FreeBSD Handbook

The FreeBSD Documentation Project

FreeBSD Handbook

by The FreeBSD Documentation Project

Published $FreeBSD: head/en_US.ISO8859-1/books/handbook/book.xml 41824 2013-06-01 17:20:11Z eadler $

Welcome to FreeBSD! This handbook covers the installation and day to day use of *FreeBSD 8.4-RELEASE* and *FreeBSD 9.1-RELEASE*. This manual is a *work in progress* and is the work of many individuals. As such, some sections may become dated and require updating. If you are interested in helping out with this project, send email to the FreeBSD documentation project mailing list (http://lists.FreeBSD.org/mailman/listinfo/freebsd-doc). The latest version of this document is always available from the FreeBSD web site (http://www.FreeBSD.org/) (previous versions of this handbook can be obtained from http://docs.FreeBSD.org/doc/). It may also be downloaded in a variety of formats and compression options from the FreeBSD FTP server (ftp://ftp.FreeBSD.org/pub/FreeBSD/doc/) or one of the numerous mirror sites. If you would prefer to have a hard copy of the handbook, you can purchase one at the FreeBSD Mall (http://www.freebsdmall.com/). You may also want to search the handbook (http://www.FreeBSD.org/search/index.html).

Copyright

Table of Contents

List of Tables

Preface

Intended Audience

The FreeBSD newcomer will find that the first section of this book guides the user through the FreeBSD installation process and gently introduces the concepts and conventions that underpin UNIX®. Working through this section requires little more than the desire to explore, and the ability to take on board new concepts as they are introduced.

Once you have traveled this far, the second, far larger, section of the Handbook is a comprehensive reference to all manner of topics of interest to FreeBSD system administrators. Some of these chapters may recommend that you do some prior reading, and this is noted in the synopsis at the beginning of each chapter.

For a list of additional sources of information, please see Appendix B.

Changes from the Third Edition

The current online version of the Handbook represents the cumulative effort of many hundreds of contributors over the past 10 years. The following are some of the significant changes since the two volume third edition was published in 2004:

- Chapter 26, DTrace, has been added with information about the powerful DTrace performance analysis tool.
- Chapter 21, File Systems Support, has been added with information about non-native file systems in FreeBSD, such as ZFS from Sun™.
- Chapter 18, Security Event Auditing, has been added to cover the new auditing capabilities in FreeBSD and explain its use.
- Chapter 23, Virtualization, has been added with information about installing FreeBSD on virtualization software.
- Chapter 2, Installing FreeBSD 9.*x* and Later, has been added to cover installation of FreeBSD using the new installation utility, **bsdinstall**.

Changes from the Second Edition (2004)

The third edition was the culmination of over two years of work by the dedicated members of the FreeBSD Documentation Project. The printed edition grew to such a size that it was necessary to publish as two separate volumes. The following are the major changes in this new edition:

- Chapter 12, Configuration and Tuning, has been expanded with new information about the ACPI power and resource management, the `cron` system utility, and more kernel tuning options.
- Chapter 15, Security, has been expanded with new information about virtual private networks (VPNs), file system access control lists (ACLs), and security advisories.
- Chapter 17, Mandatory Access Control (MAC), is a new chapter with this edition. It explains what MAC is and how this mechanism can be used to secure a FreeBSD system.
- Chapter 19, Storage, has been expanded with new information about USB storage devices, file system snapshots, file system quotas, file and network backed filesystems, and encrypted disk partitions.

- Chapter 22, Vinum, is a new chapter with this edition. It describes how to use Vinum, a logical volume manager which provides device-independent logical disks, and software RAID-0, RAID-1 and RAID-5.
- A troubleshooting section has been added to Chapter 28, PPP and SLIP.
- Chapter 29, Electronic Mail, has been expanded with new information about using alternative transport agents, SMTP authentication, UUCP, **fetchmail**, **procmail**, and other advanced topics.
- Chapter 30, Network Servers, is all new with this edition. This chapter includes information about setting up the **Apache HTTP Server**, **ftpd**, and setting up a server for Microsoft® Windows® clients with **Samba**. Some sections from Chapter 32, Advanced Networking, were moved here to improve the presentation.
- Chapter 32, Advanced Networking, has been expanded with new information about using Bluetooth® devices with FreeBSD, setting up wireless networks, and Asynchronous Transfer Mode (ATM) networking.
- A glossary has been added to provide a central location for the definitions of technical terms used throughout the book.
- A number of aesthetic improvements have been made to the tables and figures throughout the book.

Changes from the First Edition (2001)

The second edition was the culmination of over two years of work by the dedicated members of the FreeBSD Documentation Project. The following were the major changes in this edition:

- A complete Index has been added.
- All ASCII figures have been replaced by graphical diagrams.
- A standard synopsis has been added to each chapter to give a quick summary of what information the chapter contains, and what the reader is expected to know.
- The content has been logically reorganized into three parts: “Getting Started”, “System Administration”, and “Appendices”.
- Chapter 3 (“Installing FreeBSD”) was completely rewritten with many screenshots to make it much easier for new users to grasp the text.
- Chapter 4 (“UNIX Basics”) has been expanded to contain additional information about processes, daemons, and signals.
- Chapter 5 (“Installing Applications”) has been expanded to contain additional information about binary package management.
- Chapter 6 (“The X Window System”) has been completely rewritten with an emphasis on using modern desktop technologies such as **KDE** and **GNOME** on XFree86™ 4.X.
- Chapter 13 (“The FreeBSD Booting Process”) has been expanded.
- Chapter 19 (“Storage”) has been written from what used to be two separate chapters on “Disks” and “Backups”. We feel that the topics are easier to comprehend when presented as a single chapter. A section on RAID (both hardware and software) has also been added.
- Chapter 27 (“Serial Communications”) has been completely reorganized and updated for FreeBSD 4.X/5.X.
- Chapter 28 (“PPP and SLIP”) has been substantially updated.
- Many new sections have been added to Chapter 32 (“Advanced Networking”).

- Chapter 29 (“Electronic Mail”) has been expanded to include more information about configuring **sendmail**.
- Chapter 11 (“Linux® Compatibility”) has been expanded to include information about installing **Oracle®** and **SAP® R/3®**.
- The following new topics are covered in this second edition:
  - Configuration and Tuning (Chapter 12).
  - Multimedia (Chapter 8)

Organization of This Book

This book is split into five logically distinct sections. The first section, *Getting Started*, covers the installation and basic usage of FreeBSD. It is expected that the reader will follow these chapters in sequence, possibly skipping chapters covering familiar topics. The second section, *Common Tasks*, covers some frequently used features of FreeBSD. This section, and all subsequent sections, can be read out of order. Each chapter begins with a succinct synopsis that describes what the chapter covers and what the reader is expected to already know. This is meant to allow the casual reader to skip around to find chapters of interest. The third section, *System Administration*, covers administration topics. The fourth section, *Network Communication*, covers networking and server topics. The fifth section contains appendices of reference information.

Chapter 1, Introduction

Introduces FreeBSD to a new user. It describes the history of the FreeBSD Project, its goals and development model.

Chapter 3, Installation of FreeBSD 8.x and Earlier

Walks a user through the entire installation process of FreeBSD 8.*x* and earlier using **sysinstall**. Some advanced installation topics, such as installing through a serial console, are also covered.

Chapter 2, Installation of FreeBSD 9.x and Later

Walks a user through the entire installation process of FreeBSD 9.*x* and later using **bsdinstall**.

Chapter 4, UNIX Basics

Covers the basic commands and functionality of the FreeBSD operating system. If you are familiar with Linux or another flavor of UNIX then you can probably skip this chapter.

Chapter 5, Installing Applications

Covers the installation of third-party software with both FreeBSD’s innovative “Ports Collection” and standard binary packages.

Chapter 6, The X Window System

Describes the X Window System in general and using X11 on FreeBSD in particular. Also describes common desktop environments such as **KDE** and **GNOME**.

Chapter 7, Desktop Applications

Lists some common desktop applications, such as web browsers and productivity suites, and describes how to install them on FreeBSD.

Chapter 8, Multimedia

Shows how to set up sound and video playback support for your system. Also describes some sample audio and video applications.

Chapter 9, Configuring the FreeBSD Kernel

Explains why you might need to configure a new kernel and provides detailed instructions for configuring, building, and installing a custom kernel.

Chapter 10, Printing

Describes managing printers on FreeBSD, including information about banner pages, printer accounting, and initial setup.

Chapter 11, Linux Binary Compatibility

Describes the Linux compatibility features of FreeBSD. Also provides detailed installation instructions for many popular Linux applications such as **Oracle** and **Mathematica®**.

Chapter 12, Configuration and Tuning

Describes the parameters available for system administrators to tune a FreeBSD system for optimum performance. Also describes the various configuration files used in FreeBSD and where to find them.

Chapter 13, Booting Process

Describes the FreeBSD boot process and explains how to control this process with configuration options.

Chapter 14, Users and Basic Account Management

Describes the creation and manipulation of user accounts. Also discusses resource limitations that can be set on users and other account management tasks.

Chapter 15, Security

Describes many different tools available to help keep your FreeBSD system secure, including Kerberos, IPsec and OpenSSH.

Chapter 16, Jails

Describes the jails framework, and the improvements of jails over the traditional chroot support of FreeBSD.

Chapter 17, Mandatory Access Control

Explains what Mandatory Access Control (MAC) is and how this mechanism can be used to secure a FreeBSD system.

Chapter 18, Security Event Auditing

Describes what FreeBSD Event Auditing is, how it can be installed, configured, and how audit trails can be inspected or monitored.

Chapter 19, Storage

Describes how to manage storage media and filesystems with FreeBSD. This includes physical disks, RAID arrays, optical and tape media, memory-backed disks, and network filesystems.

Section 20.1, GEOM

Describes what the GEOM framework in FreeBSD is and how to configure various supported RAID levels.

Chapter 21, File Systems Support

Examines support of non-native file systems in FreeBSD, like the Z File System from Sun.

Chapter 22, Vinum

Describes how to use Vinum, a logical volume manager which provides device-independent logical disks, and software RAID-0, RAID-1 and RAID-5.

Chapter 23, Virtualization

Describes what virtualization systems offer, and how they can be used with FreeBSD.

Chapter 24, Localization

Describes how to use FreeBSD in languages other than English. Covers both system and application level localization.

Chapter 25, Updating and Upgrading FreeBSD

Explains the differences between FreeBSD-STABLE, FreeBSD-CURRENT, and FreeBSD releases. Describes which users would benefit from tracking a development system and outlines that process. Covers the methods users may take to update their system to the latest security release.

Chapter 26, DTrace

Describes how to configure and use the DTrace tool from Sun in FreeBSD. Dynamic tracing can help locate performance issues, by performing real time system analysis.

Chapter 27, Serial Communications

Explains how to connect terminals and modems to your FreeBSD system for both dial in and dial out connections.

Chapter 28, PPP and SLIP

Describes how to use PPP, SLIP, or PPP over Ethernet to connect to remote systems with FreeBSD.

Chapter 29, Electronic Mail

Explains the different components of an email server and dives into simple configuration topics for the most popular mail server software: **sendmail**.

Chapter 30, Network Servers

Provides detailed instructions and example configuration files to set up your FreeBSD machine as a network filesystem server, domain name server, network information system server, or time synchronization server.

Chapter 31, Firewalls

Explains the philosophy behind software-based firewalls and provides detailed information about the configuration of the different firewalls available for FreeBSD.

Chapter 32, Advanced Networking

Describes many networking topics, including sharing an Internet connection with other computers on your LAN, advanced routing topics, wireless networking, Bluetooth, ATM, IPv6, and much more.

Appendix A, Obtaining FreeBSD

Lists different sources for obtaining FreeBSD media on CDROM or DVD as well as different sites on the Internet that allow you to download and install FreeBSD.

Appendix B, Bibliography

This book touches on many different subjects that may leave you hungry for a more detailed explanation. The bibliography lists many excellent books that are referenced in the text.

Appendix C, Resources on the Internet

Describes the many forums available for FreeBSD users to post questions and engage in technical conversations about FreeBSD.

Appendix D, PGP Keys

Lists the PGP fingerprints of several FreeBSD Developers.

Conventions used in this book

To provide a consistent and easy to read text, several conventions are followed throughout the book.

Typographic Conventions

Italic

An *italic* font is used for filenames, URLs, emphasized text, and the first usage of technical terms.

`Monospace`

A `monospaced` font is used for error messages, commands, environment variables, names of ports, hostnames, user names, group names, device names, variables, and code fragments.

Bold

A **bold** font is used for applications, commands, and keys.

User Input

Keys are shown in **bold** to stand out from other text. Key combinations that are meant to be typed simultaneously are shown with '+' between the keys, such as:

Ctrl+**Alt**+**Del**

Meaning the user should type the **Ctrl**, **Alt**, and **Del** keys at the same time.

Keys that are meant to be typed in sequence will be separated with commas, for example:

Ctrl+**X**, **Ctrl**+**S**

Would mean that the user is expected to type the **Ctrl** and **X** keys simultaneously and then to type the **Ctrl** and **S** keys simultaneously.

Examples

Examples starting with `E:\>` indicate a MS-DOS® command. Unless otherwise noted, these commands may be executed from a "Command Prompt" window in a modern Microsoft Windows environment.

```
E:\> tools\fdimage floppies\kern.flp A:
```

Examples starting with # indicate a command that must be invoked as the superuser in FreeBSD. You can login as `root` to type the command, or login as your normal account and use su(1) to gain superuser privileges.

```
# dd if=kern.flp of=/dev/fd0
```

Examples starting with `%` indicate a command that should be invoked from a normal user account. Unless otherwise noted, C-shell syntax is used for setting environment variables and other shell commands.

```
% top
```

Acknowledgments

The book you are holding represents the efforts of many hundreds of people around the world. Whether they sent in fixes for typos, or submitted complete chapters, all the contributions have been useful.

Several companies have supported the development of this document by paying authors to work on it full-time, paying for publication, etc. In particular, BSDi (subsequently acquired by Wind River Systems (http://www.windriver.com)) paid members of the FreeBSD Documentation Project to work on improving this book full time leading up to the publication of the first printed edition in March 2000 (ISBN 1-57176-241-8). Wind River Systems then paid several additional authors to make a number of improvements to the print-output infrastructure and to add additional chapters to the text. This work culminated in the publication of the second printed edition in November 2001 (ISBN 1-57176-303-1). In 2003-2004, FreeBSD Mall, Inc (http://www.freebsdmall.com), paid several contributors to improve the Handbook in preparation for the third printed edition.

I. Getting Started

This part of the FreeBSD Handbook is for users and administrators who are new to FreeBSD. These chapters:

- Introduce you to FreeBSD.
- Guide you through the installation process.
- Teach you UNIX basics and fundamentals.
- Show you how to install the wealth of third party applications available for FreeBSD.
- Introduce you to X, the UNIX windowing system, and detail how to configure a desktop environment that makes you more productive.

We have tried to keep the number of forward references in the text to a minimum so that you can read this section of the Handbook from front to back with the minimum page flipping required.

Chapter 1 Introduction

Restructured, reorganized, and parts rewritten by Jim Mock.

1.1 Synopsis

Thank you for your interest in FreeBSD! The following chapter covers various aspects of the FreeBSD Project, such as its history, goals, development model, and so on.

After reading this chapter, you will know:

- How FreeBSD relates to other computer operating systems.
- The history of the FreeBSD Project.
- The goals of the FreeBSD Project.
- The basics of the FreeBSD open-source development model.
- And of course: where the name “FreeBSD” comes from.

1.2 Welcome to FreeBSD!

FreeBSD is a 4.4BSD-Lite based operating system for Intel (x86 and Itanium®), AMD64, Sun UltraSPARC® computers. Ports to other architectures are also underway. You can also read about the history of FreeBSD, or the current release. If you are interested in contributing something to the Project (code, hardware, funding), see the Contributing to FreeBSD (http://www.FreeBSD.org/doc/en_US.ISO8859-1/articles/contributing/index.html) article.

1.2.1 What Can FreeBSD Do?

FreeBSD has many noteworthy features. Some of these are:

- *Preemptive multitasking* with dynamic priority adjustment to ensure smooth and fair sharing of the computer between applications and users, even under the heaviest of loads.
- *Multi-user facilities* which allow many people to use a FreeBSD system simultaneously for a variety of things. This means, for example, that system peripherals such as printers and tape drives are properly shared between all users on the system or the network and that individual resource limits can be placed on users or groups of users, protecting critical system resources from over-use.
- Strong *TCP/IP networking* with support for industry standards such as SCTP, DHCP, NFS, NIS, PPP, SLIP, IPsec, and IPv6. This means that your FreeBSD machine can interoperate easily with other systems as well as act as an enterprise server, providing vital functions such as NFS (remote file access) and email services or putting your organization on the Internet with WWW, FTP, routing and firewall (security) services.
-

Memory protection ensures that applications (or users) cannot interfere with each other. One application crashing will not affect others in any way.

- FreeBSD is a *32-bit* operating system (*64-bit* on the Itanium, AMD64, and UltraSPARC) and was designed as such from the ground up.
- The industry standard *X Window System* (X11R7) provides a graphical user interface (GUI) for the cost of a common VGA card and monitor and comes with full sources.
- *Binary compatibility* with many programs built for Linux, SCO, SVR4, BSDI and NetBSD.
- Thousands of *ready-to-run* applications are available from the FreeBSD *ports* and *packages* collection. Why search the net when you can find it all right here?
- 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.
- Demand paged *virtual memory* and "merged VM/buffer cache" design efficiently satisfies applications with large appetites for memory while still maintaining interactive response to other users.
- *SMP* support for machines with multiple CPUs.
- A full complement of *C* and *C++* development tools. Many additional languages for advanced research and development are also available in the ports and packages collection.
- *Source code* for the entire system means you have the greatest degree of control over your environment. Why be locked into a proprietary solution at the mercy of your vendor when you can have a truly open system?
- Extensive *online documentation*.
- *And many more!*

FreeBSD is based on the 4.4BSD-Lite release from Computer Systems Research Group (CSRG) at the University of California at Berkeley, and carries on the distinguished tradition of BSD systems development. In addition to the fine work provided by CSRG, the FreeBSD Project has put in many thousands of hours in fine tuning the system for maximum performance and reliability in real-life load situations. As many of the commercial giants struggle to field PC operating systems with such features, performance and reliability, FreeBSD can offer them *now*!

The applications to which FreeBSD can be put are truly limited only by your own imagination. From software development to factory automation, inventory control to azimuth correction of remote satellite antennae; if it can be done with a commercial UNIX product then it is more than likely that you can do it with FreeBSD too! FreeBSD also benefits significantly from literally thousands of high quality applications developed by research centers and universities around the world, often available at little to no cost. Commercial applications are also available and appearing in greater numbers every day.

Because the source code for FreeBSD itself is generally available, the system can also be customized to an almost unheard of degree for special applications or projects, and in ways not generally possible with operating systems

from most major commercial vendors. Here is just a sampling of some of the applications in which people are currently using FreeBSD:

- *Internet Services:* The robust TCP/IP networking built into FreeBSD makes it an ideal platform for a variety of Internet services such as:
  - FTP servers
  - World Wide Web servers (standard or secure [SSL])
  - IPv4 and IPv6 routing
  - Firewalls and NAT ("IP masquerading") gateways
  - Electronic Mail servers
  - USENET News or Bulletin Board Systems
  - And more...

  With FreeBSD, you can easily start out small with an inexpensive 386 class PC and upgrade all the way up to a quad-processor Xeon with RAID storage as your enterprise grows.
- *Education:* Are you a student of computer science or a related engineering field? There is no better way of learning about operating systems, computer architecture and networking than the hands on, under the hood experience that FreeBSD can provide. 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:* With source code for the entire system available, FreeBSD is an excellent platform for research in operating systems as well as other branches of computer science. FreeBSD's freely available nature 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.
- *Networking:* Need a new router? A name server (DNS)? A firewall to keep people out of your internal network? FreeBSD can easily turn that unused 386 or 486 PC sitting in the corner into an advanced router with sophisticated packet-filtering capabilities.
- *X Window workstation:* FreeBSD is a fine choice for an inexpensive X terminal solution, using the freely available X11 server. 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.

FreeBSD is available in both source and binary form on CD-ROM, DVD, and via anonymous FTP. Please see Appendix A for more information about obtaining FreeBSD.

1.2.2 Who Uses FreeBSD?

FreeBSD is used as a platform for devices and products from many of the world's largest IT companies, including:

- Apple (http://www.apple.com/)
- Cisco (http://www.cisco.com/)
- Juniper (http://www.juniper.net/)
- NetApp (http://www.netapp.com/)

FreeBSD is also used to power some of the biggest sites on the Internet, including:

- Yahoo! (http://www.yahoo.com/)
- Yandex (http://www.yandex.ru/)
- Apache (http://www.apache.org/)
- Rambler (http://www.rambler.ru/)
- Sina (http://www.sina.com/)
- Pair Networks (http://www.pair.com/)
- Sony Japan (http://www.sony.co.jp/)
- Netcraft (http://www.netcraft.com/)
- NetEase (http://www.163.com/)
- Weathernews (http://www.weathernews.com/)

- TELEHOUSE America (http://www.telehouse.com/)
- Experts Exchange (http://www.experts-exchange.com/)

and many more.

1.3 About the FreeBSD Project

The following section provides some background information on the project, including a brief history, project goals, and the development model of the project.

1.3.1 A Brief History of FreeBSD

The FreeBSD Project had its genesis in the early part of 1993, partially as an outgrowth of the Unofficial 386BSDPatchkit by the patchkit's last 3 coordinators: Nate Williams, Rod Grimes and Jordan Hubbard.

The original goal was to produce an intermediate snapshot of 386BSD in order to fix a number of problems with it that the patchkit mechanism just was not capable of solving. The early working title for the project was 386BSD 0.5 or 386BSD Interim in reference of that fact.

386BSD was Bill Jolitz's operating system, which had been up to that point suffering rather severely from almost a year's worth of neglect. As the patchkit swelled ever more uncomfortably with each passing day, they decided to assist Bill by providing this interim "cleanup" snapshot. Those plans came to a rude halt when Bill Jolitz suddenly decided to withdraw his sanction from the project without any clear indication of what would be done instead.

The trio thought that the goal remained worthwhile, even without Bill's support, and so they adopted the name "FreeBSD" coined by David Greenman. The initial objectives were set after consulting with the system's current users and, once it became clear that the project was on the road to perhaps even becoming a reality, Jordan contacted Walnut Creek CDROM with an eye toward improving FreeBSD's distribution channels for those many unfortunates without easy access to the Internet. Walnut Creek CDROM not only supported the idea of distributing FreeBSD on CD but also went so far as to provide the project with a machine to work on and a fast Internet connection. Without Walnut Creek CDROM's almost unprecedented degree of faith in what was, at the time, a completely unknown project, it is quite unlikely that FreeBSD would have gotten as far, as fast, as it has today.

The first CD-ROM (and general net-wide) distribution was FreeBSD 1.0, released in December of 1993. This was based on the 4.3BSD-Lite ("Net/2") tape from U.C. Berkeley, with many components also provided by 386BSD and the Free Software Foundation. It was a fairly reasonable success for a first offering, and they followed it with the highly successful FreeBSD 1.1 release in May of 1994.

Around this time, some rather unexpected storm clouds formed on the horizon as Novell and U.C. Berkeley settled their long-running lawsuit over the legal status of the Berkeley Net/2 tape. A condition of that settlement was U.C. Berkeley's concession that large parts of Net/2 were "encumbered" code and the property of Novell, who had in turn acquired it from AT&T some time previously. What Berkeley got in return was Novell's "blessing" that the 4.4BSD-Lite release, when it was finally released, would be declared unencumbered and all existing Net/2 users would be strongly encouraged to switch. This included FreeBSD, and the project was given until the end of July 1994 to stop shipping its own Net/2 based product. Under the terms of that agreement, the project was allowed one last release before the deadline, that release being FreeBSD 1.1.5.1.

FreeBSD then set about the arduous task of literally re-inventing itself from a completely new and rather incomplete set of 4.4BSD-Lite bits. The “Lite” releases were light in part because Berkeley’s CSRG had removed large chunks of code required for actually constructing a bootable running system (due to various legal requirements) and the fact that the Intel port of 4.4 was highly incomplete. It took the project until November of 1994 to make this transition, at which point it released FreeBSD 2.0 to the net and on CD-ROM (in late December). Despite being still more than a little rough around the edges, the release was a significant success and was followed by the more robust and easier to install FreeBSD 2.0.5 release in June of 1995.

Since that time, FreeBSD has made a series of releases each time improving the stability, speed, and feature set of the previous version.

For now, long-term development projects continue to take place in the 10.X-CURRENT (trunk) branch, and snapshot releases of 10.X are continually made available from the snapshot server (ftp://ftp.FreeBSD.org/pub/FreeBSD/snapshots/) as work progresses.

1.3.2 FreeBSD Project Goals

Contributed by Jordan Hubbard.

The goals of the FreeBSD Project are to provide software that may be used for any purpose and without strings attached. Many of us have a significant investment in the code (and project) and would certainly not mind a little financial compensation now and then, but we are definitely not prepared to insist on it. We believe that our first and foremost “mission” is to provide code to any and all comers, and for whatever purpose, so that the code gets the widest possible use and provides the widest possible benefit. This is, I believe, one of the most fundamental goals of Free Software and one that we enthusiastically support.

That code in our source tree which falls under the GNU General Public License (GPL) or Library General Public License (LGPL) comes with slightly more strings attached, though at least on the side of enforced access rather than the usual opposite. Due to the additional complexities that can evolve in the commercial use of GPL software we do, however, prefer software submitted under the more relaxed BSD copyright when it is a reasonable option to do so.

1.3.3 The FreeBSD Development Model

Contributed by Satoshi Asami.

The development of FreeBSD is a very open and flexible process, being literally built from the contributions of hundreds of people around the world, as can be seen from our list of contributors (http://www.FreeBSD.org/doc/en_US.ISO8859-1/articles/contributors/article.html). FreeBSD’s development infrastructure allow these hundreds of developers to collaborate over the Internet. We are constantly on the lookout for new developers and ideas, and those interested in becoming more closely involved with the project need simply contact us at the FreeBSD technical discussions mailing list (http://lists.FreeBSD.org/mailman/listinfo/freebsd-hackers). The FreeBSD announcements mailing list (http://lists.FreeBSD.org/mailman/listinfo/freebsd-announce) is also available to those wishing to make other FreeBSD users aware of major areas of work.

Useful things to know about the FreeBSD Project and its development process, whether working independently or in close cooperation:

The SVN repositories

For several years, the central source tree for FreeBSD was maintained by CVS (http://www.nongnu.org/cvs/) (Concurrent Versions System), a freely available source code control tool that comes bundled with FreeBSD. In June 2008, the Project switched to using SVN (http://subversion.tigris.org) (Subversion). The switch was deemed necessary, as the technical limitations imposed by **CVS** were becoming obvious due to the rapid expansion of the source tree and the amount of history already stored. The Documentation Project and Ports Collection repositories also moved from **CVS** to **SVN** in May 2012 and July 2012, respectively. Please refer to the Synchronizing your source tree section for more information on obtaining the FreeBSD `src/` repository and Using the Ports Collection for details on obtaining the FreeBSD Ports Collection.

The committers list

The *committers* are the people who have *write* access to the Subversion tree, and are authorized to make modifications to the FreeBSD source (the term "committer" comes from the source control `commit` command, which is used to bring new changes into the repository). The best way of making submissions for review by the committers list is to use the send-pr(1) command. If something appears to be jammed in the system, then you may also reach them by sending mail to the FreeBSD committer's mailing list.

The FreeBSD core team

The *FreeBSD core team* would be equivalent to the board of directors if the FreeBSD Project were a company. The primary task of the core team is to make sure the project, as a whole, is in good shape and is heading in the right directions. Inviting dedicated and responsible developers to join our group of committers is one of the functions of the core team, as is the recruitment of new core team members as others move on. The current core team was elected from a pool of committer candidates in July 2012. Elections are held every 2 years.

Some core team members also have specific areas of responsibility, meaning that they are committed to ensuring that some large portion of the system works as advertised. For a complete list of FreeBSD developers and their areas of responsibility, please see the Contributors List (http://www.FreeBSD.org/doc/en_US.ISO8859-1/articles/contributors/article.html)

> **Note:** Most members of the core team are volunteers when it comes to FreeBSD development and do not benefit from the project financially, so "commitment" should also not be misconstrued as meaning "guaranteed support." The "board of directors" analogy above is not very accurate, and it may be more suitable to say that these are the people who gave up their lives in favor of FreeBSD against their better judgement!

