4th Edition
Covers Version 5.0

The Complete FreeBSD®

Documentation from the Source

Greg Lehey

O'REILLY®
COMMUNITY PRESS

The Complete FreeBSD

BSD is one of the foundations of modern UNIX and UNIX-like systems, and has earned a reputation for unparalleled performance and stability. It is also the source of much Internet technology. FreeBSD, an open source operating system, is by far the most popular version of BSD.

The Complete FreeBSD is a classic text published for many years by FreeBSD distributer Walnut Creek. The book is an eminently practical guide that explains not only how to get a computer up and running with the FreeBSD operating system, but how to turn it into a highly functional and secure workstation or server that can host large numbers of users and disks, support remote access, and provide web services, mail service, and other key parts of the Internet infrastructure.

This edition is based on the Version 5.0 of FreeBSD, which includes a number of new features, notably improved support for multiprocessor systems. New topics include wireless LAN support, DHCP, proxy servers, the KDE desktop, and how to write CD-ROMs.

The book is written for novices as well as moderately knowledgeable readers who have picked up some experience using UNIX. The book covers a wide range of topics central to the system administration of hubs. Among the many topics are installation and updates, backups, printers, RAID, several Internet services, firewalls, and the graphical X Window System. Many readers over the years have praised this book for anticipating their problems and providing signposts pointing toward the solutions.

The author, **Greg Lehey**, has been developing, documenting and advocating FreeBSD for nearly 10 years. He is a member of the FreeBSD core team and is known for creating Vinum, a software RAID implementation for FreeBSD.

The O'Reilly Community Press spurs the development of interesting technologies by making online documentation available in print. Key players in technical communities create and edit the content of O'Reilly Community Press titles, and O'Reilly manufactures and distributes the books. Each book reflects the knowledge and voice of the community that has created it.

oreilly.com

US $54.99 CAN $68.99

ISBN: 978-0-596-00516-0

9 780596 005160 55499

O'REILLY®
COMMUNITY PRESS

The Complete FreeBSD®

The Complete FreeBSD®

FOURTH EDITION

Documentation from the Source

Greg Lehey

O'REILLY®
COMMUNITY PRESS
Beijing · Cambridge · Farnham · Köln · Sebastopol · Taipei · Tokyo

The Complete FreeBSD®, Fourth Edition
by Greg Lehey (*grog@freebsd.org*)

Printed in the United States of America.

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

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

Editor: Andy Oram

Production Editor: Linley Dolby

Cover Designer: Edie Freedman

Printing History:

May 2003: Fourth Edition.

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

ISBN: 0-596-00516-4

[M] [4/05]

Table of Contents

Foreword

I have been a long time developer of the Berkeley Software Distributions (BSD). My involvement started in 1976, at the University of California at Berkeley. I got drawn in as an office-mate of Bill Joy, who single-handedly wrote the code for BSD and then started handling its release. Bill went on to run the Computer Systems Research Group (CSRG) which developed and released the first fully complete BSD distributions. After Bill's departure to become a founder of Sun Microsystems, I eventually rose to head the CSRG and oversee the release of the freely redistributable 4.4BSD-Lite. The 4.4BSD-Lite distribution forms the basis for all the freely distributable variants of BSD today as well as providing many of the utilities found in Linux and commercial UNIX distributions.

With the release of 4.4BSD-Lite, the University of California at Berkeley ceased further BSD development. After considering the strengths and weaknesses of different BSD development groups, I decided to do my continued development in FreeBSD because it had the largest user community. For the past ten years, therefore, I have been a member of the FreeBSD developer team.

I have always felt that it is important to use your own product. For this reason, I have always run BSD everywhere: on my workstation, on my Web/file/mail/backup server, on my laptop, and on my firewall. By necessity, I have to find tools to do my job that will run on my BSD systems. It may be easier to just run Windows and PowerPoint to do your presentations, but there are an ever increasing number of fine alternatives out there that run on FreeBSD such as the open source OpenOffice.org suite or MagicPoint.

In the old days, there were not very many people working on the BSD software. This constraint on BSD development made it easy to keep up with what BSD could do and how to manage your system. But the last decade has seen an exponential growth in the open source movement. The result has been a huge increase in the number of people working on FreeBSD and an even larger increase in the number of applications and tools that have been ported to run on FreeBSD. It has become a more than full time job just to keep track of all the system capabilities, let alone to figure out how to use them all.

Greg Lehey has done a wonderful job with this book of helping those of us that want to fully utilize the FreeBSD system to do so without having to devote our entire lives

figuring how. He has gone through and figured out each of the different tasks that you might ask your system to do. He has identified the software that you need to do the task. He explains how to configure it for your operational needs. He tells you how to monitor the resulting subsystem to make sure it is working as desired. And, he helps you to identify and fix problems that arise.

The book starts with the basics of getting the FreeBSD system up and running on your hardware, including laptops, workstations, and servers. It then explains how to customize an installation for your personal needs. This personalization includes downloading and operating the most important of the more than 8000 software packages in the FreeBSD ports collection. The book also includes a very comprehensive set of systems administration information, including the setup and operation of printers, local and external networking, the domain name system, the NFS and Samba remote filesystems, electronic mail, web surfing and hosting, and dial-up for FAX, remote login, and point-to-point network connections.

In short, this book provides everything you need to know about the FreeBSD system from the day you first pick up the software through the day you have a full suite of machines. It covers your complete range of computing needs. There is a reason that this book is so popular: as its title says, it is The Complete FreeBSD. I am very happy to see this revision which once again fulfills that mandate.

Marshall Kirk McKusick
Berkeley, California
February 2003

In this chapter:
- *The fourth edition*
- *Conventions used in this book*
- *Acknowledgments*
- *How this book was written*

Preface

FreeBSD is a high-performance operating system derived from the *Berkeley Software Distribution* (*BSD*), the version of UNIX developed at the University of California at Berkeley between 1975 and 1993. FreeBSD is not a UNIX clone. Historically and technically, it has greater rights than UNIX System V to be called *UNIX*. Legally, it may not be called UNIX, since UNIX is now a registered trade mark of The Open Group.

This book is intended to help you get FreeBSD up and running on your system and to familiarize you with it. It can't do everything, but plenty of UNIX books and online documentation are available, and a large proportion of them are directly applicable to FreeBSD. In the course of the text, I'll repeatedly point you to other documentation.

I'm not expecting you to be a guru, but I do expect you to understand the basics of using UNIX. If you've come from a Microsoft background, I'll try to make the transition a little less rocky.

The fourth edition

This book has already had quite a history. Depending on the way you count, this is the fourth or fifth edition of *The Complete FreeBSD*: the first edition of the book was called *Installing and Running FreeBSD*, and was published in March 1996. The next edition was called ''The Complete FreeBSD'', first edition. The first three editions were published by Walnut Creek CDROM, which ceased publishing activities in 2000. This is the first edition to be published by O'Reilly and Associates.

During this time, FreeBSD has changed continually, and it's difficult for a book to keep up with the change. This doesn't mean that FreeBSD has changed beyond recognition, but people have done a great job of working away those little rough edges that make the difference between a usable operating system and one that is a pleasure to use. If you come to FreeBSD from System V, you'll certainly notice the difference.

During the lifetimes of the previous editions of this book, I realised that much of the text becomes obsolete very quickly. For example, in the first edition I went to a lot of trouble

to tell people how to install from an ATAPI CD-ROM, since at the time the support was a little wobbly. Almost before the book was released, the FreeBSD team improved the support and rolled it into the base release. The result? Lots of mail messages to the `FreeBSD-questions` mailing list saying, ‘‘Where can I get *ATAPI.FLP*?’’. Even the frequently posted errata list didn’t help much.

This kind of occurrence brings home the difference in time scale between software releases and book publication. FreeBSD CD-ROMs are released several times a year. A new edition of a book every year is considered very frequent, but it obviously can’t hope to keep up with the software release cycle. As a result, this book contains less time-sensitive material than previous editions. For example, the chapter on building kernels no longer contains an in-depth discussion of the kernel build parameters. They change too frequently, and the descriptions, though correct at the time of printing, would just be confusing. Instead, the chapter now explains where to find the up-to-date information.

Another thing that we discovered was that the book was too big. The second edition contained 1,100 pages of man pages, the FreeBSD manual pages that are also installed online on the system. These printed pages were easier to read, but they had two disadvantages: firstly they were slightly out of date compared to the online version, and secondly they weighed about 1 kilogram (2.2 lbs). The book was just plain unwieldy, and some people reported that they had physically torn out the man pages from the book to make it more manageable. As a result, the third edition had only the most necessary man pages.

Times have changed since then. At the time, *The Complete FreeBSD* was the only English-language book on FreeBSD. Now there are several—see Appendix A, *Bibliography*, for more detail. In particular, the FreeBSD online handbook is available both in printed form and online at *http://www.freebsd.org/handbook/index.html*, so I have left much of the more time-sensitive issues out of this book. See the online handbook instead. Alternatively, you can print out the man pages yourself—see page 15 for details.

It’s very difficult to find a good sequence for presenting that material in this book. In many cases, there is a chicken and egg problem: what do you need to know first? Depending on what you need to do, you need to get information in different sequences. I’ve spent a lot of time trying to present the material in the best possible sequence, but inevitably you’re going to find that you’ll have to jump through one of the myriad page cross references.

Conventions used in this book

In this book, I use **bold** for the names of keys on the keyboard. We’ll see more about this in the next section.

I use *italic* for the names of UNIX utilities, directories, file names and *URI*s (*Uniform Resource Identifier*, the file naming technology of the World Wide Web), and to emphasize new terms and concepts when they are first introduced. I also use this font for comments in the examples.

I use `constant width` in examples to show the contents of files, the output from commands, program variables, actual values of keywords, for mail IDs, for the names of *Internet News* newsgroups, and in the text to represent commands.

I use *`constant width italic`* in examples to show variables for which context-specific substitutions should be made. For example, the variable *`filename`* would be replaced by an actual file name.

I use **`constant width bold`** in examples to show commands or text that would be typed in literally by the user.

In this book, I recommend the use of the Bourne shell or one of its descendents (*sh*, *bash*, *pdksh*, *ksh* or *zsh*). *sh* is in the base system, and the rest are all in the Ports Collection, which we'll look at in chapter 9. I personally use the *bash* shell. This is a personal preference, and a recommendation, but it's not the standard shell: the traditional BSD shell is the C shell (*csh*), which FreeBSD has replaced with a fuller-featured descendent, *tcsh*. In particular, the standard installation sets the `root` user up with a *csh*. See page 136 for details of how to change the shell.

In most examples, I'll show the shell prompt as `$`, but it doesn't normally matter which shell you use. In some cases, however, it does:

- Sometimes you need to be super-user, the user who can do anything. If this is necessary, I indicate it by using the prompt #.
- Sometimes the commands only work with the Bourne Shell and derivatives (*zsh*, *bash*), and they won't work with *csh*, *tcsh* and friends. In these cases I'll show the *csh* alternative with the standard *csh* prompt `%`.

> In the course of the text I'll occasionally touch on a subject that is not of absolute importance, but that may be of interest. I'll print such notes in smaller text, like this.

Describing the keyboard

One of the big differences between UNIX and other operating systems concerns the way they treat so-called ''carriage control codes.'' When UNIX was written, the standard interactive terminal was still the Teletype model KSR 35. This mechanical monstrosity printed at 10 characters per second, and the carriage control characters really did cause physical motion of the carriage. The two most important characters were *Carriage Return*, which moved the carriage (which carried the print head) to the left margin, and *Line Feed*, which turned the platen to advance the paper by the height of a line. To get to the beginning of a new line, you needed to issue both control characters. We don't have platens or carriages any more, but the characters are still there, and in many systems, including Microsoft, a line of text is terminated by a carriage return character and a line feed character. UNIX only uses a ''new line'' character, which corresponds to the line feed. This difference sometimes gives rise to confusion. We'll look at it in more detail on page 267.

