk to each other over vast assemblages of wires, fiber optics, microwaves, and the like, to make an appointment for Saturday's soccer match.* In the following description, we will deal with the means and ways by which this is accomplished, but leave out the wires, as well as the soccer part.

We define a network as a collection of *hosts* that are able to communicate with each other, often by relying on the services of a number of dedicated hosts that relay data between the participants. Hosts are often computers, but need not be; one can also think of X terminals or intelligent printers as hosts. A collection of hosts is also called a *site*.

Communication is impossible without some sort of language or code. In computer networks, these languages are collectively referred to as *protocols*. However, you shouldn't think of written protocols here, but rather of the highly formalized code of behavior observed when heads of state meet, for instance. In a very similar fashion, the protocols used in computer networks are nothing but very strict rules for the exchange of messages between two or more hosts.

* The original spirit of which (see above) still shows on some occasions in Europe.

TCP/IP Networks

Modern networking applications require a sophisticated approach to carry data from one machine to another. If you are managing a Linux machine that has many users, each of whom may wish to simultaneously connect to remote hosts on a network, you need a way of allowing them to share your network connection without interfering with each other. The approach that a large number of modern networking protocols use is called *packet switching*. A packet is a small chunk of data that is transferred from one machine to another across the network. The switching occurs as the datagram is carried across each link in the network. A packet-switched network shares a single network link among many users by alternately sending packets from one user to another across that link.

The solution that Unix systems, and subsequently many non-Unix systems, have adopted is known as TCP/IP. When learning about TCP/IP networks, you will hear the term *datagram*, which technically has a special meaning but is often used interchangeably with packet. In this section, we will have a look at underlying concepts of the TCP/IP protocols.

Introduction to TCP/IP Networks

TCP/IP traces its origins to a research project funded by the United States Defense Advanced Research Projects Agency (DARPA) in 1969. The ARPANET was an experimental network that was converted into an operational one in 1975 after it had proven to be a success.

In 1983, the new protocol suite TCP/IP was adopted as a standard, and all hosts on the network were required to use it. When ARPANET finally grew into the Internet (with ARPANET itself passing out of existence in 1990), the use of TCP/IP had spread to networks beyond the Internet itself. Many companies have now built corporate TCP/IP networks, and the Internet has become a mainstream consumer technology. It is difficult to read a newspaper or magazine now without seeing references to the Internet; almost everyone can use it now.

For something concrete to look at as we discuss TCP/IP throughout the following sections, we will consider Groucho Marx University (GMU), situated somewhere in Freedonia, as an example. Most departments run their own Local Area Networks, while some share one and others run several of them. They are all interconnected and hooked to the Internet through a single high-speed link.

Suppose your Linux box is connected to a LAN of Unix hosts at the mathematics department, and its name is **erdos**. To access a host at the physics department, say **quark**, you enter the following command:

```
$ ssh quark.school.edu
Enter password:
Last login: Wed Dec  3 18:21:25 2003 from 10.10.0.1
quark$
```

At the prompt, you enter your password. You are then given a shell* on **quark**, to which you can type as if you were sitting at the system's console. After you exit the shell, you are returned to your own machine's prompt. You have just used one of the instantaneous, interactive applications that uses TCP/IP: secure shell.

While being logged into **quark**, you might also want to run a graphical user interface application, like a word processing program, a graphics drawing program, or even a World Wide Web browser. The X Windows System is a fully network-aware graphical user environment, and it is available for many different computing systems. To tell this application that you want to have its windows displayed on your host's screen, you will need to make sure that you're SSH server and client are capable of tunneling X. To do this, you can check the *sshd_config* file on the system, which should contain a line like this:

```
X11Forwarding yes
```

If you now start your application, it will tunnel your X Window System applications so that they will be displayed on your X server instead of **quark**'s. Of course, this requires that you have X11 runnning on **erdos**. The point here is that TCP/IP allows **quark** and **erdos** to send X11 packets back and forth to give you the illusion that you're on a single system. The network is almost transparent here.

Of course, these are only examples of what you can do with TCP/IP networks. The possibilities are almost limitless, and we'll introduce you to more as you read on through the book.