Outside contributors

Last, but definitely not least, the largest group of developers are the users themselves who provide feedback and bug fixes to us on an almost constant basis. The primary way of keeping in touch with FreeBSD's more non-centralized development is to subscribe to the FreeBSD technical discussions mailing list (http://lists.FreeBSD.org/mailman/listinfo/freebsd-hackers) where such things are discussed. See Appendix C for more information about the various FreeBSD mailing lists.

The FreeBSD Contributors List (http://www.FreeBSD.org/doc/en_US.ISO8859-1/articles/contributors/article.html) is a long and growing one, so why not join it by contributing something back to FreeBSD today?

Providing code is not the only way of contributing to the project; for a more complete list of things that need doing, please refer to the FreeBSD Project web site (http://www.FreeBSD.org/index.html).

In summary, our development model is organized as a loose set of concentric circles. The centralized model is designed for the convenience of the *users* of FreeBSD, who are provided with an easy way of tracking one central code base, not to keep potential contributors out! Our desire is to present a stable operating system with a large set of coherent application programs that the users can easily install and use — this model works very well in accomplishing that.

All we ask of those who would join us as FreeBSD developers is some of the same dedication its current people have to its continued success!

1.3.4 Third Party Programs

In addition to the base distributions, FreeBSD offers a ported software collection with thousands of commonly sought-after programs. At the time of this writing, there were over 24,000 ports! The list of ports ranges from http servers, to games, languages, editors, and almost everything in between. The entire Ports Collection requires approximately 500 MB. To compile a port, you simply change to the directory of the program you wish to install, type `make install`, and let the system do the rest. The full original distribution for each port you build is retrieved dynamically so you need only enough disk space to build the ports you want. Almost every port is also provided as a pre-compiled "package", which can be installed with a simple command (`pkg_add`) by those who do not wish to compile their own ports from source. More information on packages and ports can be found in Chapter 5.

1.3.5 Additional Documentation

All recent FreeBSD versions provide an option in the installer (either sysinstall(8) or bsdinstall(8)) to install additional documentation under `/usr/local/share/doc/freebsd` during the initial system setup. Documentation may also be installed at any later time using packages as described in Section 25.4.6.2. You may view the locally installed manuals with any HTML capable browser using the following URLs:

The FreeBSD Handbook

```
/usr/local/share/doc/freebsd/handbook/index.html
```

The FreeBSD FAQ

```
/usr/local/share/doc/freebsd/faq/index.html
```

You can also view the master (and most frequently updated) copies at http://www.FreeBSD.org/.

Chapter 2 Installing FreeBSD 9.*x* and Later

Restructured, reorganized, and parts rewritten by Jim Mock. The sysinstall walkthrough, screenshots, and general copy by Randy Pratt. Updated for bsdinstall by Gavin Atkinson and Warren Block.

2.1 Synopsis

FreeBSD comes with a text-based, easy to use installation program. FreeBSD 9.0-RELEASE and later use an installation program called **bsdinstall**, while releases prior to FreeBSD 9.0-RELEASE using **sysinstall** for installation. This chapter describes the use of **bsdinstall**. The use of **sysinstall** is covered in Chapter 3.

After reading this chapter, you will know:

- How to create the FreeBSD installation media.
- How FreeBSD subdivides and refers to hard disks.
- How to start **bsdinstall**.
- The questions **bsdinstall** will ask you, what they mean, and how to answer them.

Before reading this chapter, you should:

- Read the supported hardware list that shipped with the version of FreeBSD you are installing, and verify that your hardware is supported.

> **Note:** In general, these installation instructions are written for i386™ ("PC compatible") architecture computers. Where applicable, instructions specific to other platforms will be listed. There may be minor differences between the installer and what is shown here, so use this chapter as a general guide rather than as exact literal instructions.

2.2 Hardware Requirements

2.2.1 Minimal Configuration

The minimal configuration to install FreeBSD varies with the FreeBSD version and the hardware architecture.

A summary of this information is given in the following sections. Depending on the method you choose to install FreeBSD, you may also need a supported CDROM drive, and in some cases a network adapter. This will be covered by Section 2.3.5.

2.2.1.1 FreeBSD/i386

FreeBSD/i386 requires a 486 or better processor and at least 64 MB of RAM. At least 1.1 GB of free hard drive space is needed for the most minimal installation.

Note: On old computers, increasing RAM and hard drive space is usually more effective at improving performance than installing a faster processor.

2.2.1.2 FreeBSD/amd64

There are two classes of processors capable of running FreeBSD/amd64. The first are AMD64 processors, including the AMD Athlon™64, AMD Athlon64-FX, AMD Opteron™ or better processors.

The second class of processors that can use FreeBSD/amd64 includes those using the Intel® EM64T architecture. Examples of these processors include the Intel Core™ 2 Duo, Quad, Extreme processor families, the Intel Xeon™ 3000, 5000, and 7000 sequences of processors, and the Intel Core i3, i5 and i7 processors.

If you have a machine based on an nVidia nForce3 Pro-150, you *must* use the BIOS setup to disable the IO APIC. If you do not have an option to do this, you will likely have to disable ACPI instead. There are bugs in the Pro-150 chipset for which we have not yet found a workaround.

2.2.1.3 FreeBSD/powerpc Apple® Macintosh®

All New World Apple® Macintosh® systems with built-in USB are supported. SMP is supported on machines with multiple CPUs.

A 32-bit kernel can only use the first 2 GB of RAM. FireWire® is not supported on the Blue & White PowerMac G3.

2.2.1.4 FreeBSD/sparc64

Systems supported by FreeBSD/sparc64 are listed at the FreeBSD/sparc64 (http://www.freebsd.org/platforms/sparc.html) Project.

A dedicated disk is required for FreeBSD/sparc64. It is not possible to share a disk with another operating system at this time.

2.2.2 Supported Hardware

Hardware architectures and devices supported by a FreeBSD release are listed in the Hardware Notes file. Usually named `HARDWARE.TXT`, the file is located in the root directory of the release media. Copies of the supported hardware list are also available on the Release Information (http://www.FreeBSD.org/releases/index.html) page of the FreeBSD web site.

2.3 Pre-Installation Tasks

2.3.1 Back Up Your Data

Back up all important data on the target computer where FreeBSD will be installed. Test the backups before continuing. The FreeBSD installer will ask before making changes to the disk, but once the process has started it

cannot be undone.

2.3.2 Decide Where to Install FreeBSD

If FreeBSD will be the only operating system installed, and will be allowed to use the entire hard disk, the rest of this section can be skipped. But if FreeBSD will share the disk with other operating systems, an understanding of disk layout is useful during the installation.

2.3.2.1 Disk Layouts for FreeBSD/i386 and FreeBSD/amd64

Hard disks can be divided into multiple sections. These sections are called *partitions*.

There are two ways of dividing a disk into partitions. A traditional *Master Boot Record* (MBR) holds a partition table defining up to four *primary partitions*. (For historical reasons, FreeBSD calls primary partitions *slices*.) A limit of only four partitions is restrictive for large disks, so one of these primary partitions can be made into an *extended partition*. Multiple *logical partitions* may then be created inside the extended partition. This may sound a little unwieldy, and it is.

The *GUID Partition Table* (GPT) is a newer and simpler method of partitioning a disk. GPT is far more versatile than the traditional MBR partition table. Common GPT implementations allow up to 128 partitions per disk, eliminating the need for inconvenient workarounds like logical partitions.

> **Warning:** Some older operating systems like Windows XP are not compatible with the GPT partition scheme. If FreeBSD will be sharing a disk with such an operating system, MBR partitioning is required.

FreeBSD's standard boot loader requires either a primary or GPT partition. (See Chapter 13 for more information about the FreeBSD booting process.) If all of the primary or GPT partitions are already in use, one must be freed for FreeBSD.

A minimal installation of FreeBSD takes as little as 1 GB of disk space. However, that is a *very* minimal install, leaving almost no free space. A more realistic minimum is 3 GB without a graphical environment, and 5 GB or more if a graphical user interface will be used. Third-party application software requires more space.

A variety of free and commercial partition resizing tools (http://en.wikipedia.org/wiki/List_of_disk_partitioning_software) are available. GParted Live (http://gparted.sourceforge.net/livecd.php) is a free Live CD which includes the **GParted** partition editor. **GParted** is also included with many other Linux Live CD distributions.

> **Warning:** Disk partition applications can destroy data. Make a full backup and verify its integrity before modifying disk partitions.

Resizing Microsoft Vista partitions can be difficult. A Vista installation CDROM can be useful when attempting such an operation.

Example 2-1. Using an Existing Partition

A Windows computer has a single 40 GB disk that has been split into two 20 GB partitions. Windows calls them `C:` and `D:`. The `C:` partition contains 10 GB of data, and the `D:` partition contains 5 GB of data.

Moving the data from `D:` to `C:` frees up the second partition to be used for FreeBSD.

Example 2-2. Shrinking an Existing Partition

A Windows computer has a single 40 GB disk and one large partition using the whole disk. Windows shows this 40 GB partition as a single `C:`. 15 GB of space is being used. The goal is to end up with Windows in a 20 GB partition, and have another 20 GB partition for FreeBSD.

There are two ways to do this:

1. Back up your Windows data. Then reinstall Windows, creating a 20 GB partition during the install.
2. Use a partition resizing tool like **GParted** to shrink the Windows partition and create a new partition in the freed space for FreeBSD.

Disk partitions containing different operating systems make it possible to run any one of those operating systems at a time. An alternative method that allows running multiple operating systems at the same time is covered in Chapter 23.

2.3.3 Collect Network Information

Some FreeBSD installation methods need a network connection to download files. To connect to an Ethernet network (or cable or DSL modem with an Ethernet interface), the installer will request some information about the network.

DHCP is commonly used to provide automatic network configuration. If DHCP is not available, this network information must be obtained from the local network administrator or service provider:

Network Information

1. IP address
2. Subnet mask
3. Default router IP address
4. Domain name of the local network
5. DNS server IP address(es)

2.3.4 Check for FreeBSD Errata

Although the FreeBSD Project strives to ensure that each release of FreeBSD is as stable as possible, bugs occasionally creep into the process. On very rare occasions those bugs affect the installation process. As these problems are discovered and fixed, they are noted in the FreeBSD Errata (http://www.FreeBSD.org/releases/9.1R/errata.html) on the FreeBSD web site. Check the errata before installing to make sure that there are no problems that might affect the installation.

Information and errata for all the releases can be found on the release information (http://www.FreeBSD.org/releases/index.html) section of the FreeBSD web site (http://www.FreeBSD.org/index.html).

2.3.5 Prepare the Installation Media

A FreeBSD installation is started by booting the computer with a FreeBSD installation CD, DVD, or USB memory stick. The installer is not a program that can be run from within another operating system.

In addition to the standard installation media which contains copies of all the FreeBSD installation files, there is a *bootonly* variant. Bootonly install media does not have copies of the installation files, but downloads them from the network during an install. The bootonly install CD is consequently much smaller, and reduces bandwidth usage during the install by only downloading required files.

Copies of FreeBSD installation media are available at the FreeBSD web site (http://www.FreeBSD.org/where.html#download).

> **Tip:** If you already have a copy of FreeBSD on CDROM, DVD, or USB memory stick, this section can be skipped.

FreeBSD CD and DVD images are bootable ISO files. Only one CD or DVD is needed for an install. Burn the ISO image to a bootable CD or DVD using the CD recording applications available with your current operating system.

To create a bootable memory stick, follow these steps:

1. Acquire the Memory Stick Image

   Memory stick images for FreeBSD 9.0-RELEASE and later can be downloaded from the `ISO-IMAGES/` directory at `ftp://ftp.FreeBSD.org/pub/FreeBSD/releases/`*`arch`*`/`*`arch`*`/ISO-IMAGES/`*`version`*`/FreeBSD-`*`version`*`-RELEASE-`
   Replace *`arch`* and *`version`* with the architecture and the version number which you want to install, respectively. For example, the memory stick images for FreeBSD/i386 9.0-RELEASE are available from ftp://ftp.FreeBSD.org/pub/FreeBSD/releases/i386/i386/ISO-IMAGES/9.0/FreeBSD-9.0-RELEASE-i386-memstick.img.

   > **Tip:** A different directory path is used for FreeBSD 8.*X* and earlier versions. Details of download and installation of FreeBSD 8.*X* and earlier is covered in Chapter 3.

   The memory stick image has a `.img` extension. The `ISO-IMAGES/` directory contains a number of different images, and the one needed depends on the version of FreeBSD being installed, and in some cases, the target hardware.

   > **Important:** Before proceeding, *back up* the data on the USB stick, as this procedure will *erase* it.

2. Write the Image File to the Memory Stick

Using FreeBSD to Write the Image

Warning: The example below shows `/dev/da0` as the target device where the image will be written. Be very careful that the correct device is used as the output target, or you may destroy existing data.

1. Writing the Image with dd(1)

   The `.img` file is *not* a regular file. It is an *image* of the complete contents of the memory stick. It *cannot* simply be copied like a regular file, but must be written directly to the target device with dd(1):

   ```
   # dd if=FreeBSD-9.0-RELEASE-i386-memstick.img of=/dev/da0 bs=64k
   ```

Using Windows® to Write the Image

Warning: Be sure to give the correct drive letter as the output target, or you may overwrite and destroy existing data.

1. Obtaining **Image Writer for Windows**

   Image Writer for Windows is a free application that can correctly write an image file to a memory stick. Download it from https://launchpad.net/win32-image-writer/ and extract it into a folder.

2. Writing the Image with Image Writer

   Double-click the **Win32DiskImager** icon to start the program. Verify that the drive letter shown under `Device` is the drive with the memory stick. Click the folder icon and select the image to be written to the memory stick. Click [Save] to accept the image file name. Verify that everything is correct, and that no folders on the memory stick are open in other windows. When everything is ready, click [Write] to write the image file to the memory stick.

Note: Installation from floppy disks is no longer supported.

You are now ready to start installing FreeBSD.

2.4 Starting the Installation

Important: By default, the installation will not make any changes to your disk(s) until you see the following message:

```
Your changes will now be written to disk.  If you
have chosen to overwrite existing data, it will
be PERMANENTLY ERASED. Are you sure you want to
commit your changes?
```

The install can be exited at any time prior to this warning without changing the contents of the hard drive. If you are concerned that you have configured something incorrectly you can just turn the computer off before this point, and no damage will be done.

2.4.1 Booting

2.4.1.1 Booting on i386™ and amd64

1. If you prepared a "bootable" USB stick, as described in Section 2.3.5, then plug in your USB stick before turning on the computer.

   If you are booting from CDROM, then you will need to turn on the computer, and insert the CDROM at the first opportunity.

2. Configure your machine to boot from either the CDROM or from USB, depending on the media being used for the installation. BIOS configurations allow the selection of a specific boot device. Most systems also provide for selecting a boot device during startup, typically by pressing **F10**, **F11**, **F12**, or **Escape**.

3. If your computer starts up as normal and loads your existing operating system, then either:

   1. The disks were not inserted early enough in the boot process. Leave them in, and try restarting your computer.
   2. The BIOS changes earlier did not work correctly. You should redo that step until you get the right option.
   3. Your particular BIOS does not support booting from the desired media. The Plop Boot Manager (http://www.plop.at/en/bootmanager.html) can be used to boot older computers from CD or USB media.

4. FreeBSD will start to boot. If you are booting from CDROM you will see a display similar to this (version information omitted):

```
Booting from CD-ROM...
645MB medium detected
CD Loader 1.2

Building the boot loader arguments
Looking up /BOOT/LOADER... Found
Relocating the loader and the BTX
Starting the BTX loader

BTX loader 1.00 BTX version is 1.02
Consoles: internal video/keyboard
BIOS CD is cd0
BIOS drive C: is disk0
BIOS drive D: is disk1
BIOS 636kB/261056kB available memory

FreeBSD/i386 bootstrap loader, Revision 1.1

Loading /boot/defaults/loader.conf
/boot/kernel/kernel text=0x64daa0 data=0xa4e80+0xa9e40 syms=[0x4+0x6cac0+0x4+0x88e9d]
\
```

5. The FreeBSD boot loader is displayed:

Figure 2-1. FreeBSD Boot Loader Menu

Either wait ten seconds, or press **Enter**.

2.4.1.2 Booting for Macintosh PowerPC®

On most machines, holding **C** on the keyboard during boot will boot from the CD. Otherwise, hold **Command**+**Option**+**O**+**F**, or **Windows**+**Alt**+**O**+**F** on non-Apple keyboards. At the `0 >` prompt, enter

```
boot cd:,\ppc\loader cd:0
```

For Xserves without keyboards, see Apple's support web site (http://support.apple.com/kb/TA26930) about booting into Open Firmware.

2.4.1.3 Booting for SPARC64®

Most SPARC64® systems are set up to boot automatically from disk. To install FreeBSD, you need to boot over the network or from a CDROM, which requires you to break into the PROM (OpenFirmware).

To do this, reboot the system, and wait until the boot message appears. It depends on the model, but should look about like:

```
Sun Blade 100 (UltraSPARC-IIe), Keyboard Present
Copyright 1998-2001 Sun Microsystems, Inc.  All rights reserved.
OpenBoot 4.2, 128 MB memory installed, Serial #51090132.
Ethernet address 0:3:ba:b:92:d4, Host ID: 830b92d4.
```

If your system proceeds to boot from disk at this point, you need to press **L1**+**A** or **Stop**+**A** on the keyboard, or send a `BREAK` over the serial console (using for example ~# in tip(1) or cu(1)) to get to the PROM prompt. It looks like this:

```
ok    ❶
ok {0} ❷
```

❶ This is the prompt used on systems with just one CPU.

❷ This is the prompt used on SMP systems, the digit indicates the number of the active CPU.

At this point, place the CDROM into your drive, and from the PROM prompt, type `boot cdrom`.

2.4.2 Reviewing the Device Probe Results

The last few hundred lines that have been displayed on screen are stored and can be reviewed.

To review the buffer, press **Scroll Lock**. This turns on scrolling in the display. You can then use the arrow keys, or **PageUp** and **PageDown** to view the results. Press **Scroll Lock** again to stop scrolling.

Do this now, to review the text that scrolled off the screen when the kernel was carrying out the device probes. You will see text similar to Figure 2-2, although the precise text will differ depending on the devices that you have in your computer.

Figure 2-2. Typical Device Probe Results

```
Copyright (c) 1992-2011 The FreeBSD Project.
Copyright (c) 1979, 1980, 1983, 1986, 1988, 1989, 1991, 1992, 1993, 1994
        The Regents of the University of California. All rights reserved.
FreeBSD is a registered trademark of The FreeBSD Foundation.
FreeBSD 9.0-RELEASE #0 r225473M: Sun Sep 11 16:07:30 BST 2011
    root@psi:/usr/obj/usr/src/sys/GENERIC amd64
CPU: Intel(R) Core(TM)2 Duo CPU     T9400  @ 2.53GHz (2527.05-MHz K8-class CPU)
  Origin = "GenuineIntel"  Id = 0x10676  Family = 6  Model = 17  Stepping = 6
  Features=0xbfebfbff<FPU,VME,DE,PSE,TSC,MSR,PAE,MCE,CX8,APIC,SEP,MTRR,PGE,MCA,CMOV,PAT,PSE36,CLF
  Features2=0x8e3fd<SSE3,DTES64,MON,DS_CPL,VMX,SMX,EST,TM2,SSSE3,CX16,xTPR,PDCM,SSE4.1>
  AMD Features=0x20100800<SYSCALL,NX,LM>
  AMD Features2=0x1<LAHF>
  TSC: P-state invariant, performance statistics
real memory  = 3221225472 (3072 MB)
avail memory = 2926649344 (2791 MB)
Event timer "LAPIC" quality 400
ACPI APIC Table: <TOSHIB A0064   >
FreeBSD/SMP: Multiprocessor System Detected: 2 CPUs
FreeBSD/SMP: 1 package(s) x 2 core(s)
 cpu0 (BSP): APIC ID:  0
 cpu1 (AP): APIC ID:  1
ioapic0: Changing APIC ID to 1
ioapic0 <Version 2.0> irqs 0-23 on motherboard
kbd1 at kbdmux0
acpi0: <TOSHIB A0064> on motherboard
acpi0: Power Button (fixed)
acpi0: reservation of 0, a0000 (3) failed
acpi0: reservation of 100000, b6690000 (3) failed
Timecounter "ACPI-safe" frequency 3579545 Hz quality 850
acpi_timer0: <24-bit timer at 3.579545MHz> port 0xd808-0xd80b on acpi0
cpu0: <ACPI CPU> on acpi0
ACPI Warning: Incorrect checksum in table [ASF!] - 0xFE, should be 0x9A (20110527/tbutils-282)
cpu1: <ACPI CPU> on acpi0
```

```
pcib0: <ACPI Host-PCI bridge> port 0xcf8-0xcff on acpi0
pci0: <ACPI PCI bus> on pcib0
vgapci0: <VGA-compatible display> port 0xcff8-0xcfff mem 0xff400000-0xff7fffff,0xe0000000-0xefff
agp0: <Intel GM45 SVGA controller> on vgapci0
agp0: aperture size is 256M, detected 131068k stolen memory
vgapci1: <VGA-compatible display> mem 0xffc00000-0xffcfffff at device 2.1 on pci0
pci0: <simple comms> at device 3.0 (no driver attached)
em0: <Intel(R) PRO/1000 Network Connection 7.2.3> port 0xcf80-0xcf9f mem 0xff9c0000-0xff9dffff,0x
em0: Using an MSI interrupt
em0: Ethernet address: 00:1c:7e:6a:ca:b0
uhci0: <Intel 82801I (ICH9) USB controller> port 0xcf60-0xcf7f irq 16 at device 26.0 on pci0
usbus0: <Intel 82801I (ICH9) USB controller> on uhci0
uhci1: <Intel 82801I (ICH9) USB controller> port 0xcf40-0xcf5f irq 21 at device 26.1 on pci0
usbus1: <Intel 82801I (ICH9) USB controller> on uhci1
uhci2: <Intel 82801I (ICH9) USB controller> port 0xcf20-0xcf3f irq 19 at device 26.2 on pci0
usbus2: <Intel 82801I (ICH9) USB controller> on uhci2
ehci0: <Intel 82801I (ICH9) USB 2.0 controller> mem 0xff9ff800-0xff9ffbff irq 19 at device 26.7 o
usbus3: EHCI version 1.0
usbus3: <Intel 82801I (ICH9) USB 2.0 controller> on ehci0
hdac0: <Intel 82801I High Definition Audio Controller> mem 0xff9f8000-0xff9fbfff irq 22 at device
pcib1: <ACPI PCI-PCI bridge> irq 17 at device 28.0 on pci0
pci1: <ACPI PCI bus> on pcib1
iwn0: <Intel(R) WiFi Link 5100> mem 0xff8fe000-0xff8fffff irq 16 at device 0.0 on pci1
pcib2: <ACPI PCI-PCI bridge> irq 16 at device 28.1 on pci0
pci2: <ACPI PCI bus> on pcib2
pcib3: <ACPI PCI-PCI bridge> irq 18 at device 28.2 on pci0
pci4: <ACPI PCI bus> on pcib3
pcib4: <ACPI PCI-PCI bridge> at device 30.0 on pci0
pci5: <ACPI PCI bus> on pcib4
cbb0: <RF5C476 PCI-CardBus Bridge> at device 11.0 on pci5
cardbus0: <CardBus bus> on cbb0
pccard0: <16-bit PCCard bus> on cbb0
isab0: <PCI-ISA bridge> at device 31.0 on pci0
isa0: <ISA bus> on isab0
ahci0: <Intel ICH9M AHCI SATA controller> port 0x8f58-0x8f5f,0x8f54-0x8f57,0x8f48-0x8f4f,0x8f44-0
ahci0: AHCI v1.20 with 4 3Gbps ports, Port Multiplier not supported
ahcich0: <AHCI channel> at channel 0 on ahci0
ahcich1: <AHCI channel> at channel 1 on ahci0
ahcich2: <AHCI channel> at channel 4 on ahci0
acpi_lid0: <Control Method Lid Switch> on acpi0
battery0: <ACPI Control Method Battery> on acpi0
acpi_button0: <Power Button> on acpi0
acpi_acad0: <AC Adapter> on acpi0
acpi_toshiba0: <Toshiba HCI Extras> on acpi0
acpi_tz0: <Thermal Zone> on acpi0
attimer0: <AT timer> port 0x40-0x43 irq 0 on acpi0
Timecounter "i8254" frequency 1193182 Hz quality 0
Event timer "i8254" frequency 1193182 Hz quality 100
atkbdc0: <Keyboard controller (i8042)> port 0x60,0x64 irq 1 on acpi0
atkbd0: <AT Keyboard> irq 1 on atkbdc0
kbd0 at atkbd0
atkbd0: [GIANT-LOCKED]
psm0: <PS/2 Mouse> irq 12 on atkbdc0
```

```
psm0: [GIANT-LOCKED]
psm0: model GlidePoint, device ID 0
atrtc0: <AT realtime clock> port 0x70-0x71 irq 8 on acpi0
Event timer "RTC" frequency 32768 Hz quality 0
hpet0: <High Precision Event Timer> iomem 0xfed00000-0xfed003ff on acpi0
Timecounter "HPET" frequency 14318180 Hz quality 950
Event timer "HPET" frequency 14318180 Hz quality 450
Event timer "HPET1" frequency 14318180 Hz quality 440
Event timer "HPET2" frequency 14318180 Hz quality 440
Event timer "HPET3" frequency 14318180 Hz quality 440
uart0: <16550 or compatible> port 0x3f8-0x3ff irq 4 flags 0x10 on acpi0
sc0: <System console> at flags 0x100 on isa0
sc0: VGA <16 virtual consoles, flags=0x300>
vga0: <Generic ISA VGA> at port 0x3c0-0x3df iomem 0xa0000-0xbffff on isa0
ppc0: cannot reserve I/O port range
est0: <Enhanced SpeedStep Frequency Control> on cpu0
p4tcc0: <CPU Frequency Thermal Control> on cpu0
est1: <Enhanced SpeedStep Frequency Control> on cpu1
p4tcc1: <CPU Frequency Thermal Control> on cpu1
Timecounters tick every 1.000 msec
hdac0: HDA Codec #0: Realtek ALC268
hdac0: HDA Codec #1: Lucent/Agere Systems (Unknown)
pcm0: <HDA Realtek ALC268 PCM #0 Analog> at cad 0 nid 1 on hdac0
pcm1: <HDA Realtek ALC268 PCM #1 Analog> at cad 0 nid 1 on hdac0
usbus0: 12Mbps Full Speed USB v1.0
usbus1: 12Mbps Full Speed USB v1.0
usbus2: 12Mbps Full Speed USB v1.0
usbus3: 480Mbps High Speed USB v2.0
ugen0.1: <Intel> at usbus0
uhub0: <Intel UHCI root HUB, class 9/0, rev 1.00/1.00, addr 1> on usbus0
ugen1.1: <Intel> at usbus1
uhub1: <Intel UHCI root HUB, class 9/0, rev 1.00/1.00, addr 1> on usbus1
ugen2.1: <Intel> at usbus2
uhub2: <Intel UHCI root HUB, class 9/0, rev 1.00/1.00, addr 1> on usbus2
ugen3.1: <Intel> at usbus3
uhub3: <Intel EHCI root HUB, class 9/0, rev 2.00/1.00, addr 1> on usbus3
uhub0: 2 ports with 2 removable, self powered
uhub1: 2 ports with 2 removable, self powered
uhub2: 2 ports with 2 removable, self powered
uhub3: 6 ports with 6 removable, self powered
ugen2.2: <vendor 0x0b97> at usbus2
uhub8: <vendor 0x0b97 product 0x7761, class 9/0, rev 1.10/1.10, addr 2> on usbus2
ugen1.2: <Microsoft> at usbus1
ada0 at ahcich0 bus 0 scbus1 target 0 lun 0
ada0: <Hitachi HTS543225L9SA00 FBEOC43C> ATA-8 SATA 1.x device
ada0: 150.000MB/s transfers (SATA 1.x, UDMA6, PIO 8192bytes)
ada0: Command Queueing enabled
ada0: 238475MB (488397168 512 byte sectors: 16H 63S/T 16383C)
ada0: Previously was known as ad4
ums0: <Microsoft Microsoft 3-Button Mouse with IntelliEyeTM, class 0/0, rev 1.10/3.00, addr 2> on
SMP: AP CPU #1 Launched!
cd0 at ahcich1 bus 0 scbus2 target 0 lun 0
cd0: <TEAC DV-W28S-RT 7.0C> Removable CD-ROM SCSI-0 device
```

```
cd0: 150.000MB/s transfers (SATA 1.x, ums0: 3 buttons and [XYZ] coordinates ID=0
UDMA2, ATAPI 12bytes, PIO 8192bytes)
cd0: cd present [1 x 2048 byte records]
ugen0.2: <Microsoft> at usbus0
ukbd0: <Microsoft Natural Ergonomic Keyboard 4000, class 0/0, rev 2.00/1.73, addr 2> on usbus0
kbd2 at ukbd0
uhid0: <Microsoft Natural Ergonomic Keyboard 4000, class 0/0, rev 2.00/1.73, addr 2> on usbus0
Trying to mount root from cd9660:/dev/iso9660/FREEBSD_INSTALL [ro]...
```

Check the probe results carefully to make sure that FreeBSD found all the devices you expected. If a device was not found, then it will not be listed. Kernel modules allows you to add in support for devices which are not in the `GENERIC` kernel.