It's surprising how many confusing terms exist to describe individual keys on the keyboard. My favourite is the *any* key ("`Press any key to continue`"). We won't be using the *any* key in this book, but there are a number of other keys whose names need understanding:

- The *Enter* or *Return* key. I'll call this **ENTER**.
- Control characters (characters produced by holding down the **Ctrl** key and pressing a normal keyboard key at the same time). I'll show them as, for example, **Ctrl-D** in the text, but these characters are frequently echoed on the screen as a caret (^) followed by the character entered, so in the examples, you may see things like `^D`.
- The **Alt** key, which Emacs aficionados call a **META** key, works in the same way as the **Ctrl** key, but it generates a different set of characters. These are sometimes abbreviated by prefixing the character with a tilde (~) or the characters **A-**. I personally like this method better, but to avoid confusion I'll represent the character generated by holding down the **Alt** key and pressing **D** as **Alt-D**.
- **NL** is the *new line* character. In ASCII, it is **Ctrl-J**, but UNIX systems generate it when you press the **ENTER** key. UNIX also refers to this character as `\n`, a usage which comes from the C programming language.
- **CR** is the *carriage return* character, in ASCII **Ctrl-M**. Most systems generate it with the **ENTER** key. UNIX also refers to this character as `\r`—again, this comes from the C programming language.
- **HT** is the ASCII *horizontal tab* character, **Ctrl-I**. Most systems generate it when the **TAB** key is pressed. UNIX and C also refer to this character as `\t`.

Acknowledgments

This book is based on the work of many people, first and foremost the FreeBSD documentation project. Years ago, I took significant parts from the FreeBSD handbook, in particular Chapter 7, *The tools of the trade*. The FreeBSD handbook is supplied as online documentation with the FreeBSD release—see page 12 for more information. It is subject to the BSD documentation license, a variant of the BSD software license.

Book reviewers

This book wouldn’t be the same without the help of a small group of dedicated critics who tried out what I said and pointed out that it didn’t work. In particular, I’d like to thank Jack Velte of Walnut Creek CDROM, who had the idea of this book in the first place, Jordan Hubbard and Gary Palmer for tearing the structure and content apart multiple times, and also Bob Bishop, Julian Elischer, Stefan Esser, John Fieber, Glen Foster, Poul-Henning Kamp, Michael Smith, and Nate Williams for valuable contributions (‘‘What, you expect new users to know that you have to shut down the machine before powering it off?’’).[1] Finally, special thanks to Josef Möllers, Andreas Ritter, and Jack Velte, who put early drafts of this book through its paces and actually installed FreeBSD with their help.

The second edition had much more review than the first. A number of dedicated reviewers held through for several months as I gradually cranked out usable copy. In particular, special thanks to Annelise Anderson, Sue Blake, Jonathan M. Bresler, William Bulley, Mike Cambria, Brian Clapper, Paul Coyne, Lee Crites, Jerry Dunham, Stefan Esser, Patrick Gardella, Gianmarco Giovannelli, David Kelly, Andreas Klemm, Andrew MacIntyre, Jonathan Michaels, Jörg Micheel, Marco Molteni, Charles Mott, Jay D. Nelson, Daniel J. O’Connor, Andrew Perry, Kai Peters, Wes Peters, Mark Prior, Guido van Rooij, Andrew Rutherford, Thomas Vickery and Don Wilde.

Many of the second edition reviewers came back for the third edition. In addition, thanks to John Birrell for his help with the Alpha architecture, and Michael A. Endsley for ferreting out bugs, some of which had been present since the days of *Installing and Running FreeBSD*.

The following people helped with the fourth edition: Annelise Anderson, Jonathan Arnold, Sue Blake, Doug Barton, Brian Clapper, Jerry Dunham, Matt Geddes, Jeremiah Gowdy, Daniel B. Hemmerich, Justin Heath, Peter N. M. Hansteen, Paul A. Hoadley, Ed Irvine, John Lind, Johannes Lochmann, Warner Losh, Yin Cheung ‘Yogesh’ Mar, Andrew MacIntyre, Jonathan Michaels, Ove Ruben R. Olsen, Hiten Pandya, Linh Pham, Daniel Phillips, Siegfried P Pietralla. Stephen J. Roznowski, Dan Shearer and Murray Stokely.

In addition, my thanks to the people at O’Reilly and Associates, particularly Andy Oram,

1. See page 541 for details on how to shut down the system.

with whom I had discussed this project for years before he was finally able to persuade O'Reilly that it was a good idea. Subsequently it was Andy who coordinated seeing this rather unusual project through O'Reilly channels. Emma Colby designed the cover, and David Futato provided specifications, advice, and examples for the format. Linley Dolby proofread the document after I thought it was ready, and found tens of mistakes on nearly every page, ensuring that the book is better than its predecessors.

Finally, thanks to David Lloyd for the loan of an ATA CD-R drive while writing the ATA section of Chapter 13, *Writing CD-Rs*.

How this book was written

This book was written and typeset entirely with tools supplied as standard with the FreeBSD system, including the Ports Collection. The text of this book was written with the GNU *Emacs* editor, and it was formatted on 30 June 2003 with the GNU *groff* text formatter, Version 1.18, and some heavily modified *mm* macros. The process was performed under FreeBSD 5.0-CURRENT. Even the development versions of FreeBSD are stable enough to perform heavy-duty work like professional text formatting.

The source files for this book are kept under *RCS*, the *Revision Control System* (see the man page *rcs(1)*). Here are the RCS Version IDs for the chapters of this particular book.

```
$Id: title.complete,v 4.2 2003/04/09 19:43:58 grog Exp grog $
$Id: preface.mm,v 4.21 2003/05/25 09:02:19 grog Exp $
$Id: introduction.mm,v 4.26 2003/06/30 06:47:54 grog Exp $
$Id: concepts.mm,v 4.21 2003/04/02 06:37:12 grog Exp $
$Id: quickinstall.mm,v 4.11 2003/04/09 19:26:40 grog Exp $
$Id: shareinstall.mm,v 4.12 2003/04/09 19:26:50 grog Exp $
$Id: install.mm,v 4.22 2003/06/29 04:34:08 grog Exp $
$Id: postinstall.mm,v 4.13 2003/06/29 04:30:44 grog Exp $
$Id: unixref.mm,v 4.16 2003/04/02 06:41:29 grog Exp $
$Id: unixadmin.mm,v 4.13 2003/04/02 06:50:29 grog Exp $
$Id: ports.mm,v 4.12 2003/04/02 06:43:08 grog Exp $
$Id: filesys.mm,v 4.17 2003/04/02 06:43:57 grog Exp $
$Id: disks.mm,v 4.19 2003/06/29 02:54:00 grog Exp $
$Id: vinum.mm,v 4.20 2003/06/29 04:33:42 grog Exp $
$Id: burncd.mm,v 4.14 2003/06/29 04:33:03 grog Exp $
$Id: tapes.mm,v 4.12 2003/06/29 03:06:33 grog Exp $
$Id: printers.mm,v 4.17 2003/04/02 06:48:05 grog Exp $
$Id: netintro.mm,v 4.16 2003/04/02 06:48:55 grog Exp $
$Id: netsetup.mm,v 4.21 2003/06/29 09:05:45 grog Exp $
$Id: isp.mm,v 4.10 2003/04/02 03:09:55 grog Exp $
$Id: modems.mm,v 4.10 2003/04/02 03:11:02 grog Exp $
$Id: ppp.mm,v 4.14 2003/04/02 08:14:21 grog Exp $
$Id: dns.mm,v 4.19 2003/04/02 08:43:25 grog Exp $
$Id: firewall.mm,v 4.13 2003/06/29 04:25:08 grog Exp $
$Id: netdebug.mm,v 4.17 2003/04/03 02:04:14 grog Exp $
$Id: netclient.mm,v 4.18 2003/06/30 06:12:58 grog Exp $
$Id: netserver.mm,v 4.19 2003/04/09 20:42:40 grog Exp $
$Id: mua.mm,v 4.15 2003/04/03 02:07:47 grog Exp $
$Id: mta.mm,v 4.16 2003/04/03 01:18:20 grog Exp $
$Id: xtheory.mm,v 4.14 2003/05/18 02:19:19 grog Exp $
```

```
$Id: starting.mm,v 4.23 2003/06/29 03:13:08 grog Exp $
$Id: configfiles.mm,v 4.19 2003/06/29 04:32:34 grog Exp $
$Id: current.mm,v 4.18 2003/06/29 04:29:20 grog Exp $
$Id: upgrading.mm,v 4.13 2003/06/30 06:52:25 grog Exp $
$Id: building.mm,v 4.18 2003/06/29 04:34:40 grog Exp $
$Id: biblio.mm,v 4.8 2003/06/29 06:27:59 grog Exp $
$Id: evolution.mm,v 4.13 2003/04/02 04:59:47 grog Exp grog $
$Id: tmac.Mn,v 1.18 2003/06/24 07:04:00 grog Exp grog $
```

In this chapter:

- *How to use this book*
- *FreeBSD features*
- *Licensing conditions*
- *A little history*
- *Other free UNIX-like operating systems*
- *FreeBSD system documentation*
- *Other documentation on FreeBSD*
- *The FreeBSD community*
- *Mailing lists*
- *The Berkeley daemon*

1

Introduction

FreeBSD is a free operating system derived from AT&T's *UNIX* operating system.[1] It runs on the following platforms:

- Computers based on the Intel i386 CPU architecture, including the 386, 486 and Pentium families of processors, and compatible CPUs from AMD and Cyrix.
- The Compaq/Digital Alpha processor.
- 64 bit SPARC machines from Sun Microsystems.
- In addition, significant development efforts are going towards porting FreeBSD to other hardware, notably the Intel 64 bit architecture and the IBM/Motorola PowerPC architecture.

This book describes the released versions of FreeBSD for Intel and Alpha processors. Current support for SPARC 64 processors is changing too fast for it to be practical to give details specific to this processor, but nearly everything in this book also applies to SPARC 64.

1. FreeBSD no longer contains any AT&T proprietary code, so it may be distributed freely. See page 7 for more details.

How to use this book

This book is intended for a number of different audiences. It attempts to present the material without too many forward references. It contains the following parts:

1. The first part, Chapters 1 to 6, tells you how to install FreeBSD and what to do if things go wrong.
2. Chapters 7 to 15 introduce you to life with FreeBSD, including setting up optional features and system administration.
3. Chapters 16 to 27 introduce you to FreeBSD's rich network support.
4. Finally, Chapters 28 to 33 look at system administration topics that build on all the preceding material.

In more detail, we'll discuss the following subjects:

- In the rest of this chapter, we'll look at what FreeBSD is, what you need to run it, and what resources are available, including FreeBSD's features and history, how it compares to other free UNIX-like operating systems, other sources of information about FreeBSD, the world-wide FreeBSD community, and support for FreeBSD. In addition, we'll look at the BSD's daemon emblem.
- Chapter 2, *Before you install*, discusses the installation requirements and theoretical background of installing FreeBSD.
- Chapter 3, *Quick installation*, presents a quick overview of the installation process. If you're reasonably experienced, this may be all you need to install FreeBSD.
- In Chapter 4, *Shared OS installation*, we'll look at preparing to install FreeBSD on a system that already contains another operating system.
- In Chapter 5, *Installing FreeBSD*, we'll walk through a typical installation in detail.
- Chapter 6, *Post-installation configuration*, explains the configuration you need to do after installation to get a complete functional system.
- Chapter 7, *The tools of the trade*, presents a number of aspects of FreeBSD that are of interest to newcomers (particularly from a Microsoft environment). We'll look at setting up a ''desktop,'' the concept of *users* and file naming. We'll also consider the basics of using the *shell* and editor, and how to shut down the machine.
- Chapter 8, *Taking control*, goes into more detail about the specifics of working with UNIX, such as processes, daemons, timekeeping and log files. We'll also look at features unique to FreeBSD, including multiple processor support, removable I/O devices and emulating other systems.
- Chapter 9, *The Ports Collection*, describes the thousands of free software packages that you can optionally install on a FreeBSD system.

