LINUX NETWORK ADMINISTRATOR'S GUIDE

by Olaf Kirch and Terry Dawson
# A TOUR OF LINUX NETWORK DEVICES

- Ethernet Interfaces
- The Loopback Interface
- Ethernet Interfaces
- Routing Through a Gateway
- Configuring a Gateway
- The PLIP Interface
- The SLIP and PPP Interfaces
- The Dummy Interface
- IP Alias

# CHAPTER 4 - CONFIGURING THE SERIAL HARDWARE

- Communications Software for Modem Links
- Introduction to Serial Devices
- Accessing Serial Devices
- The Serial Device Special Files
- Serial Hardware
- Using the Configuration Utilities
- The setserial Command
- The stty Command
- Serial Devices and the Login: Prompt
- Configuring the mgetty Daemon

# CHAPTER 5 - CONFIGURING TCP/IP NETWORKING

- Mounting the /proc Filesystem
- Installing the Binaries
- Setting the Hostname
- Assigning IP Addresses
- Creating Subnets
- Writing Hosts and Networks Files
- Interface Configuration for IP
- The Loopback Interface
- Ethernet Interfaces
- Routing Through a Gateway
- Configuring a Gateway
- The PLIP Interface
- The SLIP and PPP Interfaces
- The Dummy Interface
- IP Alias

# CHAPTER 6 - NAME SERVICE AND RESOLVER CONFIGURATION

- The Resolver Library
- The host.conf File
- The nsswitch.conf File
- Configuring Name Server Lookups Using resolv.conf
- Resolver Robustness

# HOW DNS WORKS

- Name Lookups with DNS
- Types of Name Servers
- The DNS Database
- Reverse Lookups

# RUNNING NAMED

- The named.boot File
- The BIND 8 host.conf File
- The DNS Database Files
- Caching-only named Configuration
- Writing the Master Files
- Verifying the Name Server Setup
- Other Useful Tools
### Chapter 17 - Electronic Mail

- **What Is a Mail Message?** .......................................................... 233
- **How Is Mail Delivered?** ............................................................ 235
- **Email Addresses** .................................................................... 236
- **RFC-822** ................................................................................ 236
- **Obsolet e Mail Formats** ............................................................ 236
- **Mixing Different Mail Formats** ............................................... 237
- **How Does Mail Routing Work?** ............................................. 237
- **Mail Routing on the Internet** .................................................. 237

### Chapter 16 - Managing Taylor UUCP

- **UUCP Transfers and Remote Execution** ................................. 213
  - The Inner Workings of uucico ...................................................... 213
  - uucico Command-Line Options ................................................ 214
- **UUCP Configuration Files** ...................................................... 215
  - A Gentle Introduction to Taylor UUCP .................................... 215
  - What UUCP Needs to Know .................................................... 217
  - Site Naming ............................................................................. 217
  - Taylor Configuration Files ..................................................... 218
  - General Configuration Options Using the config File ................ 218
  - How to Tell UUCP About Other Systems Using the sys File .... 218
  - Identifying Available Devices Through the port File ............... 222
  - How to Dial a Number Using the dial File ............................... 223
  - UUCP Over TCP ..................................................................... 223
  - Using a Direct Connection ...................................................... 224
- **Controlling Access to UUCP Features** .................................... 224
  - Command Execution ................................................................. 224
  - File Transfers .......................................................................... 225
  - Forwarding ................................................................................. 225
- **Setting Up Your System for Dialing In** ................................... 226
  - Providing UUCP Accounts ....................................................... 226
  - Protecting Yourself Against Swindlers ................................... 227
  - Be Paranoid: Call Sequence Checks ....................................... 227
  - Anonymous UUCP ................................................................... 228
- **UUCP Low-Level Protocols** .................................................. 228
  - Protocol Overview .................................................................... 228
  - Tuning the Transmission Protocol .......................................... 229
  - Selecting Specific Protocols .................................................... 229
- **Troubleshooting** .................................................................... 230
  - uucico Keeps Saying "Wrong Time to Call" ............................. 230
  - uucico Complains That the Site Is Already Locked ................. 230
  - You Can Connect to the Remote Site, but the Chat Script Fails 230
  - Your Modem Does Not Dial .................................................... 231
  - Your Modem Tries to Dial but Doesn’t Get Out ....................... 231
  - Login Succeeds, but the Handshake Fails ............................... 231
- **Log Files and Debugging** ....................................................... 231

### CHAPTER 17 - ELECTRONIC MAIL

- **Static IPX Routing Using the ipx route Command** .................. 202
- **Internal IPX Networks and Routing** .................................... 203
- **Mounting a Remote NetWare Volume** .................................. 205
  - A Simple ncpmount Example ................................................... 205
  - The ncpmount Command in Detail ......................................... 205
  - Hiding Your NetWare Login Password ................................... 207
  - A More Complex ncpmount Example ................................... 207
- **Exploring Some of the Other IPX Tools** ............................... 207
  - Server List .............................................................................. 207
  - Send Messages to NetWare Users ......................................... 208
  - Browsing and Manipulating Bindery Data ............................... 208
- **Printing to a NetWare Print Queue** ...................................... 209
  - Using nprint with the Line Printer Daemon ............................. 210
  - Managing Print Queues .......................................................... 211
- **NetWare Server Emulation** .................................................. 211

### Chapter 16 - Managing Taylor UUCP

- **UUCP Transfers and Remote Execution** ................................. 213
  - The Inner Workings of uucico ...................................................... 213
  - uucico Command-Line Options ................................................ 214
- **UUCP Configuration Files** ...................................................... 215
  - A Gentle Introduction to Taylor UUCP .................................... 215
  - What UUCP Needs to Know .................................................... 217
  - Site Naming ............................................................................. 217
  - Taylor Configuration Files ..................................................... 218
  - General Configuration Options Using the config File ................ 218
  - How to Tell UUCP About Other Systems Using the sys File .... 218
  - Identifying Available Devices Through the port File ............... 222
  - How to Dial a Number Using the dial File ............................... 223
  - UUCP Over TCP ..................................................................... 223
  - Using a Direct Connection ...................................................... 224
- **Controlling Access to UUCP Features** .................................... 224
  - Command Execution ................................................................. 224
  - File Transfers .......................................................................... 225
  - Forwarding ................................................................................. 225
- **Setting Up Your System for Dialing In** ................................... 226
  - Providing UUCP Accounts ....................................................... 226
  - Protecting Yourself Against Swindlers ................................... 227
  - Be Paranoid: Call Sequence Checks ....................................... 227
  - Anonymous UUCP ................................................................... 228
- **UUCP Low-Level Protocols** .................................................. 228
  - Protocol Overview .................................................................... 228
  - Tuning the Transmission Protocol .......................................... 229
  - Selecting Specific Protocols .................................................... 229
- **Troubleshooting** .................................................................... 230
  - uucico Keeps Saying "Wrong Time to Call" ............................. 230
  - uucico Complains That the Site Is Already Locked ................. 230
  - You Can Connect to the Remote Site, but the Chat Script Fails 230
  - Your Modem Does Not Dial .................................................... 231
  - Your Modem Tries to Dial but Doesn’t Get Out ....................... 231
  - Login Succeeds, but the Handshake Fails ............................... 231
- **Log Files and Debugging** ....................................................... 231
CHAPTER 22 - NNTP AND THE NNTPD DAEMON ................................................................. 292

THE NNTP PROTOCOL ........................................................................................................... 293

Connecting to the News Server ............................................................................................ 293
Pushing a News Article onto a Server ................................................................................... 293
Changing to NNRP Reader Mode .......................................................................................... 294
Listing Available Groups ....................................................................................................... 295
Listing Active Groups ............................................................................................................ 295
Posting an Article .................................................................................................................. 295
Listing New Articles ............................................................................................................. 296
Selecting a Group on Which to Operate ............................................................................... 296
Listing Articles in a Group ..................................................................................................... 296
Retrieving an Article Header Only ....................................................................................... 296
Retrieving an Article Body Only .......................................................................................... 297
Reading an Article from a Group .......................................................................................... 297

INSTALLING THE NNTP SERVER ....................................................................................... 298

RESTRICTING NNTP ACCESS .............................................................................................. 298
NNTP AUTHORIZATION ......................................................................................................... 299
NNTPD INTERACTION WITH C NEWS .................................................................................. 299

CHAPTER 23 - INTERNET NEWS ........................................................................................ 301

SOME INN INTERNALS ............................................................................................................ 301
NEWSREADERS AND INN ....................................................................................................... 303
INSTALLING INN .................................................................................................................... 303
CONFIGURING INN: THE BASIC SETUP .............................................................................. 303

INN CONFIGURATION FILES ................................................................................................ 304
Global Parameters .................................................................................................................. 304
Configuring Newsgroups ........................................................................................................ 305
Configuring Newsfeeds ......................................................................................................... 306
Controlling Newsreader Access ........................................................................................... 309
Expiring News Articles ......................................................................................................... 311
Handling Control Messages .................................................................................................. 312

RUNNING INN ....................................................................................................................... 314

MANAGING INN: THE CTLINND COMMAND .................................................................. 315
Add a New Group ................................................................................................................... 315
Change a Group ..................................................................................................................... 315
Remove a Group .................................................................................................................... 316
Renumber a Group ............................................................................................................... 316
Allow/Disallow Newsreaders ................................................................................................. 316
Reject Newsfeed Connections .............................................................................................. 316
Allow Newsfeed Connections .............................................................................................. 317
Disable News Server ............................................................................................................ 317
Restart News Server ............................................................................................................. 317
Display Status of a Newsfeed ............................................................................................... 317
Drop a Newsfeed .................................................................................................................. 317
Begin a Newsfeed .................................................................................................................. 318
Cancel an Article ................................................................................................................... 318

CHAPTER 24 - NEWSREADER CONFIGURATION .............................................................. 319

TIN CONFIGURATION .......................................................................................................... 319
TRN CONFIGURATION .......................................................................................................... 320
NN CONFIGURATION ........................................................................................................... 320

APPENDIX A .......................................................................................................................... 322
Preface

The Internet is now a household term in many countries. With otherwise serious people beginning to joyride along the Information Superhighway, computer networking seems to be moving toward the status of TV sets and microwave ovens. The Internet has unusually high media coverage, and social science majors are descending on Usenet newsgroups, online virtual reality environments, and the Web to conduct research on the new "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 world-wide networks has, however, made joining the global village a perfectly reasonable option for even small non-profit organizations of private computer users. Setting up an Internet host with mail and news capabilities offering dialup and ISDN access has become affordable, and the advent of DSL (Digital Subscriber Line) and Cable Modem technologies will doubtlessly continue this trend.

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 386BSD, FreeBSD, and Linux.

Linux is a freely distributable Unix clone for personal computers. It currently runs on a variety of machines that includes the Intel family of processors, but also Motorola 680x0 machines, such as the Commodore Amiga and Apple Macintosh; Sun SPARC and Ultra-SPARC machines; Compaq Alpha; MIPS, PowerPCs, such as the new generation of Apple Macintosh; and StrongARM, like the rebel.com Netwinder and 3Com Palm machines. Linux has been ported to some relatively obscure platforms, like 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 SLIP, PPP, firewalls, a full IPX implementation, and many features and some 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. We've found that beginners to Linux networking, even those with no prior exposure to Unix-like operating systems, have found 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 (with the possible exception of some of the truly Linux-specific features, such as the new Linux firewall interface, which is not well documented elsewhere) in greater depth. We've provided a bibliography for you to use 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 manual pages on topics ranging from installation to kernel programming. The LDP works include:

* **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, configuration of major software packages, and installing and upgrading software.

* **Linux System Administration 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.

More manuals are in development. For more information about the LDP you should consult their World Wide Web server at [http://www.linuxdoc.org/](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 installation and configuration of the X Window System software, or how to write in assembly language programming under Linux. These are generally located in the HOWTO subdirectory of the FTP sites listed later, or they are available on the World Wide Web at one of the many Linux Documentation Project mirror sites. See the Bibliography at the end of this book, or 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.

The bibliography of this book includes references to the HOWTO documents that are related to Linux networking.

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 FTP

If you have access to anonymous FTP, you can obtain all Linux documentation listed above from various sites, including metalab.unc.edu:/pub/Linux/docs and txs-11.mit.edu:/pub/linux/docs.

These sites are mirrored by a number of sites around the world.

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.linuxdoc.org/.

The Open Source Writers Guild (OSWG) is a project that has a scope that extends beyond Linux. The OSWG, like this book, is committed to advocating and facilitating the production of OpenSource documentation. The OSWG home site is at http://www.oswg.org:8080/oswg.

Both of these sites contain hypertext (and other) versions of many Linux related documents.

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/
P.O. Box 55549 Seattle, WA 98155-0549
1-206-782-7733
1-206-782-7191 (FAX)
sales@ssc.com

and:

Linux Systems Labs
http://www.lsl.com/
18300 Tara Drive
Clinton Township, MI 48036
1-810-987-8807
1-810-987-3562 (FAX)
sales@lsl.com

Both companies sell compendiums of Linux HOWTO documents and other Linux documentation in printed and bound form.

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

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

Learning Debian GNU/Linux
Learning Red Hat Linux
More basic than Running Linux, these books contain popular distributions on CD-ROM and offer robust directions for setting them up and using them.