After the procedure of device probing, you will see Figure 2-3. The install media can be used in three ways: to install FreeBSD, as a live CD, or to simply access a FreeBSD shell. Use the arrow keys to choose an option, and **Enter** to select.

Figure 2-3. Selecting Installation Media Mode

Selecting [Install] here will enter the installer.

2.5 Introducing bsdinstall

bsdinstall is a text-based FreeBSD installer program written by Nathan Whitehorn <`nwhitehorn@FreeBSD.org`> and introduced in 2011 for FreeBSD 9.0.

> **Note:** Kris Moore <`kmoore@FreeBSD.org`>'s **pc-sysinstall** is included with PC-BSD (http://pcbsd.org), and can also be used to install FreeBSD (http://wiki.pcbsd.org/index.php/Use_PC-BSD_Installer_to_Install_FreeBSD). Although sometimes confused with **bsdinstall**, the two are not related.

The **bsdinstall** menu system is controlled by the arrow keys, **Enter**, **Tab**, **Space**, and other keys.

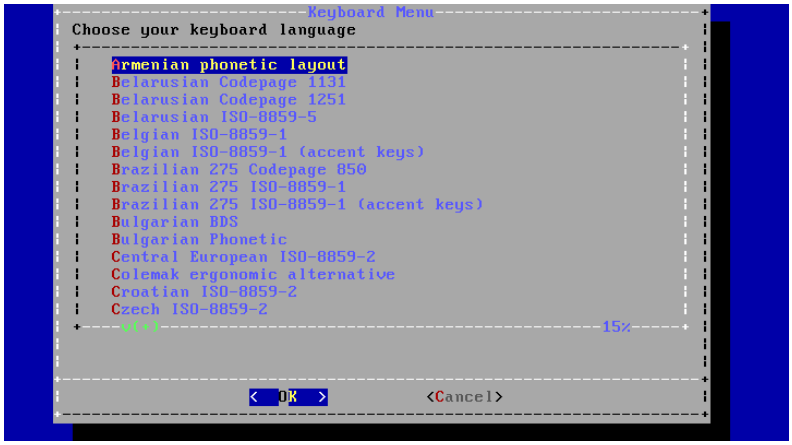
2.5.1 Selecting the Keymap Menu

Depending on the system console being used, **bsdinstall** may initially prompt to select a non-default keyboard layout.

Figure 2-4. Keymap Selection

If [YES] is selected, the following keyboard selection screen will be displayed. Otherwise, this selection menu will not be displayed, and a default keyboard mapping will be used.

Figure 2-5. Selecting Keyboard Menu



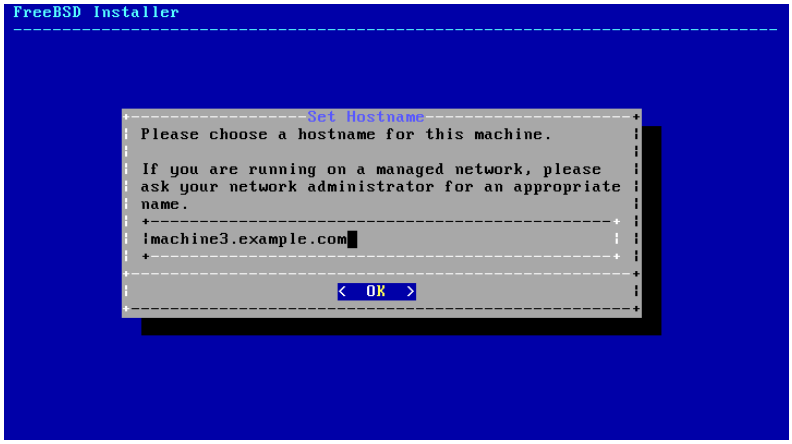
Select the keymap that most closely represents the mapping of the keyboard attached to the system, using the up/down arrow keys and pressing **Enter**.

> **Note:** Pressing **Esc** will use the default keymap. United States of America ISO-8859-1 is also a safe option if the choice of keymap is not clear.

2.5.2 Setting the Hostname

Next, **bsdinstall** will prompt for the hostname to be given to the newly installed system.

Figure 2-6. Setting the Hostname

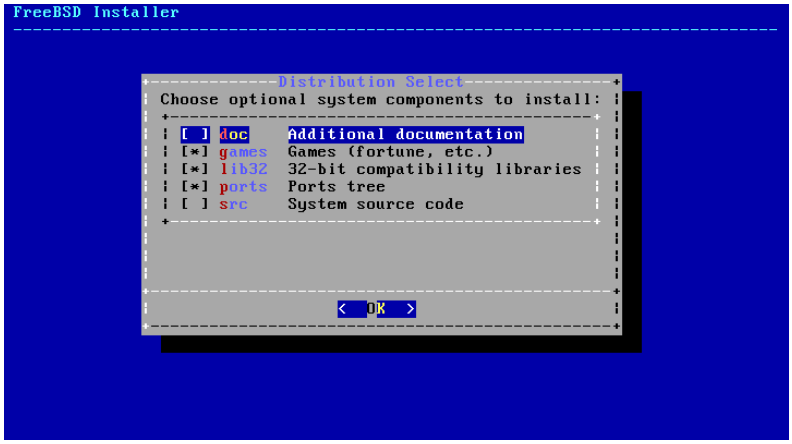


The entered hostname should be a fully-qualified hostname, such as `machine3.example.com`

2.5.3 Selecting Components to Install

Next, **bsdinstall** will prompt to select optional components to install.

Figure 2-7. Selecting Components to Install



Deciding which components to install will depend largely on the intended use of the system and the amount of disk space available. The FreeBSD Kernel and userland (collectively the “base system”) are always installed.

Depending on the type of installation, some of these components may not appear.

Optional Components

- `doc` - Additional documentation, mostly of historical interest. Documentation provided by the FreeBSD Documentation Project may be installed later.
- `games` - Several traditional BSD games, including **fortune**, **rot13**, and others.
- `lib32` - Compatibility libraries for running 32-bit applications on a 64-bit version of FreeBSD.
- `ports` - The FreeBSD Ports Collection.

  The ports collection is an easy and convenient way to install software. The Ports Collection does not contain the source code necessary to compile the software. Instead, it is a collection of files which automates the downloading, compiling and installation of third-party software packages. Chapter 5 discusses how to use the ports collection.

  > **Warning:** The installation program does not check to see if you have adequate space. Select this option only if you have adequate hard disk space. As of FreeBSD 9.0, the FreeBSD Ports Collection takes up about 500 MB of disk space. You can safely assume a larger value for more recent versions of FreeBSD.

- `src` - System source code.

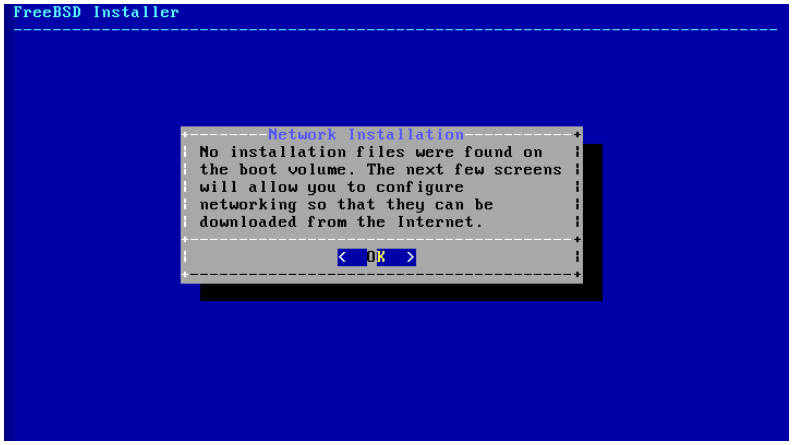
  FreeBSD comes with full source code for both the kernel and the userland. Although not required for the majority of applications, it may be required to build certain software supplied as source (for example, device drivers or kernel modules), or for developing FreeBSD itself.

  The full source tree requires 1 GB of disk space, and recompiling the entire FreeBSD system requires an additional 5 GB of space.

2.6 Installing from the Network

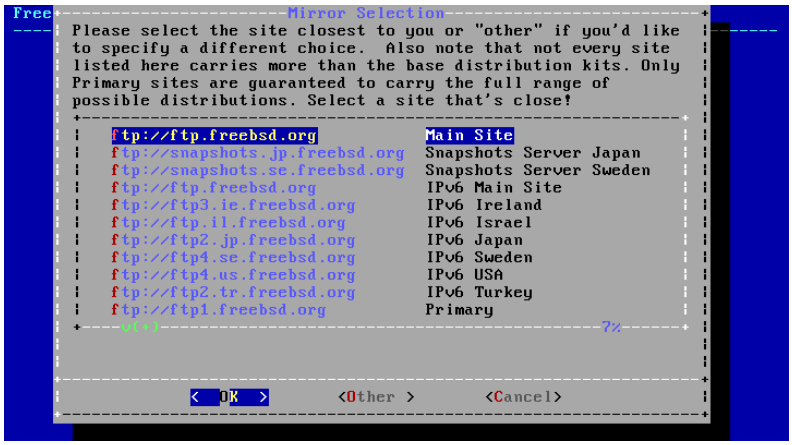
The *bootonly* installation media does not hold copies of the installation files. When a *bootonly* installation method is used, the files must be retrieved over a network connection as they are needed.

Figure 2-8. Installing from the Network



After the network connection has been configured as shown in Section 2.9.2, a mirror site is selected. Mirror sites cache copies of the FreeBSD files. Choose a mirror site located in the same region of the world as the computer on which FreeBSD is being installed. Files can be retrieved more quickly when the mirror is close to the target computer, and installation time will be reduced.

Figure 2-9. Choosing a Mirror



Installation will continue as if the installation files were located on local media.

2.7 Allocating Disk Space

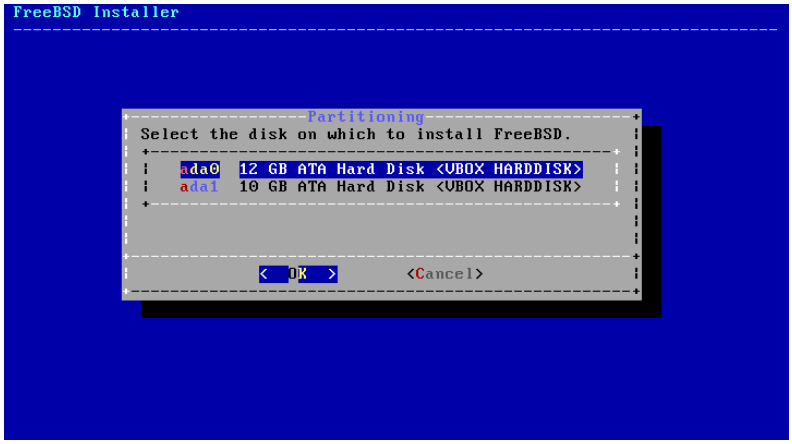
There are three ways to allocate disk space for FreeBSD. *Guided* partitioning automatically sets up disk partitions, while *Manual* partitioning allows advanced users to create customized partitions. Finally, there's the option of starting a shell where command-line programs like gpart(8), fdisk(8), and bsdlabel(8) can be used directly.

Figure 2-10. Selecting Guided or Manual Partitioning

2.7.1 Guided Partitioning

If multiple disks are connected, choose the one where FreeBSD is to be installed.

Figure 2-11. Selecting from Multiple Disks

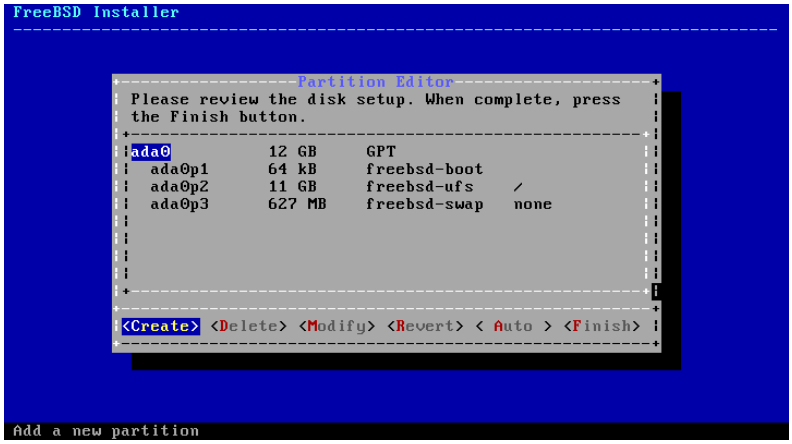


The entire disk can be allocated to FreeBSD, or just a portion of it. If [Entire Disk] is chosen, a general partition layout filling the whole disk is created. Selecting [Partition] creates a partition layout in unused space on the disk.

Figure 2-12. Selecting Entire Disk or Partition

After the partition layout has been created, review it carefully for accuracy. If a mistake has been made, selecting [Revert] will reset the partitions as they were previously, or [Auto] will recreate the automatic FreeBSD partitions. Partitions can be manually created, modified, or deleted. When the partitioning is correct, select [Finish] to continue with the installation.

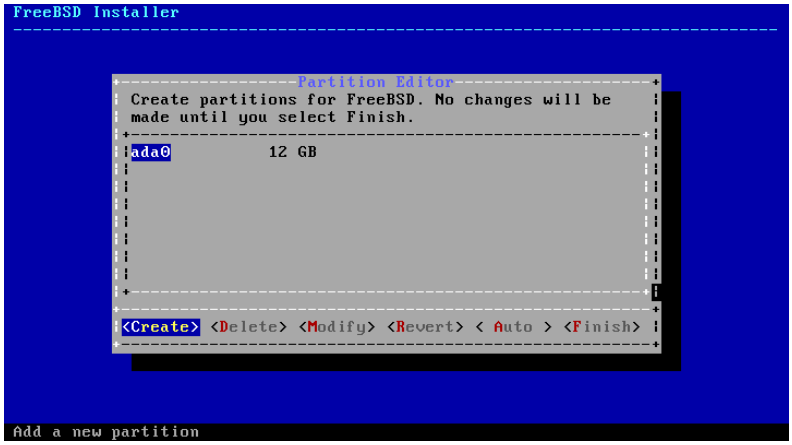
Figure 2-13. Review Created Partitions



2.7.2 Manual Partitioning

Manual partitioning goes straight to the partition editor.

Figure 2-14. Manually Create Partitions



Highlighting a drive (`ada0` in this example) and selecting [Create] displays a menu for choosing the type of *partitioning scheme*.

Figure 2-15. Manually Create Partitions

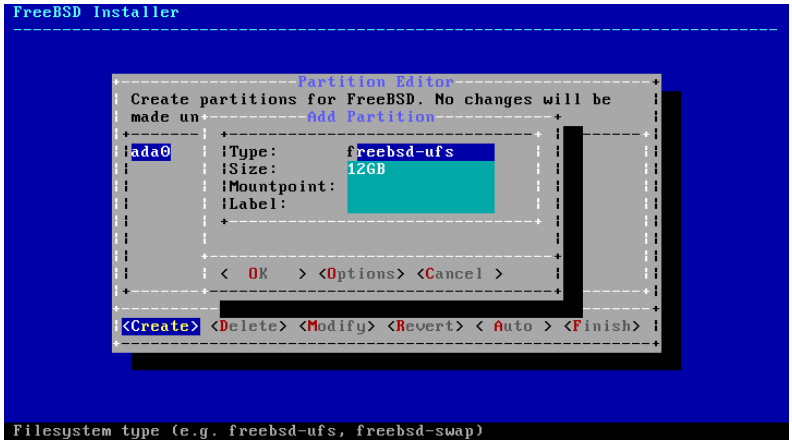
GPT partitioning is usually the most appropriate choice for PC-compatible computers. Older PC operating systems that are not compatible with GPT may require MBR partitioning instead. The other partitioning schemes are generally used for uncommon or older computer systems.

Table 2-1. Partitioning Schemes

Abbreviation	Description
APM	Apple Partition Map, used by PowerPC® Macintosh. (http://support.apple.com/kb/TA21692)
BSD	BSD Labels without an MBR, sometimes called "dangerously dedicated mode". See bsdlabel(8).
GPT	GUID Partition Table. (http://en.wikipedia.org/wiki/GUID_Partition_Table)
MBR	Master Boot Record. (http://en.wikipedia.org/wiki/Master_boot_record)
PC98	MBR variant, used by NEC PC-98 computers. (http://en.wikipedia.org/wiki/Pc9801)
VTOC8	Volume Table Of Contents, used by Sun SPARC64 and UltraSPARC computers.

After the partitioning scheme has been selected and created, selecting [Create] again will create new partitions.

Figure 2-16. Manually Create Partitions



A standard FreeBSD GPT installation uses at least three partitions:

Standard FreeBSD GPT Partitions

- `freebsd-boot` - FreeBSD boot code.
- `freebsd-ufs` - A FreeBSD UFS filesystem.
- `freebsd-swap` - FreeBSD swap space.

Another partition type worth noting is `freebsd-zfs`, used for partitions that will contain a FreeBSD ZFS filesystem. See Section 21.2. gpart(8) shows more of the available GPT partition types.

Multiple filesystem partitions can be used, and some people may prefer a traditional layout with separate partitions for the `/`, `/var`, `/tmp`, and `/usr` filesystems. See Example 2-3 for an example.

Size may be entered with common abbreviations: *K* for kilobytes, *M* for megabytes, or *G* for gigabytes.

> **Tip:** Proper sector alignment provides the best performance, and making partition sizes even multiples of 4K bytes helps to ensure alignment on drives with either 512-byte or 4K-byte sectors. Generally, using partition sizes that are even multiples of 1M or 1G is the easiest way to make sure every partition starts at an even multiple of 4K. One exception: at present, the *freebsd-boot* partition should be no larger than 512K due to boot code limitations.

A mountpoint is needed if this partition will contain a filesystem. If only a single UFS partition will be created, the mountpoint should be `/`.

A *label* is also requested. A label is a name by which this partition will be known. Drive names or numbers can change if the drive is connected to a different controller or port, but the partition label does not change. Referring to labels instead of drive names and partition numbers in files like `/etc/fstab` makes the system more tolerant of changing hardware. GPT labels appear in `/dev/gpt/` when a disk is attached. Other partitioning schemes have different label capabilities, and their labels appear in different directories in `/dev/`.

> **Tip:** Use a unique label on every filesystem to avoid conflicts from identical labels. A few letters from the computer's name, use, or location can be added to the label. "labroot" or "rootfs-lab" for the UFS root partition on the lab's computer, for example.

Example 2-3. Creating Traditional Split Filesystem Partitions

For a traditional partition layout where the `/`, `/var`, `/tmp`, and `/usr` directories are separate filesystems on their own partitions, create a GPT partitioning scheme, then create the partitions as shown. Partition sizes shown are typical for a 20G target disk. If more space is available on the target disk, larger swap or `/var` partitions may be useful. Labels shown here are prefixed with `ex` for "example", but readers should use other unique label values as described above.

By default, FreeBSD's `gptboot` expects the first UFS partition found to be the `/` partition.

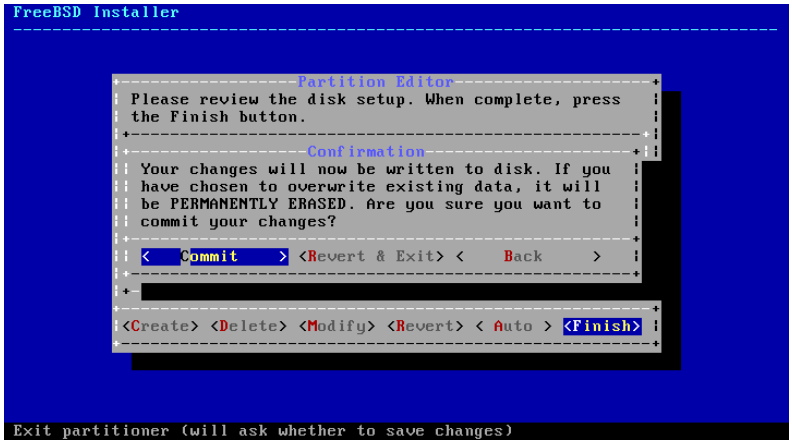
Partition Type	Size	Mountpoint	Label
`freebsd-boot`	`512K`		
`freebsd-ufs`	`2G`	`/`	`exrootfs`
`freebsd-swap`	`4G`		`exswap`
`freebsd-ufs`	`2G`	`/var`	`exvarfs`
`freebsd-ufs`	`1G`	`/tmp`	`extmpfs`
`freebsd-ufs`	accept the default (remainder of the disk)	`/usr`	`exusrfs`

After the custom partitions have been created, select [Finish] to continue with the installation.

2.8 Committing to the Installation

This is the last chance for aborting the installation to prevent changes to the hard drive.

Figure 2-17. Final Confirmation



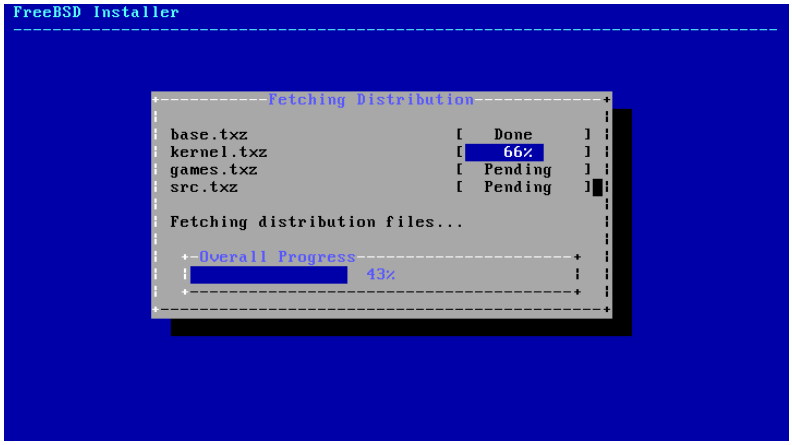
Select [Commit] and press **Enter** to proceed. If changes need to be made, select [Back] to return to the partition editor. [Revert & Exit] will exit the installer without making any changes to the hard drive.

Installation time will vary depending on the distributions chosen, installation media, and speed of the computer. There will be a series of messages displayed indicating progress.

Firstly, the installer will write the partitions to the disk, and perform a `newfs` to initialise the partitions.

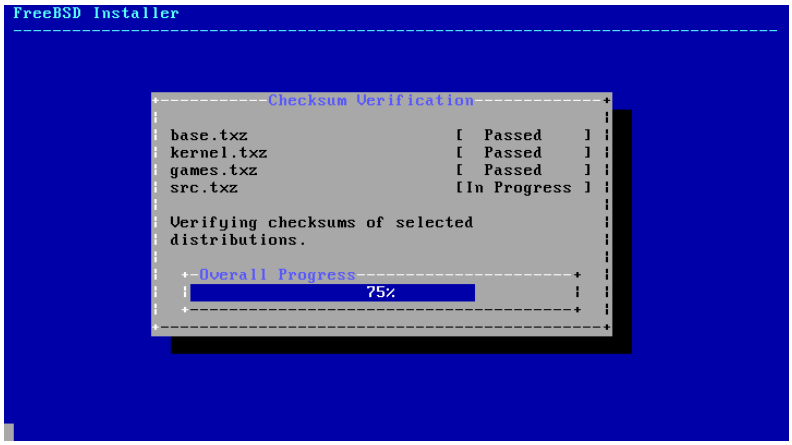
If doing a network install, **bsdinstall** will then proceed to download the required distribution files.

Figure 2-18. Fetching Distribution Files



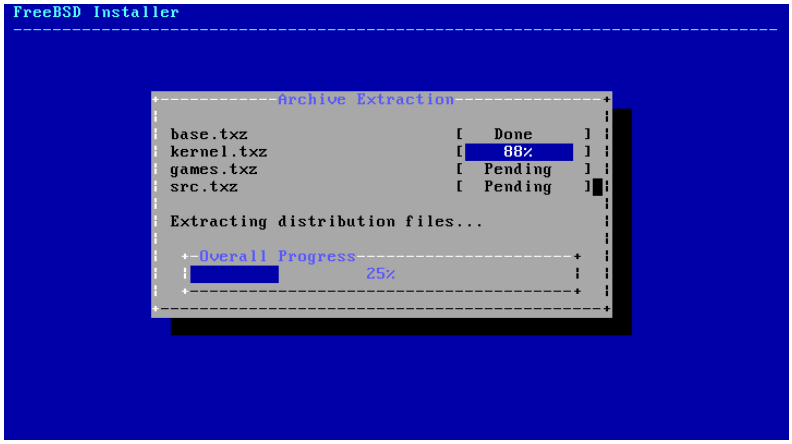
Next, the integrity of the distribution files is verified, to ensure they have not been corrupted during download or misread from the installation media.

Figure 2-19. Verifying Distribution Files



Finally, the verified distribution files are extracted to the disk.

Figure 2-20. Extracting Distribution Files



Once all requested distribution files have been extracted, **bsdinstall** will then drop straight into the post-installation configuration tasks (see Section 2.9).

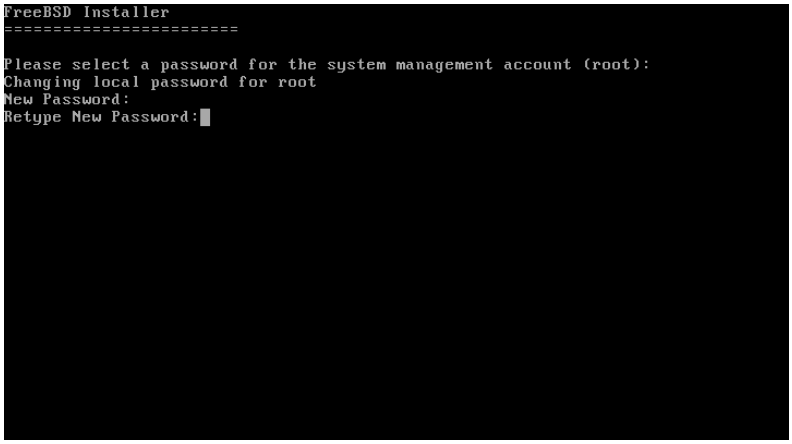
2.9 Post-Installation

Configuration of various options follows a successful installation of FreeBSD. An option can be configured by re-entering the configuration options from the final menu before booting into the newly installed FreeBSD system.

2.9.1 Setting the `root` Password

The `root` password must be set. Note that while entering the password, the characters being typed are not displayed on the screen. After the password has been entered, it must be entered again. This helps prevent typing errors.

Figure 2-21. Setting the `root` Password



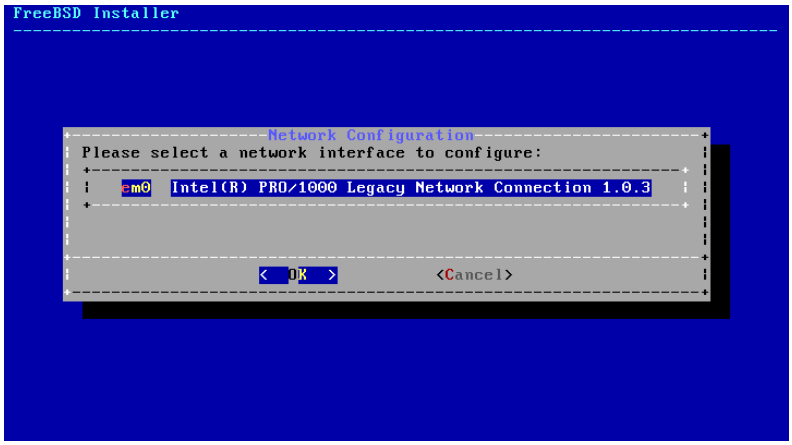
After the password has been successfully entered, the installation will continue.

2.9.2 Configuring Network Interfaces

Note: Network configuration will be skipped if it has already been done as part of a *bootonly* installation.

A list of all the network interfaces found on the computer is shown next. Select one to be configured.