- Chapter 10, *File systems and devices*, contains information about the FreeBSD directory structure and device names. You'll find the section on device names (starting on page 195) interesting even if you're an experienced UNIX hacker.
- Chapter 11, *Disks*, describes how to format and integrate hard disks, and how to handle disk errors.
- Managing disks can be a complicated affair. Chapter 12, *The Vinum Volume Manager*, describes a way of managing disk storage.
- In Chapter 13, *Writing CD-Rs*, we'll look at how to use FreeBSD to write CD-Rs.
- FreeBSD provides professional, reliable data backup services as part of the base system. Don't ever let yourself lose data because of inadequate backup provisions. Read all about it in Chapter 14, *Tapes, backups and floppy disks*.
- Chapter 15, *Printers*, describes the BSD spooling system and how to use it both on local and networked systems.
- Starting at Chapter 16, *Networks and the Internet*, we'll look at the Internet and the more important services.
- Chapter 17, *Configuring the local network*, describes how to set up local networking.
- Chapter 18, *Connecting to the Internet*, discusses the issues in selecting an Internet Service Provider (ISP) and establishing a presence on the Internet.
- Chapter 19, *Serial communications*, discusses serial hardware and the prerequisites for PPP and SLIP communications.
- In Chapter 20, *Configuring PPP*, we look at FreeBSD's two PPP implementations and what it takes to set them up.
- In Chapter 21, *The Domain Name Service*, we'll consider the use of names on the Internet.
- Security is an increasing problem on the Internet. In Chapter 22, *Firewalls, IP aliasing and proxies*, we'll look at some things we can do to improve it. We'll also look at *IP aliasing*, since it goes hand-in-hand with firewalls, and *proxy servers*.
- Networks sometimes become *notwork*s. In Chapter 23, *Network debugging*, we'll see what we can do to solve network problems.
- Chapter 24, *Basic network access: clients*, describes the client viewpoint of network access, including Web browsers, *ssh*, *ftp*, *rsync* and *nfs* clients for sharing file systems between networked computers.
- Network clients talk to network servers. We'll look at the corresponding server viewpoint in Chapter 25, *Basic network access: servers*.
- Despite the World Wide Web, traditional two-way personal communication is still very popular. We'll look at how to use mail clients in Chapter 26, *Electronic mail: clients*.

- Mail servers are an important enough topic that there's a separate Chapter 27, *Electronic mail: servers*.
- In Chapter 28, *XFree86 in depth*, we'll look at the theory behind getting X11 working.
- Chapter 29, *Starting and stopping the system*, describes how to start and stop a FreeBSD system and all the things you can do to customize it.
- In Chapter 30, *FreeBSD configuration files*, we'll look at the more common configuration files and what they should contain.
- In Chapter 31, *Keeping up to date*, we'll discuss how to ensure that your system is always running the most appropriate version of FreeBSD.
- FreeBSD keeps changing. We'll look at some aspects of what that means to you in Chapter 32, *Updating the system software*.
- Chapter 33, *Custom kernels*, discusses optional kernel features.
- Appendix A, *Bibliography*, suggests some books for further reading.
- Appendix B, *The evolution of FreeBSD*, describes the changes that have taken place in FreeBSD since it was introduced nearly ten years ago.

FreeBSD features

FreeBSD is derived from *Berkeley UNIX*, the flavour of UNIX developed by the Computer Systems Research Group at the University of California at Berkeley and previously released as the *Berkeley Software Distribution* (BSD) of UNIX.

> UNIX is a registered trademark of the Open Group, so legally, FreeBSD may not be called UNIX. The technical issues are different, of course; make up your own mind as to how much difference this makes.

Like commercial UNIX, FreeBSD provides you with many advanced features, including:

- FreeBSD uses *preemptive multitasking* with dynamic priority adjustment to ensure smooth and fair sharing of the computer between applications and users.
- FreeBSD is a *multi-user system*: many people can use a FreeBSD system simultaneously for unrelated purposes. The system shares peripherals such as printers and tape drives properly between all users on the system.

  Don't get this confused with the ''multitasking'' offered by some commercial systems. FreeBSD is a true multi-user system that protects users from each other.
- FreeBSD is secure. Its track record is borne out by the reports of the *CERT*, the leading organization dealing with computer security. See *http://www.cert.org/* for more information. The FreeBSD project has a team of security officers concerned with maintaining this lead.

- FreeBSD is reliable. It is used by ISPs around the world. FreeBSD systems regularly go several years without rebooting. FreeBSD can fail, of course, but the main causes of outages are power failures and catastrophic hardware failures.
- FreeBSD provides a complete *TCP/IP networking* implementation. This means that your FreeBSD machine can interoperate easily with other systems and also act as an enterprise server, providing vital functions such as NFS (remote file access) and electronic mail services, or putting your organization on the Internet with WWW, FTP, routing and firewall services. In addition, the Ports Collection includes software for communicating with proprietary protocols.
- *Memory protection* ensures that neither applications nor users can interfere with each other. If an application crashes, it cannot affect other running applications.
- FreeBSD includes the *XFree86* implementation of the *X11* graphical user interface.
- FreeBSD can run most programs built for versions of SCO UNIX and UnixWare, Solaris, BSD/OS, NetBSD, 386BSD and Linux on the same hardware platform.
- The FreeBSD Ports Collection includes thousands of ready-to-run applications.
- Thousands of additional and easy-to-port applications are available on the Internet. FreeBSD is source code compatible with most popular commercial UNIX systems and thus most applications require few, if any, changes to compile. Most freely available software was developed on BSD-like systems. As a result, FreeBSD is one of the easiest platforms you can port to.
- Demand paged *virtual memory* (*VM*) and ‘‘merged VM/buffer cache’’ design efficiently satisfies applications with large appetites for memory while still maintaining interactive response to other users.
- The base system contains a full complement of C, C++ and FORTRAN development tools. All commonly available programming languages, such as *perl*, *python* and *ruby*, are available. Many additional languages for advanced research and development are also available in the Ports Collection.
- FreeBSD provides the complete *source code* for the entire system, so you have the greatest degree of control over your environment. The licensing terms are the freest that you will find anywhere (‘‘Hey, use it, don’t pretend you wrote it, don’t complain to us if you have problems’’). Those are just the licensing conditions, of course. As we’ll see later in the chapter, there are plenty of people prepared to help if you run into trouble.
- Extensive *online documentation*, including traditional *man pages* and a hypertext-based *online handbook*.

FreeBSD is based on the 4.4BSD UNIX released by the Computer Systems Research Group (CSRG) at the University of California at Berkeley. The FreeBSD Project has spent many thousands of hours fine-tuning the system for maximum performance and reliability. FreeBSD’s features, performance and reliability compare very favourably with those of commercial operating systems.

Since the source code is available, you can easily customize it for special applications or projects, in ways not generally possible with operating systems from commercial vendors. You can easily start out small with an inexpensive 386 class PC and upgrade as your needs grow. Here are a few of the applications in which people currently use FreeBSD:

- *Internet Services*: the Internet grew up around Berkeley UNIX. The original TCP/IP implementation, released in 1982, was based on 4.2BSD, and nearly every current TCP/IP implementation has borrowed from it. FreeBSD is a descendent of this implementation, which has been maintained and polished for decades. It is the most mature and reliable TCP/IP available at any price. This makes it an ideal platform for a variety of Internet services such as FTP servers, World Wide Web servers, electronic mail servers, USENET news servers, DNS name servers and firewalls. With the *Samba* suite, you can replace a Microsoft file server.
- *Education:* FreeBSD is an ideal way to learn about operating systems, computer architecture and networking. A number of freely available CAD, mathematical and graphic design packages also make it highly useful to those whose primary interest in a computer is to get *other* work done.
- *Research:* FreeBSD is an excellent platform for research in operating systems as well as other branches of computer science, since the source code for the entire system is available. FreeBSD's free availability also makes it possible for remote groups to collaborate on ideas or shared development without having to worry about special licensing agreements or limitations on what may be discussed in open forums.
- *X Window workstation:* FreeBSD makes an excellent choice for an inexpensive graphical desktop solution. Unlike an X terminal, FreeBSD allows many applications to be run locally, if desired, thus relieving the burden on a central server. FreeBSD can even boot ''diskless,'' making individual workstations even cheaper and easier to administer.
- *Software Development:* The basic FreeBSD system comes with a full complement of development tools including the renowned GNU C/C++ compiler and debugger.

Licensing conditions

As the name suggests, FreeBSD is free. You don't have to pay for the code, you can use it on as many computers as you want, and you can give away copies to your friends. There are some restrictions, however. Here's the BSD license as used for all new FreeBSD code:

> Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
>
> 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

This software is provided by the FreeBSD project ‘‘as is’’ and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall the FreeBSD project or contributors be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this software, even if advised of the possibility of such damage.

The last paragraph is traditionally written in ALL CAPS, for reasons which don’t seem to have anything to do with the meaning. Older versions of the license also contained additional clauses relating to advertising.

A little history

FreeBSD is a labour of love: big commercial companies produce operating systems and charge lots of money for them; the FreeBSD project produces a professional-quality operating system and gives it away. That’s not the only difference.

In 1981, when IBM introduced their Personal Computer, the microprocessor industry was still in its infancy. They entrusted Microsoft to supply the operating system. Microsoft already had their own version of UNIX, called XENIX, but the PC had a minimum of 16 kB and no disk. UNIX was not an appropriate match for this hardware. Microsoft went looking for something simpler. The ‘‘operating system’’ they chose was correspondingly primitive: 86/DOS, a clone of Digital Research’s successful CP/M operating system, written by Tim Paterson of Seattle Computer Products and originally called *QDOS* (*Quick and Dirty Operating System*). At the time, it seemed just the thing: it ran fine without a hard disk (in fact, the original PC didn’t *have* a hard disk, not even as an option), and it didn’t use up too much memory. The only thing that they really had to do was to change the name. IBM called its version PC-DOS, while Microsoft marketed its version under the name MS-DOS.

By this time, a little further down the US West Coast, the Computer Systems Research Group (CSRG) of the University of California at Berkeley had just modified AT&T’s UNIX operating system to run on the new DEC VAX 11/780 machine, which sported virtual memory, and had turned their attention to implementing some new protocols for the ARPANET: the so-called *Internet Protocols*. The version of UNIX that they had developed was now sufficiently different from AT&T’s system that it had been dubbed *Berkeley UNIX*.

As time went on, both MS-DOS and UNIX evolved. Before long, MS-DOS was modified to handle hard disks—not well, but it handled them, and for the PC users, it was

so much better than what they had before that they ignored the inefficiencies. After all, the PC gave you your own hard disk on your desk, and you didn't have to share it with all the other people in the department. Microsoft even tried to emulate the UNIX directory structure, but succeeded only in implementing the concept of nested directories. At Berkeley, they were developing a higher performance disk subsystem, the *Fast File System*, now known as the *UNIX File System*.

By the late 80s, it was evident that Microsoft no longer intended to substantially enhance MS-DOS. New processors with support for multitasking and virtual memory had replaced the old Intel 8088 processor of the IBM PC, but they still ran MS-DOS by emulating the 8088 processor, which was now completely obsolete. The 640 kB memory limit of the original PC, which once appeared bigger than anybody would ever need, became a serious problem. In addition, people wanted to do more than one thing at a time with their computers.

A solution to both problems was obvious: move to the 32 bit address mode of the new Intel 80386 processor and introduce real multitasking, which operating systems on larger machines had had for decades. Of course, these larger machines were only physically larger. The average PC of 1990 had more memory, more disk and more processing power than just about any of the large computers of the 70s. Nevertheless, Microsoft didn't solve these problems for its ''Windows'' platform until much later, and the solutions still leave a lot to be desired.

UNIX, on the other hand, was a relatively mature operating system at the time when the PC was introduced. As a result, Microsoft-based environments have had little influence on the development of UNIX. UNIX development was determined by other factors: changes in legal regulations in the USA between 1977 and 1984 enabled AT&T first to license UNIX to other vendors, noticeably Microsoft, who announced XENIX in 1981, and then to market its own version of UNIX. AT&T developed System III in 1982, and System V in 1983. The differences between XENIX and System V were initially small, but they grew: by the mid-80s, there were four different versions of UNIX: the *Research Version*, used almost only inside AT&T, which from the eighth edition on derived from 4.1cBSD, the *Berkeley Software Distribution* (BSD) from Berkeley, the commercial *System V* from AT&T, and XENIX, which no longer interested Microsoft, and was marketed by the company that had developed it, the *Santa Cruz Operation*, or *SCO*.