Linux in a Nutshell
Another in the successful "in a Nutshell" series, this book focuses on providing a broad reference text for Linux.
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 S.S.C. Incorporated, for which details were listed previously. You can also find the magazine on the World Wide Web at [http://www.linuxjournal.com/](http://www.linuxjournal.com/).

*Linux Magazine* is a newer, independent publication. The home web site for the magazine is [http://www.linuxmagazine.com/](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. Submissions may be mailed to linux-announce@news.ornl.gov.

- **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 is a large number of specialist Linux mailing lists on which you will find many people willing to help with questions you might have.

The best-known of these are the lists hosted by Rutgers University. You may subscribe to these lists by sending an email message formatted as follows:

To: majordomo@vger.rutgers.edu
Subject: anything at all
Body:

    subscribe listname

Some of the available lists related to Linux networking are:

- **linux-net**
  - Discussion relating to Linux networking

- **linux-ppp**
  - Discussion relating to the Linux PPP implementation
Online Linux Support

There are many ways of obtaining help online, where volunteers from around the world offer expertise and services to assist users with questions and problems.

The OpenProjects IRC Network is an IRC network devoted entirely to Open Projects -- Open Source and Open Hardware alike. Some of its channels are designed to provide online Linux support services. IRC stands for Internet Relay Chat, and is a network service that allows you to talk interactively on the Internet to other users. IRC networks support multiple channels on which groups of people talk. Whatever you type in a channel is seen by all other users of that channel.

There are a number of active channels on the OpenProjects IRC network where you will find users 24 hours a day, 7 days a week who are willing and able to help you solve any Linux problems you may have, or just chat. You can use this service by installing an IRC client like **irc-II**, connecting to servername `irc.openprojects.org:6667`, and joining the `#linpeople` channel.

Linux User Groups

Many Linux User Groups around the world offer direct support to users. Many Linux User Groups engage in activities such as installation days, talks and seminars, demonstration nights, and other completely social events. Linux User Groups are a great way of meeting other Linux users in your area. There are a number of published lists of Linux User Groups. Some of the better-known ones are:

Groups of Linux Users Everywhere

- [Groups of Linux Users Everywhere](http://www.ssc.com/glue/groups/)
- [LUG list project](http://www.nlrg.nl/lugww/)
- [LUG registry](http://www.linux.org/users/)

Obtaining Linux

There is no single distribution of the Linux software; instead, there are many distributions, such as Debian, RedHat, Caldera, Corel, SuSE, 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:

Caldera

- [Caldera](http://www.caldera.com/ftp://ftp.caldera.com/)

Corel


Debian

- [Debian](http://www.debian.org/ftp://ftp.debian.org/)

RedHat

- [RedHat](http://www.redhat.com/ftp://ftp.redhat.com/)

Slackware

- [Slackware](http://www.slackware.com/ftp://ftp.slackware.com/)
Many of the popular general FTP archive sites also mirror various Linux distributions. The best-known of these sites are:

- metalab.unc.edu:/pub/Linux/distributions/
- ftp.funet.fi:/pub/Linux/mirrors/
- tsx-11.mit.edu:/pub/linux/distributions/
- mirror.aarnet.edu.au:/pub/linux/distributions/

Many of the modern distributions can be installed directly from the Internet. There is a lot of software to download for a typical installation, though, so you'd probably want to do this only if you have a high-speed, permanent network connection, or if you just need to update an existing installation.

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 you settle on your favorite one.

**File System 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 file system 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 the Linux documentation directory of all major Linux FTP sites and their mirrors, or at its home site at [http://www.pathname.com/fhs/](http://www.pathname.com/fhs/). Daniel Quinlan, the coordinator of the FHS group, can be reached at quinlan@transmeta.com.

**Standard Linux Base**

The vast number of different Linux distributions, while providing lots of healthy choice 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...
against the standard base platform, the application will work, 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/](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 joined the Linux Documentation Project 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 Shahan contributed the original sendmail mail chapter, which was completely replaced in this edition because of a new interface to the sendmail configuration.

The version of the guide that you are reading now is a revision and update prompted by O'Reilly & Associates and undertaken by Terry Dawson. Terry has been an amateur radio operator for over 20 years and has worked in the telecommunications industry for over 15 of those. He was co-author of the original NET-FAQ, and has since authored and maintained various networking-related HOWTO documents. Terry has always been an enthusiastic supporter of the Network Administrators Guide project, and added a few new chapters to this version describing features of Linux networking that have been developed since the first edition, plus a bunch of changes to bring the rest of the book up to date.

The exim chapter was contributed by Philip Hazel, who is a lead developer and maintainer of the package.

The book is organized roughly along the sequence of steps 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 rlogin and friends, the Network File System, and the Network Information System. This is followed by a chapter on how to set up your machine as a UUCP node. Most of the remaining sections is dedicated to two major applications that run on top of TCP/IP and UUCP: electronic mail and news. A special chapter has been devoted to the IPX protocol and the NCP filesystem, because these are used in many corporate environments where Linux is finding a home.

The email part features an introduction to the more intimate parts of mail transport and routing, and the myriad of addressing schemes you may be confronted with. It describes the configuration and management of exim, a mail transport agent ideal for use in most situations not requiring UUCP, and sendmail, which is for people who have to do more complicated routing involving UUCP.

The news part gives you an overview of how Usenet news works. It covers INN and C News, the two most widely used news transport software packages at the moment, and the use of NNTP to provide newsreading access to a local network. The book closes with a chapter on the care and feeding of the most popular newsreaders on Linux.

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 the section in 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. If you have the opportunity, you should also try to get the latest kernel and network release from one of the Linux FTP sites or a BBS near you. 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."

---

2 Terry Dawson can be reached at terry@linux.org.au

3 Philip Hazel can be reached at phil@cus.cam.ac.uk
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, never having imagined that it would become this successful. 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 Linux Documentation Project 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 & Associates does take a risk in publishing the Networking Guide, and it seems to have paid off for them (they've asked us to do it again). 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 is providing to 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.

Why are they publishing it? They see it as their kind of book. It's what they'd 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 Linux Documentation Project, 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 one 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 copyeditors, who got the book into the shape 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, and even then it is more unlikely that you'll do it for the price. 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. We examine IP a little more closely in Chapter 2, Issues of TCP/IP Networking, before getting 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.

4 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.
Chapter 3, *Configuring the Networking Hardware*, deals with very basic configuration issues, such as building a kernel and setting up your Ethernet card. The configuration of your serial ports is covered separately in Chapter 4, *Configuring the Serial Hardware*, because the discussion does not apply to TCP/IP networking only, but is also relevant for UUCP.

Chapter 5, *Configuring TCP/IP Networking*, helps you set up your machine for TCP/IP networking. It contains installation hints for standalone hosts with loopback enabled only, and hosts connected to an Ethernet. It also introduces you to a few useful tools you can use to test and debug your setup. Chapter 6, *Name Service and Resolver Configuration*, discusses how to configure hostname resolution and explains how to set up a name server.

Chapter 7, *Serial Line IP*, explains how to establish SLIP connections and gives a detailed reference for *dip*, a tool that allows you to automate most of the necessary steps. Chapter 8, *The Point-to-Point Protocol*, covers PPP and *pppd*, the PPP daemon.

Chapter 9, *TCP/IP Firewall*, extends our discussion on network security and describes the Linux TCP/IP firewall and its configuration tools: *ipfwadm, ipchains, and iptables*. IP firewalling provides a means of controlling who can access your network and hosts very precisely.

Chapter 10, *IP Accounting*, explains how to configure IP Accounting in Linux so you can keep track of how much traffic is going where and who is generating it.

Chapter 11, *IP Masquerade and Network Address Translation*, covers a feature of the Linux networking software called IP masquerade, 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 12, *Important Network Features*, gives a short introduction to setting up some of the most important network applications, such as *rlogin, ssh*, etc. 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 13, *The Network Information System*, and Chapter 14, *The Network File System*, discuss NIS and NFS. NIS is a tool used to distribute administrative information, such as user passwords in a local area network. NFS allows you to share filesystems between several hosts in your network.

In Chapter 15, *IPX and the NCP Filesystem*, we discuss the IPX protocol and the NCP filesystem. These allow Linux to be integrated into a Novell NetWare environment, sharing files and printers with non-Linux machines.

Chapter 16, *Managing Taylor UUCP*, gives you an extensive introduction to the administration of Taylor UUCP, a free implementation of the UUCP suite.

The remainder of the book is taken up by a detailed tour of electronic mail and Usenet news. Chapter 17, *Electronic Mail*, introduces you to the central concepts of electronic mail, like what a mail address looks like, and how the mail handling system manages to get your message to the recipient.

Chapter 18, *Sendmail*, and Chapter 19, *Getting Exim Up and Running*, cover the configuration of *sendmail* and *exim*, two mail transport agents you can use for Linux. This book explains both of them, because *exim* is easier to install for the beginner, while *sendmail* provides support for UUCP.

Chapter 20, *Netnews*, through Chapter 23, *Internet News*, explain the way news is managed in Usenet and how you install and use C News, *nntpd*, and INN: three popular software packages for managing Usenet news. After the brief introduction in Chapter 20, you can read Chapter 21, *C News*, if you want to transfer news using C News, a traditional service generally used with UUCP. The following chapters discuss more modern alternatives to C News that use the Internet-based protocol NNTP (Network News Transfer Protocol). Chapter 22, *NNTP and the nntpd Daemon* covers how to set up a simple NNTP daemon, *nntpd*, to provide news reading access for a local network, while Chapter 23 describes a more robust server for more extensive NetNews transfers, the InterNet News daemon (INN). And finally, Chapter 24, *Newsreader Configuration*, shows you how to configure and maintain various newsreaders.

**Conventions Used in This Book**
All examples presented in this book assume you are using a 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, command-line options, email addresses and pathnames, URLs, and for emphasizing new terms.

**Boldface**
Used for machine names, hostnames, site names, usernames and IDs, 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.

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

### Submitting Changes

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 & Associates, Inc.
101 Morris Street
Sebastopol, CA 95472
1-800-998-9938 (in the U.S. or Canada)
1-707-829-0515 (international or local)
1-707-829-0104 (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:


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

[http://www.oreilly.com](http://www.oreilly.com)

### Acknowledgments

This edition of the Networking Guide owes almost everything to the outstanding work of Olaf and Vince. 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 proof-read it and help iron out many mistakes, both technical and grammatical (never knew that there was such a thing as a dangling participle). Phil Hughes, John Macdonald, and Erik Ratcliffe all provided very helpful (and on the whole, quite consistent) feedback on the content of the book.

We also owe many thanks to the people at O'Reilly we've had the pleasure to work with: Sarah Jane Shangraw, who got the book into the shape you can see now; Maureen Dempsey, who copyedited the text; Rob Romano, Rhon Porter, and Chris Reilley, who created all the figures; Hanna Dyer, who designed the cover; Alicia Cech, David Futato, and Jennifer Niedherst for the internal layout; Lars Kaufman for suggesting old woodcuts as a visual theme; Judy Hoer for the index; and finally, Tim O'Reilly for the courage to take up such a project.

We are greatly indebted to Andres Sepúlveda, Wolfgang Michaelis, Michael K. Johnson, and all developers who spared the time to check the information provided in the Networking Guide. Phil Hughes, John MacDonald, and Eric Ratcliffe contributed invaluable comments on the second edition. We also wish to thank all those who read the first version of the Networking Guide and sent corrections and suggestions. You can find a hopefully complete list of contributors in the file Thanks in the online distribution. Finally, this 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.

Olaf would also like to thank the following groups and companies that printed the first edition of the Networking Guide and have donated money either to him or to the Linux Documentation Project as a whole: Linux Support Team, Erlangen, Germany; S.u.S.E. GmbH, Fuerth, Germany; and Linux System Labs, Inc., Clinton Twp., United States, RedHat Software, North Carolina, United States.

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."

The Hall of Fame

Besides those we have already mentioned, a large number of people have contributed to the Networking Guide, by reviewing it and sending us corrections and suggestions. We are very grateful.

Here is a list of those whose contributions left a trace in our mail folders.

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 will describe three types of networks in this guide. We will focus on TCP/IP most heavily because it is the most popular protocol suite in use on both Local Area Networks (LANs) and Wide Area Networks (WANs), such as the Internet. We will also take a look at UUCP and IPX. UUCP was once commonly used to transport news and mail messages over dialup telephone connections. It is less common today, but is still useful in a variety of situations. The IPX protocol is used most commonly in the Novell NetWare environment and we'll describe how to use it to connect your Linux machine into a Novell network. Each of these protocols are networking protocols and are used to carry data between host computers. We'll discuss how they are used and introduce you to their underlying principles.

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. Small agglomerations of hosts are also called sites.

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.

TCP/IP Networks

Modern networking applications require a sophisticated approach to carrying 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 uses 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 talking 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.

5 The original spirit of which (see above) still shows on some occasions in Europe
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 grown to a point at which it could almost be considered a mainstream consumer technology. It is difficult to read a newspaper or magazine now without seeing reference to the Internet; almost everyone can now use it.

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 Fredland, 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:

```
$ rlogin quark.physics
```

Welcome to the Physics Department at GMU
(ttyq2) login:

At the prompt, you enter your login name, say andres, and 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 TCP/IP provides: remote login.

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 have to set the `DISPLAY` environment variable:

```
$ DISPLAY=erdos.maths:0.0
```

```
$ export DISPLAY
```

If you now start your application, it will contact your X server instead of quark's, and display all its windows on your screen. Of course, this requires that you have X11 running 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.