We will now have a closer look at the way TCP/IP works. This information will help you understand how and why you have to configure your machine. We will start by examining the hardware and slowly work our way up.

Ethernets

The most common type of LAN hardware is known as *Ethernet*. In its simplest form, it consists of a single cable with hosts attached to it through connectors, taps, or transceivers. Simple Ethernets are relatively inexpensive to install, which together with a net transfer rate of 10, 100, 1,000, and now even 10,000 megabits per second (Mbps), accounts for much of its popularity.

Ethernets come in many flavors: *thick*, *thin*, and *twisted pair*. Older Ethernet types such as thin and thick Ethernet, rarely in use today, each use a coaxial cable, differing in diameter and the way you may attach a host to this cable. Thin Ethernet uses a T-shaped "BNC" connector, which you insert into the cable and twist onto a plug on the back of your computer. Thick Ethernet requires that you drill a small hole into

* The shell is a command-line interface to the Unix operating system. It's similar to the DOS prompt in a Microsoft Windows environment, albeit much more powerful.

the cable and attach a transceiver using a "vampire tap." One or more hosts can then be connected to the transceiver. Thin and thick Ethernet cable can run for a maximum of 200 and 500 meters, respectively, and are also called 10-base2 and 10-base5. The "base" refers to "baseband modulation" and simply means that the data is directly fed onto the cable without any modem. The number at the start refers to the speed in megabits per second, and the number at the end is the maximum length of the cable in hundreds of metres. Twisted pair uses a cable made of two pairs of copper wires and usually requires additional hardware known as *active hubs*. Twisted pair is also known as 10-baseT, the "T" meaning twisted pair. The 100 Mbps version is known as 100-baseT, and not surprisingly, 1000 Mbps is called 1000-baseT or gigabit.

To add a host to a thin Ethernet installation, you have to disrupt network service for at least a few minutes because you have to cut the cable to insert the connector. Although adding a host to a thick Ethernet system is a little complicated, it does not typically bring down the network. Twisted pair Ethernet is even simpler. It uses a device called a *hub* or *switch* that serves as an interconnection point. You can insert and remove hosts from a hub or switch without interrupting any other users at all.

Thick and thin Ethernet deployments are somewhat difficult to find anymore because they have been mostly replaced by twisted pair deployments. This has likely become a standard because of the cheap networking cards and cables—not to mention that it's almost impossible to find an old BNC connector in a modern laptop machine.

Wireless LANs are also very popular. These are based on the 802.11a/b/g specification and provide Ethernet over radio transmission. Offering similar functionality to its wired counterpart, wireless Ethernet has been subject to a number of security issues, namely surrounding encryption. However, advances in the protocol specification combined with different encryption keying methods are quickly helping to alleviate some of the more serious security concerns. Wireless networking for Linux is discussed in detail in Chapter 18.

Ethernet works like a bus system, where a host may send packets (or *frames*) of up to 1,500 bytes to another host on the same Ethernet. A host is addressed by a 6-byte address hardcoded into the firmware of its Ethernet network interface card (NIC). These addresses are usually written as a sequence of two-digit hex numbers separated by colons, as in **aa:bb:cc:dd:ee:ff**.

A frame sent by one station is seen by all attached stations, but only the destination host actually picks it up and processes it. If two stations try to send at the same time, a *collision* occurs. Collisions on an Ethernet are detected very quickly by the electronics of the interface cards and are resolved by the two stations aborting the send, each waiting a random interval and re-attempting the transmission. You'll hear lots of stories about collisions on Ethernet being a problem and that utilization of Ethernets is only about 30 percent of the available bandwidth because of them. Collisions on

Ethernet are a *normal* phenomenon, and on a very busy Ethernet network you shouldn't be surprised to see collision rates of up to about 30 percent. Ethernet networks need to be more realistically limited to about 60 percent before you need to start worrying about it.*

Other Types of Hardware

In larger installations, or in legacy corporate environments, Ethernet is usually not the only type of equipment used. There are many other data communications protocols available and in use. All of the protocols listed are supported by Linux, but due to space constraints we'll describe them briefly. Many of the protocols have HOWTO documents that describe them in detail, so you should refer to those if you're interested in exploring those that we don't describe in this book.