Figure 2-22. Choose a Network Interface



2.9.2.1 Configuring a Wireless Network Interface

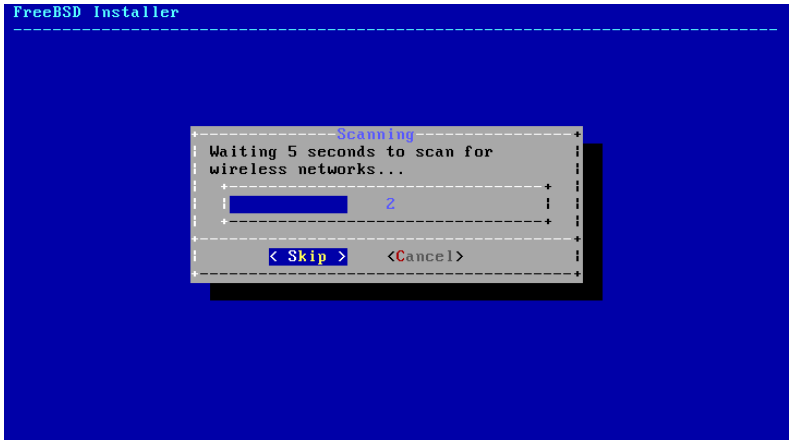
If a wireless network interface is chosen, wireless identification and security parameters must be entered to allow it to connect to the network.

Wireless networks are identified by a Service Set Identifier, or SSID. The SSID is a short, unique name given to each network.

Most wireless networks encrypt transmitted data to protect information from unauthorized viewing. WPA2 encryption is strongly recommended. Older encryption types, like WEP, offer very little security.

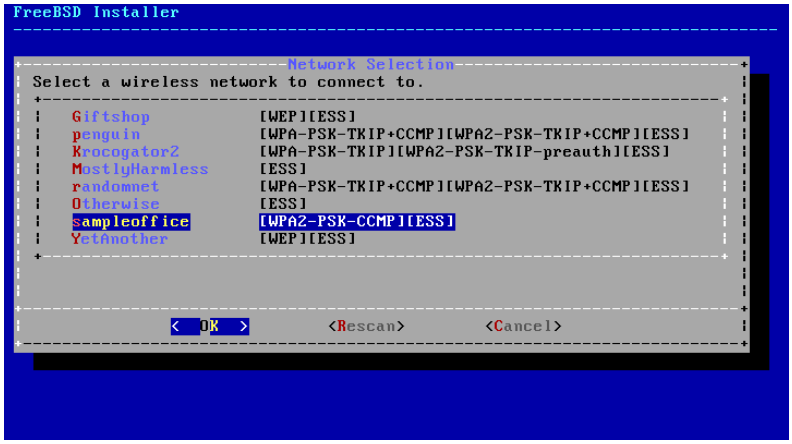
The first step in connecting to a wireless network is to scan for wireless access points.

Figure 2-23. Scanning for Wireless Access Points



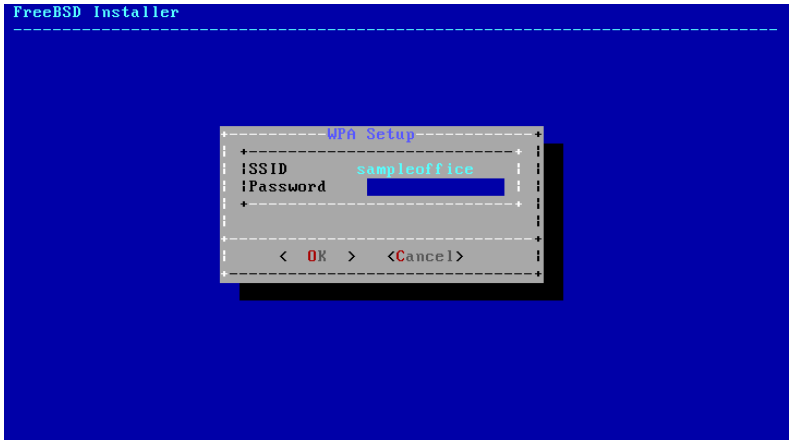
SSIDs found during the scan are listed, followed by a description of the encryption types available for that network. If the desired SSID does not appear in the list, select [Rescan] to scan again. If the desired network still does not appear, check for problems with antenna connections or try moving the computer closer to the access point. Rescan after each change is made.

Figure 2-24. Choosing a Wireless Network



The encryption information for connecting to the selected wireless network is entered after selecting the network. With WPA2, only a password (also known as the Pre-Shared Key, or PSK) is needed. Characters typed into the input box are shown as asterisks for security.

Figure 2-25. WPA2 Setup

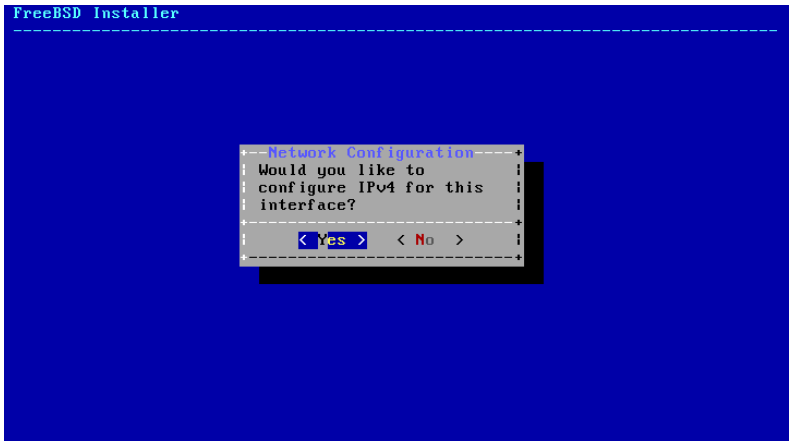


Network configuration continues after selection of the wireless network and entry of the connection information.

2.9.2.2 Configuring IPv4 Networking

Choose whether IPv4 networking is to be used. This is the most common type of network connection.

Figure 2-26. Choose IPv4 Networking



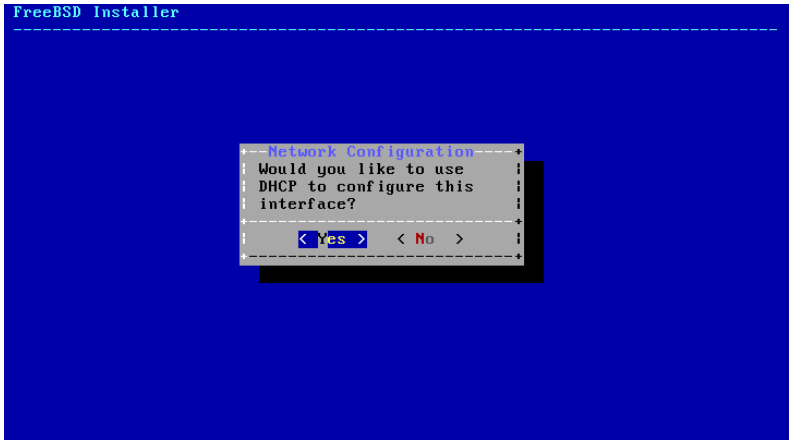
There are two methods of IPv4 configuration. *DHCP* will automatically configure the network interface correctly, and is the preferred method. *Static* configuration requires manual entry of network information.

> **Note:** Do not enter random network information, as it will not work. Obtain the information shown in Section 2.3.3 from the network administrator or service provider.

2.9.2.2.1 IPv4 DHCP Network Configuration

If a DHCP server is available, select [Yes] to automatically configure the network interface.

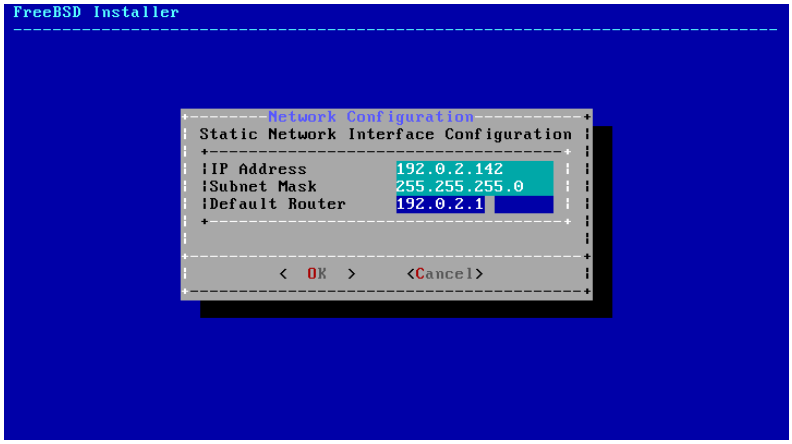
Figure 2-27. Choose IPv4 DHCP Configuration



2.9.2.2.2 IPv4 Static Network Configuration

Static configuration of the network interface requires entry of some IPv4 information.

Figure 2-28. IPv4 Static Configuration

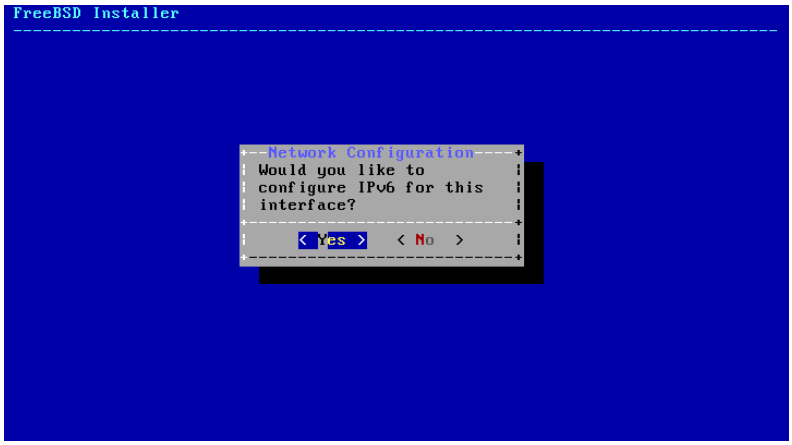


- `IP Address` - The manually-assigned IPv4 address to be assigned to this computer. This address must be unique and not already in use by another piece of equipment on the local network.
- `Subnet Mask` - The subnet mask used for the local network. Typically, this is `255.255.255.0`.
- `Default Router` - The IP address of the default router on this network. Usually this is the address of the router or other network equipment that connects the local network to the Internet. Also known as the *default gateway*.

2.9.2.3 Configuring IPv6 Networking

IPv6 is a newer method of network configuration. If IPv6 is available and desired, choose [Yes] to select it.

Figure 2-29. Choose IPv6 Networking

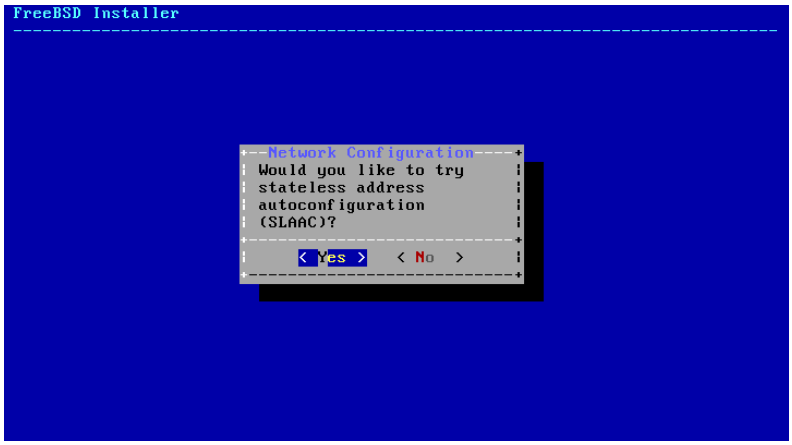


IPv6 also has two methods of configuration. *SLAAC* , or *StateLess Address AutoConfiguration*, will automatically configure the network interface correctly. *Static* configuration requires manual entry of network information.

2.9.2.3.1 IPv6 Stateless Address Autoconfiguration

SLAAC allows an IPv6 network component to request autoconfiguration information from a local router. See RFC4862 (http://tools.ietf.org/html/rfc4862) for more information.

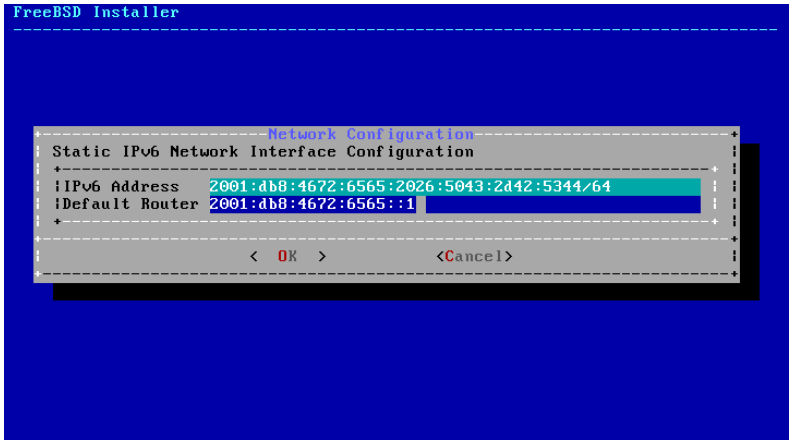
Figure 2-30. Choose IPv6 SLAAC Configuration



2.9.2.3.2 IPv6 Static Network Configuration

Static configuration of the network interface requires entry of the IPv6 configuration information.

Figure 2-31. IPv6 Static Configuration

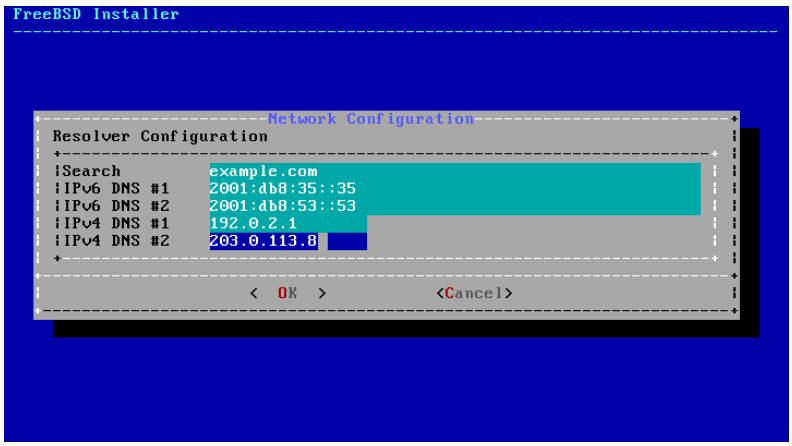


- `IPv6 Address` - The manually-assigned IP address to be assigned to this computer. This address must be unique and not already in use by another piece of equipment on the local network.
- `Default Router` - The IPv6 address of the default router on this network. Usually this is the address of the router or other network equipment that connects the local network to the Internet. Also known as the *default gateway*.

2.9.2.4 Configuring DNS

The *Domain Name System* (or *DNS*) Resolver converts hostnames to and from network addresses. If DHCP or SLAAC was used to autoconfigure the network interface, the Resolver Configuration values may already be present. Otherwise, enter the local network's domain name in the Search field. DNS #1 and DNS #2 are the IP addresses for the local DNS servers. At least one DNS server is required.

Figure 2-32. DNS Configuration



2.9.3 Setting the Time Zone

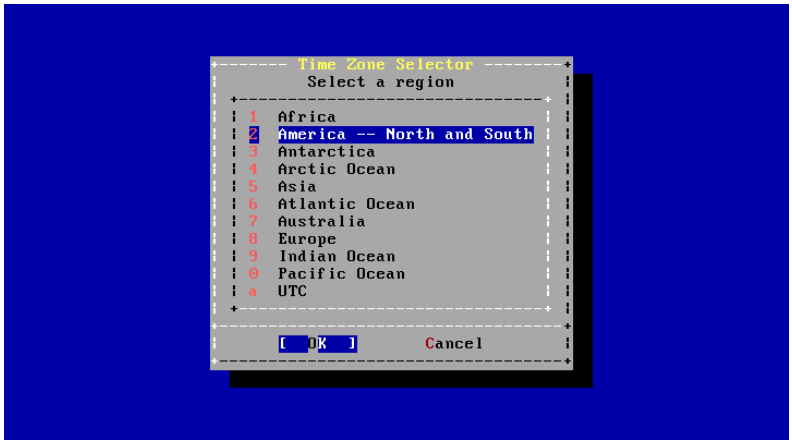
Setting the time zone for your machine will allow it to automatically correct for any regional time changes and perform other time zone related functions properly.

The example shown is for a machine located in the Eastern time zone of the United States. Your selections will vary according to your geographical location.

Figure 2-33. Select Local or UTC Clock

Select [Yes] or [No] according to how the machine's clock is configured and press **Enter**. If you do not know whether the system uses UTC or local time, select [No] to choose the more commonly-used local time.

Figure 2-34. Select a Region

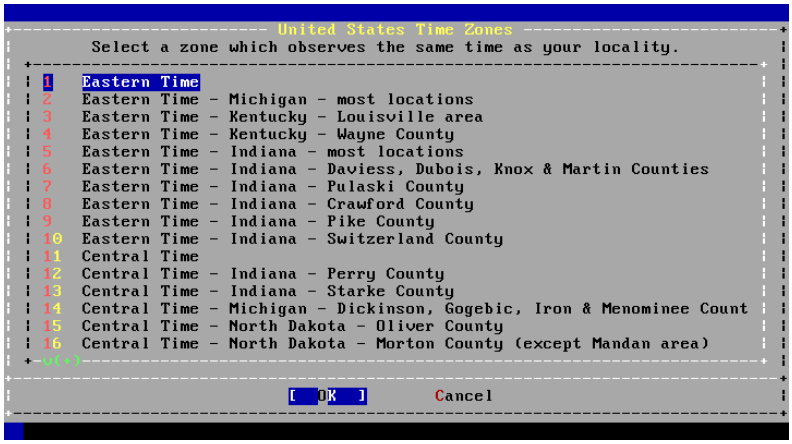


The appropriate region is selected using the arrow keys and then pressing **Enter**.

Figure 2-35. Select a Country

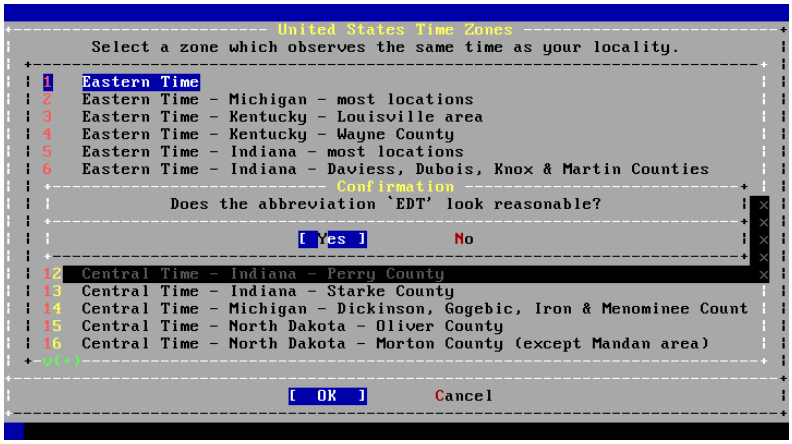
Select the appropriate country using the arrow keys and press **Enter**.

Figure 2-36. Select a Time Zone



The appropriate time zone is selected using the arrow keys and pressing **Enter**.

Figure 2-37. Confirm Time Zone

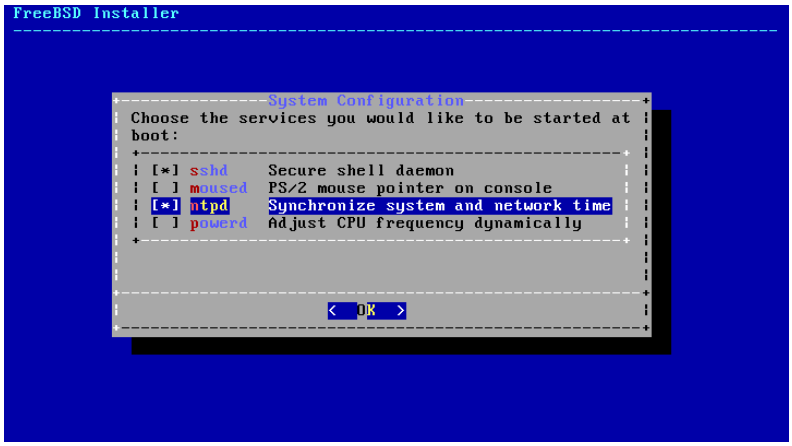


Confirm the abbreviation for the time zone is correct. If it looks okay, press **Enter** to continue with the post-installation configuration.

2.9.4 Selecting Services to Enable

Additional system services which will be started at boot can be enabled. All of these services are optional.

Figure 2-38. Selecting Additional Services to Enable



Additional Services

- `sshd` - Secure Shell (SSH) daemon for secure remote access.
- `moused` - Provides mouse usage within the system console.
- `ntpd` - Network Time Protocol (NTP) daemon for automatic clock synchronization.
- `powerd` - System power control utility for power control and energy saving.

2.9.5 Enabling Crash Dumps

bsdinstall will prompt if crash dumps should be enabled on the target system. Enabling crash dumps can be very useful in debugging issues with the system, so users are encouraged to enable crash dumps whenever possible. Select [Yes] to enable crash dumps, or [No] to proceed without crash dumps enabled.

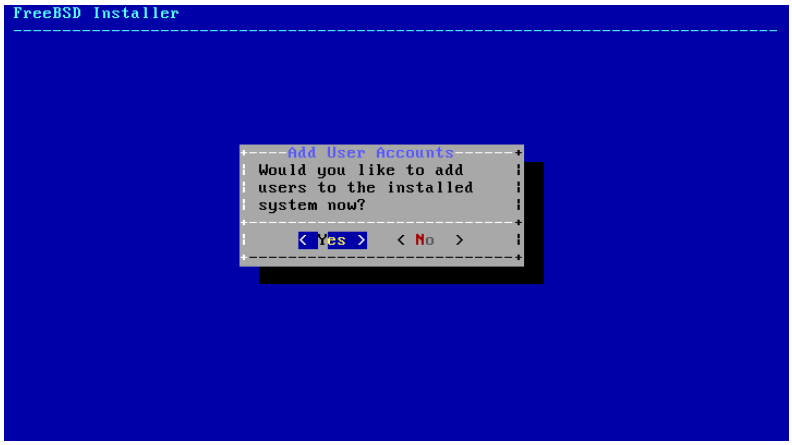
Figure 2-39. Enabling Crash Dumps

2.9.6 Add Users

Adding at least one user during the installation allows the system to be used without being logged in as `root`. When logged in as `root`, there are essentially no limits or protection on what can be done. Logging in as a normal user is safer and more secure.

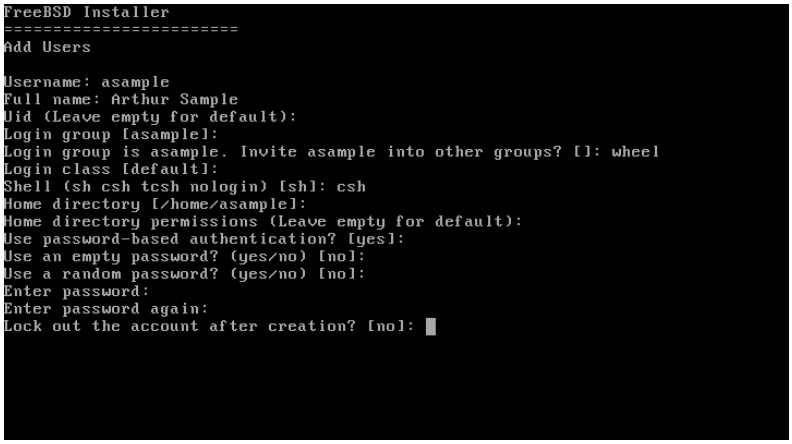
Select [Yes] to add new users.

Figure 2-40. Add User Accounts



Enter the information for the user to be added.

Figure 2-41. Enter User Information

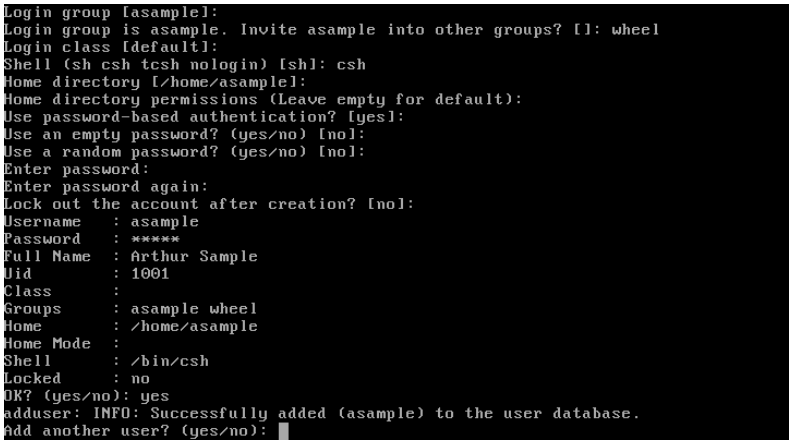


User Information

- `Username` - The name the user will enter to log in. Typically the first letter of their first name combined with their last name.
- `Full name` - The user's full name.
- `Uid` - User ID. Typically, this is left blank so the system will assign a value.
- `Login group` - The user's group. Typically left blank to accept the default.
- `Invite` *`user`* `into other groups?` - Additional groups to which the user will be added as a member.
- `Login class` - Typically left blank for the default.
- `Shell` - The interactive shell for this user. In the example, csh(1) has been chosen.
- `Home directory` - The user's home directory. The default is usually correct.
- `Home directory permissions` - Permissions on the user's home directory. The default is usually correct.
- `Use password-based authentication?` - Typically "yes".
- `Use an empty password?` - Typically "no".
- `Use a random password?` - Typically "no".
- `Enter password` - The actual password for this user. Characters typed will not show on the screen.
- `Enter password again` - The password must be typed again for verification.
- `Lock out the account after creation?` - Typically "no".

After entering everything, a summary is shown, and the system asks if it is correct. If a mistake was made during entry, enter `no` and try again. If everything is correct, enter `yes` to create the new user.

Figure 2-42. Exit User and Group Management



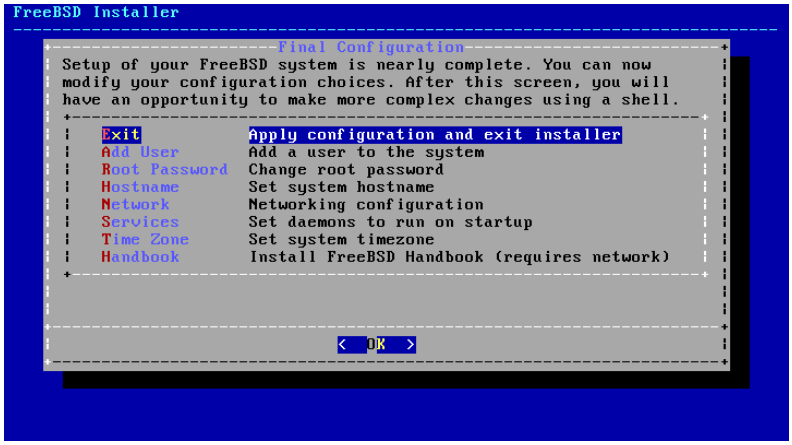
If there are more users to add, answer the "Add another user?" question with `yes`. Enter `no` to finish adding users and continue the installation.

For more information on adding users and user management, see Chapter 14.

2.9.7 Final Configuration

After everything has been installed and configured, a final chance is provided to modify settings.

Figure 2-43. Final Configuration



Use this menu to make any changes or do any additional configuration before completing the installation.

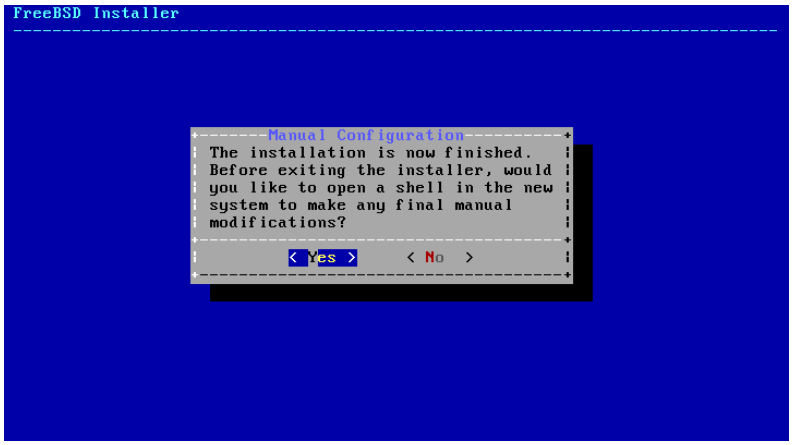
Final Configuration Options

- `Add User` - Described in Section 2.9.6.
- `Root Password` - Described in Section 2.9.1.
- `Hostname` - Described in Section 2.5.2.