Another very important application in TCP/IP networks is NFS, which stands for Network File System. It is another form of making the network transparent, because it basically allows you to treat directory hierarchies from other hosts as if they were local file systems and look like any other directories on your host. For example, all users' home directories can be kept on a central server machine from which all other hosts on the LAN mount them. The effect is that users can log in to any machine and find themselves in the same home directory. Similarly, it is possible to share large amounts of data (such as a database, documentation or application programs) among many hosts by maintaining one copy of the data on a server and allowing other hosts to access it. We will come back to NFS in Chapter 14, *The Network File System*.

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.

---

6 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.
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, or even 1,000 Megabits per second, accounts for much of its popularity.

Ethernets come in three flavors: thick, thin, and twisted pair. Thin and thick Ethernet 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 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 10base-2 and 10base-5. 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 10base-T, the "T" meaning twisted pair. The 100 Megabits per second version is known as 100base-T.

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," which serves as an interconnection point. You can insert and remove hosts from a hub without interrupting any other users at all.

Many people prefer thin Ethernet for small networks because it is very inexpensive; PC cards come for as little as US $30 (many companies are literally throwing them out now), and cable is in the range of a few cents per meter. However, for large-scale installations, either thick Ethernet or twisted pair is more appropriate. For example, the Ethernet at GMU's Mathematics Department originally chose thick Ethernet because it is a long route that the cable must take so traffic will not be disrupted each time a host is added to the network. Twisted pair installations are now very common in a variety of installations. The Hub hardware is dropping in price and small units are now available at a price that is attractive to even small domestic networks. Twisted pair cabling can be significantly cheaper for large installations, and the cable itself is much more flexible than the coaxial cables used for the other Ethernet systems. The network administrators in GMU's mathematics department are planning to replace the existing network with a twisted pair network in the coming financial year because it will bring them up to date with current technology and will save them significant time when installing new host computers and moving existing computers around.

One of the drawbacks of Ethernet technology is its limited cable length, which precludes any use of it other than for LANs. However, several Ethernet segments can be linked to one another using repeaters, bridges, or routers. Repeaters simply copy the signals between two or more segments so that all segments together will act as if they are one Ethernet. Due to timing requirements, there may not be more than four repeaters between any two hosts on the network. Bridges and routers are more sophisticated. They analyze incoming data and forward it only when the recipient host is not on the local Ethernet.

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 six-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. Utilization of Ethernet networks is more realistically limited to about 60 percent before you need to start worrying about it.

---

Other Types of Hardware

In larger installations, such as Groucho Marx University, 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.

At Groucho Marx University, each department's LAN is linked to the campus high-speed "backbone" network, which is a fiber optic cable running a network technology called Fiber Distributed Data Interface (FDDI). FDDI uses an entirely different approach to transmitting data, which basically involves sending around a number of tokens, with a station being allowed to send a frame only if it captures a token. The main advantage of a token-passing protocol is a reduction in collisions. Therefore, the protocol can more easily attain the full speed of the transmission medium, up to 100 Mbps in the case of FDDI. FDDI, being based on optical fiber, offers a significant advantage because its maximum cable length is much greater than wire-based technologies. It has limits of up to around 200 km, which makes it ideal for linking many buildings in a city, or as in GMU's case, many buildings on a campus.

Similarly, if there is any IBM computing equipment around, an IBM Token Ring network is quite likely to be installed. Token Ring is used as an alternative to Ethernet in some LAN environments, and offers the same sorts of advantages as FDDI in terms of achieving full wire speed, but at lower speeds (4 Mbps or 16 Mbps), and lower cost because it is based on wire rather than fiber. In Linux, Token Ring networking is configured in almost precisely the same way as Ethernet, so we don't cover it specifically.

Although it is much less likely today than in the past, other LAN technologies, such as ArcNet and DECnet, might be installed. Linux supports these too, but we don't cover them here.

Many national networks operated by Telecommunications companies support packet switching protocols. Probably the most popular of these is a standard named X.25. Many Public Data Networks, like Tymnet in the U.S., Austpacc in Australia, and Datex-P in Germany offer this service. X.25 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 PAD (Packet Assembler Disassembler). 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 experimental implementation of the X.25 protocol available for Linux.

A more recent protocol commonly offered 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 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, ATM (Asynchronous Transfer Mode) 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 provide 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," and are 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. Both are linked to the FDDI campus backbone network.

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, *Issues of TCP/IP Networking*, 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 are IP addresses, and finally, there are hardware addresses, like the 6-byte Ethernet address. All these addresses somehow have to match so that when you type `rlogin 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 SLIP, or *Serial Line IP*. A modification of SLIP known as CSLIP, or *Compressed SLIP*, performs compression of IP headers to make better use of the relatively low bandwidth provided by most serial links. Another serial protocol is PPP, or the *Point-to-Point Protocol*. 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.

### 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 `rlogin` process on *erdos* and the shell process on *quark*. Thus, the information sent to and fro must be split up 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 ten people on your Ethernet started downloading the latest release of Netscape's 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 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 `rlogin` example, the client application (`rlogin`) opens a port on `erdos` and connects to port 513 on `quark`, to which the `rlogind` server is known to listen. This action establishes a TCP connection. Using this connection, `rlogind` 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 `rlogin` 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 `rlogin`, the overhead involved is prohibitive for applications like NFS, which instead uses a sibling protocol of TCP called UDP, or **User Datagram Protocol**. 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 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 name server 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 `rlogin` client will use the local port 1023, and the second one will use port 1022. 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 `rlogin`, these numbers have to be administered centrally. This is done by the IETF (Internet Engineering Task Force), 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 513, for example, is different from UDP port 513. In fact, these ports serve as access points for two different services, namely `rlogin` (TCP) and `rwho` (UDP).

**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 XNS (Xerox Networking System) 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.

UUCP Networks

Unix-to-Unix Copy (UUCP) started out as a package of programs that transferred files over serial lines, scheduled those transfers, and initiated execution of programs on remote sites. It has undergone major changes since its first implementation in the late seventies, but it is still rather spartan in the services it offers. Its main application is still in Wide Area Networks, based on periodic dialup telephone links.

UUCP was first developed by Bell Laboratories in 1977 for communication between their Unix development sites. In mid-1978, this network already connected over 80 sites. It was running email as an application, as well as remote printing. However, the system’s central use was in distributing new software and bug fixes. Today, UUCP is not confined solely to the Unix environment. There are free and commercial ports available for a variety of platforms, including AmigaOS, DOS, and Atari's TOS.

One of the main disadvantages of UUCP networks is that they operate in batches. Rather than having a permanent connection established between hosts, it uses temporary connections. A UUCP host machine might dial in to another UUCP host only once a day, and then only for a short period of time. While it is connected, it will transfer all of the news, email, and files that have been queued, and then disconnect. It is this queuing that limits the sorts of applications that UUCP can be applied to. In the case of email, a user may prepare an email message and post it. The message will stay queued on the UUCP host machine until it dials in to another UUCP host to transfer the message. This is fine for network services such as email, but is no use at all for services such as rlogin.

Despite these limitations, there are still many UUCP networks operating all over the world, run mainly by hobbyists, which offer private users network access at reasonable prices. The main reason for the longtime popularity of UUCP was that it was very cheap compared to having your computer directly connected to the Internet. To make your computer a UUCP node, all you needed was a modem, a working UUCP implementation, and another UUCP node that was willing to feed you mail and news. Many people were prepared to provide UUCP feeds to individuals because such connections didn’t place much demand on their existing network.

We cover the configuration of UUCP in a chapter of its own later in the book, but we won’t focus on it too heavily, as it’s being replaced rapidly with TCP/IP, now that cheap Internet access has become commonly available in most parts of the world.

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 peo-
ple, 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, which we'll discuss in Chapter 15, IPX and the NCP Filesystem), Appletalk (for Apple networks) and AX.25, NetRom, and Rose (for amateur radio networks). Other standard Net-4 features include IP firewalls, IP accounting (discussed later in Chapter 9, TCP/IP Firewall and Chapter 10, IP Accounting), and IP Masquerade (discussed later in Chapter 11, IP Masquerade and Network Address Translation). IP tunnelling in a couple of different flavors and advanced policy routing are supported. A very large variety of Ethernet devices is 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 an implementation of the SMB filesystem, which interoperates with applications like lanmanager and Microsoft Windows, called Samba, written by Andrew Tridgell, 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 quite mature 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.

**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.kernel.org in /pub/linux/kernel/v$2.$x/, where $x$ is an even number. The latest experimental Linux kernels can be found on ftp.kernel.org in /pub/linux/kernel/v$2.$y/, where $y$ is an odd number. There are Linux kernel source mirrors all over the world. It is now hard to imagine Linux without standard network support.

---

8 Alan can be reached at alan@lxorg.u.ukuu.org.uk
9 NCP is the protocol on which Novell file and print services are based
### 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, like mail and news, 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 log files 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, like `inn` or C News, 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 log file 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 Chapter 19, *Getting Exim Up and Running* or Chapter 18, *Sendmail*.

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 maintenance. 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 name server.

### 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 discussing the context in which they may occur and some common defenses against them.

This section will discuss a few examples and basic techniques for dealing with system security. Of course, the topics covered cannot treat all security issues you may be faced with in detail; they merely serve to illustrate the problems that may arise. Therefore, reading a good book on security is an absolute must, especially in a networked system.

System security starts with good system administration. This includes checking the ownership and permissions of all vital files and directories and monitoring use of privileged accounts. The COPS program, for instance, will check your file system and common configuration files for unusual permissions or other anomalies. It is also wise to use a password suite that enforces certain rules on the users' passwords that make them hard to guess. The shadow password suite, for instance, requires a password to have at least five letters and to contain both upper- and lowercase numbers, as well as non-alphabetic characters.

When making a service accessible to the network, make sure to give it "least privilege"; don't permit it to do things that aren't required for it to work as designed. For example, you should make programs setuid to `root` or some other privileged account only when necessary. Also, if you want to use a service for only a very limited application, don't hesitate to configure it as restrictively as your special application allows. For instance, if you want to allow diskless hosts to boot from your machine, you must provide *Trivial File Transfer Protocol* (TFTP) so that they can download basic configuration files from the `/boot` directory. However, when used unrestrictively, TFTP allows users anywhere in the world to download any world-readable file from your system. If this is not what you want, restrict TFTP service to the `/boot` directory.\footnote{We will come back to this topic in Chapter 12, *Important Network Features*}

You might also want to restrict certain services to users from certain hosts, say from your local network. In Chapter 12, we introduce `tcptr`, which does this for a variety of network applications. More sophisticated methods of restricting access to particular hosts or services will be explored later in Chapter 9.
Another important point is to avoid "dangerous" software. Of course, any software you use can be dangerous because software may have bugs that clever people might exploit to gain access to your system. Things like this happen, and there's no complete protection against it. This problem affects free software and commercial products alike. However, programs that require special privilege are inherently more dangerous than others, because any loophole can have drastic consequences. If you install a setuid program for network purposes, be doubly careful to check the documentation so that you don't create a security breach by accident.

Another source of concern should be programs that enable login or command execution with limited authentication. The rlogin, rsh, and rexec commands are all very useful, but offer very limited authentication of the calling party. Authentication is based on trust of the calling host name obtained from a name server (we'll talk about these later), which can be faked. Today it should be standard practice to disable the r commands completely and replace them with the ssh suite of tools. The ssh tools use a much more reliable authentication method and provide other services, such as encryption and compression, as well.

You can never rule out the possibility that your precautions might fail, regardless of how careful you have been. You should therefore make sure you detect intruders early. Checking the system log files is a good starting point, but the intruder is probably clever enough to anticipate this action and will delete any obvious traces he or she left. However, there are tools like tripwire, written by Gene Kim and Gene Spafford, that allow you to check vital system files to see if their contents or permissions have been changed. tripwire computes various strong checksums over these files and stores them in a database. During subsequent runs, the checksums are recomputed and compared to the stored ones to detect any modifications.

---

11 There have been commercial Unix systems (that you have to pay lots of money for) that came with a setuid-root shell script, which allowed users to gain root privilege using a simple standard trick

12 In 1988, the RTM worm brought much of the Internet to a grinding halt, partly by exploiting a gaping hole in some programs including the sendmail program. This hole has long since been fixed
Chapter 2 - Issues of TCP/IP Networking

In this chapter we turn to the configuration decisions you'll need to make when connecting your Linux machine to a TCP/IP network, including dealing with IP addresses, hostnames, and routing issues. This chapter gives you the background you need in order to understand what your setup requires, while the next chapters cover the tools you will use.

To learn more about TCP/IP and the reasons behind it, refer to the three-volume set Internetworking with TCP/IP, by Douglas R. Comer (Prentice Hall). For a more detailed guide to managing a TCP/IP network, see TCP/IP Network Administration by Craig Hunt (O'Reilly).

Networking Interfaces

To hide the diversity of equipment that may be used in a networking environment, TCP/IP defines an abstract interface through which the hardware is accessed. This interface offers a set of operations that is the same for all types of hardware and basically deals with sending and receiving packets.

For each peripheral networking device, a corresponding interface has to be present in the kernel. For example, Ethernet interfaces in Linux are called by such names as eth0 and eth1; PPP (discussed in Chapter 8, The Point-to-Point Protocol) interfaces are named ppp0 and ppp1; and FDDI interfaces are given names like fddi0 and fddi1. These interface names are used for configuration purposes when you want to specify a particular physical device in a configuration command, and they have no meaning beyond this use.