One older and quickly disappearing technology is IBM's Token Ring network. Token Ring is used as an alternative to Ethernet in some LAN environments, and runs at lower speeds (4 Mbps or 16 Mbps). In Linux, Token Ring networking is configured in almost precisely the same way as Ethernet, so we don't cover it specifically.

Many national networks operated by telecommunications companies support packet-switching protocols. Previously, the most popular of these was a standard named X.25. It defines a set of networking protocols that describes how data terminal equipment, such as a host, communicates with data communications equipment (an X.25 switch). X.25 requires a synchronous data link and therefore special synchronous serial port hardware. It is possible to use X.25 with normal serial ports if you use a special device called a *Packet Assembler Disassembler* (PAD). The PAD is a standalone device that provides asynchronous serial ports and a synchronous serial port. It manages the X.25 protocol so that simple terminal devices can make and accept X.25 connections. X.25 is often used to carry other network protocols, such as TCP/IP. Since IP datagrams cannot simply be mapped onto X.25 (or vice versa), they are encapsulated in X.25 packets and sent over the network. There is an implementation of the X.25 protocol available for Linux, but it will not be discussed in depth here.

A protocol commonly used by telecommunications companies is called *Frame Relay*. The Frame Relay protocol shares a number of technical features with the X.25 protocol, but is much more like the IP protocol in behavior. Like X.25, Frame Relay requires special synchronous serial hardware. Because of their similarities, many cards support both of these protocols. An alternative is available that requires no

* The Ethernet FAQ at *http://www.faqs.org/faqs/LANs/ethernet-faq/*talks about this issue, and a wealth of detailed historical and technical information is available at Charles Spurgeon's Ethernet web site at *http://www.ethermanage.com/ethernet/ethernet.htm/*.

special internal hardware, again relying on an external device called a Frame Relay Access Device (FRAD) to manage the encapsulation of Ethernet packets into Frame Relay packets for transmission across a network. Frame Relay is ideal for carrying TCP/IP between sites. Linux provides drivers that support some types of internal Frame Relay devices.

If you need higher-speed networking that can carry many different types of data, such as digitized voice and video, alongside your usual data, *Asynchronous Transfer Mode* (ATM) is probably what you'll be interested in. ATM is a new network technology that has been specifically designed to provide a manageable, high-speed, low-latency means of carrying data and control over the Quality of Service (QoS). Many telecommunications companies are deploying ATM network infrastructure because it allows the convergence of a number of different network services into one platform, in the hope of achieving savings in management and support costs. ATM is often used to carry TCP/IP. The *Networking HOWTO* offers information on the Linux support available for ATM.

Frequently, radio amateurs use their radio equipment to network their computers; this is commonly called *packet radio*. One of the protocols used by amateur radio operators is called AX.25 and is loosely derived from X.25. Amateur radio operators use the AX.25 protocol to carry TCP/IP and other protocols, too. AX.25, like X.25, requires serial hardware capable of synchronous operation, or an external device called a *Terminal Node Controller* to convert packets transmitted via an asynchronous serial link into packets transmitted synchronously. There are a variety of different sorts of interface cards available to support packet radio operation; these cards are generally referred to as being "Z8530 SCC based," named after the most popular type of communications controller used in the designs. Two of the other protocols that are commonly carried by AX.25 are the NetRom and Rose protocols, which are network layer protocols. Since these protocols run over AX.25, they have the same hardware requirements. Linux supports a fully featured implementation of the AX.25, NetRom, and Rose protocols. The *AX25 HOWTO* is a good source of information on the Linux implementation of these protocols.

Other types of Internet access involve dialing up a central system over slow but cheap serial lines (telephone, ISDN, and so on). These require yet another protocol for transmission of packets, such as SLIP or PPP, which will be described later.

The Internet Protocol

Of course, you wouldn't want your networking to be limited to one Ethernet or one point-to-point data link. Ideally, you would want to be able to communicate with a host computer regardless of what type of physical network it is connected to. For example, in larger installations such as Groucho Marx University, you usually have a number of separate networks that have to be connected in some way. At GMU, the math department runs two Ethernets: one with fast machines for professors and graduates, and another with slow machines for students.