One casualty of UNIX's maturity was the CSRG in Berkeley. UNIX was too mature to be considered an object of research, and the writing was on the wall: the CSRG would close down. Some people decided to port Berkeley UNIX to the PC—after all, SCO had ported its version of UNIX to the PC years earlier. In the Berkeley tradition, however, they wanted to give it away. The industry's reaction was not friendly. In 1992, AT&T's subsidiary *USL* (*UNIX Systems Laboratories*) filed a lawsuit against *Berkeley Software Design, Inc.* (*BSDI*), the manufacturer of the BSD/386 operating system, later called BSD/OS, a system very similar to FreeBSD. They alleged distribution of AT&T source code in violation of licence agreements. They subsequently extended the case to the University of California at Berkeley. The suit was settled out of court, and the exact conditions were not all disclosed. The only one that became public was that BSDI would migrate their source base to the newer 4.4BSD-Lite sources, a thing that they were

preparing to do in any case. Although not involved in the litigation, it was suggested to FreeBSD that they should also move to 4.4BSD-Lite, which was done with the release of FreeBSD release 2.0 in late 1994.

Now, in the early 21st century, FreeBSD is the best known of the BSD operating systems, one that many consider to follow in the tradition of the CSRG. I can think of no greater honour for the development team. It was developed on a shoestring budget, yet it manages to outperform commercial operating systems by an order of magnitude.

The end of the UNIX wars

In the course of the FreeBSD project, a number of things have changed about UNIX. Sun Microsystems moved from a BSD base to a System V base in the late 80s, a move that convinced many people that BSD was dead and that System V was the future. Things turned out differently: in 1992, AT&T sold USL to Novell, Inc., who had introduced a product based on System V.4 called UnixWare. Although UnixWare has much better specifications than SCO's old System V.3 UNIX, it was never a success, and Novell finally sold their UNIX operation to SCO. SCO itself was then bought out by Caldera (which recently changed its name back to SCO), while the ownership of the UNIX trade mark has passed to the Open Group. System V UNIX is essentially dead: current commercial versions of UNIX have evolved so far since System V that they can't be considered the same system. By contrast, BSD is alive and healthy, and lives on in FreeBSD, NetBSD, OpenBSD and Apple's Mac OS X.

The importance of the AT&T code in the earlier versions of FreeBSD was certainly overemphasized in the lawsuit. All of the disputed code was over 10 years old at the time, and none of it was of great importance. In January 2002, Caldera released all ''ancient'' versions of UNIX under a BSD license. These specifically included all versions of UNIX from which BSD was derived: the first to seventh editions of Research UNIX and 32V, the predecessor to 3BSD. As a result, all versions of BSD, including those over which the lawsuit was conducted, are now freely available.

Other free UNIX-like operating systems

FreeBSD isn't the only free UNIX-like operating system available—it's not even the best-known one. The best-known free UNIX-like operating system is undoubtedly Linux, but there are also a number of other BSD-derived operating systems. We'll look at them first:

- *386/BSD* was the original free BSD operating system, introduced by William F. Jolitz in 1992. It never progressed beyond a test stage: instead, two derivative operating systems arose, FreeBSD and NetBSD. 386/BSD has been obsolete for years.
- *NetBSD* is an operating system which, to the casual observer, is almost identical to FreeBSD. The main differences are that NetBSD concentrates on hardware independence, whereas FreeBSD concentrates on performance. FreeBSD also tries harder to be easy to understand for a beginner. You can find more information about NetBSD at *http://www.NetBSD.org*.

- *OpenBSD* is a spin-off of NetBSD that focuses on security. It's also very similar to FreeBSD. You can find more information at *http://www.OpenBSD.org*.
- Apple computer introduced Version 10 (X) of its *Mac OS* in early 2001. It is a big deviation from previous versions of Mac OS: it is based on a Mach microkernel with a BSD environment. The base system (Darwin) is also free. FreeBSD and Darwin are compatible at the user source code level.

You could get the impression that there are lots of different, incompatible BSD versions. In fact, from a user viewpoint they're all very similar to each other, much more than the individual distributions of Linux, which we'll look at next.

FreeBSD and Linux

In 1991, Linus Torvalds, then a student in Helsinki, Finland, decided he wanted to run UNIX on his home computer. At that time the BSD sources were not freely available, and so Linus wrote his own version of UNIX, which he called Linux.

Linux is a superb example of how a few dedicated, clever people can produce an operating system that is better than well-known commercial systems developed by a large number of trained software engineers. It is better even than a number of commercial UNIX systems.

Obviously, I prefer FreeBSD over Linux, or I wouldn't be writing this book, but the differences between FreeBSD and Linux are more a matter of philosophy rather than of concept. Here are a few contrasts:

Table 1-1: Differences between FreeBSD and Linux

FreeBSD	Linux
FreeBSD is a direct descendent of the original UNIX, though it contains no residual AT&T code.	Linux is a clone and never contained any AT&T code.
FreeBSD is a complete operating system, maintained by a central group of software developers under the Concurrent Versions System which maintains a complete history of the project development. There is only one distribution of FreeBSD.	Linux is a kernel, personally maintained by Linus Torvalds and a few trusted companions. The non-kernel programs supplied with Linux are part of a *distribution*, of which there are several. Distributions are not completely compatible with each other.
The FreeBSD development style emphasizes accountability and documentation of changes.	The Linux kernel is maintained by a small number of people who keep track of all changes. Unofficial patches abound.
The kernel supplied with a specific release of FreeBSD is clearly defined.	Linux distributions often have subtly different kernels. The differences are not always documented.

FreeBSD aims to be a stable production environment.	Many versions of Linux are still ‘‘bleeding edge’’ development environments. This is changing rapidly, however.
As a result of the centralized development style, FreeBSD is straightforward and easy to install.	The ease of installation of Linux depends on the *distribution*. If you switch from one distribution of Linux to another, you’ll have to learn a new set of installation tools.
FreeBSD is still relatively unknown, since its distribution was initially restricted due to the AT&T lawsuits.	Linux did not have any lawsuits to contend with, so for some time it was thought to be the only free UNIX-type system available.
As a result of the lack of knowledge of FreeBSD, relatively little commercial software is available for it.	A growing amount of commercial software is becoming available for Linux.
As a result of the smaller user base, FreeBSD is less likely to have drivers for brand-new boards than Linux.	Just about any new board will soon have a driver for Linux.
Because of the lack of commercial applications and drivers for FreeBSD, FreeBSD runs most Linux programs, whether commercial or not.	Linux appears not to need to be able to run FreeBSD programs.
FreeBSD is licensed under the BSD license—see page 6. There are very few restrictions on its use.	Linux is licensed under the GNU General Public License. Further details are at *http://www.gnu.org/licenses/gpl.html*. By comparison with the BSD license, it imposes significant restrictions on what you can do with the source code.
FreeBSD has aficionados who are prepared to flame anybody who dares suggest that it’s not better than Linux.	Linux has aficionados who are prepared to flame anybody who dares suggest that it’s not better than FreeBSD.

In summary, Linux is also a very good operating system. For many, it’s better than FreeBSD.

FreeBSD system documentation

FreeBSD comes with a considerable quantity of documentation which we'll look at in the following few pages:

- The FreeBSD Documentation Project maintains a collection of ''books,'' documents in HTML or PDF format which can also be accessed online. They're installed in the directory hierarchy */usr/share/doc*.
- The traditional UNIX document format is *man pages*, individual documents describing specific functionality. They're short and to the point of being cryptic, but if you know what you're looking for, they have just the right amount of detail. They're not a good introduction.
- The GNU project introduced their own document format, *GNU info*. Some GNU programs have no other form of documentation.

Reading online documentation

You'll find a number of HTML documents in the directory */usr/share/doc/en/books*:

- */usr/share/doc/en/books/faq/index.html* contains the FreeBSD *FAQ* (*Frequently Asked Questions*). It's just what it says it is: a list of questions that people frequently ask about FreeBSD, with answers of course.
- */usr/share/doc/en/books/fdp-primer/index.html* is a primer for the *FreeBSD Documentation Project*,
- */usr/share/doc/en/books/handbook/index.html* is the FreeBSD *online handbook*. It contains a lot of information specifically about FreeBSD, including a deeper discussion of many topics in this book.
- */usr/share/doc/en/books/porters-handbook/index.html* is a handbook for contributors to the FreeBSD Ports Collection, which we'll discuss in Chapter 9, *The Ports Collection*.
- */usr/share/doc/en/books/ppp-primer/index.html* contains a somewhat dated document about setting up PPP. If you have trouble with Chapter 20, *Configuring PPP*, you may find it useful.

In addition to the directory */usr/share/doc/en/books*, there's also a directory */usr/share/doc/en/articles* with a number of shorter items of documentation.

Note the component *en* in the pathnames above. That stands for *English*. A number of these books are also installed in other languages: change *en* to *de* for a German version, to *es* for Spanish, to *fr* for French, to *ja* for Japanese, to *ru* for Russian, or to *zh* for Chinese. Translation efforts are continuing, so you may find documentation in other languages as well.

If you're running X, you can use a browser like *mozilla* to read the documents. If you don't have X running yet, use *lynx*. Both of these programs are included in the CD-ROM distribution. To install them, use *sysinstall*, which is described on page 92.

lynx is not a complete substitute for complete web browsers such as *mozilla*: since it is text-only, it is not capable of displaying the large majority of web pages correctly. It's good enough for reading most of the FreeBSD online documentation, however.

In each case, you start the browser with the name of the document, for example:

```
$ lynx /usr/share/doc/en/books/handbook/index.html
$ mozilla /usr/share/doc/en/books/handbook/index.html &
```

Enter the & after the invocation of *mozilla* to free up the window in which you invoke it: *mozilla* opens its own window.

If you haven't installed the documentation, you can still access it from the Live Filesystem CD-ROM. Assuming the CD-ROM is mounted on */cdrom*, choose the file */cdrom/usr/share/doc/en/books/handbook/index.html*.

Alternatively, you can print out the handbook. This is a little more difficult, and of course you'll lose the hypertext references, but you may prefer it in this form. To format the handbook for printing, you'll need a PostScript printer or *ghostscript*. See page 271 for more details of how to print PostScript.

The printable version of the documentation doesn't usually come with the CD-ROM distribution. You can pick it up with *ftp* (see page 433) from *ftp://ftp.FreeBSD.ORG/pub/FreeBSD/doc/*, which has the same directory structure as described above. For example, you would download the handbook in PostScript form from *ftp://ftp.FreeBSD.ORG/pub/FreeBSD/doc/en/books/handbook/book.ps.bz2*.

The online manual

The most comprehensive documentation on FreeBSD is the online manual, usually referred to as the *man pages*. Nearly every program, file, library function, device or interface on the system comes with a short reference manual explaining the basic operation and various arguments. If you were to print it out, it would run to well over 8,000 pages.

When online, you view the man pages with the command *man*. For example, to learn more about the command *ls*, type:

```
$ man ls
LS(1)                     FreeBSD Reference Manual                     LS(1)

NAME
     ls - list directory contents

SYNOPSIS
     ls [-ACFLRTacdfiloqrstu1] [ file ... ]

DESCRIPTION
     For each operand that names a file of a type other than directory, ls
```

```
     displays its name as well as any requested, associated information.  For
     each operand that names a file of type directory, ls displays the names.
(etc)
```

In this particular example, with the exception of the first line, the text in **constant width bold** is not input, it's the way it appears on the screen.

The online manual is divided up into sections numbered:

1. User commands
2. System calls and error numbers
3. Functions in the C libraries
4. Device drivers
5. File formats
6. Games and other diversions
7. Miscellaneous information
8. System maintenance and operation commands
9. Kernel interface documentation

In some cases, the same topic may appear in more than one section of the online manual. For example, there is a user command *chmod* and a system call `chmod()`. In this case, you can tell the *man* command which you want by specifying the section number:

```
$ man 1 chmod
```

This command displays the manual page for the user command chmod. References to a particular section of the online manual are traditionally placed in parentheses in written documentation. For example, *chmod(1)* refers to the user command *chmod*, and *chmod(2)* means the system call.

This is fine if you know the name of the command and forgot how to use it, but what if you can't recall the command name? You can use *man* to search for keywords in the command descriptions by using the `-k` option, or by starting the program *apropos*:

```
$ man -k mail
$ apropos mail
```

Both of these commands do the same thing: they show the names of the man pages that have the keyword *mail* in their descriptions.