Before being used by TCP/IP networking, an interface must be assigned an IP address that serves as its identification when communicating with the rest of the world. This address is different from the interface name mentioned previously; if you compare an interface to a door, the address is like the nameplate pinned on it.

Other device parameters may be set, like the maximum size of datagrams that can be processed by a particular piece of hardware, which is referred to as Maximum Transfer Unit (MTU). Other attributes will be introduced later. Fortunately, most attributes have sensible defaults.

IP Addresses

As mentioned in Chapter 1, Introduction to Networking, the IP networking protocol understands addresses as 32-bit numbers. Each machine must be assigned a number unique to the networking environment. If you are running a local network that does not have TCP/IP traffic with other networks, you may assign these numbers according to your personal preferences. There are some IP address ranges that have been reserved for such private networks. These ranges are listed in Table 2.1. However, for sites on the Internet, numbers are assigned by a central authority, the Network Information Center (NIC).

IP addresses are split up into four eight-bit numbers called octets for readability. For example, quark.physics.groucho.edu has an IP address of 0x954C0C04, which is written as 149.76.12.4. This format is often referred to as dotted quad notation.

Another reason for this notation is that IP addresses are split into a network number, which is contained in the leading octets, and a host number, which is the remainder. When applying to the NIC for IP addresses, you are not assigned an address for each single host you plan to use. Instead, you are given a network number and allowed to assign all valid IP addresses within this range to hosts on your network according to your preferences.

13 The version of the Internet Protocol most frequently used on the Internet is Version 4. A lot of effort has been expended in designing a replacement called IP Version 6. IPv6 uses a different addressing scheme and larger addresses. Linux has an implementation of IPv6, but it isn't ready to document it in this book yet. The Linux kernel support for IPv6 is good, but a large number of network applications need to be modified to support it as well. Stay tuned.

14 Frequently, IP addresses will be assigned to you by the provider from whom you buy your IP connectivity. However, you may also apply to the NIC directly for an IP address for your network by sending email to hostmaster@internic.net, or by using the form at http://www.internic.net/.
The size of the host part depends on the size of the network. To accommodate different needs, several classes of networks, defining different places to split IP addresses, have been defined. The class networks are described here:

Class A
Class A comprises networks 1.0.0.0 through 127.0.0.0. The network number is contained in the first octet. This class provides for a 24-bit host part, allowing roughly 1.6 million hosts per network.

Class B
Class B contains networks 128.0.0.0 through 191.255.0.0; the network number is in the first two octets. This class allows for 16,320 nets with 65,024 hosts each.

Class C
Class C networks range from 192.0.0.0 through 223.255.255.0, with the network number contained in the first three octets. This class allows for nearly 2 million networks with up to 254 hosts.

Classes D, E, and F
Addresses falling into the range of 224.0.0.0 through 254.0.0.0 are either experimental or are reserved for special purpose use and don't specify any network. IP Multicast, which is a service that allows material to be transmitted to many points on an internet at one time, has been assigned addresses from within this range.

If we go back to the example in Chapter 1, we find that 149.76.12.4, the address of quark, refers to host 12.4 on the class B network 149.76.0.0.

You may have noticed that not all possible values in the previous list were allowed for each octet in the host part. This is because octets 0 and 255 are reserved for special purposes. An address where all host part bits are 0 refers to the network, and an address where all bits of the host part are 1 is called a broadcast address. This refers to all hosts on the specified network simultaneously. Thus, 149.76.255.255 is not a valid host address, but refers to all hosts on network 149.76.0.0.

A number of network addresses are reserved for special purposes. 0.0.0.0 and 127.0.0.0 are two such addresses. The first is called the default route, and the latter is the loopback address. The default route has to do with the way the IP routes datagrams.

Network 127.0.0.0 is reserved for IP traffic local to your host. Usually, address 127.0.0.1 will be assigned to a special interface on your host, the loopback interface, which acts like a closed circuit. Any IP packet handed to this interface from TCP or UDP will be returned to them as if it had just arrived from some network. This allows you to develop and test networking software without ever using a "real" network. The loopback network also allows you to use networking software on a standalone host. This may not be as uncommon as it sounds; for instance, many UUCP sites don't have IP connectivity at all, but still want to run the INN news system. For proper operation on Linux, INN requires the loopback interface.

Some address ranges from each of the network classes have been set aside and designated "reserved" or "private" address ranges. These addresses are reserved for use by private networks and are not routed on the Internet. They are commonly used by organizations building their own intranet, but even small networks often find them useful. The reserved network addresses appear in Table 2.1.

### Table 2.1: IP Address Ranges Reserved for Private Use

<table>
<thead>
<tr>
<th>Class</th>
<th>Networks</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>10.0.0.0 through 10.255.255.255</td>
</tr>
<tr>
<td>B</td>
<td>172.16.0.0 through 172.31.0.0</td>
</tr>
<tr>
<td>C</td>
<td>192.168.0.0 through 192.168.255.0</td>
</tr>
</tbody>
</table>
Address Resolution

Now that you've seen how IP addresses are composed, you may be wondering how they are used on an Ethernet or Token Ring network to address different hosts. After all, these protocols have their own addresses to identify hosts that have absolutely nothing in common with an IP address, don't they? Right.

A mechanism is needed to map IP addresses onto the addresses of the underlying network. The mechanism used is the Address Resolution Protocol (ARP). In fact, ARP is not confined to Ethernet or Token Ring, but is used on other types of networks, such as the amateur radio AX.25 protocol. The idea underlying ARP is exactly what most people do when they have to find Mr. X in a throng of 150 people: the person who wants him calls out loudly enough that everyone in the room can hear them, expecting him to respond if he is there. When he responds, we know which person he is.

When ARP wants to find the Ethernet address corresponding to a given IP address, it uses an Ethernet feature called broadcasting, in which a datagram is addressed to all stations on the network simultaneously. The broadcast datagram sent by ARP contains a query for the IP address. Each receiving host compares this query to its own IP address and if it matches, returns an ARP reply to the inquiring host. The inquiring host can now extract the sender's Ethernet address from the reply.

You may wonder how a host can reach an Internet address that may be on a different network halfway around the world. The answer to this question involves routing, namely finding the physical location of a host in a network. We will discuss this issue further in the next section.

Let's talk a little more about ARP. Once a host has discovered an Ethernet address, it stores it in its ARP cache so that it doesn't have to query for it again the next time it wants to send a datagram to the host in question. However, it is unwise to keep this information forever; the remote host's Ethernet card may be replaced because of technical problems, so the ARP entry becomes invalid. Therefore, entries in the ARP cache are discarded after some time to force another query for the IP address.

Sometimes it is also necessary to find the IP address associated with a given Ethernet address. This happens when a diskless machine wants to boot from a server on the network, which is a common situation on Local Area Networks. A diskless client, however, has virtually no information about itself -- except for its Ethernet address! So it broadcasts a message containing a request asking a boot server to provide it with an IP address. There's another protocol for this situation named Reverse Address Resolution Protocol (RARP). Along with the BOOTP protocol, it serves to define a procedure for bootstrapping diskless clients over the network.

IP Routing

We now take up the question of finding the host that datagrams go to based on the IP address. Different parts of the address are handled in different ways; it is your job to set up the files that indicate how to treat each part.

IP Networks

When you write a letter to someone, you usually put a complete address on the envelope specifying the country, state, and Zip Code. After you put it in the mailbox, the post office will deliver it to its destination: it will be sent to the country indicated, where the national service will dispatch it to the proper state and region. The advantage of this hierarchical scheme is obvious: wherever you post the letter, the local postmaster knows roughly which direction to forward the letter, but the postmaster doesn't care which way the letter will travel once it reaches its country of destination.

IP networks are structured similarly. The whole Internet consists of a number of proper networks, called autonomous systems. Each system performs routing between its member hosts internally so that the task of delivering a datagram is reduced to finding a path to the destination host's network. As soon as the datagram is handed to any host on that particular network, further processing is done exclusively by the network itself.
Subnetworks

This structure is reflected by splitting IP addresses into a host and network part, as explained previously. By default, the destination network is derived from the network part of the IP address. Thus, hosts with identical IP network numbers should be found within the same network.

It makes sense to offer a similar scheme inside the network, too, since it may consist of a collection of hundreds of smaller networks, with the smallest units being physical networks like Ethernets. Therefore, IP allows you to subdivide an IP network into several subnets.

A subnet takes responsibility for delivering datagrams to a certain range of IP addresses. It is an extension of the concept of splitting bit fields, as in the A, B, and C classes. However, the network part is now extended to include some bits from the host part. The number of bits that are interpreted as the subnet number is given by the so-called subnet mask, or netmask. This is a 32-bit number too, which specifies the bit mask for the network part of the IP address.

The campus network of Groucho Marx University is an example of such a network. It has a class B network number of 149.76.0.0, and its netmask is therefore 255.255.0.0.

Internally, GMU's campus network consists of several smaller networks, such as various departments' LANs. So the range of IP addresses is broken up into 254 subnets, 149.76.1.0 through 149.76.254.0. For example, the department of Theoretical Physics has been assigned 149.76.12.0. The campus backbone is a network in its own right, and is given 149.76.1.0. These subnets share the same IP network number, while the third octet is used to distinguish between them. They will thus use a subnet mask of 255.255.255.0.

Figure 2.1 shows how 149.76.12.4, the address of quark, is interpreted differently when the address is taken as an ordinary class B network and when used with subnetting.

Figure 2.1: Subnetting a class B network

It is worth noting that subnetting (the technique of generating subnets) is only an internal division of the network. Subnets are generated by the network owner (or the administrators). Frequently, subnets are created to reflect existing boundaries, be they physical (between two Ethernets), administrative (between two departments), or geographical (between two locations), and authority over each subnet is delegated to some contact person. However, this structure affects only the network's internal behavior, and is completely invisible to the outside world.

Gateways

Subnetting is not only a benefit to the organization; it is frequently a natural consequence of hardware boundaries. The viewpoint of a host on a given physical network, such as an Ethernet, is a very limited one: it can only talk to the host of the network it is on. All other hosts can be accessed only through special-purpose machines.

15  Autonomous systems are slightly more general. They may comprise more than one IP network.
called gateways. A gateway is a host that is connected to two or more physical networks simultaneously and is configured to switch packets between them.

Figure 2.2 shows part of the network topology at Groucho Marx University (GMU). Hosts that are on two subnets at the same time are shown with both addresses.

**Figure 2.2: A part of the net topology at Groucho Marx University**

![Network topology diagram]

Different physical networks have to belong to different IP networks for IP to be able to recognize if a host is on a local network. For example, the network number 149.76.4.0 is reserved for hosts on the mathematics LAN. When sending a datagram to quark, the network software on erdos immediately sees from the IP address 149.76.12.4 that the destination host is on a different physical network, and therefore can be reached only through a gateway (sophus by default).

*sophus* itself is connected to two distinct subnets: the Mathematics department and the campus backbone. It accesses each through a different interface, *eth0* and *fddi0*, respectively. Now, what IP address do we assign it? Should we give it one on subnet 149.76.1.0, or on 149.76.4.0?

The answer is: "both." *sophus* has been assigned the address 149.76.1.1 for use on the 149.76.1.0 network and address 149.76.4.1 for use on the 149.76.4.0 network. A gateway must be assigned one IP address for each network it belongs to. These addresses -- along with the corresponding netmask -- are tied to the interface through which the subnet is accessed. Thus, the interface and address mapping for *sophus* would look like this:
Interface Address Netmask

Newsreaders can deliver news when a user posts an article. Since the handling of newsreaders deserves special attention, we will come back to this a little later.

Very small news sites should consider a caching NNTP server program like **leafnode**, which is available at http://wpxx02.toxi.uni-wuerzburg.de/~krasel/leafnode.html.
When receiving an article, \texttt{innd} first looks up its message ID in the \texttt{history} file. Duplicate articles are dropped and the occurrences are optionally logged. The same goes for articles that are too old or lack some required header field, such as \texttt{Subject}.

If \texttt{innd} finds that the article is acceptable, it looks at the \texttt{Newsgroups} header line to find out what groups it has been posted to. If one or more of these groups are found in the \texttt{active} file, the article is filed to disk. Otherwise, it is filed to the special group junk.

Individual articles are kept below \texttt{/var/spool/news}, also called the \texttt{news spool}. Each newsgroup has a separate directory, in which each article is stored in a separate file. The file names are consecutive numbers, so that an article in \texttt{comp.risks} may be filed as \texttt{comp/risks/217}, for instance. When \texttt{innd} finds that the directory it wants to store the article in does not exist, it creates it automatically.

Apart from storing articles locally, you may also want to pass them on to outgoing feeds. This is governed by the \texttt{newsfeeds} file that lists all downstream sites along with the newsgroups that should be fed to them.

Just like \texttt{innd}'s receiving end, the processing of outgoing news is handled by a single interface, too. Instead of doing all the transport-specific handling itself, \texttt{innd} relies on various backends to manage the transmission of articles to other news servers. Outgoing facilities are collectively dubbed \texttt{channels}. Depending on its purpose, a channel can have different attributes that determine exactly what information \texttt{innd} passes on to it.

For an outgoing NNTP feed, for instance, \texttt{innd} might fork the \texttt{innxmit} program at startup, and, for each article that should be sent across that feed, pass its message ID, size, and filename to \texttt{innxmit}'s standard input. For an outgoing UUCP feed, on the other hand, it might write the article's size and file name to a special logfile, which is head by a different process at regular intervals in order to create batches and queue them to the UUCP subsystem.