- `Network` - Described in Section 2.9.2.
- `Services` - Described in Section 2.9.4.
- `Time Zone` - Described in Section 2.9.3.
- `Handbook` - Download and install the FreeBSD Handbook (which is what you are reading now).

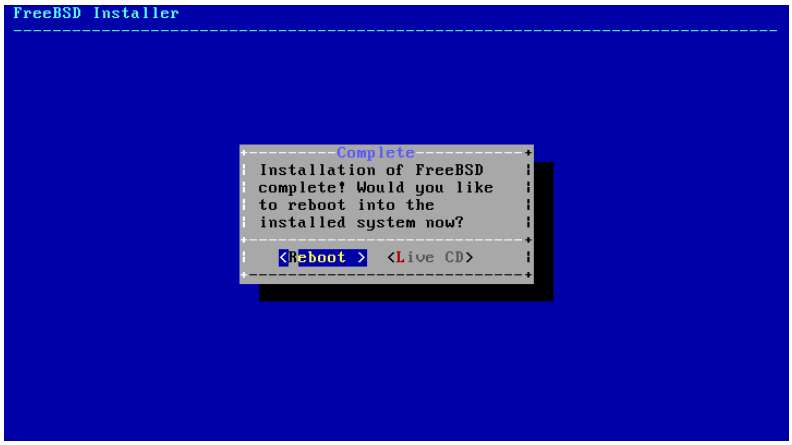
After any final configuration is complete, select Exit to leave the installation.

Figure 2-44. Manual Configuration



bsdinstall will prompt if there are any additional configuration that needs to be done before rebooting into the new system. Select [Yes] to exit to a shell within the new system, or [No] to proceed to the last step of the installation.

Figure 2-45. Complete the Installation



If further configuration or special setup is needed, selecting [Live CD] will boot the install media into Live CD mode.

When the installation is complete, select [Reboot] to reboot the computer and start the new FreeBSD system. Do not forget to remove the FreeBSD install CD, DVD, or USB memory stick, or the computer may boot from it again.

2.9.8 FreeBSD Booting and Shutdown

2.9.8.1 FreeBSD/i386 Booting

As FreeBSD boots, many informational messages are displayed. Most will scroll off the screen; this is normal. After the system finishes booting, a login prompt is displayed. Messages that scrolled off the screen can be reviewed by pressing **Scroll-Lock** to turn on the *scroll-back buffer*. The **PgUp**, **PgDn**, and arrow keys can be used to scroll back through the messages. Pressing **Scroll-Lock** again unlocks the display and returns to the normal screen.

At the `login:` prompt, enter the username added during the installation, `asample` in the example. Avoid logging in as `root` except when necessary.

The scroll-back buffer examined above is limited in size, so not all of the messages may have been visible. After logging in, most of them can be seen from the command line by typing `dmesg | less` at the prompt. Press **q** to return to the command line after viewing.

Typical boot messages (version information omitted):

```
Copyright (c) 1992-2011 The FreeBSD Project.
Copyright (c) 1979, 1980, 1983, 1986, 1988, 1989, 1991, 1992, 1993, 1994
        The Regents of the University of California. All rights reserved.
FreeBSD is a registered trademark of The FreeBSD Foundation.

    root@farrell.cse.buffalo.edu:/usr/obj/usr/src/sys/GENERIC amd64
CPU: Intel(R) Core(TM)2 Duo CPU     E8400  @ 3.00GHz (3007.77-MHz K8-class CPU)
  Origin = "GenuineIntel"  Id = 0x10676  Family = 6  Model = 17  Stepping = 6
  Features=0x783fbff<FPU,VME,DE,PSE,TSC,MSR,PAE,MCE,CX8,APIC,SEP,MTRR,PGE,MCA,CMOV,PAT,PSE36,MMX,
  Features2=0x209<SSE3,MON,SSSE3>
  AMD Features=0x20100800<SYSCALL,NX,LM>
  AMD Features2=0x1<LAHF>
real memory  = 536805376 (511 MB)
avail memory = 491819008 (469 MB)
Event timer "LAPIC" quality 400
ACPI APIC Table: <VBOX   VBOXAPIC>
ioapic0: Changing APIC ID to 1
ioapic0 <Version 1.1> irqs 0-23 on motherboard
kbd1 at kbdmux0
acpi0: <VBOX VBOXXSDT> on motherboard
acpi0: Power Button (fixed)
acpi0: Sleep Button (fixed)
Timecounter "ACPI-fast" frequency 3579545 Hz quality 900
acpi_timer0: <32-bit timer at 3.579545MHz> port 0x4008-0x400b on acpi0
cpu0: <ACPI CPU> on acpi0
pcib0: <ACPI Host-PCI bridge> port 0xcf8-0xcff on acpi0
pci0: <ACPI PCI bus> on pcib0
isab0: <PCI-ISA bridge> at device 1.0 on pci0
isa0: <ISA bus> on isab0
atapci0: <Intel PIIX4 UDMA33 controller> port 0x1f0-0x1f7,0x3f6,0x170-0x177,0x376,0xd000-0xd00f a
ata0: <ATA channel 0> on atapci0
ata1: <ATA channel 1> on atapci0
vgapci0: <VGA-compatible display> mem 0xe0000000-0xe0ffffff irq 18 at device 2.0 on pci0
em0: <Intel(R) PRO/1000 Legacy Network Connection 1.0.3> port 0xd010-0xd017 mem 0xf0000000-0xf001
em0: Ethernet address: 08:00:27:9f:e0:92
pci0: <base peripheral> at device 4.0 (no driver attached)
```

```
pcm0: <Intel ICH (82801AA)> port 0xd100-0xd1ff,0xd200-0xd23f irq 21 at device 5.0 on pci0
pcm0: <SigmaTel STAC9700/83/84 AC97 Codec>
ohci0: <OHCI (generic) USB controller> mem 0xf0804000-0xf0804fff irq 22 at device 6.0 on pci0
usbus0: <OHCI (generic) USB controller> on ohci0
pci0: <bridge> at device 7.0 (no driver attached)
acpi_acad0: <AC Adapter> on acpi0
atkbdc0: <Keyboard controller (i8042)> port 0x60,0x64 irq 1 on acpi0
atkbd0: <AT Keyboard> irq 1 on atkbdc0
kbd0 at atkbd0
atkbd0: [GIANT-LOCKED]
psm0: <PS/2 Mouse> irq 12 on atkbdc0
psm0: [GIANT-LOCKED]
psm0: model IntelliMouse Explorer, device ID 4
attimer0: <AT timer> port 0x40-0x43,0x50-0x53 on acpi0
Timecounter "i8254" frequency 1193182 Hz quality 0
Event timer "i8254" frequency 1193182 Hz quality 100
sc0: <System console> at flags 0x100 on isa0
sc0: VGA <16 virtual consoles, flags=0x300>
vga0: <Generic ISA VGA> at port 0x3c0-0x3df iomem 0xa0000-0xbffff on isa0
atrtc0: <AT realtime clock> at port 0x70 irq 8 on isa0
Event timer "RTC" frequency 32768 Hz quality 0
ppc0: cannot reserve I/O port range
Timecounters tick every 10.000 msec
pcm0: measured ac97 link rate at 485193 Hz
em0: link state changed to UP
usbus0: 12Mbps Full Speed USB v1.0
ugen0.1: <Apple> at usbus0
uhub0: <Apple OHCI root HUB, class 9/0, rev 1.00/1.00, addr 1> on usbus0
cd0 at ata1 bus 0 scbus1 target 0 lun 0
cd0: <VBOX CD-ROM 1.0> Removable CD-ROM SCSI-0 device
cd0: 33.300MB/s transfers (UDMA2, ATAPI 12bytes, PIO 65534bytes)
cd0: Attempt to query device size failed: NOT READY, Medium not present
ada0 at ata0 bus 0 scbus0 target 0 lun 0
ada0: <VBOX HARDDISK 1.0> ATA-6 device
ada0: 33.300MB/s transfers (UDMA2, PIO 65536bytes)
ada0: 12546MB (25694208 512 byte sectors: 16H 63S/T 16383C)
ada0: Previously was known as ad0
Timecounter "TSC" frequency 3007772192 Hz quality 800
Root mount waiting for: usbus0
uhub0: 8 ports with 8 removable, self powered
Trying to mount root from ufs:/dev/ada0p2 [rw]...
Setting hostuuid: 1848d7bf-e6a4-4ed4-b782-bd3f1685d551.
Setting hostid: 0xa03479b2.
Entropy harvesting: interrupts ethernet point_to_point kickstart.
Starting file system checks:
/dev/ada0p2: FILE SYSTEM CLEAN; SKIPPING CHECKS
/dev/ada0p2: clean, 2620402 free (714 frags, 327461 blocks, 0.0% fragmentation)
Mounting local file systems:.
vboxguest0 port 0xd020-0xd03f mem 0xf0400000-0xf07fffff,0xf0800000-0xf0803fff irq 20 at device 4.0
vboxguest: loaded successfully
Setting hostname: machine3.example.com.
Starting Network: lo0 em0.
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> metric 0 mtu 16384
```

```
        options=3<RXCSUM,TXCSUM>
        inet6 ::1 prefixlen 128
        inet6 fe80::1%lo0 prefixlen 64 scopeid 0x3
        inet 127.0.0.1 netmask 0xff000000
        nd6 options=21<PERFORMNUD,AUTO_LINKLOCAL>
em0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
        options=9b<RXCSUM,TXCSUM,VLAN_MTU,VLAN_HWTAGGING,VLAN_HWCSUM>
        ether 08:00:27:9f:e0:92
        nd6 options=29<PERFORMNUD,IFDISABLED,AUTO_LINKLOCAL>
        media: Ethernet autoselect (1000baseT <full-duplex>)
        status: active
Starting devd.
Starting Network: usbus0.
DHCPREQUEST on em0 to 255.255.255.255 port 67
DHCPACK from 10.0.2.2
bound to 192.168.1.142 -- renewal in 43200 seconds.
add net ::ffff:0.0.0.0: gateway ::1
add net ::0.0.0.0: gateway ::1
add net fe80::: gateway ::1
add net ff02::: gateway ::1
ELF ldconfig path: /lib /usr/lib /usr/lib/compat /usr/local/lib
32-bit compatibility ldconfig path: /usr/lib32
Creating and/or trimming log files.
Starting syslogd.
No core dumps found.
Clearing /tmp (X related).
Updating motd:.
Configuring syscons: blanktime.
Generating public/private rsa1 key pair.
Your identification has been saved in /etc/ssh/ssh_host_key.
Your public key has been saved in /etc/ssh/ssh_host_key.pub.
The key fingerprint is:
10:a0:f5:af:93:ae:a3:1a:b2:bb:3c:35:d9:5a:b3:f3 root@machine3.example.com
The key's randomart image is:
+--[RSA1 1024]----+
|    o..          |
|   o . .         |
|  .   o          |
|       o         |
|    o   S        |
|   + + o         |
|o . + *          |
|o+ ..+ .         |
|==o..o+E         |
+-----------------+
Generating public/private dsa key pair.
Your identification has been saved in /etc/ssh/ssh_host_dsa_key.
Your public key has been saved in /etc/ssh/ssh_host_dsa_key.pub.
The key fingerprint is:
7e:1c:ce:dc:8a:3a:18:13:5b:34:b5:cf:d9:d1:47:b2 root@machine3.example.com
The key's randomart image is:
+--[ DSA 1024]----+
|       ..     . .|
```

```
|      o  .   . + |
|     . ..   . E .|
|    . .  o o . . |
|     +  S = .    |
|    +  . = o     |
|     +  . * .    |
|    . .  o .     |
|      .o. .      |
+-----------------+
Starting sshd.
Starting cron.
Starting background file system checks in 60 seconds.

Thu Oct  6 19:15:31 MDT 2011

FreeBSD/amd64 (machine3.example.com) (ttyv0)

login:
```

Generating the RSA and DSA keys may take some time on slower machines. This happens only on the initial boot-up of a new installation, and only if **sshd** is set to start automatically. Subsequent boots will be faster.

FreeBSD does not install graphical environments by default, but many are available. See Chapter 6 for more information.

2.9.9 FreeBSD Shutdown

Proper shutdown of a FreeBSD computer helps protect data and even hardware from damage. Do not just turn off the power. If the user is a member of the `wheel` group, become the superuser by typing `su` at the command line and entering the `root` password. Otherwise, log in as `root` and use `shutdown -p now`. The system will close down cleanly and turn itself off.

The **Ctrl**+**Alt**+**Del** key combination can be used to reboot the system, but is not recommended during normal operation.

2.10 Troubleshooting

The following section covers basic installation troubleshooting, such as common problems people have reported.

2.10.1 What to Do If Something Goes Wrong

Due to various limitations of the PC architecture, it is impossible for probing to be 100% reliable, however, there are a few things you can do if it fails.

Check the Hardware Notes (http://www.FreeBSD.org/releases/index.html) document for your version of FreeBSD to make sure your hardware is supported.

If your hardware is supported and you still experience lock-ups or other problems, you will need to build a custom kernel. This will allow you to add in support for devices which are not present in the `GENERIC` kernel. The

kernel on the boot disks is configured assuming that most hardware devices are in their factory default configuration in terms of IRQs, IO addresses, and DMA channels. If your hardware has been reconfigured, you will most likely need to edit the kernel configuration and recompile to tell FreeBSD where to find things.

It is also possible that a probe for a device not present will cause a later probe for another device that is present to fail. In that case, the probes for the conflicting driver(s) should be disabled.

> **Note:** Some installation problems can be avoided or alleviated by updating the firmware on various hardware components, most notably the motherboard. Motherboard firmware is usually referred to as the BIOS. Most motherboard and computer manufacturers have a website for upgrades and upgrade information.
>
> Manufacturers generally advise against upgrading the motherboard BIOS unless there is a good reason for doing so, like a critical update. The upgrade process *can* go wrong, leaving the BIOS incomplete and the computer inoperative.

2.10.2 Troubleshooting Questions and Answers

1. My system hangs while probing hardware during boot, or it behaves strangely during install.

FreeBSD makes extensive use of the system ACPI service on the i386, amd64, and ia64 platforms to aid in system configuration if it is detected during boot. Unfortunately, some bugs still exist in both the ACPI driver and within system motherboards and BIOS firmware. ACPI can be disabled by setting the `hint.acpi.0.disabled` hint in the third stage boot loader:

```
set hint.acpi.0.disabled="1"
```

This is reset each time the system is booted, so it is necessary to add `hint.acpi.0.disabled="1"` to the file `/boot/loader.conf`. More information about the boot loader can be found in Section 13.1.

2.11 Using the Live CD

A live CD of FreeBSD is available on the same CD as the main installation program. This is useful for those who are still wondering whether FreeBSD is the right operating system for them and want to test some of the features before installing.

> **Note:** The following points should be noted while using the live CD:
>
> - To gain access to the system, authentication is required. The username is `root`, and the password is blank.
> - As the system runs directly from the CD, performance will be significantly slower than that of a system installed on a hard disk.
> - The live CD provides a command prompt and not a graphical interface.

Chapter 3 Installing FreeBSD 8.*x*

Restructured, reorganized, and parts rewritten by Jim Mock. The sysinstall walkthrough, screenshots, and general copy by Randy Pratt.

3.1 Synopsis

FreeBSD provides a text-based, easy to use installation program. FreeBSD 9.0-RELEASE and later use the installation program known as bsdinstall(8) while FreeBSD 8.*X* uses sysinstall(8). This chapter describes how to use sysinstall(8). The use of bsdinstall(8) is covered in Chapter 2.

After reading this chapter, you will know:

- How to create the FreeBSD installation media.
- How FreeBSD refers to and subdivides hard disks.
- How to start sysinstall(8).
- The questions sysinstall(8) asks, what they mean, and how to answer them.

Before reading this chapter, you should:

- Read the supported hardware list that shipped with the version of FreeBSD to install, and verify that the system's hardware is supported.

> **Note:** In general, these installation instructions are written for the i386 and FreeBSD/amd64 architectures. Where applicable, instructions specific to other platforms will be listed. There may be minor differences between the installer and what is shown here. This chapter should be used as a general guide rather than a literal installation manual.

3.2 Hardware Requirements

3.2.1 Minimal Configuration

The minimal configuration to install FreeBSD varies with the FreeBSD version and the hardware architecture.

A summary of this information is given in the following sections. Depending on the method chosen to install FreeBSD, a floppy drive, CDROM drive, or network adapter may be needed. Instructions on how to prepare the installation media can be found in Section 3.3.7.

3.2.1.1 FreeBSD/i386 and FreeBSD/pc98

Both FreeBSD/i386 and FreeBSD/pc98 require a 486 or better processor, at least 24 MB of RAM, and at least 150 MB of free hard drive space for the most minimal installation.

Note: In the case of older hardware, installing more RAM and more hard drive space is often more important than a faster processor.

3.2.1.2 FreeBSD/amd64

There are two classes of processors capable of running FreeBSD/amd64. The first are AMD64 processors, including the AMD Athlon64, AMD Athlon64-FX, and AMD Opteron or better processors.

The second class of processors includes those using the Intel EM64T architecture. Examples of these processors include the Intel Core 2 Duo, Quad, Extreme processor families, and the Intel Xeon 3000, 5000, and 7000 sequences of processors.

If the machine is based on an nVidia nForce3 Pro-150, the BIOS setup *must* be used to disable the IO APIC. If this option does not exist, disable ACPI instead as there are bugs in the Pro-150 chipset.

3.2.1.3 FreeBSD/sparc64

To install FreeBSD/sparc64, use a supported platform (see Section 3.2.2).

A dedicated disk is needed for FreeBSD/sparc64 as it is not possible to share a disk with another operating system at this time.

3.2.2 Supported Hardware

A list of supported hardware is provided with each FreeBSD release in the FreeBSD Hardware Notes. This document can usually be found in a file named `HARDWARE.TXT`, in the top-level directory of a CDROM or FTP distribution, or in sysinstall(8)'s documentation menu. It lists, for a given architecture, which hardware devices are known to be supported by each release of FreeBSD. Copies of the supported hardware list for various releases and architectures can also be found on the Release Information (http://www.FreeBSD.org/releases/index.html) page of the FreeBSD website.

3.3 Pre-installation Tasks

3.3.1 Inventory the Computer

Before installing FreeBSD it is recommended to inventory the components in the computer. The FreeBSD installation routines will show components such as hard disks, network cards, and CDROM drives with their model number and manufacturer. FreeBSD will also attempt to determine the correct configuration for these devices, including information about IRQ and I/O port usage. Due to the vagaries of computer hardware, this process is not always completely successful, and FreeBSD may need some manual configuration.

If another operating system is already installed, use the facilities provided by that operating systems to view the hardware configuration. If the settings of an expansion card are not obvious, check if they are printed on the card

itself. Popular IRQ numbers are 3, 5, and 7, and I/O port addresses are normally written as hexadecimal numbers, such as `0x330`.

It is recommended to print or write down this information before installing FreeBSD. It may help to use a table, as seen in this example:

Table 3-1. Sample Device Inventory

Device Name	IRQ	I/O port(s)	Notes
First hard disk	N/A	N/A	40 GB, made by Seagate, first IDE master
CDROM	N/A	N/A	First IDE slave
Second hard disk	N/A	N/A	20 GB, made by IBM, second IDE master
First IDE controller	14	0x1f0	
Network card	N/A	N/A	Intel 10/100
Modem	N/A	N/A	3Com® 56K faxmodem, on COM1
…			

Once the inventory of the components in the computer is complete, check if it matches the hardware requirements of the FreeBSD release to install.

3.3.2 Make a Backup

If the computer contains valuable data, ensure it is backed up, and that the backup has been tested before installing FreeBSD. The FreeBSD installer will prompt before writing any data to disk, but once that process has started, it cannot be undone.

3.3.3 Decide Where to Install FreeBSD

If FreeBSD is to be installed on the entire hard disk, skip this section.

However, if FreeBSD will co-exist with other operating systems, a rough understanding of how data is laid out on the disk is useful.

3.3.3.1 Disk Layouts for FreeBSD/i386

A PC disk can be divided into discrete chunks known as *partitions*. Since FreeBSD also has partitions, naming can quickly become confusing. Therefore, these disk chunks are referred to as slices in FreeBSD. For example, the FreeBSD version of fdisk(8) refers to slices instead of partitions. By design, the PC only supports four partitions per disk. These partitions are called *primary partitions*. To work around this limitation and allow more than four partitions, a new partition type was created, the *extended partition*. A disk may contain only one extended partition. Special partitions, called *logical partitions*, can be created inside this extended partition.

Each partition has a *partition ID*, which is a number used to identify the type of data on the partition. FreeBSD partitions have the partition ID of `165`.

In general, each operating system will identify partitions in a particular way. For example, Windows, assigns each primary and logical partition a *drive letter*, starting with `C:`.

FreeBSD must be installed into a primary partition. If there are multiple disks, a FreeBSD partition can be created on all, or some, of them. When FreeBSD is installed, at least one partition must be available. This might be a blank partition or it might be an existing partition whose data can be overwritten.

If all the partitions on all the disks are in use, free one of them for FreeBSD using the tools provided by an existing operating system, such as Windows `fdisk`.

If there is a spare partition, use that. If it is too small, shrink one or more existing partitions to create more available space.

A minimal installation of FreeBSD takes as little as 100 MB of disk space. However, that is a *very* minimal install, leaving almost no space for files. A more realistic minimum is 250 MB without a graphical environment, and 350 MB or more for a graphical user interface. If other third-party software will be installed, even more space is needed.

You can use a tool such as **GParted** to resize your partitions and make space for FreeBSD. **GParted** is known to work on NTFS and is available on a number of Live CD Linux distributions, such as SystemRescueCD (http://www.sysresccd.org/).

> **Warning:** Incorrect use of a shrinking tool can delete the data on the disk. Always have a recent, working backup before using this type of tool.

Example 3-1. Using an Existing Partition Unchanged

Consider a computer with a single 4 GB disk that already has a version of Windows installed, where the disk has been split into two drive letters, `C:` and `D:`, each of which is 2 GB in size. There is 1 GB of data on `C:`, and 0.5 GB of data on `D:`.

This disk has two partitions, one per drive letter. Copy all existing data from `D:` to `C:`, which will free up the second partition, ready for FreeBSD.

Example 3-2. Shrinking an Existing Partition

Consider a computer with a single 4 GB disk that already has a version of Windows installed. When Windows was installed, it created one large partition, a `C:` drive that is 4 GB in size. Currently, 1.5 GB of space is used, and FreeBSD should have 2 GB of space.

In order to install FreeBSD, either:

1. Backup the Windows data and then reinstall Windows, asking for a 2 GB partition at install time.
2. Use one of the tools described above to shrink your Windows partition.

3.3.4 Collect the Network Configuration Details

Before installing from an FTP site or an NFS server, make note of the network configuration. The installer will prompt for this information so that it can connect to the network to complete the installation.

3.3.4.1 Connecting to an Ethernet Network or Cable/DSL Modem

If using an Ethernet network or an Internet connection using an Ethernet adapter via cable or DSL, the following information is needed:

1. IP address
2. IP address of the default gateway
3. Hostname
4. DNS server IP addresses
5. Subnet Mask

If this information is unknown, ask the system administrator or service provider. Make note if this information is assigned automatically using *DHCP*.

3.3.4.2 Connecting Using a Modem

If using a dialup modem, FreeBSD can still be installed over the Internet, it will just take a very long time.

You will need to know:

1. The phone number to dial the Internet Service Provider (ISP)
2. The COM: port the modem is connected to
3. The username and password for the ISP account

3.3.5 Check for FreeBSD Errata

Although the FreeBSD Project strives to ensure that each release of FreeBSD is as stable as possible, bugs do occasionally creep into the process. On rare occasions those bugs affect the installation process. As these problems are discovered and fixed, they are noted in the FreeBSD Errata (http://www.FreeBSD.org/releases/9.1R/errata.html), which is found on the FreeBSD website. Check the errata before installing to make sure that there are no late-breaking problems to be aware of.

Information about all releases, including the errata for each release, can be found on the release information (http://www.FreeBSD.org/releases/index.html) section of the FreeBSD website (http://www.FreeBSD.org/index.html).

3.3.6 Obtain the FreeBSD Installation Files

The FreeBSD installer can install FreeBSD from files located in any of the following places:

Local Media

- A CDROM or DVD
- A USB Memory Stick
- A MS-DOS partition on the same computer
- Floppy disks (FreeBSD/pc98 only)

Network

- An FTP site through a firewall or using an HTTP proxy
- An NFS server
- A dedicated parallel or serial connection

If installing from a purchased FreeBSD CD/DVD, skip ahead to Section 3.3.7.

To obtain the FreeBSD installation files, skip ahead to Section 3.13 which explains how to prepare the installation media. After reading that section, come back here and read on to Section 3.3.7.

3.3.7 Prepare the Boot Media

The FreeBSD installation process is started by booting the computer into the FreeBSD installer. It is not a program that can be run within another operating system. The computer normally boots using the operating system installed on the hard disk, but it can also be configured to boot from a CDROM or from a USB disk.

> **Tip:** If installing from a CD/DVD to a computer whose BIOS supports booting from the CD/DVD, skip this section. The FreeBSD CD/DVD images are bootable and can be used to install FreeBSD without any other special preparation.

To create a bootable memory stick, follow these steps:

1. Acquire the Memory Stick Image

   Memory stick images for FreeBSD 8.*X* can be downloaded from the `ISO-IMAGES/` directory at `ftp://ftp.FreeBSD.org/pub/FreeBSD/releases/`*`arch`*`/ISO-IMAGES/`*`version`*`/FreeBSD-`*`version`*`-RELEASE-`*`arch`*`-`
   Replace *`arch`* and *`version`* with the architecture and the version number to install. For example, the memory stick images for FreeBSD/i386 8.4-RELEASE are available from ftp://ftp.FreeBSD.org/pub/FreeBSD/releases/i386/ISO-IMAGES/8.4/FreeBSD-8.4-RELEASE-i386-memstick.img.

   > **Tip:** A different directory path is used for FreeBSD 9.0-RELEASE and later versions. How to download and install FreeBSD 9.*X* is covered in Chapter 2.

   The memory stick image has a `.img` extension. The `ISO-IMAGES/` directory contains a number of different images and the one to use depends on the version of FreeBSD and the type of media supported by the hardware being installed to.

   > **Important:** Before proceeding, *back up* the data on the USB stick, as this procedure will *erase* it.

2. Write the Image File to the Memory Stick

Using FreeBSD to Write the Image

> **Warning:** The example below lists `/dev/da0` as the target device where the image will be written. Be very careful that you have the correct device as the output target, or you may destroy your existing data.

1. Writing the Image with dd(1)

   The `.img` file is *not* a regular file that can just be copied to the memory stick. It is an image of the complete contents of the disk. This means that dd(1) must be used to write the image directly to the disk:

   ```
   # dd if=FreeBSD-8.4-RELEASE-i386-memstick.img of=/dev/da0 bs=64k
   ```

   If an `Operation not permitted` error is displayed, make certain that the target device is not in use, mounted, or being automounted by another program. Then try again.

Using Windows to Write the Image

> **Warning:** Make sure to use the correct drive letter as the output target, as this command will overwrite and destroy any existing data on the specified device.

1. Obtaining **Image Writer for Windows**

   Image Writer for Windows is a free application that can correctly write an image file to a memory stick. Download it from https://launchpad.net/win32-image-writer/ and extract it into a folder.

2. Writing the Image with Image Writer

   Double-click the **Win32DiskImager** icon to start the program. Verify that the drive letter shown under `Device` is the drive with the memory stick. Click the folder icon and select the image to be written to the memory stick. Click Save to accept the image file name. Verify that everything is correct, and that no folders on the memory stick are open in other windows. Finally, click Write to write the image file to the drive.

To create the boot floppy images for a FreeBSD/pc98 installation, follow these steps:

1. Acquire the Boot Floppy Images

   The FreeBSD/pc98 boot disks can be downloaded from the floppies directory, `ftp://ftp.FreeBSD.org/pub/FreeBSD/releases/pc98/`*`version`*`-RELEASE/floppies/`. Replace *`version`* with the version number to install.

   The floppy images have a `.flp` extension. `floppies/` contains a number of different images. Download `boot.flp` as well as the number of files associated with the type of installation, such as `kern.small*` or `kern*`.

Important: The FTP program must use *binary mode* to download these disk images. Some web browsers use *text* or *ASCII* mode, which will be apparent if the disks are not bootable.

2. Prepare the Floppy Disks

   Prepare one floppy disk per downloaded image file. It is imperative that these disks are free from defects. The easiest way to test this is to reformat the disks. Do not trust pre-formatted floppies. The format utility in Windows will not tell about the presence of bad blocks, it simply marks them as "bad" and ignores them. It is advised to use brand new floppies.

   Important: If the installer crashes, freezes, or otherwise misbehaves, one of the first things to suspect is the floppies. Write the floppy image files to new disks and try again.