This connection is handled by a dedicated host called a *gateway* that handles incoming and outgoing packets by copying them between the two Ethernets and the FDDI fiber optic cable. For example, if you are at the math department and want to access **quark** on the physics department's LAN from your Linux box, the networking software will not send packets to **quark** directly because it is not on the same Ethernet. Therefore, it has to rely on the gateway to act as a forwarder. The gateway (named **sophus**) then forwards these packets to its peer gateway **niels** at the physics department, using the backbone network, with **niels** delivering it to the destination machine. Data flow between **erdos** and **quark** is shown in Figure 1-1.

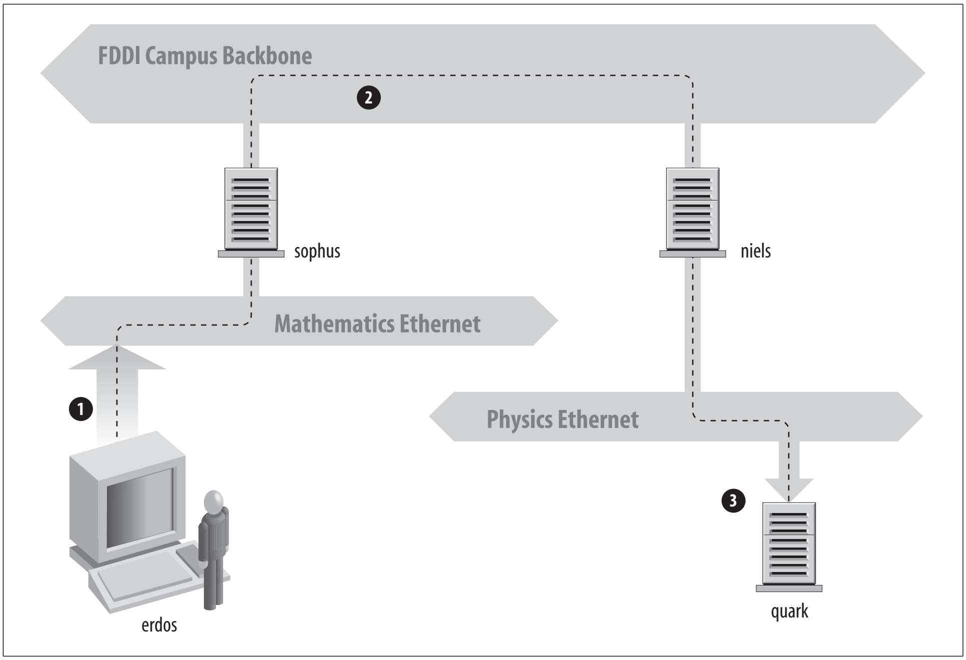


Figure 1-1. The three steps of sending a datagram from erdos to quark

This scheme of directing data to a remote host is called *routing*, and packets are often referred to as datagrams in this context. To facilitate things, datagram exchange is governed by a single protocol that is independent of the hardware used: IP, or *Internet Protocol*. In Chapter 2, we will cover IP and the issues of routing in greater detail.

The main benefit of IP is that it turns physically dissimilar networks into one apparently homogeneous network. This is called internetworking, and the resulting "meta-network" is called an *internet*. Note the subtle difference here between *an* internet and *the* Internet. The latter is the official name of one particular global internet.

Of course, IP also requires a hardware-independent addressing scheme. This is achieved by assigning each host a unique 32-bit number called the *IP address*. An IP address is usually written as four decimal numbers, one for each 8-bit portion, separated by dots. For example, **quark** might have an IP address of **0x954C0C04**, which would be written as **149.76.12.4**. This format is also called *dotted decimal notation* and sometimes *dotted quad notation*. It is increasingly going under the name IPv4 (for Internet Protocol, Version 4) because a new standard called IPv6 offers much more flexible addressing, as well as other modern features. It will be at least a year after the release of this edition before IPv6 is in use.

You will notice that we now have three different types of addresses: first there is the host's name, like **quark**, then there is an IP address, and finally, there is a hardware address, such as the 6-byte Ethernet address. All these addresses somehow have to match so that when you type *ssh quark*, the networking software can be given **quark**'s IP address; and when IP delivers any data to the physics department's Ethernet, it somehow has to find out what Ethernet address corresponds to the IP address.