Besides these two examples, there are other types of channels that are not strictly outgoing feeds. These are used, for instance, when archiving certain newsgroups, or when generating overview information. Overview information is intended to help newsreaders thread articles more efficiently. Old-style newsreaders had to scan all articles separately in order to obtain the header information required for threading. This would put an immense strain on the server machine, especially when using NNTP; furthermore, it was very slow. The overview mechanism alleviates this problem by prerecording all relevant headers in a separate file (called \texttt{.overview}) for each newsgroup. This information can then be picked up by newsreaders either by reading it directly from the spool directory, or by using the \texttt{XOVER} command when connected via NNTP. INN has the \texttt{innd} daemon feed all articles to the \texttt{overchan} command, which is attached to the daemon through a channel. We'll see how this is done when we discuss configuring news feeds later.

\begin{itemize}
\item [136] This is indicated by the \texttt{Date} header field; the limit is usually two weeks.
\item [137] Threading 1,000 articles when talking to a loaded server could easily take around five minutes, which only the most dedicated Usenet addict would find acceptable.
\end{itemize}
**Newsreaders and INN**

Newsreaders running on the same machine as the server (or having mounted the server's news spool via NFS) can read articles from the spool directly. To post an article composed by the user, they invoke the `inews` program, which adds any header fields that are missing and forwards them to the daemon via NNTP.

Alternatively, newsreaders can access the server remotely via NNTP. This type of connection is handled differently from NNTP-based news feeds, to avoid tying up the daemon. Whenever a newsreader connects to the NNTP server, `innd` forks a separate program called `nnrpd`, which handles the session while `innd` returns to the more important things (receiving incoming news, for example). You may be wondering how the `innd` process can distinguish between an incoming news feed and a connecting newsreader. The answer is quite simple: the NNTP protocol requires that an NNTP-based newsreader issue a `mode reader` command after connecting to the server; when this command is received, the server starts the `nnrpd` process, hands the connection to it, and returns to listening for connections from another news server. There used to be at least one DOS-based newsreader which was not configured to do this, and hence failed miserably when talking to INN, because `innd` itself does not recognize any of the commands used to read news if it doesn't know the connection is from a news reader.

We'll talk a little more about newsreader access to INN under "Controlling Newsreader Access," later in the chapter.

**Installing INN**

Before diving into INN's configuration, let's talk about its installation. Read this section, even if you've installed INN from one of the various Linux distributions; it contains some hints about security and compatibility.

Linux distributions included Version INN-1.4sec for quite some time. Unfortunately, this version had two subtle security problems. Modern versions don't have these problems and most distributions include a precompiled Linux binary of INN Version 2 or later.

If you choose, you can build INN yourself. You can obtain the source from ftp.isc.org in the `/isc/inn/` directory. Building INN requires that you edit a configuration file that tells INN some detail about your operating system, and some features may require minor modifications to the source itself.

Compiling the package itself is pretty simple; there's a script called `BUILD` that will guide you through the process. The source also contains extensive documentation on how to install and configure INN.

After installing all binaries, some manual fixups may be required to reconcile INN with any other applications that may want to access its `rnews` or `inews` programs. UUCP, for instance, expects to find the `rnews` program in `/usr/bin` or `/bin`, while INN installs it in `/usr/lib/bin` by default. Make sure `/usr/lib/bin` is in the default search path, or that there are symbolic links pointing to the actual location of the `rnews` and `inews` commands.

**Configuring INN: the Basic Setup**

One of the greatest obstacles beginners may face is that INN requires a working network setup to function properly, even when running on a standalone host. Therefore, it is essential that your kernel supports TCP/IP networking when running INN, and that you have set up the loopback interface as explained in Chapter 5, *Configuring TCP/IP Networking*.

Next, you have to make sure that `innd` is started at boot time. The default INN installation provides a script file called `boot` in the `/etc/news/` directory. If your distribution uses the SystemV-style `init` package, all you have to do is create a symbolic link from your `/etc/init.d/inn` file pointing to `/etc/news/boot`. For other flavors of `init`, you have to make sure `/etc/news/boot` is executed from one of your `rc` scripts. Since INN requires networking support, the startup script should be run after the network interfaces are configured.

---

138 The name apparently stands for NetNews Read & Post Daemon.
**INN Configuration Files**

Having completed these general tasks, you can now turn to the really interesting part of INN: its configuration files. All these files reside in `/etc/news`. Some changes to configurations files were introduced in Version 2, and it is Version 2 that we describe here. If you're running an older version, you should find this chapter useful to guide you in upgrading your configuration. During the next few sections, we will discuss them one by one, building the Virtual Brewery's configuration as an example.

If you want to find out more about the features of individual configuration files, you can also consult the manual pages; the INN distribution contains individual manual pages for each of them.

**Global Parameters**

There are a number of INN parameters that are global in nature; they are relevant to all newsgroups carried.

**The inn.conf file**

INN's main configuration file is `inn.conf`. Among other things, it determines the name by which your machine is known on Usenet. Version 2 of INN allows a baffling number of parameters to be configured in this file. Fortunately, most parameters have default values that are reasonable for most sites. The `inn.conf(5)` file details all of the parameters, and you should read it carefully if you experience any problems.

A simple example `inn.conf` might look like:

```plaintext
# Sample inn.conf for the Virtual Brewery
server: vlager.vbrew.com
domain: vbrew.com
fromhost: vbrew.com
pathhost: news.vbrew.com
organization: The Virtual Brewery
mta: /usr/sbin/sendmail -oi %s
moderatormailer: %s@uunet.uu.net
#
# Paths to INN components and files.
#
pathnews: /usr/lib/news
pathbin: /usr/lib/news/bin
pathfilter: /usr/lib/news/bin/filter
pathcontrol: /usr/lib/news/bin/control
pathdb: /var/lib/news
pathetc: /etc/news
pathrun: /var/run/news
pathlog: /var/log/news
pathhttp: /var/log/news
pathtmp: /var/tmp
pathtmpool: /var/spool/news
patharticles: /var/spool/news/articles
pathoverview: /var/spool/news/overview
pathoutgoing: /var/spool/news/outgoing
pathincoming: /var/spool/news/incoming
patharchive: /var/spool/news/archive
pathuniover: /var/spool/news/uniover
overviewname: .overview
```

The first line tells the programs `rnews` and `inews` which host to contact when delivering articles. This entry is absolutely crucial; to pass articles to `innd`, they have to establish an NNTP connection with the server.

The `domain` keyword should specify the domain portion of the host's fully qualified domain name. A couple of programs must look up your host's fully qualified domain name; if your resolver library returns the unqualified hostname only, the name given in the `domain` attribute is tacked onto it. It's not a problem to configure it either way, so it's best to define `domain`. 
The next line defines what hostname inews is going to use when adding a From: line to articles posted by local users. Most newsreaders use the From: field when composing a reply mail message to the author of an article. If you omit this field, it will default to your news host's fully qualified domain name. This is not always the best choice. You might, for example, have news and mail handled by two different hosts. In this case, you would supply the fully qualified domain name of your mail host after the fromhost statement.

The pathhost line defines the hostname INN is to add to the Path: header field whenever it receives an article. In most cases, you will want to use the fully qualified domain name of your news server; you can then omit this field since that is the default. Occasionally you may want to use a generic name, such as news.vbrew.com, when serving a large domain. Doing this allows you to move the news system easily to a different host, should you choose to at some time.

The next line contains the organization keyword. This statement allows you to configure what text inews will put into the Organization: line of articles posted by your local users. Formally, you would place a description of your organization or your organization's name in full here. Should you not wish to be so formal, it is fashionable for organizations with a sense of humor to exhibit it here.

The organization keyword is mandatory and specifies the pathname of the mail transport agent that will be used for posting moderator messages. %s is replaced by the moderator email address.

The moderatormailer entry defines a default address used when a user tries to post to a moderated newsgroup. The list of moderator addresses for each newsgroup is usually kept in a separate file, but you will have a hard time keeping track of all of them. The moderatormailer entry is therefore consulted as a last resort; if it is defined, inews will replace the %s string with the (slightly transformed) newsgroup name and send the entire article to this address. For instance, when posting to soc.feminism, the article is mailed to soc-feminism@uunet.uu.net, given the above configuration. At UUNET, there should be a mail alias installed for each of these submissions addresses that automatically forwards all messages to the appropriate moderator.

Finally, each of the remaining entries specifies the location of some component file or executable belonging to INN. If you've installed INN from a package, these paths should have been configured for you. If you're installing from source, you'll need to ensure that they reflect where you've installed INN.

**Configuring Newsgroups**

The news administrator on a system is able to control which newsgroups users have access to. INN provides two configuration files that allow the administrator to decide which newsgroups to support and provide descriptions for them.

**The active and newsgroups files**

The active and newsgroups files are used to store and describe the newsgroups hosted by this news server. They list which newsgroups we are interested in receiving and serving articles for, and administrative information about them. These files are found in the /var/lib/news/ directory.

The active file determines which newsgroups this server supports. Its syntax is straightforward. Each line in the active file has four fields delimited by whitespace:

name himark lomark flags

The name field is the name of the newsgroup. The himark field is the highest number that has been used for an article in that newsgroup. The lomark field is the lowest active number in use in the newsgroup. To illustrate how this works, consider the follow scenario. Imagine that we have a newly created newsgroup: himark and lomark are both 0 because there are no articles. If we post 5 articles, they will be numbered 1 through 5. himark will now equal 5, the highest numbered article, and lomark will equal 1, the lowest active article. If article 5 is cancelled there will be no change; himark will remain at 5 to ensure that that article number is not reallocated and lomark will remain at 1, the lowest active article. If we now cancel article 1, himark will remain unchanged, but lomark will now equal 2, because 1 is no longer active. If we now post a new article, it will be assigned article number 6, so himark will now equal 6. Article 5 has been in use, so we won't reassign it. lomark remains at 2. This mechanism allows us to easily allocate unique article numbers for new articles and to calculate approximately how many active articles there are in the group: himark-lomark.

The field may contain one of the following:
Posting directly to this news server is allowed.

Posting directly to this news server is not allowed. This prevents newsreaders from posting directly to this news server. New articles may only be received from other news servers.

The group is moderated. Any articles posted to this newsgroup are forwarded to the newsgroup moderator for approval before they enter the newsgroup. Most newsgroups are unmoderated.

Articles in this group are not kept, but only passed on. This causes the news server to accept the article, but all it will do with it is pass it to the "up-stream" news servers. It will not make the articles available to newsreaders reading from this server.

Articles cannot be posted to this newsgroup. The only way that news articles are delivered to this server is by feeding them from another news server. Newsreaders may not directly write articles to this server.

Articles are locally filed into the `foo.bar` group.

In our simple server configuration we'll carry a small number of newsgroups, so our /var/lib/news/active file will look like:

```plaintext
control 000000000 000000001 y
junk 000000000 000000001 y
rec.crafts.brewing 000000000 000000001 y
rec.crafts.brewing.ales 000000000 000000001 y
rec.crafts.brewing.badtaste 000000000 000000001 y
rec.crafts.brewing.brandy 000000000 000000001 y
rec.crafts.brewing.champagne 000000000 000000001 y
rec.crafts.brewing.private 000000000 000000001 y
```

The `himark` and `lomark` numbers in this example are those you would use when creating new newsgroups. The `himark` and `lomark` numbers will look quite different for a newsgroup that has been active for some time.

The newsgroups file is even simpler. It provides one-line descriptions of newsgroups. Some newsreaders are able to read and present this information to a user to help them decide whether they want to subscribe.

The format of the newsgroups file is simply:

```plaintext
name description
```

The `name` field is the name of a newsgroup, and the `<description>` is a single line description of that newsgroup.

We want to describe the newsgroups that our server supports, so we'll build our newsgroups file as follows:

```plaintext
rec.crafts.brewing.ales Home brewing Ales and Lagers
rec.crafts.brewing.badtaste Home brewing foul tasting brews
rec.crafts.brewing.brandy Home brewing your own Brandy
rec.crafts.brewing.champagne Home brew your own Champagne
rec.crafts.brewing.private The Virtual Brewery home brewers group
```

### Configuring Newsfeeds

INN provides the news administrator the ability to control which newsgroups are forwarded on to other news servers and how they will be forwarded. The most common method uses the NNTP protocol described earlier, but INN also allows newsfeeds via other protocols, such as UUCP.

#### The newsfeeds file

The newsfeeds file determines where news articles will be sent. It normally resides in the /etc/news/ directory.

The format of the newsfeeds is a little complicated at first. We'll describe the general layout here, and the newsfeeds(5) manual page describes what we leave out. The format is as follows:
# newsfeeds file format
site:pattern:flags:param
site2:pattern2:\flags2:param2

Each news feed to a site is described by a single line, or may be spread across multiple lines using the \ continuation character. The : characters delimit the fields in each line. The # character at the start of a line marks that line as a comment.

The site field names the site to which this feed description relates. The sitename can be coded any way you like and doesn't have to be the domain name of the site. The site name will be used later and will refer to an entry in a table that supplies the hostname to the innxmit program that transmits the news articles by NNTP to the remote server. You may have multiple entries for each site; each entry will be treated individually.