3. Write the Image Files to the Floppy Disks

   The `.flp` files are *not* regular files that can be copied to the disk. They are images of the complete contents of the disk. Specific tools must be used to write the images directly to the disk.

   FreeBSD provides a tool called `rawrite` for creating the floppies on a computer running Windows. This tool can be downloaded from `ftp://ftp.FreeBSD.org/pub/FreeBSD/releases/pc98/` *`version`*`-RELEASE/tools/` on the FreeBSD FTP site. Download this tool, insert a floppy, then specify the filename to write to the floppy drive:

   ```
   C:\> rawrite boot.flp A:
   ```

   Repeat this command for each `.flp` file, replacing the floppy disk each time, being sure to label the disks with the name of the file. Adjust the command line as necessary, depending on where the `.flp` files are located.

   When writing the floppies on a UNIX-like system, such as another FreeBSD system, use dd(1) to write the image files directly to disk. On FreeBSD, run:

   ```
   # dd if=boot.flp of=/dev/fd0
   ```

   On FreeBSD, `/dev/fd0` refers to the first floppy disk. Other UNIX variants might have different names for the floppy disk device, so check the documentation for the system as necessary.

You are now ready to start installing FreeBSD.

3.4 Starting the Installation

Important: By default, the installer will not make any changes to the disk(s) until after the following message:

```
Last Chance: Are you SURE you want continue the installation?

If you're running this on a disk with data you wish to save then WE
STRONGLY ENCOURAGE YOU TO MAKE PROPER BACKUPS before proceeding!

We can take no responsibility for lost disk contents!
```

The install can be exited at any time prior to this final warning without changing the contents of the hard drive. If there is a concern that something is configured incorrectly, turn the computer off before this point, and no damage will be done.

3.4.1 Booting

3.4.1.1 Booting for the i386

1. Turn on the computer. As it starts it should display an option to enter the system set up menu, or BIOS, commonly reached by keys like **F2**, **F10**, **Del**, or **Alt**+**S**. Use whichever keystroke is indicated on screen. In some cases the computer may display a graphic while it starts. Typically, pressing **Esc** will dismiss the graphic and display the boot messages.
2. Find the setting that controls which devices the system boots from. This is usually labeled as the "Boot Order" and commonly shown as a list of devices, such as `Floppy`, `CDROM`, `First Hard Disk`, and so on.

   If booting from the CD/DVD, make sure that the CDROM drive is selected. If booting from a USB disk, make sure that it is selected instead. When in doubt, consult the manual that came with the computer or its motherboard.

   Make the change, then save and exit. The computer should now restart.
3. If using a prepared a "bootable" USB stick, as described in Section 3.3.7, plug in the USB stick before turning on the computer.

   If booting from CD/DVD, turn on the computer, and insert the CD/DVD at the first opportunity.

   > **Note:** For FreeBSD/pc98, installation boot floppies are available and can be prepared as described in Section 3.3.7. The first floppy disc will contain `boot.flp`. Put this floppy in the floppy drive to boot into the installer.

   If the computer starts up as normal and loads the existing operating system, then either:

   1. The disks were not inserted early enough in the boot process. Leave them in, and try restarting the computer.
   2. The BIOS changes did not work correctly. Redo that step until the right option is selected.
   3. That particular BIOS does not support booting from the desired media.
4. FreeBSD will start to boot. If booting from CD/DVD, messages will be displayed, similar to these:

```
Booting from CD-Rom...
645MB medium detected
CD Loader 1.2

Building the boot loader arguments
Looking up /BOOT/LOADER... Found
Relocating the loader and the BTX
Starting the BTX loader

BTX loader 1.00 BTX version is 1.02
Consoles: internal video/keyboard
```

```
BIOS CD is cd0
BIOS drive C: is disk0
BIOS drive D: is disk1
BIOS 636kB/261056kB available memory

FreeBSD/i386 bootstrap loader, Revision 1.1

Loading /boot/defaults/loader.conf
/boot/kernel/kernel text=0x64daa0 data=0xa4e80+0xa9e40 syms=[0x4+0x6cac0+0x4+0x88e9d]
\
```

If booting from floppy disc, a display similar to this will be shown:

```
Booting from Floppy...
Uncompressing ... done

BTX loader 1.00  BTX version is 1.01
Console: internal video/keyboard
BIOS drive A: is disk0
BIOS drive C: is disk1
BIOS 639kB/261120kB available memory

FreeBSD/i386 bootstrap loader, Revision 1.1

Loading /boot/defaults/loader.conf
/kernel text=0x277391 data=0x3268c+0x332a8 |

Insert disk labelled "Kernel floppy 1" and press any key...
```

Remove the `boot.flp` floppy, insert the next floppy, and press **Enter**. When prompted, insert the other disks as required.

5. The boot process will then display the FreeBSD boot loader menu:

Figure 3-1. FreeBSD Boot Loader Menu

Either wait ten seconds, or press **Enter**.

3.4.1.2 Booting for SPARC64

Most SPARC64 systems are set to boot automatically from disk. To install FreeBSD, boot over the network or from a CD/DVD and wait until the boot message appears. The message depends on the model, but should look similar to:

```
Sun Blade 100 (UltraSPARC-IIe), Keyboard Present
Copyright 1998-2001 Sun Microsystems, Inc.  All rights reserved.
OpenBoot 4.2, 128 MB memory installed, Serial #51090132.
Ethernet address 0:3:ba:b:92:d4, Host ID: 830b92d4.
```

If the system proceeds to boot from disk, press **L1**+**A** or **Stop**+**A** on the keyboard, or send a `BREAK` over the serial console using ~# in tip(1) or cu(1) to get to the PROM prompt. It looks like this:

```
ok     ❶
ok {0} ❷
```

❶ This is the prompt used on systems with just one CPU.

❷ This is the prompt used on SMP systems and the digit indicates the number of the active CPU.

At this point, place the CD/DVD into the drive and from the PROM prompt, type `boot cdrom`.

3.4.2 Reviewing the Device Probe Results

The last few hundred lines that have been displayed on screen are stored and can be reviewed.

To review this buffer, press **Scroll Lock** to turn on scrolling in the display. Use the arrow keys or **PageUp** and **PageDown** to view the results. Press **Scroll Lock** again to stop scrolling.

Do this now, to review the text that scrolled off the screen when the kernel was carrying out the device probes. Text similar to Figure 3-2 will be displayed, although it will differ depending on the devices in the computer.

Figure 3-2. Typical Device Probe Results

```
avail memory = 253050880 (247120K bytes)
Preloaded elf kernel "kernel" at 0xc0817000.
Preloaded mfs_root "/mfsroot" at 0xc0817084.
md0: Preloaded image </mfsroot> 4423680 bytes at 0xc03ddcd4

md1: Malloc disk
Using $PIR table, 4 entries at 0xc00fde60
npx0: <math processor> on motherboard
npx0: INT 16 interface
pcib0: <Host to PCI bridge> on motherboard
pci0: <PCI bus> on pcib0
pcib1:<VIA 82C598MVP (Apollo MVP3) PCI-PCI (AGP) bridge> at device 1.0 on pci0
pci1: <PCI bus> on pcib1
pci1: <Matrox MGA G200 AGP graphics accelerator> at 0.0 irq 11
isab0: <VIA 82C586 PCI-ISA bridge> at device 7.0 on pci0
isa0: <iSA bus> on isab0
atapci0: <VIA 82C586 ATA33 controller> port 0xe000-0xe00f at device 7.1 on pci0
ata0: at 0x1f0 irq 14 on atapci0
```

```
ata1: at 0x170 irq 15 on atapci0
uhci0 <VIA 83C572 USB controller> port 0xe400-0xe41f irq 10 at device 7.2 on pci
0
usb0: <VIA 83572 USB controller> on uhci0
usb0: USB revision 1.0
uhub0: VIA UHCI root hub, class 9/0, rev 1.00/1.00, addr1
uhub0: 2 ports with 2 removable, self powered
pci0: <unknown card> (vendor=0x1106, dev=0x3040) at 7.3
dc0: <ADMtek AN985 10/100BaseTX> port 0xe800-0xe8ff mem 0xdb000000-0xeb0003ff ir
q 11 at device 8.0 on pci0
dc0: Ethernet address: 00:04:5a:74:6b:b5
miibus0: <MII bus> on dc0
ukphy0: <Generic IEEE 802.3u media interface> on miibus0
ukphy0: 10baseT, 10baseT-FDX, 100baseTX, 100baseTX-FDX, auto
ed0: <NE2000 PCI Ethernet (RealTek 8029)> port 0xec00-0xec1f irq 9 at device 10.
0 on pci0
ed0 address 52:54:05:de:73:1b, type NE2000 (16 bit)
isa0: too many dependant configs (8)
isa0: unexpected small tag 14
orm0: <Option ROM> at iomem 0xc0000-0xc7fff on isa0
fdc0: <NEC 72065B or clone> at port 0x3f0-0x3f5,0x3f7 irq 6 drq2 on isa0
fdc0: FIFO enabled, 8 bytes threshold
fd0: <1440-KB 3.5” drive> on fdc0 drive 0
atkbdc0: <Keyboard controller (i8042)> at port 0x60,0x64 on isa0
atkbd0: <AT Keyboard> flags 0x1 irq1 on atkbdc0
kbd0 at atkbd0
psm0: <PS/2 Mouse> irq 12 on atkbdc0
psm0: model Generic PS/@ mouse, device ID 0
vga0: <Generic ISA VGA> at port 0x3c0-0x3df iomem 0xa0000-0xbffff on isa0
sc0: <System console> at flags 0x100 on isa0
sc0: VGA <16 virtual consoles, flags=0x300>
sio0 at port 0x3f8-0x3ff irq 4 flags 0x10 on isa0
sio0: type 16550A
sio1 at port 0x2f8-0x2ff irq 3 on isa0
sio1: type 16550A
ppc0: <Parallel port> at port 0x378-0x37f irq 7 on isa0
pppc0: SMC-like chipset (ECP/EPP/PS2/NIBBLE) in COMPATIBLE mode
ppc0: FIFO with 16/16/15 bytes threshold
plip0: <PLIP network interface> on ppbus0
ad0: 8063MB <IBM-DHEA-38451> [16383/16/63] at ata0-master UDMA33
acd0: CD-RW <LITE-ON LTR-1210B> at ata1-slave PIO4
Mounting root from ufs:/dev/md0c
/stand/sysinstall running as init on vty0
```

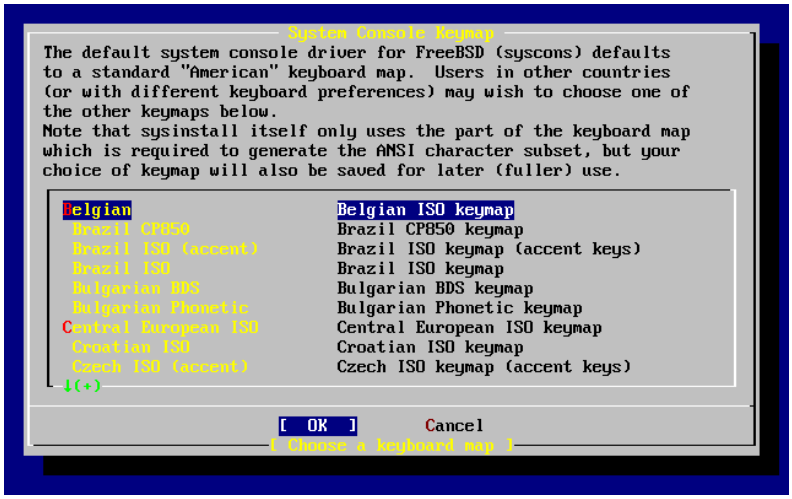
Check the probe results carefully to make sure that FreeBSD found all the devices. If a device was not found, it will not be listed. A custom kernel can be used to add in support for devices which are not in the `GENERIC` kernel.

After the device probe, the menu shown in Figure 3-3 will be displayed. Use the arrow key to choose a country, region, or group. Then press **Enter** to set the country.

Figure 3-3. Selecting Country Menu

If United States is selected as the country, the standard American keyboard map will be used. If a different country is chosen, the following menu will be displayed. Use the arrow keys to choose the correct keyboard map and press **Enter**.

Figure 3-4. Selecting Keyboard Menu



After the country selection, the sysinstall(8) main menu will display.

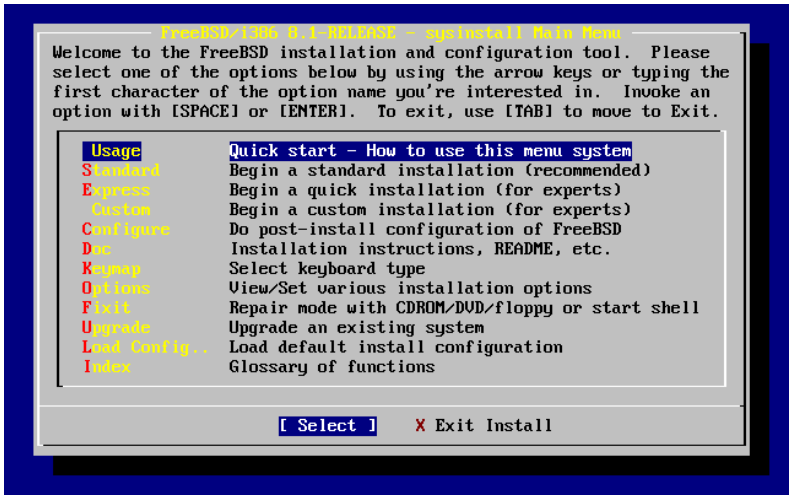
3.5 Introducing sysinstall(8)

The FreeBSD 8.X installer, sysinstall(8), is console based and is divided into a number of menus and screens that can be used to configure and control the installation process.

This menu system is controlled by the arrow keys, **Enter**, **Tab**, **Space**, and other keys. To view a detailed description of these keys and what they do, ensure that the Usage entry is highlighted and that the [Select] button is selected, as shown in Figure 3-5, then press **Enter**.

The instructions for using the menu system will be displayed. After reviewing them, press **Enter** to return to the Main Menu.

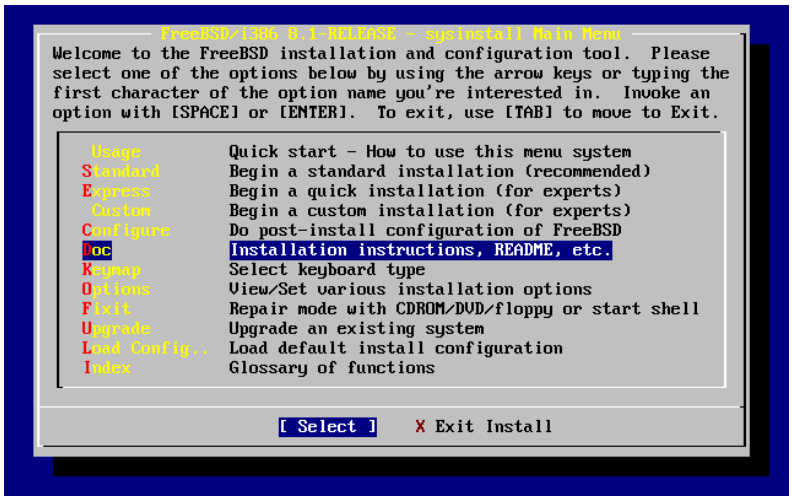
Figure 3-5. Selecting Usage from Sysinstall Main Menu



3.5.1 Selecting the Documentation Menu

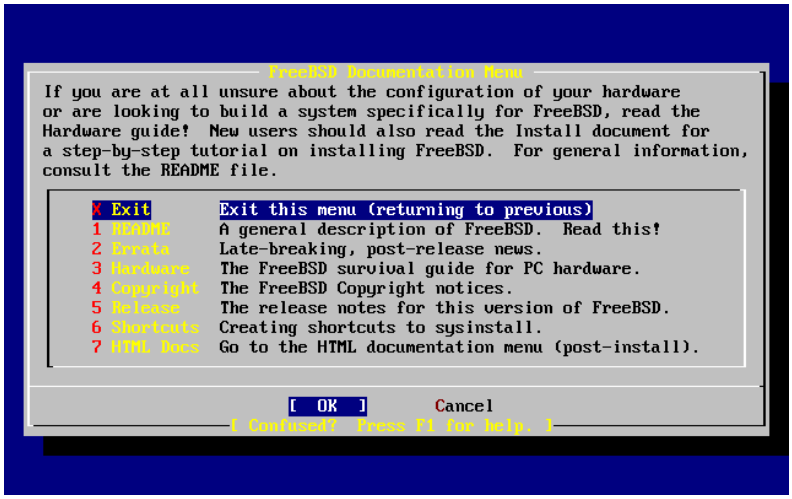
From the Main Menu, select Doc with the arrow keys and press **Enter**.

Figure 3-6. Selecting Documentation Menu



This will display the Documentation Menu.

Figure 3-7. Sysinstall Documentation Menu



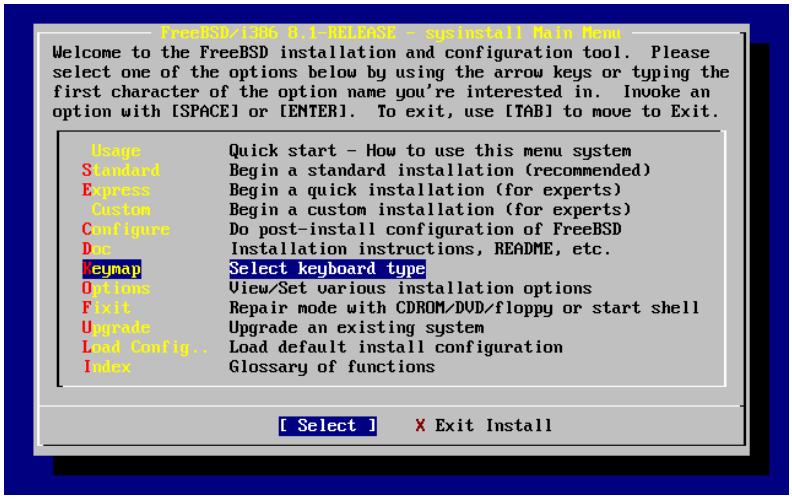
It is important to read the documents provided. To view a document, select it with the arrow keys and press **Enter**. When finished reading a document, press **Enter** to return to the Documentation Menu.

To return to the Main Installation Menu, select Exit with the arrow keys and press **Enter**.

3.5.2 Selecting the Keymap Menu

To change the keyboard mapping, use the arrow keys to select Keymap from the menu and press **Enter**. This is only required when using a non-standard or non-US keyboard.

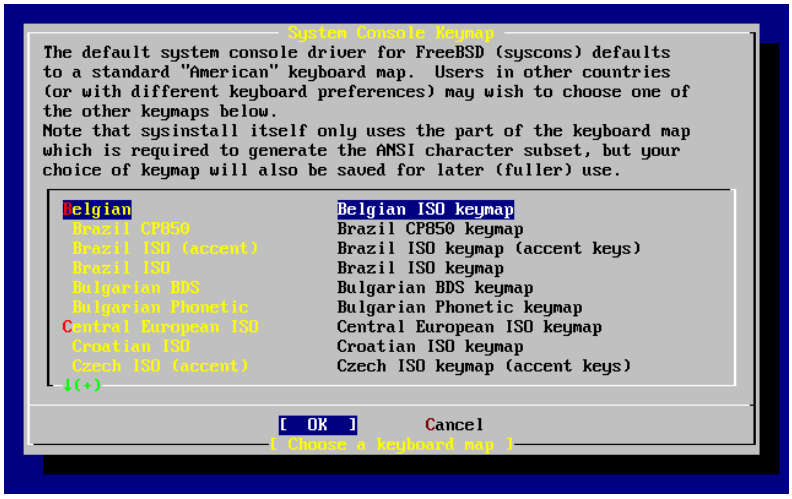
Figure 3-8. Sysinstall Main Menu



A different keyboard mapping may be chosen by selecting the menu item using the up and down arrow keys and pressing **Space**. Pressing **Space** again will unselect the item. When finished, choose the [OK] using the arrow keys and press **Enter**.

Only a partial list is shown in this screen representation. Selecting [Cancel] by pressing **Tab** will use the default keymap and return to the Main Install Menu.

Figure 3-9. Sysinstall Keymap Menu



3.5.3 Installation Options Screen

Select Options and press **Enter**.

Figure 3-10. Sysinstall Main Menu

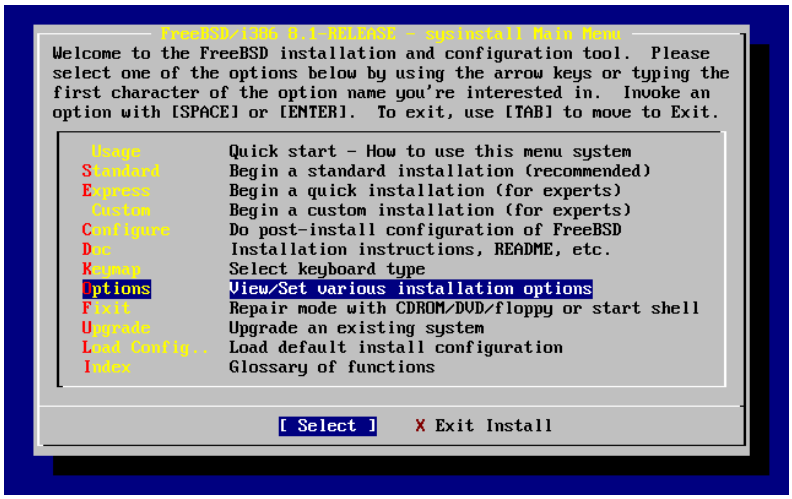
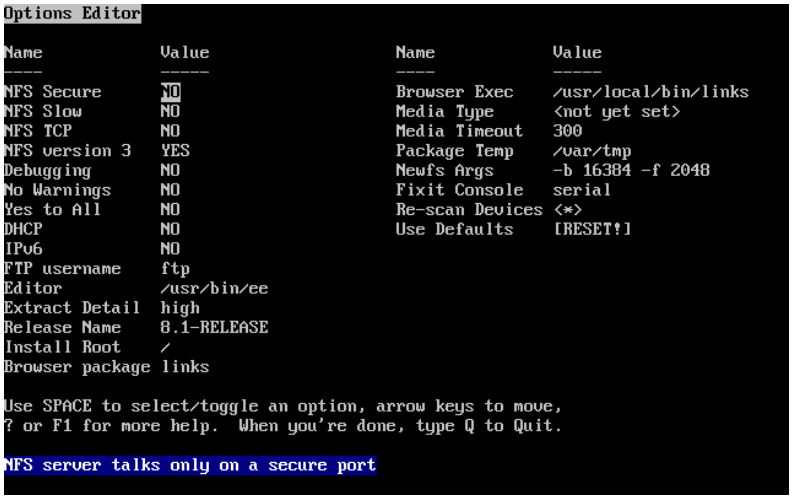


Figure 3-11. Sysinstall Options



The default values are usually fine for most users and do not need to be changed. The release name will vary according to the version being installed.

The description of the selected item will appear at the bottom of the screen highlighted in blue. Notice that one of the options is Use Defaults to reset all values to startup defaults.

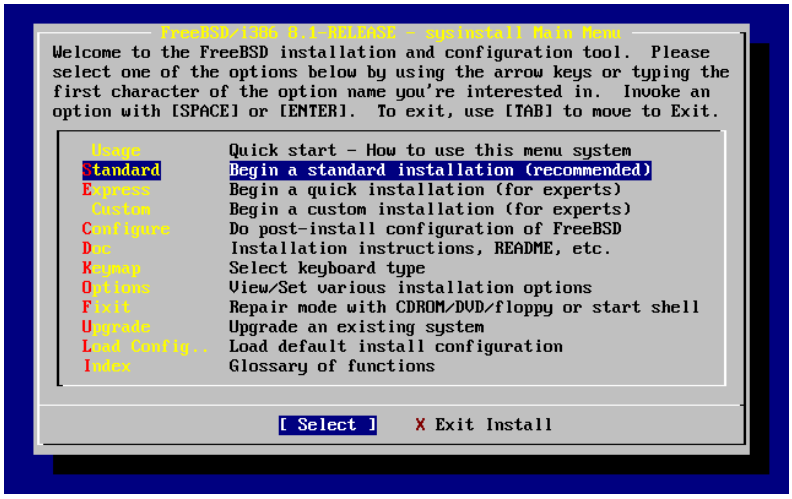
Press **F1** to read the help screen about the various options.

Press **Q** to return to the Main Install menu.

3.5.4 Begin a Standard Installation

The Standard installation is the option recommended for those new to UNIX or FreeBSD. Use the arrow keys to select Standard and then press **Enter** to start the installation.

Figure 3-12. Begin Standard Installation



3.6 Allocating Disk Space

The first task is to allocate disk space for FreeBSD, and label that space so that sysinstall(8) can prepare it. In order to do this you need to know how FreeBSD expects to find information on the disk.

3.6.1 BIOS Drive Numbering

Before installing and configuring FreeBSD it is important to be aware how FreeBSD deals with BIOS drive mappings.

In a PC running a BIOS-dependent operating system such as Microsoft Windows, the BIOS is able to abstract the normal disk drive order and the operating system goes along with the change. This allows the user to boot from a disk drive other than the "primary master". This is especially convenient for users buy an identical second hard drive, and perform routine copies of the first drive to the second drive. If the first drive fails, is attacked by a virus, or is scribbled upon by an operating system defect, they can easily recover by instructing the BIOS to logically swap the drives. It is like switching the cables on the drives, without having to open the case.

Systems with SCSI controllers often include BIOS extensions which allow the SCSI drives to be re-ordered in a similar fashion for up to seven drives.

A user who is accustomed to taking advantage of these features may become surprised when the results with FreeBSD are not as expected. FreeBSD does not use the BIOS, and does not know the “logical BIOS drive mapping”. This can lead to perplexing situations, especially when drives are physically identical in geometry and have been made as data clones of one another.

When using FreeBSD, always restore the BIOS to natural drive numbering before installing FreeBSD, and then leave it that way. If drives need to be switched around, take the time to open the case and move the jumpers and cables.

An Illustration from the Files of Bill and Fred's Exceptional Adventures:

Bill breaks-down an older Wintel box to make another FreeBSD box for Fred. Bill installs a single SCSI drive as SCSI unit zero and installs FreeBSD on it.

Fred begins using the system, but after several days notices that the older SCSI drive is reporting numerous errors.

To address the situation, Bill grabs an identical SCSI drive and installs this drive as SCSI unit four and makes an image copy from drive zero to drive four. Now that the new drive is installed and functioning, Bill decides to start using it, so he uses features in the SCSI BIOS to re-order the disk drives so that the system boots from SCSI unit four. FreeBSD boots and runs just fine.

Fred continues his work and soon decides that it is time to upgrade to a newer version of FreeBSD. Bill removes SCSI unit zero because it was a bit flaky and replaces it with another identical disk drive. Bill then installs the new version of FreeBSD onto the new SCSI unit zero and the installation goes well.

Fred uses the new version of FreeBSD for a few days, and certifies that it is good enough for use in the engineering department. It is time to copy all of his work from the old version, so Fred mounts SCSI unit four which should contain the latest copy of the older FreeBSD version. Fred is dismayed to find that none of his work is present on SCSI unit four.

It turns out that when Bill made an image copy of the original SCSI unit zero onto SCSI unit four, unit four became the "new clone". When Bill re-ordered the SCSI BIOS so that he could boot from SCSI unit four, FreeBSD was still running on SCSI unit zero. Making this kind of BIOS change causes some or all of the boot and loader code to be fetched from the selected BIOS drive. But when the FreeBSD kernel drivers take over, the BIOS drive numbering is ignored, and FreeBSD transitions back to normal drive numbering. In this example, the system continued to operate on the original SCSI unit zero, and all of Fred's data was there, not on SCSI unit four. The fact that the system appeared to be running on SCSI unit four was simply an artifact of human expectations.

Fortunately, the older SCSI unit zero was retrieved and all of Fred's work was restored.

Although SCSI drives were used in this illustration, the concepts apply equally to IDE drives.