We will deal with these situations in Chapter 2. For now, it's enough to remember that these steps of finding addresses are called *hostname resolution*, for mapping hostnames onto IP addresses, and *address resolution*, for mapping the latter to hardware addresses.

IP over Serial Lines

On serial lines, a "de facto" standard exists known as *Serial Line IP* (SLIP). A modification of SLIP known as *Compressed SLIP* (CSLIP), performs compression of IP headers to make better use of the relatively low bandwidth provided by most serial links. Another serial protocol is *Point-to-Point Protocol* (PPP). PPP is more modern than SLIP and includes a number of features that make it more attractive. Its main advantage over SLIP is that it isn't limited to transporting IP datagrams, but is designed to allow just about any protocol to be carried across it. This book discusses PPP in Chapter 6.

The Transmission Control Protocol

Sending datagrams from one host to another is not the whole story. If you log in to **quark**, you want to have a reliable connection between your *ssh* process on **erdos** and the shell process on **quark**. Thus, the information sent to and fro must be split into packets by the sender and reassembled into a character stream by the receiver. Trivial as it seems, this involves a number of complicated tasks.

A very important thing to know about IP is that, by intent, it is not reliable. Assume that 10 people on your Ethernet started downloading the latest release of the Mozilla web browser source code from GMU's FTP server. The amount of traffic generated might be too much for the gateway to handle because it's too slow and it's tight on memory. Now if you happen to send a packet to **quark**, **sophus** might be out of buffer space for a moment and therefore unable to forward it. IP solves this problem by simply discarding it. The packet is irrevocably lost. It is therefore the responsibility of the communicating hosts to check the integrity and completeness of the data and retransmit it in case of error.

This process is performed by yet another protocol, *Transmission Control Protocol* (TCP), which builds a reliable service on top of IP. The essential property of TCP is that it uses IP to give you the illusion of a simple connection between the two processes on your host and the remote machine so that you don't have to care about how and along which route your data actually travels. A TCP connection works essentially like a two-way pipe that both processes may write to and read from. Think of it as a telephone conversation.

TCP identifies the end points of such a connection by the IP addresses of the two hosts involved and the number of a *port* on each host. Ports may be viewed as attachment points for network connections. If we are to strain the telephone example a little more, and you imagine that cities are like hosts, one might compare IP addresses to area codes (where numbers map to cities), and port numbers to local codes (where numbers map to individual people's telephones). An individual host may support many different services, each distinguished by its own port number.

In the *ssh* example, the client application (*ssh*) opens a port on **erdos** and connects to port 22 on **quark**, to which the sshd server is known to listen. This action establishes a TCP connection. Using this connection, *sshd* performs the authorization procedure and then spawns the shell. The shell's standard input and output are redirected to the TCP connection so that anything you type to *ssh* on your machine will be passed through the TCP stream and be given to the shell as standard input.

The User Datagram Protocol

Of course, TCP isn't the only user protocol in TCP/IP networking. Although suitable for applications like *ssh*, the overhead involved is prohibitive for applications like NFS, which instead uses a sibling protocol of TCP called *User Datagram Protocol* (UDP). Just like TCP, UDP allows an application to contact a service on a certain port of the remote machine, but it doesn't establish a connection for this. Instead, you use it to send single packets to the destination service—hence its name.

Assume that you want to request a small amount of data from a database server. It takes at least three datagrams to establish a TCP connection, another three to send and confirm a small amount of data each way, and another three to close the connection. UDP provides us with a means of using only two datagrams to achieve almost the same result. UDP is said to be connectionless, and it doesn't require us to establish and close a session. We simply put our data into a datagram and send it to the server; the server formulates its reply, puts the data into a datagram addressed back to us, and transmits it back. While this is both faster and more efficient than TCP for simple transactions, UDP was not designed to deal with datagram loss. It is up to the application, a nameserver, for example, to take care of this.