Alternatively, you may browse through the */usr/bin* directory, which contains most of the system executables. You'll see lots of file names, but you don't have any idea what they do. To find out, enter one of the lines:

```
$ cd /usr/bin; man -f *
$ cd /usr/bin; whatis *
```

Both of these commands do the same thing: they print out a one-line summary of the purpose of the program:

```
$ cd /usr/bin; man -f *
a2p(1)           - Awk to Perl translator
addftinfo(1)     - add information to troff font files for use with groff
apply(1)         - apply a command to a set of arguments
apropos(1)       - search the whatis database
...etc
```

Printing man pages

If you prefer to have man pages in print, rather than on the screen, you can do this in two different ways:

- The simpler way is to redirect the output to the spooler:

  ```
  $ man ls | lpr
  ```

  This gives you a printed version that looks pretty much like the original on the screen, except that you may not get bold or underlined text.

- You can get typeset output with *troff*:

  ```
  $ man -t ls | lpr
  ```

  This gives you a properly typeset version of the man page, but it requires that your spooling system understand PostScript—see page 271 for more details of printing PostScript, even on printers that don't understand PostScript.

GNU info

The Free Software Foundation has its own online hypertext browser called *info*. Many FSF programs come with either no man page at all, or with an excuse for a man page (*gcc*, for example). To read the online documentation, you need to browse the *info* files with the *info* program, or from *Emacs* with the *info* mode. To start *info*, simply type:

```
$ info
```

In *Emacs*, enter **CTRL-h i** or **ALT-x** `info`. Whichever way you start *info*, you can get brief introduction by typing **h**, and a quick command reference by typing **?**.

Other documentation on FreeBSD

FreeBSD users have access to probably more top-quality documentation than just about any other operating system. Remember that word UNIX is trademarked. Sure, the lawyers tell us that we can't refer to FreeBSD as UNIX, because UNIX belongs to the Open Group. That doesn't make the slightest difference to the fact that nearly every book on UNIX applies more directly to FreeBSD than any other flavour of UNIX. Why?

Commercial UNIX vendors have a problem, and FreeBSD doesn't help them: why should people buy their products when you can get it free from the FreeBSD Project (or, for that matter, from other free UNIX-like operating systems such as NetBSD, OpenBSD and Linux)? One obvious reason would be ''value-added features.'' So they add features or fix weak points in the system, put a copyright on the changes, and help lock their customers in to their particular implementation. As long as the changes are really useful, this is legitimate, but it does make the operating system less compatible with ''standard UNIX,'' and the books about standard UNIX are less applicable.

In addition, many books are written by people with an academic background. In the UNIX world, this means that they are more likely than the average user to have been exposed to BSD. Many general UNIX books handle primarily BSD, possibly with an additional chapter on the commercial System V version.

In Appendix A, *Bibliography*, you'll find a list of books that I find worthwhile. I'd like to single out some that I find particularly good, and that I frequently use myself:

- *UNIX Power Tools*, by Jerry Peek, Tim O'Reilly, and Mike Loukides, is a superb collection of interesting information, including a CD-ROM. Recommended for everybody, from beginners to experts.
- *UNIX for the Impatient*, by Paul W. Abrahams and Bruce R. Larson, is more similar to this book, but it includes a lot more material on specific products, such as shells and the *Emacs* editor.
- The *UNIX System Administration Handbook*, by Evi Nemeth, Garth Snyder, Scott Seebass, and Trent R. Hein, is one of the best books on systems administration I have seen. It covers a number different UNIX systems, including an older version of FreeBSD.

There are also many active Internet groups that deal with FreeBSD. Read about them in the online handbook.

The FreeBSD community

FreeBSD was developed by a world-wide group of developers. It could not have happened without the Internet. Many of the key players have never even met each other in person; the main means of communication is via the Internet. If you have any kind of Internet connection, you can participate as well. If you don't have an Internet connection, it's about time you got one. The connection doesn't have to be complete: if you can receive email, you can participate. On the other hand, FreeBSD includes all the software you need for a complete Internet connection, not the very limited subset that most PC-based ''Internet'' packages offer you.

Mailing lists

As it says in the copyright, FreeBSD is supplied as-is, without any support liability. If you're on the Internet, you're not alone, however. Liability is one thing, but there are plenty of people prepared to help you, most for free, some for fee. A good place to start is with the mailing lists. There are a number of mailing lists that you can join. Some of the more interesting ones are:

- `FreeBSD-questions@FreeBSD.org` is the list to which you may send general questions, in particular on how to use FreeBSD. If you have difficulty understanding anything in this book, for example, this is the right place to ask. It's also the list to use if you're not sure which is the most appropriate.
- `FreeBSD-newbies@FreeBSD.org` is a list for newcomers to FreeBSD. It's intended for people who feel a little daunted by the system and need a bit of reassurance. It's not the right place to ask any kind of technical question.
- `FreeBSD-hackers@FreeBSD.org` is a technical discussion list.
- `FreeBSD-current@FreeBSD.org` is an obligatory list for people who run the development version of FreeBSD, called `FreeBSD-CURRENT`.
- `FreeBSD-stable@FreeBSD.org` is a similar list for people who run the more recent stable version of FreeBSD, called `FreeBSD-STABLE`. We'll talk about these versions on page 582. Unlike the case for `FreeBSD-CURRENT` users, it's not obligatory for `FreeBSD-STABLE` users to subscribe to `FreeBSD-stable`.

You can find a complete list of FreeBSD mailing lists on the web site, currently at *http://www.FreeBSD.org/doc/en_US.ISO8859-1/books/handbook/eresources.html*. This address is part of the online handbook and may change when the handbook is modified; follow the link *Mailing Lists* from *http://www.FreeBSD.org/* if it is no longer valid, or if you can't be bothered typing in the URI.

The mailing lists are run by *mailman* (in the Ports Collection). Join them via the web interface mentioned above. You will receive a mail message from *mailman* asking you to confirm your subscription by replying to the message. You don't need to put anything in

the reply: the reply address is used once only, and you're the only person who will ever see it, so the system knows that it's you by the fact that you replied at all. You also have the option of confirming via a web interface with a specially generated URI. Similar considerations apply in this case.

FreeBSD mailing lists can have a very high volume of traffic. The FreeBSD-questions mailing list, for example, has thousands of subscribers, and many of them are themselves mailing lists. It receives over a hundred messages every day. That's about a million messages a day in total for just one mailing list, so when you sign up for a mailing list, be sure to read the charter. You can find the URI from the *mailman* confirmation message. It's also a good idea to ''lurk'' (listen, but not say anything) on the mailing list a while before posting anything: each list has its own traditions.

When submitting a question to `FreeBSD-questions`, consider the following points:

1. Remember that nobody gets paid for answering a FreeBSD question. They do it of their own free will. You can influence this free will positively by submitting a well-formulated question supplying as much relevant information as possible. You can influence this free will negatively by submitting an incomplete, illegible, or rude question. It's perfectly possible to send a message to FreeBSD-questions and not get an answer even if you follow these rules. It's much more possible to not get an answer if you don't.

2. Not everybody who answers FreeBSD questions reads every message: they look at the subject line and decide whether it interests them. Clearly, it's in your interest to specify a subject. ''FreeBSD problem'' or ''Help'' aren't enough. If you provide no subject at all, many people won't bother reading it. If your subject isn't specific enough, the people who can answer it may not read it.

3. When sending a new message, well, send a new message. Don't just reply to some other message, erase the old content and change the subject line. That leaves an `In-Reply-To:` header which many mail readers use to thread messages, so your message shows up as a reply to some other message. People often delete messages a whole thread at a time, so apart from irritating people, you also run a chance of having the message deleted unread.

4. Format your message so that it is legible, and PLEASE DON'T SHOUT!!!!!. It's really painful to try to read a message written full of typos or without any line breaks. A lot of badly formatted messages come from bad mailers or badly configured mailers. The following mailers are known to send out badly formatted messages without you finding out about them:

   Eudora
   exmh
   Microsoft Exchange
   Microsoft Internet Mail
   Microsoft Outlook
   Netscape

As you can see, the mailers in the Microsoft world are frequent offenders. If at all possible, use a UNIX mailer. If you must use a mailer under Microsoft environments, make sure it is set up correctly. Try not to use MIME: a lot of people use mailers which don't get on very well with MIME.

For further information on this subject, check out *http://www.lemis.com/email.html*.

5. Make sure your time and time zone are set correctly. This may seem a little silly, since your message still gets there, but many of the people you are trying to reach get several hundred messages a day. They frequently sort the incoming messages by subject and by date, and if your message doesn't come before the first answer, they may assume they missed it and not bother to look.

6. Don't include unrelated questions in the same message. Firstly, a long message tends to scare people off, and secondly, it's more difficult to get all the people who can answer all the questions to read the message.

7. Specify as much information as possible. This is a difficult area: the information you need to submit depends on the problem. Here's a start:

   - If you get error messages, don't say ''I get error messages'', say (for example) ''I get the error message *No route to host*''.
   - If your system panics, don't say ''My system panicked'', say (for example) ''my system panicked with the message *free vnode isn't*''.
   - If you have difficulty installing FreeBSD, please tell us what hardware you have, particularly if you have something unusual.
   - If, for example, you have difficulty getting PPP to run, describe the configuration. Which version of PPP do you use? What kind of authentication do you have? Do you have a static or dynamic IP address? What kind of messages do you get in the log file? See Chapter 20, *Configuring PPP*, for more details in this particular case.

8. If you don't get an answer immediately, or if you don't even see your own message appear on the list immediately, don't resend the message. Wait at least 24 hours. The FreeBSD mailer offloads messages to a number of subordinate mailers around the world. Usually the messages come through in a matter of seconds, but sometimes it can take several hours for the mail to get through.

9. If you do all this, and you still don't get an answer, there could be other reasons. For example, the problem is so complicated that nobody knows the answer, or the person who does know the answer was offline. If you don't get an answer after, say, a week, it might help to re-send the message. If you don't get an answer to your second message, though, you're probably not going to get one from this forum. Resending the same message again and again will only make you unpopular.

How to follow up to a question

Often you will want to send in additional information to a question you have already sent. The best way to do this is to reply to your original message. This has three advantages:

1. You include the original message text, so people will know what you're talking about. Don't forget to trim unnecessary text, though.
2. The text in the subject line stays the same (you did remember to put one in, didn't you?). Many mailers will sort messages by subject. This helps group messages together.
3. The message reference numbers in the header will refer to the previous message. Some mailers, such as mutt, can thread messages, showing the exact relationships between the messages.

There are more suggestions, in particular for answering questions, at *http://www.lemis.com/questions.html*. See also Chapter 26, *Electronic mail: clients* for more information about sending mail messages. You may also like to check out the FreeBSD web site at *http://www.FreeBSD.org/* and the support page at *http://www.FreeBSD.org/support.html*.

In addition, a number of companies offer support for FreeBSD. See the web page *http://www.FreeBSD.org/commercial/consulting_bycat.html* for some possibilities.

Unsubscribing from the mailing lists

There's a lot of traffic on the mailing lists, particularly on `FreeBSD-questions`. You may find you can't take it and want to get out again. Again, you unsubscribe from the list either via the web or via a special mail address, ***not*** by sending mail to the the list. Each message you get from the mailing lists finishes with the following text:

```
freebsd-questions@freebsd.org mailing list
http://lists.freebsd.org/mailman/listinfo/freebsd-questions
To unsubscribe, send any mail to "freebsd-questions-unsubscribe@freebsd.org"
```

Don't be one of those people who send the unsubscribe request to the mailing list instead.

User groups

But how about meeting FreeBSD users face to face? There are a number of user groups around the world. If you live in a big city, chances are that there's one near you. Check *http://www.FreeBSD.org/support.html#user* for a list. If you don't find one, consider taking the initiative and starting one.

In addition, USENIX holds an annual conference, the *BSDCon*, which deals with technical aspects of the BSD operating systems. It's also a great opportunity to get to know other users from around the world. If you're in Europe, there is also a BSDCon Europe, which at the time of writing was not run by USENIX. See *http://www.eurobsdcon.org* for more details.

Reporting bugs

If you find something wrong with FreeBSD, we want to know about it, so that we can fix it. To report a bug, use the *send-pr* program to send it as a mail message.

There used to be a web form at *http://www.FreeBSD.org/send-pr.html*, but it has been closed down due to abuse.