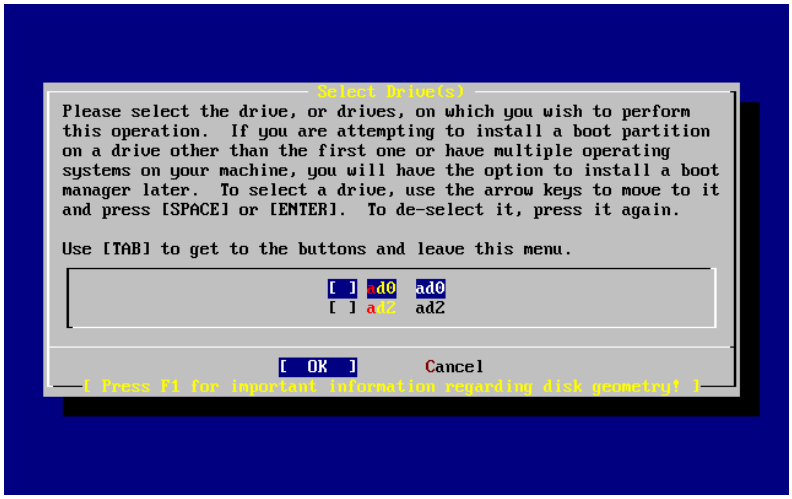
3.6.2 Creating Slices Using FDisk

After choosing to begin a standard installation in sysinstall(8), this message will appear:

```
                              Message
In the next menu, you will need to set up a DOS-style ("fdisk")
partitioning scheme for your hard disk. If you simply wish to devote
all disk space to FreeBSD (overwriting anything else that might be on
the disk(s) selected) then use the (A)ll command to select the default
partitioning scheme followed by a (Q)uit. If you wish to allocate only
free space to FreeBSD, move to a partition marked "unused" and use the
(C)reate command.
                              [  OK  ]

                      [ Press enter or space ]
```

Press **Enter** and a list of all the hard drives that the kernel found when it carried out the device probes will be displayed. Figure 3-13 shows an example from a system with two IDE disks called `ad0` and `ad2`.

Figure 3-13. Select Drive for FDisk



Note that `ad1` is not listed here.

Consider two IDE hard disks where one is the master on the first IDE controller and one is the master on the second IDE controller. If FreeBSD numbered these as `ad0` and `ad1`, everything would work.

But if a third disk is later added as the slave device on the first IDE controller, it would now be `ad1`, and the previous `ad1` would become `ad2`. Because device names are used to find filesystems, some filesystems may no longer appear correctly, requiring a change to the FreeBSD configuration.

To work around this, the kernel can be configured to name IDE disks based on where they are and not the order in which they were found. With this scheme, the master disk on the second IDE controller will *always* be `ad2`, even if there are no `ad0` or `ad1` devices.

This configuration is the default for the FreeBSD kernel, which is why the display in this example shows `ad0` and `ad2`. The machine on which this screenshot was taken had IDE disks on both master channels of the IDE controllers and no disks on the slave channels.

Select the disk on which to install FreeBSD, and then press [OK]. **FDisk** will start, with a display similar to that shown in Figure 3-14.

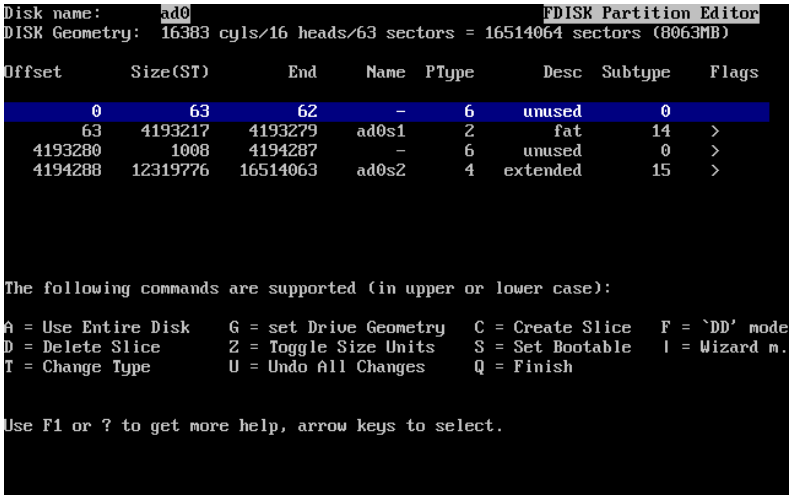
The **FDisk** display is broken into three sections.

The first section, covering the first two lines of the display, shows details about the currently selected disk, including its FreeBSD name, the disk geometry, and the total size of the disk.

The second section shows the slices that are currently on the disk, where they start and end, how large they are, the name FreeBSD gives them, and their description and sub-type. This example shows two small unused slices which are artifacts of disk layout schemes on the PC. It also shows one large FAT slice, which appears as `C:` in Windows, and an extended slice, which may contain other drive letters in Windows.

The third section shows the commands that are available in **FDisk**.

Figure 3-14. Typical Default FDisk Partitions



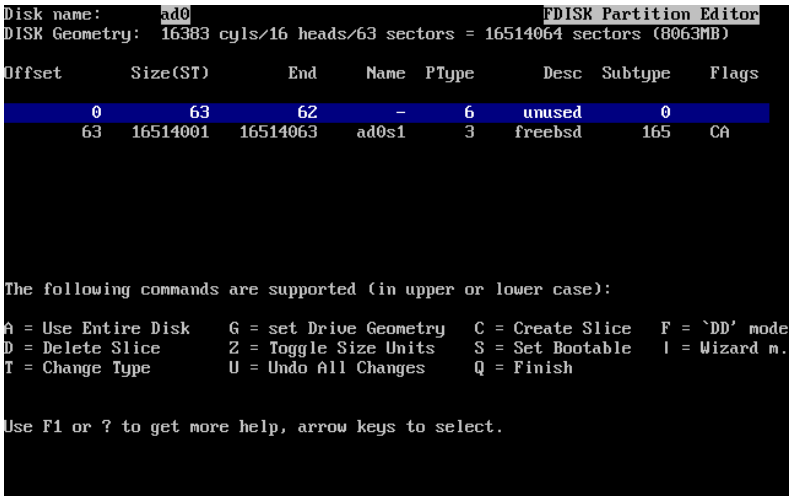
This step varies, depending on how the disk is to be sliced.

To install FreeBSD to the entire disk, which will delete all the other data on this disk, press **A**, which corresponds to the Use Entire Disk option. The existing slices will be removed and replaced with a small area flagged as `unused` and one large slice for FreeBSD. Then, select the newly created FreeBSD slice using the arrow keys and press **S** to mark the slice as being bootable. The screen will then look similar to Figure 3-15. Note the `A` in the `Flags` column, which indicates that this slice is *active*, and will be booted from.

If an existing slice needs to be deleted to make space for FreeBSD, select the slice using the arrow keys and press **D**. Then, press **C** to be prompted for the size of the slice to create. Enter the appropriate value and press **Enter**. The default value in this box represents the largest possible slice to make, which could be the largest contiguous block of unallocated space or the size of the entire hard disk.

If you have already made space for FreeBSD then you can press **C** to create a new slice. Again, you will be prompted for the size of slice you would like to create.

Figure 3-15. Fdisk Partition Using Entire Disk



When finished, press **Q**. Any changes will be saved in sysinstall(8), but will not yet be written to disk.

3.6.3 Install a Boot Manager

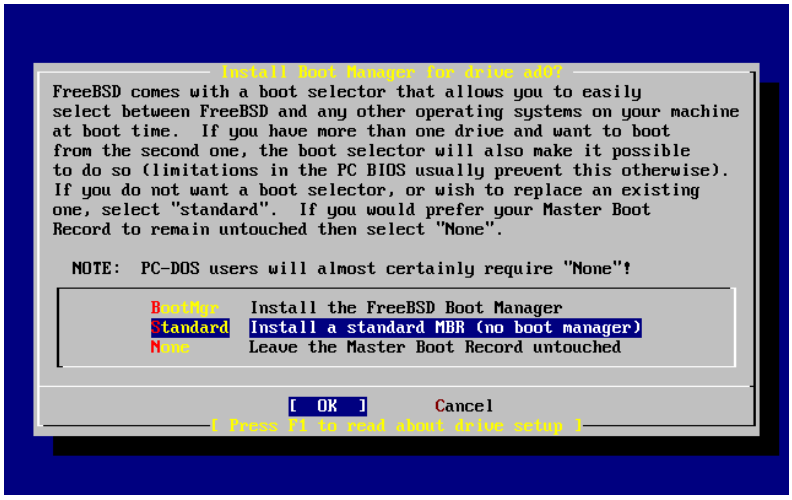
The next menu provides the option to install a boot manager. In general, install the FreeBSD boot manager if:

- There is more than one drive and FreeBSD will be installed onto a drive other than the first one.
- FreeBSD will be installed alongside another operating system on the same disk, and you want to choose whether to start FreeBSD or the other operating system when the computer starts.

If FreeBSD is going to be the only operating system on this machine, installed on the first hard disk, then the Standard boot manager will suffice. Choose None if using a third-party boot manager capable of booting FreeBSD.

Make a selection and press **Enter**.

Figure 3-16. Sysinstall Boot Manager Menu



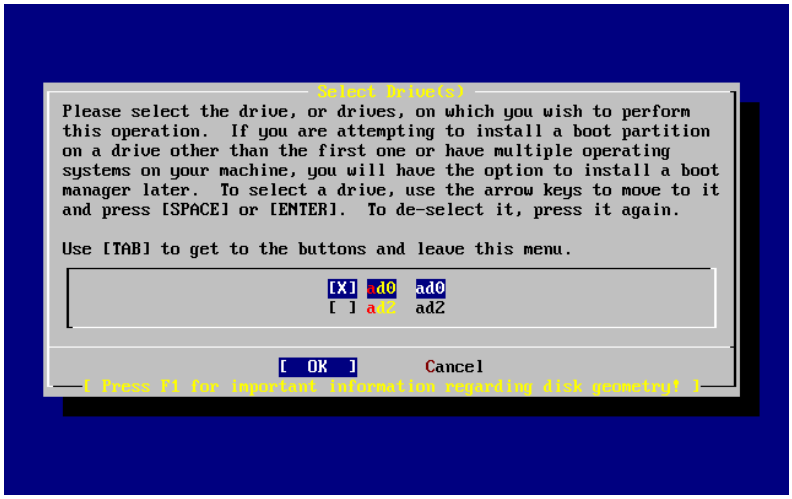
The help screen, reached by pressing **F1**, discusses the problems that can be encountered when trying to share the hard disk between operating systems.

3.6.4 Creating Slices on Another Drive

If there is more than one drive, it will return to the Select Drives screen after the boot manager selection. To install FreeBSD on to more than one disk, select another disk and repeat the slice process using **FDisk**.

> **Important:** If installing FreeBSD on a drive other than the first drive, the FreeBSD boot manager needs to be installed on both drives.

Figure 3-17. Exit Select Drive



Use **Tab** to toggle between the last drive selected, [OK], and [Cancel].

Press **Tab** once to toggle to [OK], then press **Enter** to continue with the installation.

3.6.5 Creating Partitions Using Disklabel

Next, create some partitions inside each slice. Remember that each partition is lettered, from a through to h, and that partitions b, c, and d have conventional meanings that should be adhered to.

Certain applications can benefit from particular partition schemes, especially when laying out partitions across more than one disk. However, for a first FreeBSD installation, do not give too much thought to how to partition the disk. It is more important to install FreeBSD and start learning how to use it. You can always re-install FreeBSD to change the partition scheme after becoming more familiar with the operating system.

The following scheme features four partitions: one for swap space and three for filesystems.

Table 3-2. Partition Layout for First Disk

Partition	Filesystem	Size	Description
a	/	1 GB	This is the root filesystem. Every other filesystem will be mounted somewhere under this one. 1 GB is a reasonable size for this filesystem as user files should not be stored here and a regular FreeBSD install will put about 128 MB of data here.

Partition	Filesystem	Size	Description
b	N/A	2-3 x RAM	The system's swap space is kept on the b partition. Choosing the right amount of swap space can be a bit of an art. A good rule of thumb is that swap space should be two or three times as much as the available physical memory (RAM). There should be at least 64 MB of swap, so if there is less than 32 MB of RAM in the computer, set the swap amount to 64 MB. If there is more than one disk, swap space can be put on each disk. FreeBSD will then use each disk for swap, which effectively speeds up the act of swapping. In this case, calculate the total amount of swap needed and divide this by the number of disks to give the amount of swap to put on each disk.
e	/var	512 MB to 4096 MB	/var contains files that are constantly varying, such as log files and other administrative files. Many of these files are read from or written to extensively during FreeBSD's day-to-day running. Putting these files on another filesystem allows FreeBSD to optimize the access of these files without affecting other files in other directories that do not have the same access pattern.
f	/usr	Rest of disk (at least 8 GB)	All other files will typically be stored in /usr and its subdirectories.

Warning: The values above are given as example and should be used by experienced users only. Users are encouraged to use the automatic partition layout called `Auto Defaults` by the FreeBSD partition editor.

If installing FreeBSD on to more than one disk, create partitions in the other configured slices. The easiest way to do this is to create two partitions on each disk, one for the swap space, and one for a filesystem.

Table 3-3. Partition Layout for Subsequent Disks

Partition	Filesystem	Size	Description
b	N/A	See description	Swap space can be split across each disk. Even though the a partition is free, convention dictates that swap space stays on the b partition.
e	/disk*n*	Rest of disk	The rest of the disk is taken up with one big partition. This could easily be put on the a partition, instead of the e partition. However, convention says that the a partition on a slice is reserved for the filesystem that will be the root (/) filesystem. Following this convention is not necessary, but sysinstall(8) uses it, so following it makes the installation slightly cleaner. This filesystem can be mounted anywhere; this example mounts it as /disk*n*, where *n* is a number that changes for each disk.

Having chosen the partition layout, create it using sysinstall(8).

```
                                Message
Now, you need to create BSD partitions inside of the fdisk
partition(s) just created. If you have a reasonable amount of disk
space (1GB or more) and don't have any special requirements, simply
use the (A)uto command to allocate space automatically. If you have
more specific needs or just don't care for the layout chosen by
(A)uto, press F1 for more information on manual layout.

                                [  OK  ]
                          [ Press enter or space ]
```

Press **Enter** to start the FreeBSD partition editor, called **Disklabel**.

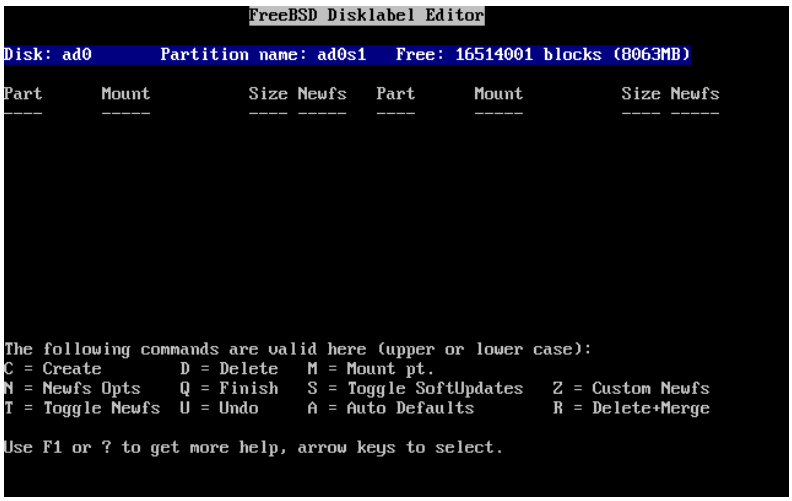
Figure 3-18 shows the display when **Disklabel** starts. The display is divided into three sections.

The first few lines show the name of the disk being worked on and the slice that contains the partitions to create. At this point, **Disklabel** calls this the `Partition name` rather than slice name. This display also shows the amount of free space within the slice; that is, space that was set aside in the slice, but that has not yet been assigned to a partition.

The middle of the display shows the partitions that have been created, the name of the filesystem that each partition contains, their size, and some options pertaining to the creation of the filesystem.

The bottom third of the screen shows the keystrokes that are valid in **Disklabel**.

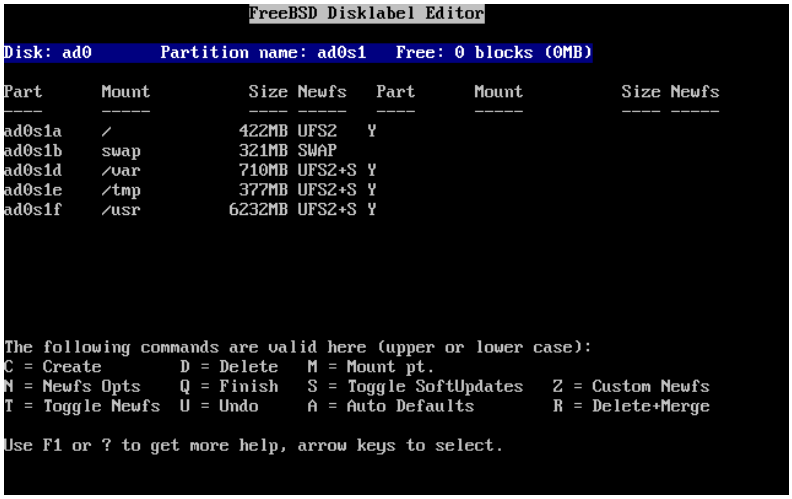
Figure 3-18. Sysinstall Disklabel Editor



Disklabel can automatically create partitions and assign them default sizes. The default sizes are calculated with the help of an internal partition sizing algorithm based on the disk size. Press **A** to see a display similar to that shown in Figure 3-19. Depending on the size of the disk, the defaults may or may not be appropriate.

> **Note:** The default partitioning assigns `/tmp` its own partition instead of being part of the `/` partition. This helps avoid filling the `/` partition with temporary files.

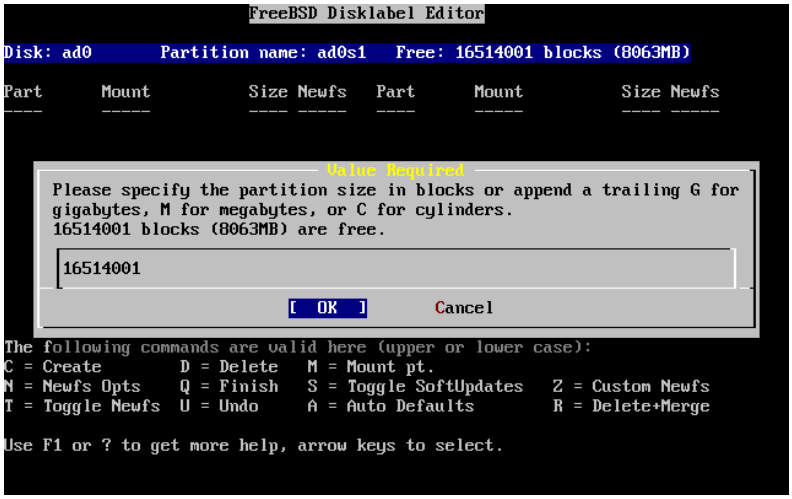
Figure 3-19. Sysinstall Disklabel Editor with Auto Defaults



To replace the default partitions, use the arrow keys to select the first partition and press **D** to delete it. Repeat this to delete all the suggested partitions.

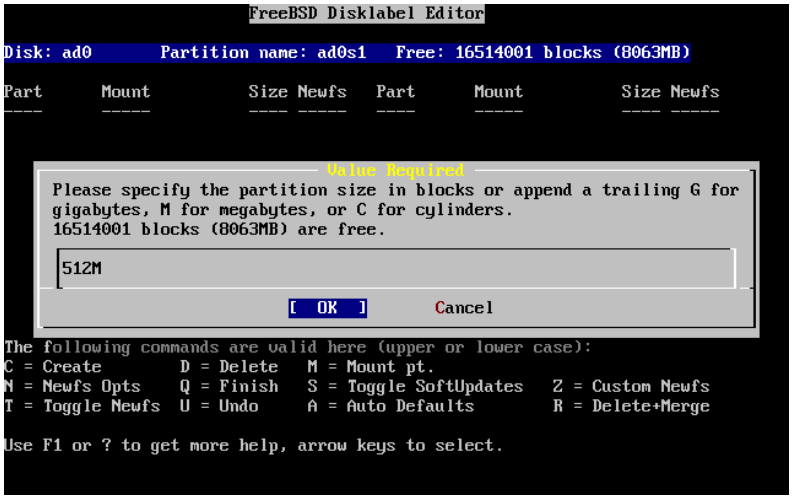
To create the first partition, `a`, mounted as `/`, make sure the proper disk slice at the top of the screen is selected and press **C**. A dialog box will appear, prompting for the size of the new partition, as shown in Figure 3-20. The size can be entered as the number of disk blocks to use or as a number followed by either `M` for megabytes, `G` for gigabytes, or `C` for cylinders.

Figure 3-20. Free Space for Root Partition



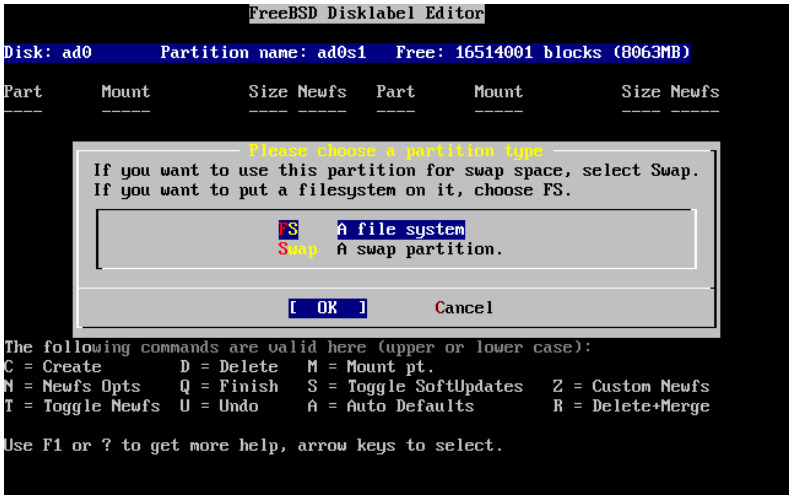
The default size shown will create a partition that takes up the rest of the slice. If using the partition sizes described in the earlier example, delete the existing figure using **Backspace**, and then type in **`512M`**, as shown in Figure 3-21. Then press [OK].

Figure 3-21. Edit Root Partition Size



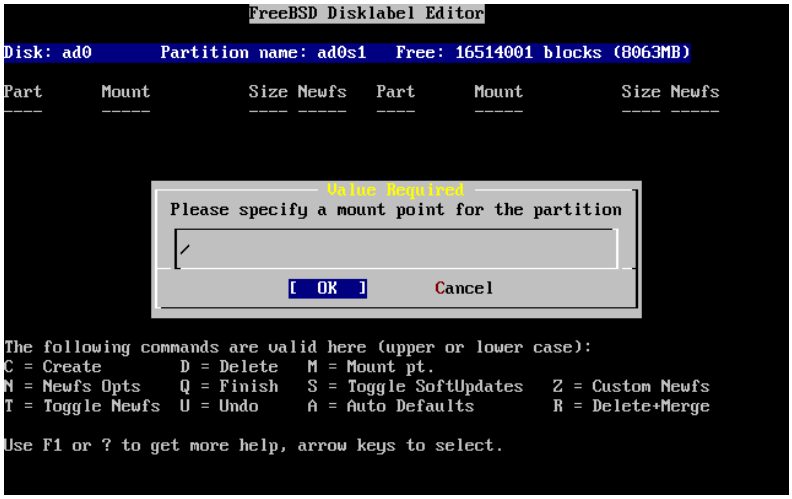
After choosing the partition's size, the installer will ask whether this partition will contain a filesystem or swap space. The dialog box is shown in Figure 3-22. This first partition will contain a filesystem, so check that FS is selected and press **Enter**.

Figure 3-22. Choose the Root Partition Type



Finally, tell **Disklabel** where the filesystem will be mounted. The dialog box is shown in Figure 3-23. Type `/`, and then press **Enter**.

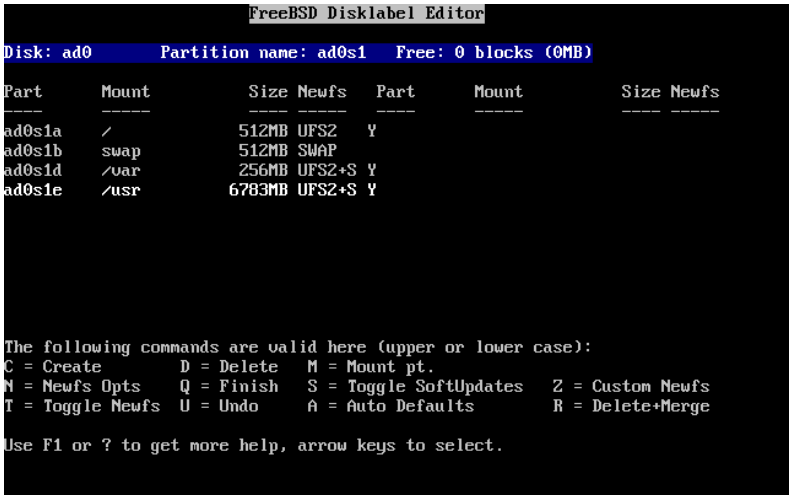
Figure 3-23. Choose the Root Mount Point



The display will then update to show the newly created partition. Repeat this procedure for the other partitions. When creating the swap partition, it will not prompt for the filesystem mount point. When creating the final partition, `/usr`, leave the suggested size as is to use the rest of the slice.

The final FreeBSD DiskLabel Editor screen will appear similar to Figure 3-24, although the values chosen may be different. Press **Q** to finish.

Figure 3-24. Sysinstall Disklabel Editor



3.7 Choosing What to Install

3.7.1 Select the Distribution Set

Deciding which distribution set to install will depend largely on the intended use of the system and the amount of disk space available. The predefined options range from installing the smallest possible configuration to everything. Those who are new to UNIX or FreeBSD should select one of these canned options. Customizing a distribution set is typically for the more experienced user.

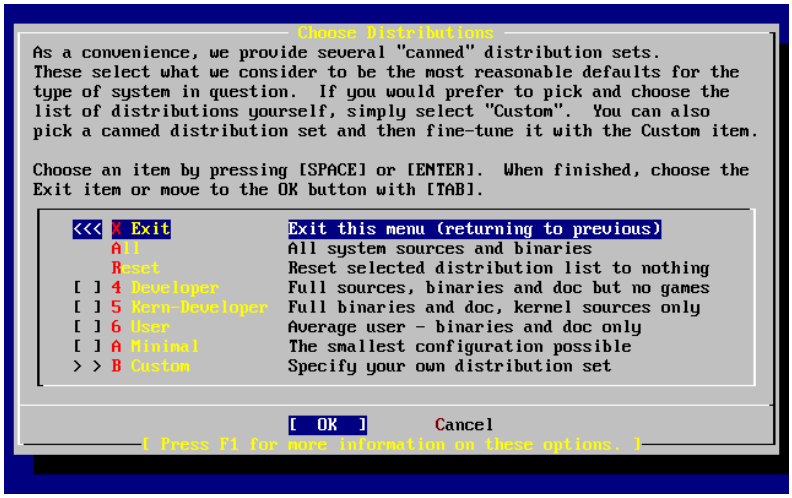
Press **F1** for more information on the distribution set options and what they contain. When finished reviewing the help, press **Enter** to return to the Select Distributions Menu.

If a graphical user interface is desired, the configuration of **Xorg** and selection of a default desktop must be done after the installation of FreeBSD. More information regarding the installation and configuration of a **Xorg** can be found in Chapter 6.

If compiling a custom kernel is anticipated, select an option which includes the source code. For more information on why a custom kernel should be built or how to build a custom kernel, see Chapter 9.

The most versatile system is one that includes everything. If there is adequate disk space, select All, as shown in Figure 3-25, by using the arrow keys and pressing **Enter**. If there is a concern about disk space, consider using an option that is more suitable for the situation. Do not fret over the perfect choice, as other distributions can be added after installation.

Figure 3-25. Choose Distributions



3.7.2 Installing the Ports Collection

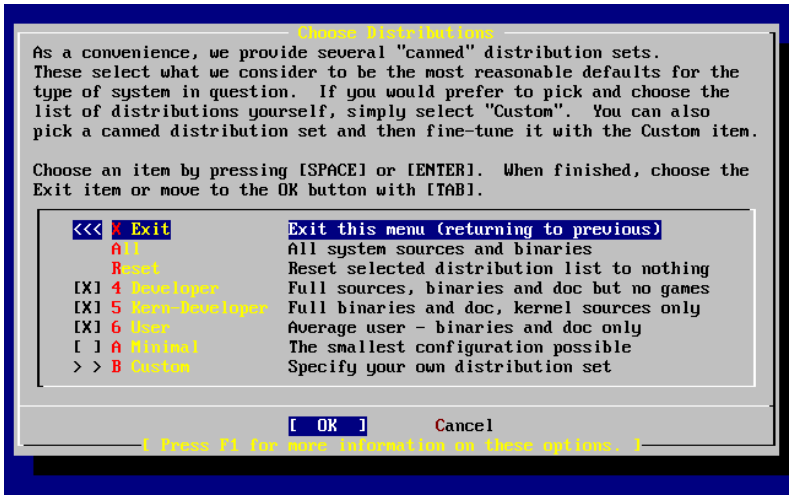
After selecting the desired distribution, an opportunity to install the FreeBSD Ports Collection is presented. The Ports Collection is an easy and convenient way to install software as it provides a collection of files that automate the downloading, compiling, and installation of third-party software packages. Chapter 5 discusses how to use the Ports Collection.