More on Ports

Ports may be viewed as attachment points for network connections. If an application wants to offer a certain service, it attaches itself to a port and waits for clients (this is also called *listening* on the port). A client who wants to use this service allocates a port on its local host and connects to the server's port on the remote host. The same port may be open on many different machines, but on each machine only one process can open a port at any one time.

An important property of ports is that once a connection has been established between the client and the server, another copy of the server may attach to the server port and listen for more clients. This property permits, for instance, several concurrent remote logins to the same host, all using the same port 513. TCP is able to tell these connections from one another because they all come from different ports or hosts. For example, if you log in twice to **quark** from **erdos**, the first *ssh* client may use the local port 6464, and the second one could use port 4235. Both, however, will connect to the same port 513 on **quark**. The two connections will be distinguished by use of the port numbers used at **erdos**.

This example shows the use of ports as rendezvous points, where a client contacts a specific port to obtain a specific service. In order for a client to know the proper port number, an agreement has to be reached between the administrators of both systems on the assignment of these numbers. For services that are widely used, such as *ssh*, these numbers have to be administered centrally. This is done by the Internet Engineering Task Force (IETF), which regularly releases an RFC titled *Assigned Numbers* (RFC-1700). It describes, among other things, the port numbers assigned to well-known services. Linux uses a file called */etc/services* that maps service names to numbers.

It is worth noting that, although both TCP and UDP connections rely on ports, these numbers do not conflict. This means that TCP port 22, for example, is different from UDP port 22.

The Socket Library

In Unix operating systems, the software performing all the tasks and protocols described above is usually part of the kernel, and so it is in Linux. The programming interface most common in the Unix world is the Berkeley Socket Library. Its name derives from a popular analogy that views ports as sockets and connecting to a port as plugging in. It provides the bind call to specify a remote host, a transport protocol, and a service that a program can connect or listen to (using connect, listen, and accept). The socket library is somewhat more general in that it provides not only a class of TCP/IP-based sockets (the *AF_INET* sockets), but also a class that handles connections local to the machine (the *AF_UNIX* class). Some implementations can also handle other classes, like the Xerox Networking System (XNS) protocol or X.25.

In Linux, the socket library is part of the standard *libc* C library. It supports the *AF_INET* and *AF_INET6* sockets for TCP/IP and *AF_UNIX* for Unix domain sockets. It also supports *AF_IPX* for Novell's network protocols, *AF_ X25* for the X.25 network protocol, *AF_ATMPVC* and *AF_ATMSVC* for the ATM network protocol and *AF_AX25*, *AF_NETROM*, and *AF_ ROSE* sockets for Amateur Radio protocol support. Other protocol families are being developed and will be added in time.

Linux Networking

As it is the result of a concerted effort of programmers around the world, Linux wouldn't have been possible without the global network. So it's not surprising that in the early stages of development, several people started to work on providing it with network capabilities. A UUCP implementation was running on Linux almost from the very beginning, and work on TCP/IP-based networking started around autumn 1992, when Ross Biro and others created what has now become known as Net-1.

After Ross quit active development in May 1993, Fred van Kempen began to work on a new implementation, rewriting major parts of the code. This project was known as Net-2. The first public release, Net-2d, was made in the summer of 1993 (as part of the 0.99.10 kernel), and has since been maintained and expanded by several people, most notably Alan Cox. Alan's original work was known as Net-2Debugged. After heavy debugging and numerous improvements to the code, he changed its name to Net-3 after Linux 1.0 was released. The Net-3 code was further developed for Linux 1.2 and Linux 2.0. The 2.2 and later kernels use the Net-4 version network support, which remains the standard official offering today.

The Net-4 Linux Network code offers a wide variety of device drivers and advanced features. Standard Net-4 protocols include SLIP and PPP (for sending network traffic over serial lines), PLIP (for parallel lines), IPX (for Novell compatible networks), Appletalk (for Apple networks) and AX.25, NetRom, and Rose (for amateur radio networks). Other standard Net-4 features include IP firewalling (discussed in Chapter 7), IP accounting (Chapter 8), and IP Masquerade (Chapter 9). IP tunneling in a couple of different flavors and advanced policy routing are supported. A very large variety of Ethernet devices are supported, in addition to support for some FDDI, Token Ring, Frame Relay, and ISDN, and ATM cards.