The Berkeley daemon

The little daemon at the right symbolizes BSD. It is included with kind permission of Marshall Kirk McKusick, one of the leading members of the former Computer Sciences Research Group at the University of California at Berkeley, and owner of the daemon's copyright. Kirk also wrote the foreword to this book.

The daemon has occasionally given rise to a certain amount of confusion. In fact, it's a joking reference to processes that run in the background—see Chapter 8, *Taking control*, page 150, for a description. The outside world occasionally sees things differently, as the following story indicates:

```
Newsgroups: alt.humor.best-of-usenet
Subject: [comp.org.usenix] A Great Daemon Story
```

```
From: Rob Kolstad <kolstad@bsdi.com>
Newsgroups: comp.org.usenix
Subject: A Great Daemon Story
```

Linda Branagan is an expert on daemons. She has a T-shirt that sports the daemon in tennis shoes that appears on the cover of the 4.3BSD manuals and *The Design and Implementation of the 4.3BSD UNIX Operating System* by S. Leffler, M. McKusick, M. Karels, J. Quarterman, Addison Wesley Publishing Company, Reading, MA 1989.

She tells the following story about wearing the 4.3BSD daemon T-shirt:

Last week I walked into a local ‘‘home style cookin’ restaurant/watering hole’’ in Texas to pick up a take-out order. I spoke briefly to the waitress behind the counter, who told me my order would be done in a few minutes.

So, while I was busy gazing at the farm implements hanging on the walls, I was approached by two ‘‘natives.’’ These guys might just be the original Texas rednecks.

‘‘Pardon us, ma’am. Mind if we ask you a question?’’

Well, people keep telling me that Texans are real friendly, so I nodded.

‘‘Are you a Satanist?’’

Well, at least they didn’t ask me if I liked to party.

‘‘Uh, no, I can’t say that I am.’’

‘‘Gee, ma’am. Are you sure about that?’’ they asked.

I put on my biggest, brightest Dallas Cowboys cheerleader smile and said, ‘‘No, I’m positive. The closest I’ve ever come to Satanism is watching Geraldo.’’

‘‘Hmmm. Interesting. See, we was just wondering why it is you have the lord of darkness on your chest there.’’

I was this close to slapping one of them and causing a scene—then I stopped and noticed the shirt I happened to be wearing that day. Sure enough, it had a picture of a small, devilish-looking creature that has for some time now been associated with a certain operating system. In this particular representation, the creature was wearing sneakers.

They continued: ‘‘See, ma’am, we don’t exactly appreciate it when people show off pictures of the devil. Especially when he’s lookin’ so friendly.’’

These idiots sounded terrifyingly serious.

Me: ‘‘Oh, well, see, this isn’t really the devil, it’s just, well, it’s sort of a mascot.

Native: ‘‘And what kind of football team has the devil as a mascot?’’

Me: ‘‘Oh, it’s not a team. It’s an operating—uh, a kind of computer.’’

I figured that an ATM machine was about as much technology as these guys could handle, and I knew that if I so much as uttered the word ‘‘UNIX’’ I would only make things worse.

Native: ‘‘Where does this satanical computer come from?’’

Me: ‘‘California. And there’s nothing satanical about it really.’’

Somewhere along the line here, the waitress noticed my predicament—but these guys probably outweighed her by 600 pounds, so all she did was look at me sympathetically and run off into the kitchen.

Native: ‘‘Ma’am, I think you’re lying. And we’d appreciate it if you’d leave the premises now.’’

Fortunately, the waitress returned that very instant with my order, and they agreed that it would be okay for me to actually pay for my food before I left. While I was at the cash register, they amused themselves by talking to each other.

Native #1: ‘‘Do you think the police know about these devil computers?’’

Native #2: ‘‘If they come from California, then the FBI oughta know about ’em.’’

They escorted me to the door. I tried one last time: ‘‘You’re really blowing this all out of proportion. A lot of people use this ‘kind of computers.’ Universities, researchers, businesses. They’re actually very useful.’’

Big, big, *big* mistake. I should have guessed at what came next.

Native: ‘‘Does the government use these devil computers?’’

Me: ‘‘Yes.’’

Another *big* boo-boo.

Native: ‘‘And does the government pay for ’em? With our tax dollars?’’

I decided that it was time to jump ship.

Me: ‘‘No. Nope. Not at all. Your tax dollars never entered the picture at all. I promise. No sir, not a penny. Our good Christian congressmen would never let something like that happen. Nope. Never. Bye.’’

Texas. What a country.

The daemon tradition goes back quite a way. As recently as 1996, after the publication of the first edition of this book, the following message went through the `FreeBSD-chat` mailing list:

To: "Jonathan M. Bresler" <jmb@freefall.freebsd.org>
Cc: obrien@antares.aero.org (Mike O’Brien),
joerg_wunsch@uriah.heep.sax.de,
chat@FreeBSD.org, juphoff@tarsier.cv.nrao.edu
Date: Tue, 07 May 1996 16:27:20 -0700
Sender: owner-chat@FreeBSD.org

> details and gifs PLEASE!

If you insist. :-)

Sherman, set the Wayback Machine for around 1976 or so (see Peter Salus’ *A Quarter Century of UNIX* for details), when the first really national UNIX meeting was held in Urbana, Illinois. This would be after the ‘‘forty people in a Brooklyn classroom’’ meeting held by Mel Ferentz (yeah I was at that too) and the more-or-less simultaneous West Coast meeting(s) hosted by SRI, but before the UNIX Users Group was really incorporated as a going concern.

I knew Ken Thompson and Dennis Ritchie would be there. I was living in Chicago at the time, and so was comic artist Phil Foglio, whose star was just beginning to rise. At that time I was a bonded locksmith. Phil’s roommate had unexpectedly split town, and he was the only one who knew the combination to the wall safe in their apartment. This is the only apartment I’ve ever seen that had a wall safe, but it sure did have one, and Phil had some stuff locked in there. I didn’t hold out much hope, since safes are far beyond where I was (and am) in my locksmithing sphere of competence, but I figured ‘‘no guts no glory’’ and told him I’d give it a whack. In return, I told him, he could do some T-shirt art for me. He readily agreed.

Wonder of wonders, this safe was vulnerable to the same algorithm that Master locks used to be susceptible to. I opened it in about 15 minutes of manipulation. It was my greatest moment as a locksmith and Phil was overjoyed. I went down to my lab and shot some Polaroid snaps of the PDP-11 system I was running UNIX on at the time, and gave it to Phil with some descriptions of the visual puns I wanted: pipes, demons with forks running along the pipes, a ‘‘bit bucket’’ named */dev/null*, all that.

What Phil came up with is the artwork that graced the first decade’s worth of ‘‘UNIX T-shirts,’’ which were made by a Ma and Pa operation in a Chicago suburb. They turned out transfer art using a 3M color copier in their basement. Hence, the PDP-11 is reversed (the tape drives are backwards) but since Phil left off the front panel, this was hard to tell. His trademark signature was photo-reversed, but was

recopied by the T-shirt people and ‘‘re-forwardized,’’ which is why it looks a little funny compared to his real signature.

Dozens and dozens of these shirts were produced. Bell Labs alone accounted for an order of something like 200 for a big picnic. However, only four (4) REAL originals were produced: these have a distinctive red collar and sleeve cuff. One went to Ken, one to Dennis, one to me, and one to my then-wife. I now possess the latter two shirts. Ken and Dennis were presented with their shirts at the Urbana conference.

People ordered these shirts direct from the Chicago couple. Many years later, when I was living in LA, I got a call from Armando Stettner, then at DEC, asking about that now-famous artwork. I told him I hadn’t talked to the Illinois T-shirt makers in years. At his request I called them up. They’d folded the operation years ago and were within days of discarding all the old artwork. I requested its return, and duly received it back in the mail. It looked strange, seeing it again in its original form, a mirror image of the shirts with which I and everyone else were now familiar.

I sent the artwork to Armando, who wanted to give it to the Ultrix marketing people. They came out with the Ultrix poster that showed a nice shiny Ultrix machine contrasted with the chewing-gum-and-string PDP-11 UNIX people were familiar with. They still have the artwork, so far as I know.

I no longer recall the exact contents of the letter I sent along with the artwork. I did say that as far as I knew, Phil had no residual rights to the art, since it was a ‘work made for hire’, though nothing was in writing (and note this was decades before the new copyright law). I do not now recall if I explicitly assigned all rights to DEC. What is certain is that John Lassiter’s daemon, whether knowingly borrowed from the original, or created by parallel evolution, postdates the first horde of UNIX daemons by at least a decade and probably more. And if Lassiter’s daemon looks a lot like a Phil Foglio creation, there’s a reason.

I have never scanned in Phil’s artwork; I’ve hardly ever scanned in anything, so I have no GIFs to show. But I have some very very old UNIX T-shirts in startlingly good condition. Better condition than I am at any rate: I no longer fit into either of them.

Mike O’Brien
creaky antique

Note the date of this message: it appeared since the first edition of this book. Since then, the daemon image has been scanned in, and you can find a version at *http://www.mckusick.com/beastie/shirts/usenix.html*.

In this chapter:
- *Using old hardware*
- *PC Hardware*
- *How the system detects hardware*
- *Configuring ISA cards*
- *PCMCIA, PC Card and CardBus*
- *Universal Serial Bus*
- *Disks*
- *Disk data layout*
- *Making the file systems*
- *Disk size limitations*
- *Display hardware*
- *The hardware*
- *Compaq/Digital Alpha machines*
- *The CD-ROM distribution*

2

Before you install

FreeBSD runs on just about any modern PC, Alpha or 64 bit SPARC machine. You can skip this chapter and the next and move to chapter 3, and you'll have a very good chance of success. Nevertheless, it makes things easier to know the contents of this chapter before you start. If you do run into trouble, it will give you the background information you need to solve the trouble quickly and simply.

FreeBSD also runs on most Intel-based laptops; in general the considerations above apply for laptops as well. In the course of the book we'll see examples of where laptops require special treatment.

Most of the information here applies primarily to Intel platforms. We'll look at the Compaq Alpha architecture on page 42. The first release of FreeBSD to support the SPARC 64 architecture is 5.0, and support is still a little patchy. At the time of going to press, it's not worth describing, since it will change rapidly. The instructions on the CD-ROM distribution are currently the best source of information on running FreeBSD on SPARC 64.

Using old hardware

FreeBSD runs on all relatively recent machines. In addition, a lot of older hardware that is available for a nominal sum, or even for free, runs FreeBSD quite happily, though you may need to take more care in the installation.

FreeBSD does not support all PC hardware: the PC has been on the market for over 20 years, and it has changed a lot in that time. In particular:

- FreeBSD does not support 8 bit and 16 bit processors. These include the 8086 and 8088, which were used in the IBM PC and PC-XT and clones, and the 80286, used in the IBM PC-AT and clones.
- The FreeBSD kernel no longer supports ST-506 and ESDI drives. You're unlikely to have any of these: they're now so old that most of them have failed. The *wd* driver still includes support for them, but it hasn't been tested, and if you want to use this kind of drive you might find it better to use FreeBSD Release 3. See page 31 to find out how to identify these drives. You can get Release 3 of FreeBSD from *ftp://ftp.FreeBSD.org/pub/FreeBSD/releases/i386/3.x-STABLE*. You'll have to perform a network installation.
- Memory requirements for FreeBSD have increased significantly in the last few years, and you should consider 16 MB a minimum size, though nobody has recently checked whether it wouldn't install in, say, 12 MB. FreeBSD Release 3 still runs in 4 MB, though you need 5 MB for installation.

If you're planning to install FreeBSD on an old machine, consider the following to be an absolute minimum:

- PC with 80386 CPU, Alpha-based machine with SRM firmware.
- 16 MB memory (Intel) or 24 MB (Alpha).
- 80 MB free disk space (Intel). Nobody has tried an installation on an Alpha or SPARC machine with less than 500 MB, though you can probably reduce this value significantly.

You don't absolutely need a keyboard and display board: many FreeBSD machines run server tasks with neither keyboard nor display. Even then, though, you may find it convenient to put a display board in the machine to help in case you run into trouble.