The pattern field specifies which news groups are to be sent to this site. The default is to send all groups, so if that is what you want, just make this field empty. This field is usually a comma-delimited list of pattern-matching expressions. The * character matches zero or more of any character, the . character has no special significance, the ! character (if used at the start of an expression) performs a logical NOT, and the @ character at the start of a newsgroup name means "Do not forward any articles that are posted or crossposted to this group." The list is read and parsed from left to right, so you should ensure that you place the more specific rules first. The pattern:

rec.crafts.brewing*,!rec.crafts.brewing.poison,@rec.crafts.brewing.private

would send all of the rec.crafts.brewing news hierarchy except the rec.crafts.brewing.poison. It would not feed any articles that were either posted or crossposted to the rec.crafts.brewing.private newsgroup; these articles will be trapped and available only to those people who use this server. If you reversed the first two patterns, the first pattern would be overridden by the second and you would end up feeding articles for the rec.crafts.brewing.poison newsgroup. The same is true of the first and last patterns; you must always place the more specific patterns before any less specific patterns for them to take effect.

The flags field controls and places constraints on the feed of news articles to this site. The flags field is a comma delimited list can contain any of the items from the following list, delimited by commands:

<size
  Article must be less then size bytes.

A|items
  Article checks. items can be one or more of d (must have Distribution header) or p (don't check for site in Path header).

B|high|low
  Internal buffer size before writing to output.

H[count]
  Article must have less then count hops; the default is 1.

I|size
  Internal buffer size (for a file feed).

M|pattern
  Only moderated groups that match the pattern.

N|pattern
  Only unmoderated groups that match the pattern.

S|size
  Start spooling if more than size bytes get queued.

T|type
  Feed types: f (file), m (funnel; the param field names the entry that articles will be funneled to), p (pipe to program), c (send to stdin channel of the param field's subprocess), and x (like c, but handles commands on stdin).
The \texttt{param} field has special coding that is dependent on the type of feed. In the most common configuration it is where you specify the name of the output file to which you will write the outgoing feed. In other configurations you can leave it out. In yet other configurations it takes on different meanings. If you want to do something unusual, the \texttt{newsfeeds(5)} manual page will explain the use of the \texttt{param} field in some detail.

There is a special site name that should be coded as \texttt{ME} and should be the first entry in the file. This entry is used to control the default settings for your news feeds. If the \texttt{ME} entry has a distribution list associated with it, this list will be prepended to each of the other site entries before they are sent. This allows you to, for example, declare some newsgroups to be automatically fed, or automatically blocked from feeding, without having to repeat the pattern in each site entry.

We mentioned earlier that it was possible to use some special feeds to generate thread data that makes the newsreader's job easier. We'll do this by exploiting the \texttt{overchan} command that is part of the INN distribution. To do this, we've created a special local feed called \texttt{overview} that will pass the news articles to the \texttt{overchan} command for processing into overview data.

Our news server will provide only one external news feed, which goes to the Groucho Marx University, and they receive articles for all newsgroups except the \texttt{control} and \texttt{junk} newsgroups, the \texttt{rec.crafts.brewing.private} newsgroup, which will be kept locally, and the \texttt{rec.crafts.brewing.poison} newsgroup, which we don't want people from our brewery seen posting to.

We'll use the \texttt{nntpsend} command to transport the news via NNTP to the \texttt{news.groucho.edu} server. \texttt{nntpsend} requires us to use the "file" delivery method and to write the article's pathname and article ID. Note that we've set the \texttt{param} field to the name of the output file. We'll talk a little more about the \texttt{nntpsend} command in a moment. Our resulting newsfeed's configuration is:

\begin{verbatim}
# /etc/news/newsfeeds file for the Virtual Brewery
#
# Send all newsgroups except the control and junk ones by default
ME:!control,!junk::
#
# Generate overview data for any newsreaders to use.
overview::Tc,WO:/usr/lib/news/bin/overchan
#
# Feed the Groucho Marx University everything except our private newsgroup
# and any articles posted to the rec.crafts.brewing.poison newsgroup.
gmarxu:!rec.crafts.brewing.poison,@rec.crafts.brewing.private:\
    Tf,Wnm:news.groucho.edu
#
\end{verbatim}

### The \texttt{nntpsend.ctl} file

The \texttt{nntpsend} program manages the transmission of news articles using the NNTP protocol by calling the \texttt{innxmit} command. We saw a simple use of the \texttt{nntpsend} command earlier, but it too has a configuration file that provides us with some flexibility in how we configure our news feeds.

The \texttt{nntpsend} command expects to find batch files for the sites it will feed. It expects those batch files to be named \texttt{/var/spool/news/out.going/sitename}. \texttt{inn} creates these batch files when acting on an entry in the \texttt{newsfeeds}, which we saw in the previous sections. We specified the sitename as the filename in the \texttt{param} field, and that satisfies the \texttt{nntpsend} command's input requirements.

The \texttt{nntpsend} command has a configuration file called \texttt{nntpsend.ctl} that is usually stored in the \texttt{/etc/news/} directory.
The `nntpsend.ctl` file allows us to associate a fully qualified domain name, some news feed size constraints, and a number of transmission parameters with a news feed site name. The sitename is a means of uniquely identifying a logical feed of articles. The general format of the file is:

```
sitename:fqdn:max_size:[args]
```

The following list describes the elements of this format:

- **sitename**
  - The sitename as supplied in the `newsfeeds` file

- **fqdn**
  - The fully qualified domain name of the news server to which we will be feeding the news articles

- **max_size**
  - The maximum volume of news to feed in any single transfer

- **args**
  - Additional arguments to pass to the `innxmit` command

Our sample configuration requires a very simple `nntpsend.ctl` file. We have only one news feed. We'll restrict the feed to a maximum of 2 MB of traffic and we'll pass an argument to the `innxmit` that sets a 3-minute (180 second) timeout. If we were a larger site and had many news feeds, we'd simply create new entries for each new feed site that looked much the same as this one:

```
# /etc/news/nntpsend.ctl
#
gmarxu:news.groucho.edu:2m:-t 180
#
```

### Controlling Newsreader Access

Not so many years ago, it was common for organizations to provide public access to their news servers. Today it is difficult to locate public news servers; most organizations carefully control who has access to their servers, typically restricting access to users supported on their network. INN provides configuration files to control this access.

#### The `incoming.conf` file

We mentioned in our introduction to INN that it achieves some of its efficiency and size by separating the news feed mechanism from the newsreading mechanism. The `/etc/news/incoming.conf` file is where you specify which hosts will be feeding you news using the NNTP protocol, as well as where you define some parameters that control the way articles are fed to you from these hosts. Any host not listed in this file that connects to the news socket will not be handled by the `innd` daemon; instead, it will be handled by the `nnrpd` daemon.

The `/etc/news/incoming.conf` file syntax is very simple, but it takes a moment to come to terms with. Three types of valid entries are allowed: key/value pairs, which are how you specify attributes and their values; peers, which is how you specify the name of a host allowed to send articles to us using NNTP; and groups, a means of applying key/value pairs to groups of peers. Key/value pairs can have three different types of scope. Global pairs apply to every peer defined in the file. Group pairs apply to all peers defined within that group. Peer pairs apply only to that one peer. Specific definitions override less specific ones: therefore, peer definitions override group definitions, which in turn override global pairs.

Curly brace characters (`{}`) are used to delimit the start and end of the `group` and `peer` specifications. The `#` character marks the rest of the line it appears on as a comment. Key/value pairs are separated by the colon character and appear one to a line.

A number of different keys may be specified. The more common and useful are:

- **hostname**
  - This key specifies a comma-separated list of fully qualified names or IP addresses of the peers that we'll allow to send us articles. If this key is not supplied, the hostname defaults to the label of the peer.
streaming

This key determines whether streaming commands are allowed from this host. It is a Boolean value that defaults to true.

max-connections

This key specifies the maximum number of connections allowed from this group or peer. A value of zero means unlimited (which can also be specified using none).

password

This key allows you to specify the password that must be used by a peer if it is to be allowed to transfer news. The default is to not require a password.

patterns

This key specifies the newsgroups that we accept from the associated peer. This field is coded according to precisely the same rules as we used in our newsfeeds file.

In our example we have only one host that we are expecting to feed us news: our upstream news provider at Groucho Marx University. We'll have no password, but we will ensure that we don't accept any articles for our private newsgroup from outside. Our hosts.nntp looks like:

```
# Virtual Brewery incoming.conf file.

# Global settings
streaming: true
max-connections: 5

# Allow NNTP posting from our local host.
peer ME {
    hostname: "localhost, 127.0.0.1"
}

# Allow groucho to send us all newsgroup except our local ones.
peer groucho {
    hostname: news.groucho.edu
    patterns: !rec.crafts.brewing.private
}
```

The nnrp.access file

We mentioned earlier that newsreaders, and in fact any host not listed in the hosts.nntp, that connect to the INN news server are handled by the nnrpcl program. nnrpcl uses the /etc/news/nnrp.access file to determine who is allowed to make use of the news server, and what permissions they should have.

The nnrp.access file has a similar structure to the other configuration files we've looked at. It comprises a set of patterns used to match against the connecting host's domain name or IP address, and fields that determine what access and permission it should be given. Each entry should appear on a line by itself, and fields are separated by colons. The last entry in this file that matches the connecting host will be the one used, so again, you should put general patterns first and follow them with more specific ones later in the file. The five fields of each entry in the order they should appear are:

- Hostname or IP address
- Permissions
- Username

- Hostname or IP address
  This field conforms to wildmat(3) pattern-matching rules. It is a pattern that describes the connecting host's name or IP address.

- Permissions
  This field determines what permissions the matching host should be granted. There are two permissions you may configure: R gives read permissions, and P gives posting permissions.

- Username
  This field is optional and allows you to specify a username that an NNTP client must log into the server before being allowed to post news articles. This field may be left blank. No user authentication is required to read articles.
Password

This field is optional and is the password accompanying the username field. Leaving this field blank means that no password is required to post articles.

Newsgroups

This field is a pattern specifying which newsgroups the client is allowed to access. The pattern follows the same rules as those used in the newsfeeds file. The default for this field is no newsgroups, so you would normally have a pattern configured here.

In the virtual brewery example, we will allow any NNTP client in the Virtual Brewery domain to both read and post to all newsgroups. We will allow any NNTP client read-only access to all newsgroups except our private internal newsgroup. Our nnrp.access file will look like this:

```
# Virtual Brewery - nnrp.access
# We will allow public reading of all newsgroups except our private one.
*:R::*,!rec.crafts.brewing.private

# Any host with the Virtual Brewery domain may Read and Post to all
# newsgroups
*.vbrew.com:RP::*
```

Expire News Articles

When news articles are received by a news server, they are stored to disk. News articles need to be available to users for some period of time to be useful, so a large operating news server can consume lots of disk space. To ensure that the disk space is used effectively, you can opt to delete news articles automatically after a period of time. This is called article expiration. Naturally, INN provides a means of automatically expiring news articles.

The expire.ctl file

The INN server uses a program called expire to delete expired news articles. The expire program in turn uses a file called /etc/news/expire.ctl to configure the rules that govern article expiration.

The syntax of /etc/news/expire.ctl is fairly simple. As with most configuration files, empty lines or lines beginning with the # character are ignored. The general idea is that you specify one rule per line. Each rule defines how article expiration will be performed on newsgroups matching a supplied pattern. The rule syntax looks like this:

```
pattern:modflag:keep:default:purge
```

The following list describes the fields:

- **pattern**
  This field is a comma-delimited list of patterns matching names of newsgroups. The wildmat(3) routine is used to match these patterns. The last rule matching a newsgroup name is the one that is applied, so if you want to specify wildcard (*) rules, they should be listed first in this file.

- **modflag**
  This flag describes how this rule applies to moderated newsgroups. It can be coded with an M to mean that this rule applies only to moderated newsgroups, a U to mean that this rule applies only to unmoderated newsgroups, or an A to mean that this rule ignores the moderated status and applies to all groups.

- **keep**
  This field allows you to specify the minimum time an article with an "Expires" header will be kept before it is expired. The units are days, and are a floating point, so you may specify values like 7.5 for seven-and-a-half days. You may also specify never if you wish articles to stay in a newsgroup forever.

- **default**
  This field is the most important. This field allows you to specify the time an article without an Expires header will be kept. Most articles won't have an Expires header. This field is coded in the same way as the keep field, with never meaning that articles without Expires headers will never be expired.
This field allows you to specify the maximum time an article with an Expires header will be kept before it is expired. The coding of this field is the same as for the keep field.

Our requirements are simple. We will keep all articles in all newsgroups for 14 days by default, and between 7 and 21 days for articles that have an Expires header. The rec.crafts.brewing.private newsgroup is our internal newsgroup, so we'll make sure we don't expire any articles from it:

```
# expire.ctl file for the Virtual Brewery

# Expire all articles in 14 days by default, 7-21 days for those with
# Expires: headers
*:A:7:14:21

# This is a special internal newsgroup, which we will never expire.
```

We will mention one special type of entry you may have in your /etc/news/expires.ctl file. You may have exactly one line that looks like this:

```
/re/remember/:days
```

This entry allows you to specify the minimum number of days that an article will be remembered in the history file, irrespective of whether the article itself has been expired or not. This might be useful if one of the sites that is feeding you articles is infrequent and has a habit of sending you old articles every now and again. Setting the /remember/ field helps to prevent the upstream server from sending you the article again, even if it has already been expired from your server. If your server remembers it has already received the article, it will reject attempts to resend it. It is important to remember that this setting has no effect at all on article expiration; it affects only the time that details of an article are kept in the history database.