Additionally, there are a number of other features that greatly enhance the flexibility of Linux. These features include interoperability with the Microsoft Windows network environment, in a project called Samba, discussed in Chapter 16, and an implementation of the Novell NCP (NetWare Core Protocol).*

Different Streaks of Development

There have been, at various times, varying network development efforts active for Linux.

Fred continued development after Net-2Debugged was made the official network implementation. This development led to the Net-2e, which featured a much revised design of the networking layer. Fred was working toward a standardized Device Driver Interface (DDI), but the Net-2e work has ended now.

Yet another implementation of TCP/IP networking came from Matthias Urlichs, who wrote an ISDN driver for Linux and FreeBSD. For this driver, he integrated some of the BSD networking code in the Linux kernel. That project, too, is no longer being worked on.

There has been a lot of rapid change in the Linux kernel networking implementation, and change is still the watchword as development continues. Sometimes this means that changes also have to occur in other software, such as the network configuration tools. While this is no longer as large a problem as it once was, you may still find that upgrading your kernel to a later version means that you must upgrade your network configuration tools, too. Fortunately, with the large number of Linux distributions available today, this is a quite simple task.

The Net-4 network implementation is now a standard and is in use at a very large number of sites around the world. Much work has been done on improving the performance of the Net-4 implementation, and it now competes with the best implementations available for the same hardware platforms. Linux is proliferating in the Internet Service Provider environment, and is often used to build cheap and reliable World Wide Web servers, mail servers, and news servers for these sorts of organizations. There is now sufficient development interest in Linux that it is managing to keep abreast of networking technology as it changes, and current releases of the Linux kernel offer the next generation of the IP protocol, IPv6, as a standard offering, which will be discussed at greater detail in Chapter 13.

Where to Get the Code

It seems odd now to remember that in the early days of the Linux network code development, the standard kernel required a huge patch kit to add the networking support to it. Today, network development occurs as part of the mainstream Linux kernel development process. The latest stable Linux kernels can be found on *ftp://ftp.*

* NCP is the protocol on which Novell file and print services are based.

kernel.org in */pub/linux/kernel/v2.x/*, where *x* is an even number. The latest experimental Linux kernels can be found on *ftp://ftp.kernel.org* in */pub/linux/kernel/v2.y/*, where y is an odd number. The kernel.org distributions can also be accessed via HTTP at *http://www.kernel.org*. There are Linux kernel source mirrors all over the world.

Maintaining Your System

Throughout this book, we will mainly deal with installation and configuration issues. Administration is, however, much more than that—after setting up a service, you have to keep it running, too. For most services, only a little attendance will be necessary, while some, such as mail, require that you perform routine tasks to keep your system up to date. We will discuss these tasks in later chapters.

The absolute minimum in maintenance is to check system and per-application logfiles regularly for error conditions and unusual events. Often, you will want to do this by writing a couple of administrative shell scripts and periodically running them from *cron*. The source distributions of some major applications contain such scripts. You only have to tailor them to suit your needs and preferences.

The output from any of your *cron* jobs should be mailed to an administrative account. By default, many applications will send error reports, usage statistics, or logfile summaries to the *root* account. This makes sense only if you log in as *root* frequently; a much better idea is to forward *root*'s mail to your personal account by setting up a mail alias as described in Chapters 11 and 12.

However carefully you have configured your site, Murphy's Law guarantees that some problem *will* surface eventually. Therefore, maintaining a system also means being available for complaints. Usually, people expect that the system administrator can at least be reached via email as *root*, but there are also other addresses that are commonly used to reach the person responsible for a specific aspect of maintenence. For instance, complaints about a malfunctioning mail configuration will usually be addressed to *postmaster*, and problems with the news system may be reported to *newsmaster* or *usenet*. Mail to *hostmaster* should be redirected to the person in charge of the host's basic network services, and the DNS name service if you run a nameserver.

System Security

Another very important aspect of system administration in a network environment is protecting your system and users from intruders. Carelessly managed systems offer malicious people many targets. Attacks range from password guessing to Ethernet snooping, and the damage caused may range from faked mail messages to data loss or violation of your users' privacy. We will mention some particular problems when