When I say *absolute* minimum, I mean it. You can't do very much with such a minimal system, but for some purposes it might be adequate. You can improve the performance of such a minimal system significantly by adding memory. Before you go to the trouble to even try such a minimal installation, consider the cost of another 16 MB of memory. And you can pick up better machines than this second-hand for $50. Is the hassle worth it?

To get full benefits from a desktop or laptop FreeBSD system (but not from a machine used primarily as a server), you should be running the X Window system. This uses more memory. Consider 32 MB a usable minimum here, though thanks to FreeBSD's virtual memory system, this is not such a hard limit as it is with some other systems.

> The speed of a virtual memory-based system such as FreeBSD depends at least as much on memory performance as on processor performance. If you have, say, a 486DX-33 and 16 MB of memory, upgrading memory to 32 MB will probably buy you more performance than upgrading the motherboard to a Pentium 100 and keeping the 16 MB memory. This applies for a usual mix of programs, in particular, programs that don't perform number crunching.

Any SPARC 64 machine runs FreeBSD acceptably, as the machines are relatively new. If you're running Intel or Alpha, consider the following the minimum for getting useful work done with FreeBSD and X:

- PC with 80486DX/2-66, or Alpha-based machine
- 32 MB memory (i386) or 64 MB (Alpha)
- SVGA display board with 2 MB memory, 1024x768
- Mouse
- 200 MB free disk space

> Your mileage may vary. During the review phase of an earlier edition of this book, one of the reviewers stated that he was very happy with his machine, which has a 486-33 processor, 16 MB main memory, and 1 MB memory on his display board. He said that it ran a lot faster than his Pentium 100 at work, which ran Microsoft. The moral: if your hardware doesn't measure up to the recommended specification, don't be discouraged. Try it out anyway.

Beyond this minimum, FreeBSD supports a large number of other hardware components.

Device drivers

The FreeBSD kernel is the only part of the system that can access the hardware. It includes *device drivers*, which control the function of peripheral devices such as disks, displays and network boards. When you install new hardware, you need a driver for it.

There are two ways to get a driver into the kernel: you can build a kernel that includes the driver code, or you can load a driver module (*Kernel Loadable Module* or *kld*) into the kernel at run time. Not all drivers are available as klds. If you need one of these drivers, and it's not included in the standard kernel, you have to build a new kernel. We look at building kernels in Chapter 33.

The kernel configuration supplied with FreeBSD distributions is called `GENERIC` after the name of the configuration file that describes it. It contains support for most common devices, though support for some older hardware is missing, usually because it conflicts with more modern drivers. For a full list of currently supported hardware, read the web page *http://www.FreeBSD.org/releases/* and select the link *Hardware Notes* for the release you're interested in. This file is also available on installed FreeBSD systems as */usr/share/doc/en_US.ISO_8859-1/books/faq/hardware.html*. It is also available in other languages; see the subdirectories of */usr/share/doc*.

PC Hardware

This section looks at the information you need to understand to install FreeBSD on the i386 architecture. In particular, in the next section we'll look at how FreeBSD detects hardware, and what to do if your hardware doesn't correspond to the system's expectations. On page 31 we'll see how FreeBSD and other PC operating systems handle disk space, and how to set up your disk for FreeBSD.

Some of this information also applies to the Alpha and SPARC 64 architectures. We'll look at the differences for the Alpha architecture on page 42. Currently the SPARC 64 implementation is changing too fast to describe it in a meaningful manner.

Since the original PC, a number of hardware standards have come, and some have gone:

- The original PC had an 8 bit bus. Very few of these cards are still available, but they are compatible with the ISA bus (see the next item).
- The PC AT, introduced in 1984, had a 16 bit 80286 processor. To support this processor, the bus was widened to 16 bits. This bus came to be known as the *Industry Standard Architecture*, or *ISA*. This standard is still not completely dead, and many new motherboards support it. Most older motherboards have a number of ISA slots.
- The ISA bus has a number of severe limitations, notably poor performance. This became a problem very early. In 1985, IBM introduced the PS/2 system, which addressed this issue with a new bus, the so-called *Microchannel Architecture* or *MCA*. Although successful for IBM, MCA was not adopted by other manufacturers, and FreeBSD does not support it at all. IBM no longer produces products based on MCA.
- In parallel to MCA, other manufacturers introduced a bus called the *Extended Industry Standard Architecture*, or *EISA*. As the name suggests, it is a higher-performance extension of ISA, and FreeBSD supports it. Like MCA, it is obsolete.
- EISA still proved to be not fast enough for good graphics performance. In the late 80s, a number of *local bus* solutions appeared. They had better performance, but some were very unreliable. FreeBSD supported most of them, but you can't rely on it. It's best to steer clear of them.
- Finally, in the early 1990s, Intel brought out a new bus called *Peripheral Component Interconnect*, or *PCI*. PCI is now the dominant bus on a number of architectures. Most modern PC add-on boards are PCI.

  Compared to earlier buses, PCI is much faster. Most boards have a 32 bit wide data bus, but there is also a 64 bit PCI standard. PCI boards also contain enough intelligence to enable the system to configure them, which greatly simplifies installation of the system or of new boards.
- Modern motherboards also have an *AGP* (*Accelerated Graphics Port*) slot specifically designed to support exactly one graphic card. As the name implies, it's faster even than PCI, but it's optimized for graphics only. FreeBSD supports it, of course; otherwise it couldn't run on modern hardware.
- Most laptops have provision for external plug-in cards that conform to the *PC Card* (formerly called *PCMCIA*) or *CardBus* standards. These cards are designed to be inserted into and removed from a running system. FreeBSD has support for these cards; we'll look at them in more detail on page 30.
- More and more, the basic serial and parallel ports installed on early PCs are being replaced by a *Universal Serial Bus* or *USB*. We'll look at it on page 31.

How the system detects hardware

When the system starts, each driver in the kernel examines the system to find any hardware that it might be able to control. This examination is called *probing*. Depending on the driver and the nature of the hardware it supports, the probe may be clever enough to set up the hardware itself, or to recognize its hardware no matter how it has been set up, or it may expect the hardware to be set up in a specific manner in order to find it. In general, you can expect PCI drivers to be able to set up the card to work correctly. In the case of ISA or EISA cards, you may not be as lucky.

Configuring ISA cards

ISA cards are rapidly becoming obsolete, but sometimes they're still useful:

- ISA graphics cards are very slow in comparison with modern graphic cards, but if you just want a card for maintenance on a server machine that normally doesn't display anything, this is an economical alternative.
- Some ISA disk controllers can be useful, but they are sharply limited in performance.
- ISA Ethernet cards may be a choice for low-volume networking.
- Many ISA serial cards and built-in modems are still available.

Most ISA cards require some configuration. There are four main parameters that you may need to set for PC controller boards:

1. The *port address* is the address of the first of possibly several control registers that the driver uses to communicate with the board. It is normally specified in hexadecimal, for example `0x320`.

   > If you come from a Microsoft background, you might be more used to the notation `320H`. The notation `0x320` comes from the C programming language. You'll see a lot of it in UNIX.

   Each board needs its own address or range of addresses. The ISA architecture has a sharply limited address range, and one of the most frequent causes of problems when installing a board is that the port addresses overlap with those of another board.

   Beware of boards with a large number of registers. Typical port addresses end in (hexadecimal) `0`. Don't rely on being able to take any unoccupied address ending in `0`, though: some boards, such as Novell NE2000 compatible Ethernet boards, occupy up to 32 registers—for example, from `0x320` to `0x33f`. Note also that a number of addresses, such as the serial and parallel ports, often end in `8`.

2. Boards use an *Interrupt Request*, also referred to as *IRQ*, to get the attention of the driver when a specific event happens. For example, when a serial interface reads a character, it generates an interrupt to tell the driver to collect the character. Interrupt requests can sometimes be shared, depending on the driver and the hardware. There are even fewer interrupt requests than port addresses: a total of 15, of which a number

are reserved by the motherboard. You can usually expect to be able to use IRQs 3, 4, 5, 7, 9, 10, 11 and 12. IRQ 2 is special: due to the design of the original IBM PC/AT, it is the same thing as IRQ 9. FreeBSD refers to this interrupt as IRQ 9.

As if the available interrupts weren't already restricted enough, ISA and PCI boards use the same set of interrupt lines. PCI cards can share interrupt lines between multiple boards, and in fact the PCI standard only supports four interrupts, called INTA, INTB, INTC and INTD. In the PC architecture they map to four of the 15 ISA interrupts. PCI cards are self-configuring, so all you need to do is to ensure that PCI and ISA interrupts don't conflict. You normally set this up in a BIOS setup menu.

3. Some high-speed devices perform *Direct Memory Access*, also known as *DMA*, to transfer data to or from memory without CPU intervention. To transfer data, they assert a *DMA Request* (DRQ) and wait for the bus to reply with a *DMA Acknowledge* (DACK). The combination of DRQ and DACK is sometimes called a *DMA Channel*. The ISA architecture supplies 7 DMA channels, numbered 0 to 3 (8 bit) and 5 to 7 (16 bit). The floppy driver uses DMA channel 2. DMA channels may not be shared.
4. Finally, controllers may have on-board memory, sometimes referred to as *I/O memory* or *IOmem*. It is usually located at addresses between `0xa0000` and `0xeffff`.

If the driver only looks at specific board configurations, you can set the board to match what the driver expects, typically by setting jumpers or using a vendor-supplied diagnostic program to set on-board configuration memory, or you can build a kernel to match the board settings.

PCMCIA, PC Card and CardBus

Laptops don't have enough space for normal PCI expansion slots, though many use a smaller PCI card format. It's more common to see *PC Card* or *CardBus* cards, though. PC Card was originally called *PCMCIA*, which stands for *Personal Computer Memory Card International Association*: the first purpose of the bus was to expand memory. Nowadays memory expansion is handled by other means, and PC Card cards are usually peripherals such as network cards, modems or disks. It's true that you can insert compact flash memory for digital cameras into a PC Card adapter and access it from FreeBSD, but even in this case, the card looks like a disk, not a memory card.

The original PC Card standard already has one foot in the grave: it's a 16 bit bus that doesn't work well with modern laptops. The replacement standard has a 32 bit wide bus and is called *CardBus*. The cards look almost identical, and most modern laptops support both standards. In this book I'll use use the term *PC Card* to include CardBus unless otherwise stated. FreeBSD Release 5 includes completely new PC Card code. It now supports both 16 bit PC Card and 32 bit CardBus cards.

PC Card offers one concept that conventional cards don't: the cards are *hot swappable*. You can insert them and remove them in a running system. This poses a number of potential problems, some of which are only partially solved.

PC Card and CardBus cards

PC Card and CardBus both use the same form factor cards: they are 54 mm wide and at least 85 mm long, though some cards, noticeably wireless networking cards, are up to 120 mm long and project beyond the casing of the laptop. The wireless cards contain an antenna in the part of the card that projects from the machine.

PC Card cards can have one of three standard thicknesses:

- *Type 1* cards are 3.3 mm thick. They're very uncommon.
- *Type 2* cards are 5 mm thick. These are the most common type, and most laptops take two of them.
- *Type 3* cards are 10.5 mm thick. In most laptops you can normally insert either one type 3 card or two type 2 cards.

The `GENERIC` FreeBSD kernel contains support for PC Card, so you don't need to build a new kernel.

Universal Serial Bus

The *Universal Serial Bus* (*USB*) is a new way of connecting external peripherals, typically those that used to be connected by serial or parallel ports. It's much faster than the old components: the old serial interface had a maximum speed of 115,200 bps, and the maximum you can expect to transfer over the parallel port is about 1 MB/s. By comparison, current USB implementations transfer data at up to 12 Mb/s, and a version with 480 Mb/s is in development.

As the name states, USB is a *bus*: you can connect multiple devices to a bus. Currently the most common devices are mid-speed devices such as printers and scanners, but you can connect just about anything, including keyboards, mice, Ethernet cards and mass storage devices.

Disks

A number of different disks have been used on PCs:

- *ST-506* disks are the oldest. You can recognize them by the fact that they have two cables: a *control cable* that usually has connections for two disks, and a thinner *data cable* that is not shared with any other disk. They're just about completely obsolete by now, but FreeBSD Release 3 still supports them with the *wd* driver. These disks are sometimes called by their modulation format, *Modified Frequency Modulation* or *MFM*. A variant of MFM that offers about 50% more storage is *RLL* or *Run Length Limited* modulation. From the operating system point of view, there is no difference between MFM and RLL.