### Handling Control Messages

Just as with C News, INN can automatically process control messages. INN provides a powerful configuration mechanism to control what action will occur for each of a variety of control messages, and an access control mechanism to control who can initiate actions against which newsgroups.

#### The control.ctl file

The control.ctl file is fairly simple in structure. The syntax rules for this file are much the same as for the other INN configuration files. Lines beginning with # are ignored, lines may be continued using /, and fields are delimited by :.

When a control message is received, it is tested against each rule in turn. The last rule in the file that matches the message is the rule that will be used, so you should put any generic rules at the start of the file and more specific rules at the end of the file. The general syntax of the file is:

```
message:from:newsgroups:action
```

The meanings of each of the fields are:

- **message**
  - This is the name of the control message. Typical control messages are described later.

- **from**
  - This is a shell-style pattern matching the email address of the person sending the message. The email address is converted to lowercase before comparison.

- **newsgroups**
  - If the control message is newgroup or rmgroup, this field is a shell-style pattern matching the newsgroup created or removed.

- **action**
  - This field specifies what action to take for any message matching the rule. There are quite a number of actions we can take; they are described in the next list.
The message field of each line can have one of the following values:

**checkgroups**
This message requests that news administrators resynchronize their active newsgroups database against the list of newsgroups supplied in the control message.

**newgroup**
This message requests the creation of a new newsgroup. The body of the control message should contain a short description of the purpose of the newsgroup to be created.

**rmgroup**
requests that a newsgroup be removed.

**sendsys**
This message requests that the sys file of this news server be transmitted by mail to the originator of the control message. RFC-1036 states that it is a requirement of Usenet membership that this information be publicly available because it is used to keep the map of Usenet up to date.

**version**
This message requests that the hostname and version of news server software be returned to the originator of the control message.

**all**
This is a special coding that will match any control message.

The message field may include the following actions:

**doit**
The requested command is performed. In many cases, a mail message will be sent to the administrator to advise them that the action has taken place.

**doit=**
This is the same as the doit action except that a log message will be written to the file log file. If the specified file is mail, the log entry is sent by email. If the specified file is the null string, the log message is written to /dev/null and is equivalent to using the unqualified doit action. If the file name begins with a / character, the name is taken to be an absolute filename for the logfile; otherwise, the specified name is translated to /var/log/news/file.log.

**doifarg**
The requested command is performed if the command has an argument. If the command has no argument, the control message is ignored.

**drop**
The requested command is ignored.

**log**
A log message is sent to the stderr output of the innd process. This is normally directed out to the /var/log/news/errlog file.

**log=**
This is the same as a log action, except the logfile is specified as per the rules given for the doit= file action.

**mail**
An email message is sent to the news administrator containing the requested command details. No other action takes place.

**verify-**
If an action begins with the string "verify-", then the control message is authenticated using PGP (or GPG).

---

139  PGP and GPG are tools designed to authenticate or encrypt messages using public key techniques. GPG is the GNU free version of PGP. GPG may be found at [http://www.gnupg.org/](http://www.gnupg.org/) and PGP may be found at [http://www.pgp.com/](http://www.pgp.com/).
So that you can see what a control.ctl file would look like in practice, here is a very short illustrative sample:

```bash
## Sample /etc/news/control.ctl
##
## Warning: You should not use this file, it is illustrative only.

## Control Message Handling
all:*:*:mail
cHECKGROUPS:*:*:mail
ihave:*:*:drop
sendme:*:*:drop
sendsys:*:*:log=sendsys
senduuname:*:*:log=senduuname
version:*:*:log=version
newgroup:*:*:mail
rmgroup:*:*:mail

## Handle control messages for the eight most important news hierarchies
## COMP, HUMANITIES, MISC, NEWS, REC, SCI, SOC, TALK
checkgroups:*:comp.*|humanities.*|misc.*|news.*|rec.*|sci.*|soc.*|talk.*:drop
newgroup:*:comp.*|humanities.*|misc.*|news.*|rec.*|sci.*|soc.*|talk.*:drop
rmgroup:*:comp.*|humanities.*|misc.*|news.*|rec.*|sci.*|soc.*|talk.*:drop
checkgroups:group-admin@isc.org:*:verify-news.announce.newgroups
newgroup:group-admin@isc.org:comp.*|misc.*|news.*:verify-news.announce.newgroups
newgroup:group-admin@isc.org:rec.*|sci.*|soc.*:verify-news.announce.newgroups
newgroup:group-admin@isc.org:talk.*|humanities.*:verify-news.announce.newgroups
newgroup:gnu@prep.ai.mit.edu:gnu.*:doit
newgroup:news@ai.mit.edu:gnu.*:doit
rmgroup:gnu@prep.ai.mit.edu:gnu.*:doit
rmgroup:news@ai.mit.edu:gnu.*:doit

## GNU ( Free Software Foundation )
newgroup:gnu@prep.ai.mit.edu:gnu.*:doit
newgroup:news@ai.mit.edu:gnu.*:doit
rmgroup:gnu@prep.ai.mit.edu:gnu.*:doit
rmgroup:news@ai.mit.edu:gnu.*:doit

## LINUX (Newsfeed from news.lameter.com)
checkgroups:christoph@lameter.com:linux.*:doit
newgroup:christoph@lameter.com:linux.*:doit
rmgroup:christoph@lameter.com:linux.*:doit
```

### Running INN

The inn source package provides a script suitable for starting inn at boot time. The script is usually called /usr/lib/news/bin/rc.news. The script reads arguments from another script, usually called /usr/lib/news/innshellvars, which contains definitions of the filenames and filepaths that inn will use to locate components it needs. It is generally considered a good idea to execute inn with the permissions of a non-root user, such as news.

To ensure that inn is started at boot time, you should check that /usr/lib/news/innshellvars is configured correctly and then call the /usr/lib/news/bin/rc.news script from a script executed at boot time.

Additionally, there are administrative tasks that must be performed periodically. These tasks are usually configured to be executed by the cron command. The best way to do this is to add the appropriate commands to your
/etc/crontab file, or even better, create a file suitable for the /etc/cron.d directory, if your distribution provides one. An example of such a file might look like:

```
# Example /etc/cron.d/inn file, as used in the Debian distribution.
#
SHELL=/bin/sh
PATH=/usr/lib/news/bin:/sbin:/bin:/usr/sbin:/usr/bin
# Expire old news and overview entries nightly, generate reports.
15 0 * * * news news.daily expireover lowmark delayrm
# Every hour, run an rnews -U. This is not only for UUCP sites, but
# also to process queued up articles put there by in.nmrpd in case
# innd wasn't accepting any articles.
10 * * * * news rnews -U
```

These commands will ensure that old news is automatically expired each day, and that any queued articles are processed each hour. Note also that they are executed with the permissions of the news user.

**Managing INN: The ctlinnd Command**

The INN news server comes with a command to manage its day-to-day operation. The ctlinnd command can be used to manipulate newsgroups and newsgroup feeds, to obtain the status, of the server, and to reload, stop, and start the server.

You'd normally get a summary of the ctlinnd command syntax using:

```
# ctlinnd -h
```

We'll cover some of the more important uses of ctlinnd here; please consult the ctlinnd manual page for more detail.

**Add a New Group**

Use the following syntax to add a new group:

```
ctlinnd newgroup group rest creator
```

The arguments are defined as follows:

- `group`
  - The name of the group to create.

- `rest`
  - This argument should be coded in the same way as the flags field of the active file. It defaults to y if not supplied.

- `creator`
  - The name of the person creating the group. Enclose it in quotes if there are any spaces in the name.

**Change a Group**

Use the following syntax to change a group:

```
ctlinnd changegroup group rest
```

The arguments are defined as follows:

- `group`
  - The name of the group to change.
This argument should be coded in the same way as the flags field of the active file.
This command is useful to change the moderation status of a group.

**Remove a Group**

Use the following syntax to remove a group:
```bash
cctl innd rmgroup group
```
The argument is defined as follows:
```bash
group
```
The name of the group to remove.
This command removes the specified newsgroup from the active file. It has no effect on the news spool. All articles in the spool for the specified group will be expired in the usual fashion, but no new articles will be accepted.

**Renumber a Group**

Use the following syntax to renumber a group:
```bash
cctl innd renumber group
```
The argument is defined as follows:
```bash
group
```
The name of the group to renumber. If a group is an empty string, all groups are renumbered.
This command updates the low-water mark for the specified group.

**Allow/Disallow Newsreaders**

Use the following syntax to allow or disallow newsreaders:
```bash
cctl innd readers flag text
```
The arguments are defined as follows:
```bash
flag
```
Specifying n causes all newsreader connections to be disallowed. Specifying y allows newsreader connections.

```bash
text
```
The text supplied will be given to newsreaders who attempt to connect, and usually describes the reason for disabling newsreader access. When reenabling newsreader access, this field must be either an empty string or a copy of the text supplied when the newsreader was disabled.
This command does not affect incoming newsfeeds. It only controls connections from newsreaders.

**Reject Newsfeed Connections**

Use the following syntax to reject newsfeed connections:
```bash
cctl innd reject reason
```
The argument is defined as follows:
```bash
reason
```
The text supplied should explain why incoming connections to innd are rejected.
This command does not affect connections that are handed off to nnrpd (i.e., newsreaders); it only affects connections that would be handled by innrd directly, such as remote newsfeeds.

**Allow Newsfeed Connections**

Use the following syntax to allow newsfeed connections:

```bash
ctlinnd allow reason
```

The argument is defined as follows:

`reason`

The supplied text must be the same as that supplied to the preceding `reject` command or an empty string.

This command reverses the effect of a `reject` command.

**Disable News Server**

Use the following syntax to disable the news server:

```bash
ctlinnd throttle reason
```

The argument is defined as follows:

`reason`

The reason for throttling the server.

This command is simultaneously equivalent to a `newsreaders no` and a `reject`, and is useful when emergency work is performed on the news database. It ensures that nothing attempts to update it while you are working on it.

**Restart News Server**

Use the following syntax to restart the news server:

```bash
ctlinnd go reason
```

The argument is defined as follows:

`reason`

The reason given when stopping the server. If this field is an empty string, the server will be reenabled unconditionally. If a reason is given, only those functions disabled with a reason matching the supplied text will be restarted.

This command is used to restart a server function after a `throttle`, `pause`, or `reject` command.

**Display Status of a Newsfeed**

Use the following syntax to display the status of a newsfeed:

```bash
ctlinnd feedinfo site
```

The argument is defined as follows:

`sites`

The site name (taken from the `newsfeeds` file) for which you wish to display the newsfeed's status.

**Drop a Newsfeed**

Use the following syntax to drop a newsfeed:
ctlinnd drop site

The argument is defined as follows:

site

The name of the site (taken from the newsfeeds file) to which feeds are dropped. If this field is an empty string, all active feeds will be dropped.

Dropping a newsfeed to a site halts any active feeds to the site. It is not a permanent change. This command would be useful if you've modified the feed details for a site and a feed to that site is active.

**Begin a Newsfeed**

Use the following syntax to begin a newsfeed:

ctlinnd begin site

The argument is defined as follows:

site

The name of the site from the newsfeeds file to which feeds are started. If a feed to the site is already active, a drop command is done first automatically.

This command causes the server to reread the newsfeeds file, locate the matching entry, and commence a newsfeed to the named site using the details found. You can use this command to test a new news feed to a site after you've added or modified its entry in the newsfeeds file.

**Cancel an Article**

Use the following syntax to cancel an article:

ctlinnd cancel Message-Id

The argument is defined as follows:

Message-ID

The ID of the article to be cancelled.

This command causes the specified article to be deleted from the server. It does not generate a cancel message.
Chapter 24 - Newsreader Configuration

A newsreader is a program that users invoke to view, store, and create news articles. Several newsreaders have been ported to Linux. We will describe the basic setup for the three most popular newsreaders: tin, trn, and nn.

One of the most effective newsreaders is:

$ find /var/spool/news -name '[0-9]*' -exec cat {} \; | more

This is the way Unix die-hard read their news.

Most newsreaders, however, are much more sophisticated. They usually offer a full-screen interface with separate levels for displaying all groups the user has subscribed to, an overview of all articles in each group, and individual articles. Many web browsers double as newsreaders, but if you want to use a standalone newsreader, this chapter explains how to configure two classic ones: trn and nn.

At the newsgroup level, most newsreaders display a list of articles, showing their subject lines and authors. In big groups, it is difficult for the user to keep track of articles relating to each other, although it is possible to identify responses to earlier articles.

A response usually repeats the original article's subject, prepending it with Re:. Additionally, the References: header line should include the message ID of the article on which the response is directly following up. Sorting articles by these two criteria generates small clusters (in fact, trees) of articles, which are called threads.

One of the tasks of writing a newsreader is devising an efficient scheme of threading, because the time required for this is proportional to the square of the number of articles.

We will not go into how the user interfaces are built here. All newsreaders currently available for Linux have a good help function; please refer to it for more details.

In the following sections, we will deal only with administrative tasks. Most of these relate to the creation of threads databases and accounting.

**tin Configuration**

The most versatile newsreader with respect to threading is tin. It was written by Iain Lea and is loosely modeled on an older newsreader named tass (written by Rich Skrenta). It does its threading when the user enters the newsgroup, and it is pretty fast unless you're getting posts via NNTP.