- *ESDI* (*Enhanced Small Device Interface*) disks were designed to work around some of the limitations of ST-506 drives. They also use the same cabling as ST-506, but they are not hardware compatible, though most ESDI controllers understand ST-506 commands. They are also obsolete, but the *wd* driver in FreeBSD Release 3 supports them, too.

- *IDE* (*Integrated Device Electronics*), now frequently called *ATA* (*AT Attachment*), is the current low-cost PC disk interface. It supports two disks connected by a single 40 or 80 conductor flat cable. The connectors for both cables are the same, but the 80 conductor cable is needed for the 66 MHz, 100 MHz and 133 MHz transfer rates supported by recent disk drives.

  All modern IDE disks are so-called *EIDE* (*Enhanced IDE*) drives. The original IDE disks were limited by the PC BIOS standard to a size of 504 MB (1024 * 16 * 63 * 512, or 528,482,304 bytes). EIDE drives exceed this limit by several orders of magnitude.

  A problem with older IDE controllers was that they used *programmed I/O* or *PIO* to perform the transfer. In this mode, the CPU is directly involved in the transfer to or from the disk. Older controllers transferred a byte at a time, but more modern controllers can transfer in units of 32 bits. Either way, disk transfers use a large amount of CPU time with programmed I/O, and it's difficult to achieve the transfer rates of modern IDE drives, which can be as high as 100 MB/s. During such transfers, the system appears to be unbearably slow: it ''grinds to a halt.''

  To solve this problem, modern chipsets offer DMA transfers, which almost completely eliminate CPU overhead. There are two kinds of DMA, each with multiple possible transfer modes. The older *DMA* mode is no longer in use. It handled transfer rates between 2.1 MB/s and 16.7 MB/s. The newer *UDMA* (*Ultra DMA*) mode supports transfer rates between 16.7 MB/s and 133 MB/s. Current disks use UDMA33 (33 MHz transfer rate), which is the fastest rate you can use with a 40 conductor cable, and UDMA66 (66 MHz), UDMA100 (100 MHz) and UDMA-133 (133 MHz) with an 80 conductor cable. To get this transfer rate, both the disk and the disk controller must support the rate. FreeBSD supports all UDMA modes.

  Another factor influencing IDE performance is the fact that most IDE controllers and disks can only perform one transfer at a time. If you have two disks on a controller, and you want to access both, the controller serializes the requests so that a request to one drive completes before the other starts. This results in worse performance than on a SCSI chain, which does not have this restriction. If you have two disks and two controllers, it's better to put one disk on each controller. This situation is gradually changing, so when choosing hardware it's worth checking on current support for *tagged queueing*, which allows concurrent transfers.

- *SCSI* is the *Small Computer Systems Interface*. It's usually pronounced ''scuzzy.'' It is used for disks, tapes, CD-ROMs and also other devices such as scanners and printers. The SCSI controller is more correctly called a *host adapter*. Like IDE, SCSI has evolved significantly over time. SCSI devices are connected by a single flat cable, with 50 conductors (''narrow SCSI,'' which connects a total of 8 devices) or 68

conductors (‘‘wide SCSI,’’ which also connects up to 16 devices). Some SCSI devices have subdevices, for example CD-ROM changers.

SCSI drives have a reputation for much higher performance than IDE. This is mainly because nearly all SCSI host adapters support DMA, whereas in the past IDE controllers usually used programmed I/O. In addition, SCSI host adapters can perform transfers from multiple units at the same time, whereas IDE controllers can only perform one transfer at a time. Typical SCSI drives are still faster than IDE drives, but the difference is nowhere near as large as it used to be. Narrow SCSI can support transfer rates of up to 40 MB/s (Ultra 2), and wide SCSI can support rates of up to 320 MB/s (Ultra 320). These speeds are not necessarily faster than IDE: you can connect more than seven times as many devices to a wide SCSI chain.

Disk data layout

Before you install FreeBSD, you need to decide how you want to use the disk space available to you. If desired, FreeBSD can coexist with other operating systems on the Intel platform. In this section, we’ll look at the way data is laid out on disk, and what we need to do to create FreeBSD file systems on disk.

PC BIOS and disks

The basics of disk drives are relatively straightforward: data is stored on one or more rotating disks with a magnetic coating similar in function to the coating on an audio tape. Unlike a tape, however, disk heads do not touch the surface: the rotating disk produces an air pressure against the head, which keeps it floating very close to the surface. The disk has (usually) one *read/write head* for each surface to transfer data to and from the system. People frequently talk about the number of heads, not the number of surfaces, though strictly speaking this is incorrect: if there are two heads per surface (to speed up access), you’re still interested in the number of surfaces, not the number of heads.

While transferring data, the heads are stationary, so data is written on disks in a number of concentric circular *tracks*. Logically, each track is divided into a number of *sectors*, which nowadays almost invariably contain 512 bytes. A single positioning mechanism moves the heads from one track to another, so at any one time all the tracks under the current head position can be accessed without repositioning. This group of tracks is called a *cylinder*.

Since the diameter of the track differs from one track to the other, so does the storage capacity per track. Nevertheless, for the sake of simplicity, older drives, such as ST-506 (MFM and RLL) drives, had a fixed number of sectors per track. To perform a data transfer, you needed to tell the drive which cylinder, head and sector to address. This mode of addressing is thus called *CHS* addressing.

Modern disks have a varying number of sectors per track on different parts of the disk to optimize the storage space, and for the same reason they normally store data on the disk in much larger units than sectors. Externally, they translate the data into units of sectors,

and they also optionally maintain the illusion of ‘‘tracks’’ and ‘‘heads,’’ though the values have nothing to do with the internal organization of the disk. Nevertheless, BIOS setup routines still give you the option of specifying information about disk drives in terms of the numbers of cylinders, heads and sectors, and some insist on it. In reality, modern disk drives address sectors sequentially, so-called *Logical Block Addressing* or *LBA*. CHS addressing has an additional problem: various standards have limited the size of disks to 504 MB or 8 GB. We’ll look at that in more detail on page 39.

SCSI drives are a different matter: the system BIOS normally doesn’t know anything about them. They are always addressed in LBA mode. It’s up to the host adapter to interrogate the drive and find out how much space is on it. Typically, the host adapter has a BIOS that interrogates the drive and finds its dimensions. The values it determines may not be correct: the PC BIOS 1 GB address limit (see page 39) might bite you. Check your host adapter documentation for details.

Disk partitioning

The PC BIOS divides the space on a disk into up to four *partitions*, headed by a *partition table*. For Microsoft systems, each partition may be either a *primary partition* that contains a file system (a ‘‘drive’’ in Microsoft terminology), or an *extended partition* that contains multiple file systems (or ‘‘logical partitions’’).

FreeBSD does not use the PC BIOS partition table directly. It maintains its own partitioning scheme with its own partition table. On the PC platform, it places this partition table in a single PC BIOS partition, rather in the same way that a PC BIOS extended partition contains multiple ‘‘logical partitions.’’ It refers to PC BIOS partitions as ‘‘slices.’’

> This double usage of the word *partition* is really confusing. In this book, I follow BSD usage, but I continue to refer to the PC BIOS partition table by that name.

Partitioning offers the flexibility that other operating systems need, so it has been adopted by all operating systems that run on the PC platform. Figure 2-1 shows a disk with all four slices allocated. The *Partition Table* is the most important data structure. It contains information about the size, location and type of the slices (PC partitions). The PC BIOS allows one of these slices to be designated as *active*: at system startup time, its bootstrap record is used to start the system.

The partition table of a boot disk also contains a *Master Boot Record* (*MBR*), which is responsible for finding the correct slice and booting it. The MBR and the partition table take up the first sector on disk, and many people consider them to be the same thing. You only need an MBR on disks from which you boot the system.

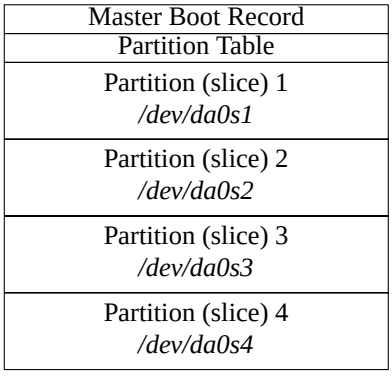

Figure 2-1: Partition table

PC usage designates at least one slice as the *primary partition*, the `C:` drive. Another slice may be designated as an *extended partition* that contains the other ‘‘drives’’ (all together in one slice).

UNIX systems have their own form of partitioning which predates the PC and is not compatible with the PC method. As a result, all versions of UNIX that can coexist with Microsoft implement their own partitioning within a single slice (PC BIOS partition). This is conceptually similar to an extended partition. FreeBSD systems define up to eight partitions per slice. They can be used for the following purposes:

- A partition can be a *file system*, a structure in which UNIX stores files.
- It can be used as a *swap partition*. FreeBSD uses virtual memory: the total addressed memory in the system can exceed the size of physical memory, so we need space on disk to store memory pages that don’t fit into physical memory. Swap is a separate partition for performance reasons: you can use files for swap, like Microsoft does, but it is much less efficient.
- The partition may be used by other system components. For example, the *Vinum* volume manager uses special partitions as building blocks for volumes. We’ll look at Vinum on page 221.
- The partition may not be a real partition at all. For example, partition `c` refers to the entire slice, so it overlaps all the rest. For obvious reasons, the partitions that represent file systems and swap space (`a`, `b`, and `d` through `h`) should not overlap.

Block and character devices

Traditional UNIX treats disk devices in two different ways. As we have seen, you can think of a disk as a large number of sequential blocks of data. Looking at it like this doesn’t give you a file system—it’s more like treating it as a tape. UNIX calls this kind of access *raw* access. You’ll also hear the term *character device*.

Normally, of course, you want files on your disk: you don't care where they are, you just want to be able to open them and manipulate them. In addition, for performance reasons the system keeps recently accessed data in a *buffer cache*. This involves a whole lot more work than raw devices. These devices are called *block devices*.

By contrast with UNIX, Linux originally did not have character disk devices. Starting with Release 4.0, FreeBSD has taken the opposite approach: there are now no user-accessible block devices any more. There are a number of reasons for this:

- Having two different names for devices is confusing. In older releases of FreeBSD, you could recognize block and character devices in an `ls -l` listing by the letters `b` and `c` at the beginning of the permissions. For example, in FreeBSD 3.1 you might have seen:

  ```
  $ ls -l /dev/rwd0s1a   /dev/wd0s1a
  crw-r-----  1 root  operator    3, 131072 Oct 31 19:59 /dev/rwd0s1a
  brw-r-----  1 root  operator    0, 131072 Oct 31 19:59 /dev/wd0s1a
  ```

  wd is the old name for the current *ad* disks. The question is: when do you use which one? Even compared to UNIX System V, the rules were different.

- Nearly all access to disk goes via the file system, and user-accessible block devices add complication.

- If you write to a block device, you don't automatically write to the disk, only into buffer cache. The system decides when to write to disk. If there's a problem writing to disk, there's no way to notify the program that performed the write: it might even already have finished. You can demonstrate this very effectively by comparing the way FreeBSD and Linux write to a floppy disk. It takes 50 seconds to write a complete floppy disk—the speed is determined by the hardware, so the FreeBSD copy program finishes after 50 seconds. With Linux, though, the program runs only for a second or two, after which it finishes and you get your prompt back. In the meantime, the system flushes the data to floppy: you still need to wait a total of 50 seconds. If you remove the floppy in this time, you obviously lose data.

The removal of block devices caused significant changes to device naming. In older releases of FreeBSD, the device name was the name of the block device, and the raw (character) device had the letter `r` at the beginning of the name, as shown in the example above.

Let's look more carefully at how BSD names its partitions:

- Like all other devices, the *device nodes*, the entries that describe the devices, are stored in the directory */dev*. Unlike traditional UNIX and older releases of FreeBSD, FreeBSD Release 5 includes the *device file system* or *devfs*, which creates the device nodes automatically, so you don't need to worry about creating them yourself.

- Next comes the name of the driver. As we have seen, FreeBSD has drivers for IDE and friends (`ad`), SCSI disks (`da`) and floppy disks (`fd`). For SCSI disks, we now have the name */dev/da*.