On a 486DX50, it takes roughly 30 seconds to thread 1,000 articles when reading directly from disk. It would take more than 5 minutes over NNTP to reach a loaded news server. You may improve this time by regularly updating your index file by invoking tin with the -u option, so that when you next start tin to read news the threads already exist. Alternatively, you can invoke tin with the -U option to read news. When invoked this way, tin forks a background process to build the index files while you are reading news.

Usually, tin dumps its threading databases in the user's home directory below .tin/index. This may be costly in terms of resources, however, so you should keep a single copy of them in a central location. This may be achieved by making tin setuid to news, for example. tin will then keep all thread databases below /var/spool/news/index. For any file access or shell escape, it will reset its effective uid to the real uid of the user who invoked it.

The version of tin included in some Linux distributions is compiled without NNTP support, but most do have it now. When invoked as rtin or with the -r option, tin tries to connect to the NNTP server specified in the file /etc/nntpserver or in the NNTPSERVER environment variable. The nntpserver file simply contains the server's name on a single line.

---

140 Things improve drastically if the NNTP server does the threading itself and lets the client retrieve the threads databases; INN does this, for instance.

141 This is the reason why you will get ugly error messages when invoking tin as superuser. But you shouldn't do routine work as root, anyway.
**trn Configuration**

*trn* is also the successor to an older newsreader, namely *rn* (which means *read news*). The "t" in its name stands for "threaded." It was written by Wayne Davidson.

Unlike *tin*, *trn* has no provision for generating its threading database at runtime. Instead, it uses those prepared by a program called *mthreads* that has to be invoked regularly from *cron* to update the index files.

You can still access new articles if you're not running *mthreads*, but you will have all those "A GENUINE INVESTMENT OPPORTUNITY" articles scattered across your article selection menu, instead of a single thread you may easily skip.

To turn on threading for particular newsgroups, invoke *mthreads* with the list of newsgroups on the command line. The format of the list is the same as the one in the C News *sys* file:

```bash
$mthreads 'comp,rec,!rec.games.go'
```

This command enables threading for all of comp and rec, except for rec.games.go (people who play Go don't need fancy threads). After that, you simply invoke *mthreads* with no options at all to make it thread any newly arrived articles. Threading of all groups found in your *active* file can be turned on by invoking *mthreads* with a group list of all.

If you're receiving news during the night, you will customarily run *mthreads* once in the morning, but you can also to do so more frequently if necessary. Sites that have very heavy traffic may want to run *mthreads* in daemon mode. When it is started at boot time using the `-d` option, it puts itself in the background, wakes up every ten minutes to check if there are any newly arrived articles, and threads them. To run *mthreads* in daemon mode, put the following line in your *rc.news* script:

```bash
/usr/local/bin/rn/mthreads -deav
```

The `-a` option makes *mthreads* automatically turn on threading for new groups as they are created; `-v` enables verbose log messages to the *mthreads* log file *mt.log* in the directory where you have *trn* installed.

Old articles that are no longer available must be removed from the index files regularly. By default, only articles with a number below the low-water mark will be removed. Articles above this number that have been expired (because the oldest article has been assigned a long expiration date by an *Expires:* header field) may nevertheless be removed by giving *mthreads* the `-e` option to force an "enhanced" expiry run. When *mthreads* is running in daemon mode, the `-e` option makes *mthreads* put in such an enhanced expiry run once a day, shortly after midnight.

**nn Configuration**

*nn*, written by Kim F. Storm, claims to be a newsreader whose ultimate goal is not to read news. Its name stands for "No News," and its motto is "No news is good news. *nn* is better."

To achieve this ambitious goal, *nn* comes with a large assortment of maintenance tools that not only allow thread generation, but also extensive database consistency checks, accounting, gathering of usage statistics, and access restrictions. There is also an administration program called *nnadmin*, which allows you to perform these tasks interactively. It is very intuitive, so we will not dwell on these aspects, but deal only with the generation of the index files.

The *nn* threads database manager is called *nnmaster*. It is usually run as a daemon, started from an *rc* file at boot time. It is invoked as:

```bash
/usr/local/lib/nn/nnmaster -l -r -C
```

This enables threading for all newsgroups present in your *active* file.

Equivalently, you may invoke *nnmaster* periodically from *cron*, giving it a list of groups to act upon. This list is very similar to the subscription list in the *sys* file, except that it uses blanks instead of commas. Instead of

---

142 Note that C News (described in Chapter 21, *C News*) doesn't update this low-water mark automatically; you have to run *updatemin* to do so.
the fake group name all, an empty argument of "" should be used to denote all groups. A sample invocation looks like this:

```
# /usr/local/lib/nn/nnmaster !rec.games.go rec comp
```

Note that the order is significant. The leftmost group specification that matches always wins. Thus, if we had put !rec.games.go after rec, all articles from this group would have been threaded nevertheless.

nn offers several methods to remove expired articles from its databases. The first is to update the database by scanning the newsgroup directories and discarding the entries whose corresponding article has exceeded its expiration date. This is the default operation obtained by invoking nnmaster with the -E option. It is reasonably quick, unless you're doing this via NNTP.

The second method behaves exactly like a default expiration run of mthreads; it removes only those entries that refer to articles with numbers below the low-water mark in the active file. It may be enabled using the -e option.

Finally, the third strategy discards the entire database and recollects all articles. It may be enabled using the -E3 option.

The list of groups to be expired is given by the -F option in the same fashion as above. However, if you have nnmaster running as daemon, you must kill it (using -k) before expiration can take place, and restart it with the original options afterward. Thus the proper command to run expiration on all groups using the first method is:

```
# nnmaster -kF ""
# nnmaster -lrC
```

There are many more flags that fine-tune the nn's behavior. If you are concerned about removing bad articles or assembling article digests, read the nnmaster manual page.

nnmaster relies on a file named GROUPS, which is located in /var/lib/nn. If it does not exist when nnmaster is first run, it is created. For each newsgroup, it contains a line that begins with the group's name, optionally followed by a time stamp and flags. You may edit these flags to enable certain behavior for the group in question, but you may not change the order in which the groups appear. The flags allowed and their effects are detailed in the nnmaster manual page, too.

---

143 Their order has to agree with that of the entries in the (binary) MASTER file.
Appendix A

Example Network: The Virtual Brewery

Throughout this book we've used the following example that is a little less complex than Groucho Marx University and may be closer to the tasks you will actually encounter.

The Virtual Brewery is a small company that brews, as the name suggests, virtual beer. To manage their business more efficiently, the virtual brewers want to network their computers, which all happen to be PCs running the brightest and shiniest production Linux kernel. Figure A.1 shows the network configuration.

On the same floor, just across the hall, there’s the Virtual Winery, which works closely with the brewery. The vintners run an Ethernet of their own. Quite naturally, the two companies want to link their networks once they are operational. As a first step, they want to set up a gateway host that forwards datagrams between the two subnets. Later, they also want to have a UUCP link to the outside world, through which they exchange mail and news. In the long run, they also want to set up PPP connections to connect to offsite locations and to the Internet.

The Virtual Brewery and the Virtual Winery each have a class C subnet of the Brewery's class B network, and gateway to each other via the host vlager, which also supports the UUCP connection. Figure A.2 shows the configuration.

Figure A.1: The Virtual Brewery and Virtual Winery subnets

![Virtual Brewery and Virtual Winery subnets](image1)

Figure A.2: The Virtual Brewery Network

![Virtual Brewery Network](image2)

Connecting the Virtual Subsidiary Network

The Virtual Brewery grows and opens a branch in another city. The subsidiary runs an Ethernet of its own using the IP network number 172.16.3.0, which is subnet 3 of the Brewery's class B network. The host vlager acts as the gateway for the Brewery network and will support the PPP link, its peer at the new branch is called vbourbon and has an IP address of 172.16.3.1. This network is illustrated in Figure A.2.
Appendix B - Useful Cable Configurations

If you wish to connect two computers together and you don't have an Ethernet network, you will need either a serial null modem cable, or a PLIP parallel cable.

These cables can be bought off the shelf, but are much cheaper and fairly simple to make yourself.

**A PLIP Parallel Cable**

To make a parallel cable to use for PLIP, you will need two 25-pin connectors (called DB-25) and a cable with at least eleven conductors. The cable must not be any longer than 15 meters (50 feet). The cable may or may not have a shield, but if you are building a long cable, it is probably a good idea to have one.

If you look at the connector, you should be able to read tiny numbers at the base of each pin -- from 1 for the pin at the top left (if you hold the broader side up) to 25 for the pin at the bottom right. For the null printer cable, you have to connect the following pins of both connectors with each other, as shown in Figure B.1.

All remaining pins remain unconnected. If the cable is shielded, the shield should be connected to the DB-25’s metallic shell on just one end.

**A Serial NULL Modem Cable**

A serial null modem cable will work for both SLIP and PPP. Again, you will need two DB-25 connectors. This time your cable requires only eight conductors.

You may have seen other NULL modem cable designs, but this one allows you to use hardware flow control -- which is far superior to XON/XOFF flow control -- or none at all. The conductor configuration is shown in Figure B.2.

Again, if you have a shield, you should connect it to the first pin at one end only.

*Figure B.1: Parallel PLIP cable*
Figure B.2: Serial NULL-Modem cable

Pins face away from you, that is, into the page.
Appendix C - Copyright Information

Copyright © 1993 Olaf Kirch Copyright © 2000 Terry Dawson Copyright on O'Reilly printed version © 2000 O'Reilly & Associates

The online version of this book, which at this time of printing contains exactly the same text as the O'Reilly printed version, is available under the GNU FDL. Rights to reprint the document under the FDL include the right to print and distribute printed copies of the online version. Rights to copy the O'Reilly printed version are reserved. You can find the online copy of the license at http://www.oreilly.com/catalog/linag2/book/copyright.html. The book is available at http://www.linuxdoc.org/LDP/nag/nag.html and may be reposted by others at other locations.

Permission is granted to copy, print, distribute, and modify the online document under the terms of the GNU Free Documentation License, Version 1.1, or any later version published by the Free Software Foundation; with the Invariant Sections being the Acknowledgments (in the Preface) and Appendix C, Linux Network Administrator's Guide, Second Edition, Copyright Information. Further acknowledgments can be added outside the Invariant Section. The Front-Cover Text must read:

Linux Network Administrator's Guide
by Olaf Kirch and Terry Dawson
Copyright © 1993 Olaf Kirch
Copyright © 2000 Terry Dawson
Copyright on O'Reilly printed version © 2000 O'Reilly & Associates

The following is a copy of the GNU Free Documentation License, which is also at http://www.gnu.org/copyleft/fdl.html.

Version 1.1, March 2000
Copyright © 2000 Free Software Foundation, Inc.
59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The purpose of this License is to make a manual, textbook, or other written document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft," which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

Applicability and Definitions

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document," below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you."
A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not "Transparent" is called "Opaque."

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats that do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

**Verbatim Copying**

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

**Copying in Quantity**

If you publish printed copies of the Document numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.
If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

** Modifications **

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

1. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

2. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).

3. State on the Title page the name of the publisher of the Modified Version, as the publisher.

4. Preserve all the copyright notices of the Document.

5. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

6. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

7. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

8. Include an unaltered copy of this License.

9. Preserve the section entitled "History," and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

10. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

11. In any section entitled "Acknowledgements" or "Dedications," preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

12. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
13. Delete any section entitled "Endorsements." Such a section may not be included in the Modified Version.

14. Do not retitle any existing section as "Endorsements" or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section entitled "Endorsements," provided it contains nothing but endorsements of your Modified Version by various parties -- for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

**Combining Documents**

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled "History" in the various original documents, forming one section entitled "History"; likewise combine any sections entitled "Acknowledgements," and any sections entitled "Dedications." You must delete all sections entitled "Endorsements."

**Collections of Documents**

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

**Aggregation with Independent Works**

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggre-
gate,” and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document's Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

**Translation**

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

**Termination**

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

**Future Revisions of this License**

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See [http://www.gnu.org/copyleft/](http://www.gnu.org/copyleft/).

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.'
Appendix D

SAGE: The System Administrators Guild

If you are not getting everything you need from posting to comp.os.linux.* groups and reading documentation, maybe it's time to consider joining SAGE, the System Administrators Guild, sponsored by USENIX. The main goal of SAGE is to advance system administration as a profession. SAGE brings together system and network administrators to foster professional and technical development, share problems and solutions, and communicate with users, management, and vendors on system administration topics.

Current SAGE initiatives include:

- Co-sponsoring the highly successful annual System Administration Conferences (LISA) with USENIX.
- Publishing Job Descriptions for System Administrators, edited by Tina Darmohray, the first in a series of very practical booklets and resource guides covering system administration issues and techniques.
- Creating an archive site, ftp.sage.usenix.org, for papers from the System Administration Conferences and sysadmin-related documentation.
- Establishing working groups in areas important to system administrators, such as jobs, publications, policies, electronic information distribution, education, vendors, and standards.

To learn more about the USENIX Association and its Special Technical Group, SAGE, contact the USENIX Association office at (510) 528-8649 in the U.S., or by email to office@usenix.org. To receive information electronically, contact info@usenix.org. Annual SAGE membership is $25 (you must also be a member of USENIX). Members enjoy free subscriptions to login: and Computing Systems, a quarterly refereed technical journal; discounts on conference and symposia registration; and savings on SAGE publication purchases and other services.