The installation program does not check to see if you have adequate space. Select this option only if you have adequate hard disk space. As of FreeBSD 9.1, the FreeBSD Ports Collection takes up about 500 MB of disk space. You can safely assume a larger value for more recent versions of FreeBSD.

```
                        User Confirmation Requested
Would you like to install the FreeBSD ports collection?

This will give you ready access to over 24,000 ported software packages,
at a cost of around 500 MB of disk space when "clean" and possibly much
more than that if a lot of the distribution tarballs are loaded
(unless you have the extra CDs from a FreeBSD CD/DVD distribution
available and can mount it on /cdrom, in which case this is far less
of a problem).

The Ports Collection is a very valuable resource and well worth having
on your /usr partition, so it is advisable to say Yes to this option.

For more information on the Ports Collection & the latest ports,
visit:
    http://www.FreeBSD.org/ports

                                [ Yes ]     No
```

Select [Yes] with the arrow keys to install the Ports Collection or [No] to skip this option. Press **Enter** to continue. The Choose Distributions menu will redisplay.

Figure 3-26. Confirm Distributions



Once satisfied with the options, select Exit with the arrow keys, ensure that [OK] is highlighted, and press **Enter** to continue.

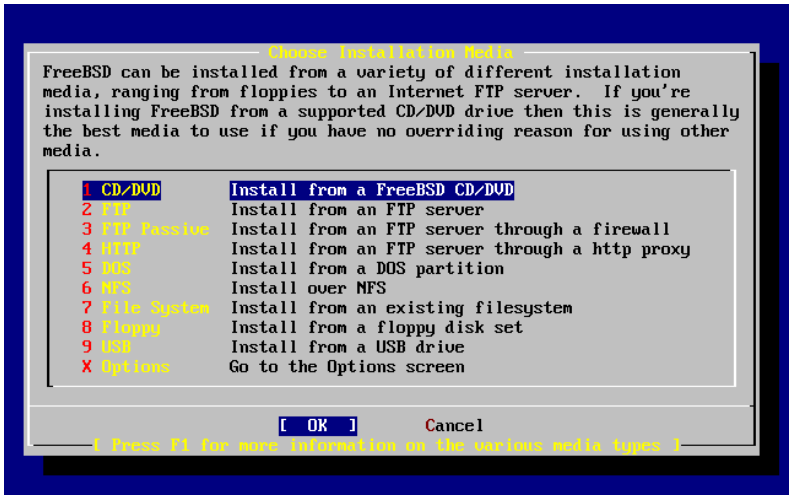
3.8 Choosing the Installation Media

If installing from a CD/DVD, use the arrow keys to highlight Install from a FreeBSD CD/DVD. Ensure that [OK] is highlighted, then press **Enter** to proceed with the installation.

For other methods of installation, select the appropriate option and follow the instructions.

Press **F1** to display the Online Help for installation media. Press **Enter** to return to the media selection menu.

Figure 3-27. Choose Installation Media



FTP Installation Modes: There are three FTP installation modes to choose from: active FTP, passive FTP, or via a HTTP proxy.

FTP Active: Install from an FTP server

This option makes all FTP transfers use “Active” mode. This will not work through firewalls, but will often work with older FTP servers that do not support passive mode. If the connection hangs with passive mode (the default), try using active mode.

FTP Passive: Install from an FTP server through a firewall

This option instructs sysinstall(8) to use passive mode for all FTP operations. This allows the user to pass through firewalls that do not allow incoming connections on random TCP ports.

FTP via a HTTP proxy: Install from an FTP server through a http proxy

This option instructs sysinstall(8) to use the HTTP protocol to connect to a proxy for all FTP operations. The proxy will translate the requests and send them to the FTP server. This allows the user to pass through firewalls that do not allow FTP, but offer a HTTP proxy. In this case, specify the proxy in addition to the FTP server.

For a proxy FTP server, give the name of the server as part of the username, after an “@” sign. The proxy server then “fakes” the real server. For example, to install from `ftp.FreeBSD.org`, using the proxy FTP server `foo.example.com`, listening on port 1234, go to the options menu, set the FTP username to

`ftp@ftp.FreeBSD.org` and the password to an email address. As the installation media, specify FTP (or passive FTP, if the proxy supports it), and the URL `ftp://foo.example.com:1234/pub/FreeBSD`.

Since `/pub/FreeBSD` from `ftp.FreeBSD.org` is proxied under `foo.example.com`, the proxy will fetch the files from `ftp.FreeBSD.org` as the installer requests them.

3.9 Committing to the Installation

The installation can now proceed if desired. This is also the last chance for aborting the installation to prevent changes to the hard drive.

```
                      User Confirmation Requested
 Last Chance! Are you SURE you want to continue the installation?

 If you're running this on a disk with data you wish to save then WE
 STRONGLY ENCOURAGE YOU TO MAKE PROPER BACKUPS before proceeding!

 We can take no responsibility for lost disk contents!

                             [ Yes ]    No
```

Select [Yes] and press **Enter** to proceed.

The installation time will vary according to the distribution chosen, installation media, and the speed of the computer. There will be a series of messages displayed, indicating the status.

The installation is complete when the following message is displayed:

```
                                Message

Congratulations! You now have FreeBSD installed on your system.

We will now move on to the final configuration questions.
For any option you do not wish to configure, simply select No.

If you wish to re-enter this utility after the system is up, you may
do so by typing: /usr/sbin/sysinstall.

                                 [ OK ]

                       [  Press enter or space  ]
```

Press **Enter** to proceed with post-installation configurations.

Selecting [No] and pressing **Enter** will abort the installation so no changes will be made to the system. The following message will appear:

```
                                Message
Installation complete with some errors.  You may wish to scroll
through the debugging messages on VTY1 with the scroll-lock feature.
You can also choose "No" at the next prompt and go back into the
installation menus to retry whichever operations have failed.
```

```
[ OK ]
```

This message is generated because nothing was installed. Pressing **Enter** will return to the Main Installation Menu to exit the installation.

3.10 Post-installation

Configuration of various options can be performed after a successful installation. An option can be configured by re-entering the configuration menus before booting the new FreeBSD system or after boot using sysinstall(8) and then selecting the Configure menu.

3.10.1 Network Device Configuration

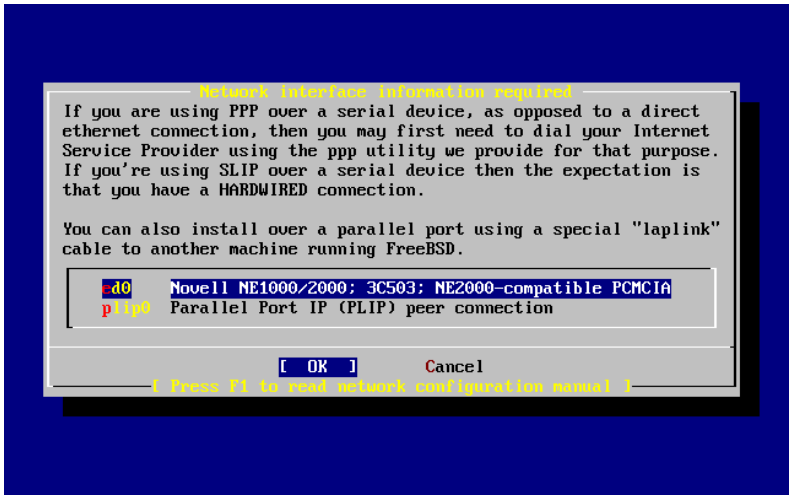
If PPP was previously configured for an FTP install, this screen will not display and can be configured after boot as described above.

For detailed information on Local Area Networks and configuring FreeBSD as a gateway/router refer to the Advanced Networking chapter.

```
                    User Confirmation Requested
  Would you like to configure any Ethernet or PPP network devices?

                              [ Yes ]   No
```

To configure a network device, select [Yes] and press **Enter**. Otherwise, select [No] to continue.

Figure 3-28. Selecting an Ethernet Device



Select the interface to be configured with the arrow keys and press **Enter**.

```
                    User Confirmation Requested
        Do you want to try IPv6 configuration of the interface?
```

```
                    Yes   [ No ]
```

In this private local area network, the current Internet type protocol (IPv4) was sufficient and [No] was selected with the arrow keys and **Enter** pressed.

If connected to an existing IPv6 network with an RA server, choose [Yes] and press **Enter**. It will take several seconds to scan for RA servers.

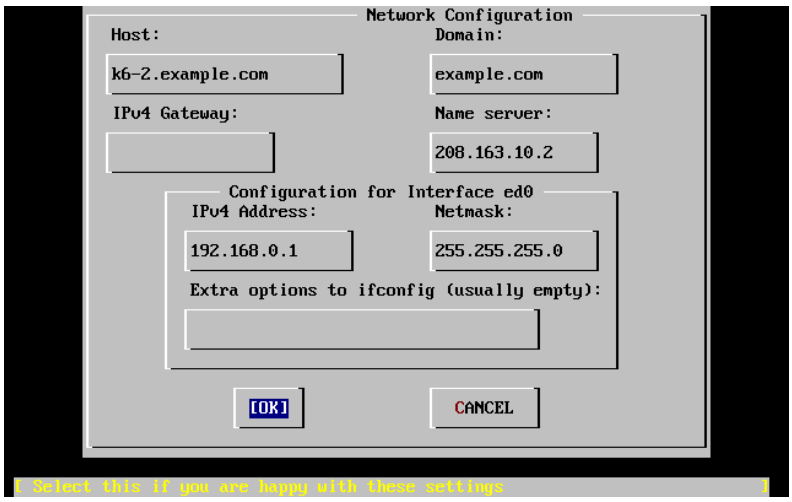
```
                    User Confirmation Requested
Do you want to try DHCP configuration of the interface?

                    Yes   [ No ]
```

If Dynamic Host Configuration Protocol DHCP) is not required, select [No] with the arrow keys and press **Enter**.

Selecting [Yes] will execute dhclient(8) and, if successful, will fill in the network configuration information automatically. Refer to Section 30.6 for more information.

The following Network Configuration screen shows the configuration of the Ethernet device for a system that will act as the gateway for a Local Area Network.

Figure 3-29. Set Network Configuration for *ed0*



Use **Tab** to select the information fields and fill in appropriate information:

Host

The fully-qualified hostname, such as `k6-2.example.com` in this case.

Domain

The name of the domain that the machine is in, such as `example.com` for this case.

IPv4 Gateway

IP address of host forwarding packets to non-local destinations. This must be filled in if the machine is a node on the network. *Leave this field blank* if the machine is the gateway to the Internet for the network. The IPv4

Gateway is also known as the default gateway or default route.

Name server

IP address of the local DNS server. There is no local DNS server on this private local area network so the IP address of the provider's DNS server (`208.163.10.2`) was used.

IPv4 address

The IP address to be used for this interface was `192.168.0.1`

Netmask

The address block being used for this local area network is `192.168.0.0` - `192.168.0.255` with a netmask of `255.255.255.0`.

Extra options to ifconfig(8)

Any additional interface-specific options to ifconfig(8). There were none in this case.

Use **Tab** to select [OK] when finished and press **Enter**.

```
                    User Confirmation Requested
        Would you like to bring the ed0 interface up right now?

                              [ Yes ]   No
```

Choosing [Yes] and pressing **Enter** will bring the machine up on the network so it is ready for use. However, this does not accomplish much during installation, since the machine still needs to be rebooted.

3.10.2 Configure Gateway

```
                     User Confirmation Requested
      Do you want this machine to function as a network gateway?

                              [ Yes ]    No
```

If the machine will be acting as the gateway for a local area network and forwarding packets between other machines, select [Yes] and press **Enter**. If the machine is a node on a network, select [No] and press **Enter** to continue.

3.10.3 Configure Internet Services

```
                      User Confirmation Requested
Do you want to configure inetd and the network services that it provides?

                               Yes   [ No ]
```

If [No] is selected, various services will not be enabled. These services can be enabled after installation by editing `/etc/inetd.conf` with a text editor. See Section 30.2.1 for more information.

Otherwise, select [Yes] to configure these services during install. An additional confirmation will display:

```
                      User Confirmation Requested
```

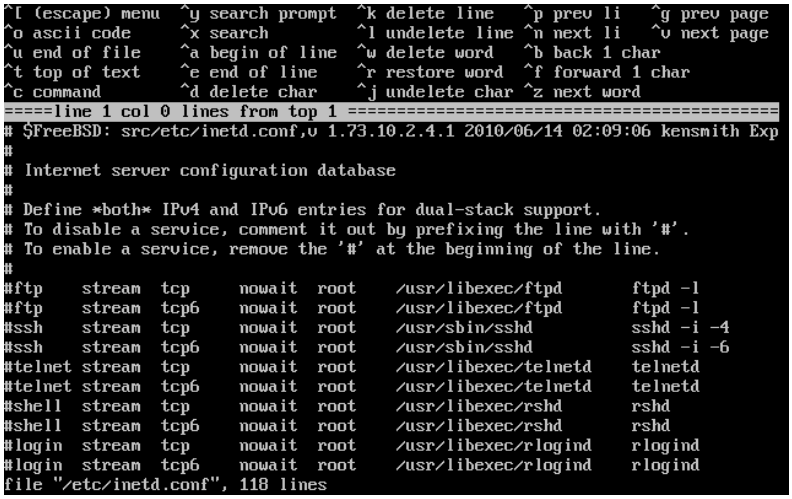
```
The Internet Super Server (inetd) allows a number of simple Internet
services to be enabled, including finger, ftp and telnetd.  Enabling
these services may increase risk of security problems by increasing
the exposure of your system.

With this in mind, do you wish to enable inetd?

                                [ Yes ]   No
```

Select [Yes] to continue.

```
                        User Confirmation Requested
inetd(8) relies on its configuration file, /etc/inetd.conf, to determine
which of its Internet services will be available.  The default FreeBSD
inetd.conf(5) leaves all services disabled by default, so they must be
specifically enabled in the configuration file before they will
function, even once inetd(8) is enabled.  Note that services for
IPv6 must be separately enabled from IPv4 services.

Select [Yes] now to invoke an editor on /etc/inetd.conf, or [No] to
use the current settings.

                                [ Yes ]   No
```

Selecting [Yes] allows services to be enabled by deleting the # at the beginning of the lines representing those services.

Figure 3-30. Editing `inetd.conf`



Once the edits are complete, press **Esc** to display a menu which will exit the editor and save the changes.

3.10.4 Enabling SSH Login

```
                        User Confirmation Requested
```

```
Would you like to enable SSH login?
         Yes        [  No  ]
```

Selecting [Yes] will enable sshd(8), the daemon for **OpenSSH**. This allows secure remote access to the machine. For more information about **OpenSSH**, see Section 15.10.

3.10.5 Anonymous FTP

```
                    User Confirmation Requested
Do you want to have anonymous FTP access to this machine?

                             Yes    [ No ]
```

3.10.5.1 Deny Anonymous FTP

Selecting the default [No] and pressing **Enter** will still allow users who have accounts with passwords to use FTP to access the machine.

3.10.5.2 Allow Anonymous FTP

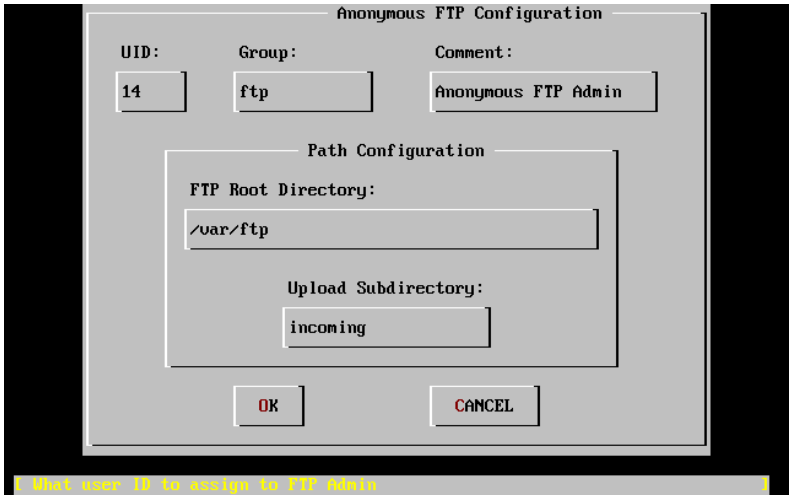
Anyone can access the machine if anonymous FTP connections are allowed. The security implications should be considered before enabling this option. For more information about security, see Chapter 15.

To allow anonymous FTP, use the arrow keys to select [Yes] and press **Enter**. An additional confirmation will display:

```
                      User Confirmation Requested
Anonymous FTP permits un-authenticated users to connect to the system
FTP server, if FTP service is enabled.  Anonymous users are
restricted to a specific subset of the file system, and the default
configuration provides a drop-box incoming directory to which uploads
are permitted.  You must separately enable both inetd(8), and enable
ftpd(8) in inetd.conf(5) for FTP services to be available.  If you
did not do so earlier, you will have the opportunity to enable inetd(8)
again later.

If you want the server to be read-only you should leave the upload
directory option empty and add the -r command-line option to ftpd(8)
in inetd.conf(5)

Do you wish to continue configuring anonymous FTP?

                         [ Yes ]         No
```

This message indicates that the FTP service will also have to be enabled in `/etc/inetd.conf` to allow anonymous FTP connections. Select [Yes] and press **Enter** to continue. The following screen will display:

Figure 3-31. Default Anonymous FTP Configuration



Use **Tab** to select the information fields and fill in appropriate information:

UID

The user ID to assign to the anonymous FTP user. All files uploaded will be owned by this ID.

Group

Which group to place the anonymous FTP user into.

Comment

String describing this user in `/etc/passwd`.

FTP Root Directory

Where files available for anonymous FTP will be kept.

Upload Subdirectory

Where files uploaded by anonymous FTP users will go.

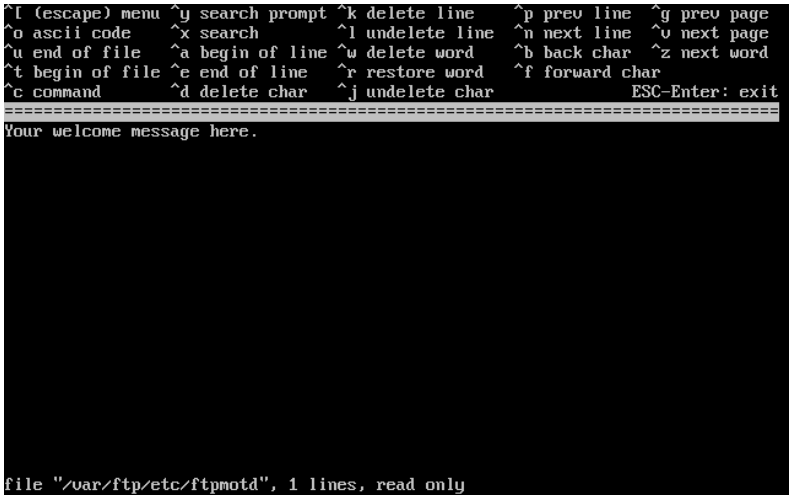
The FTP root directory will be put in `/var` by default. If there is not enough room there for the anticipated FTP needs, use `/usr` instead by setting the FTP root directory to `/usr/ftp`.

Once satisfied with the values, press **Enter** to continue.

```
                    User Confirmation Requested
        Create a welcome message file for anonymous FTP users?

                              [ Yes ]    No
```

If [Yes] is selected, press **Enter** and the cu(1) editor will automatically start.

Figure 3-32. Edit the FTP Welcome Message



Use the instructions to change the message. Note the file name location at the bottom of the editor screen.

Press **Esc** and a pop-up menu will default to a) leave editor. Press **Enter** to exit and continue. Press **Enter** again to save any changes.

3.10.6 Configure the Network File System

The Network File System (NFS) allows sharing of files across a network. A machine can be configured as a server, a client, or both. Refer to Section 30.3 for more information.

3.10.6.1 NFS Server

```
                    User Confirmation Requested
 Do you want to configure this machine as an NFS server?

                                Yes    [ No ]
```

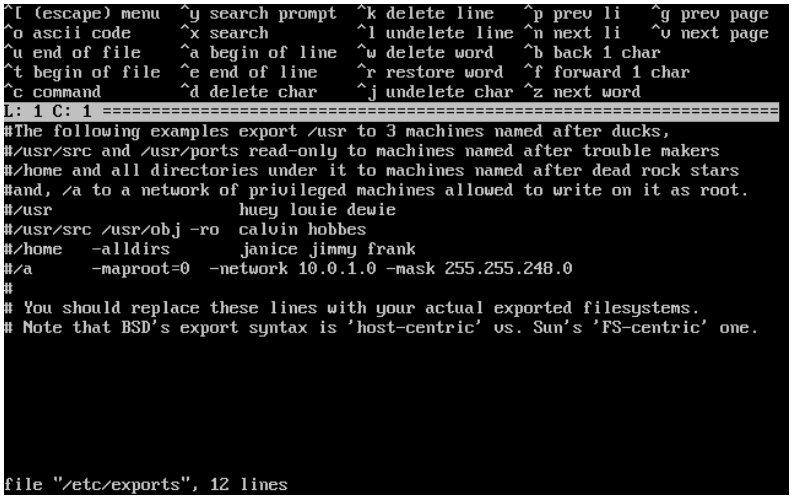
If there is no need for a NFS server, select [No] and press **Enter**.

If [Yes] is chosen, a message will pop-up indicating that `/etc/exports` must be created.

```
                                Message
Operating as an NFS server means that you must first configure an
/etc/exports file to indicate which hosts are allowed certain kinds of
access to your local filesystems.
Press [Enter] now to invoke an editor on /etc/exports
                                [ OK ]
```

Press **Enter** to continue. A text editor will start, allowing `/etc/exports` to be edited.

Figure 3-33. Editing `exports`



Use the instructions to add the exported filesystems. Note the file name location at the bottom of the editor screen.

Press **Esc** and a pop-up menu will default to a) leave editor. Press **Enter** to exit and continue.

3.10.6.2 NFS Client

The NFS client allows the machine to access NFS servers.

```
                        User Confirmation Requested
 Do you want to configure this machine as an NFS client?

                                  Yes   [ No ]
```

With the arrow keys, select [Yes] or [No] as appropriate and press **Enter**.

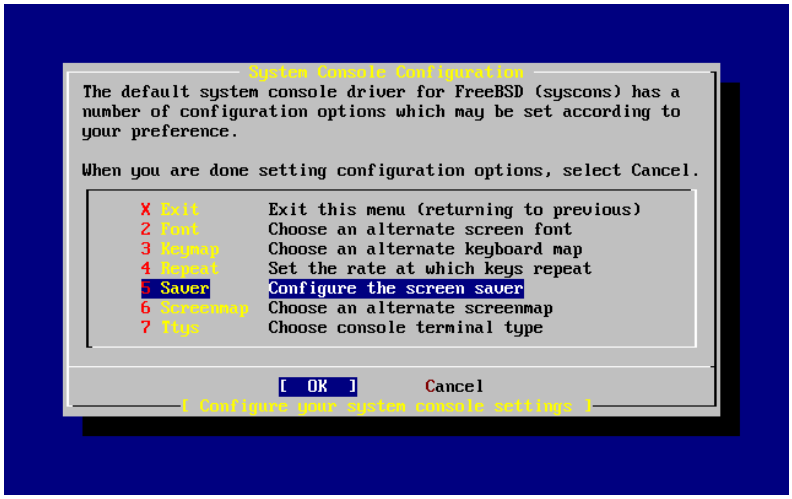
3.10.7 System Console Settings

There are several options available to customize the system console.

```
                      User Confirmation Requested
       Would you like to customize your system console settings?

                                  [ Yes ]  No
```

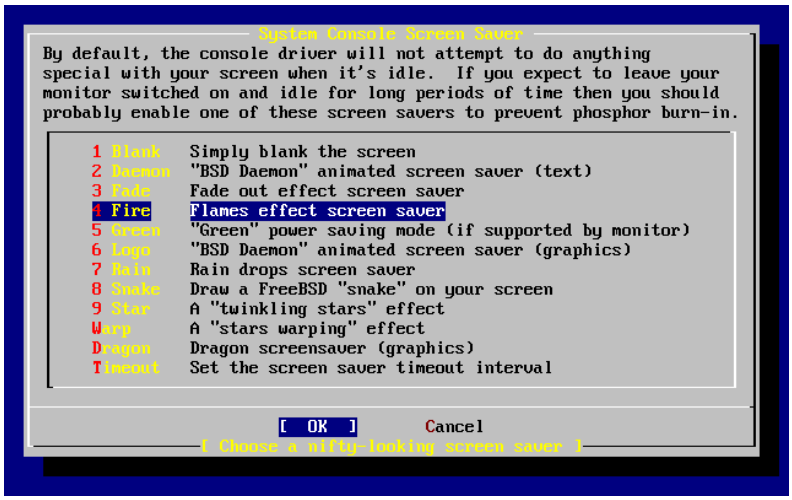
To view and configure the options, select [Yes] and press **Enter**.

Figure 3-34. System Console Configuration Options



A commonly used option is the screen saver. Use the arrow keys to select Saver and then press **Enter**.

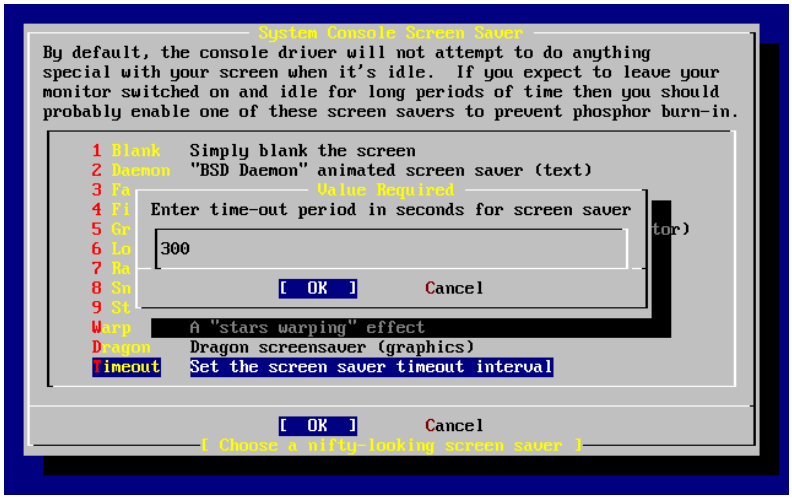
Figure 3-35. Screen Saver Options



Select the desired screen saver using the arrow keys and then press **Enter**. The System Console Configuration menu will redisplay.

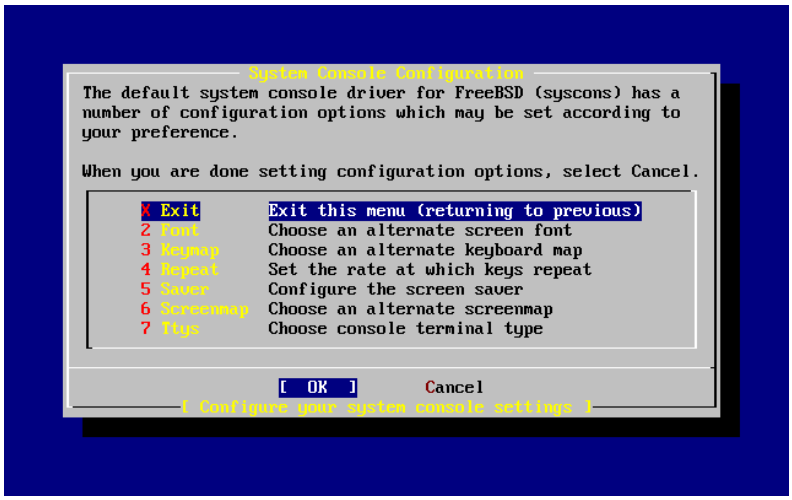
The default time interval is 300 seconds. To change the time interval, select Saver again. At the Screen Saver Options menu, select Timeout using the arrow keys and press **Enter**. A pop-up menu will appear:

Figure 3-36. Screen Saver Timeout



The value can be changed, then select [OK] and press **Enter** to return to the System Console Configuration menu.

Figure 3-37. System Console Configuration Exit



Select Exit and press **Enter** to continue with the post-installation configuration.

3.10.8 Setting the Time Zone

Setting the time zone allows the system to automatically correct for any regional time changes and perform other time zone related functions properly.

The example shown is for a machine located in the Eastern time zone of the United States. The selections will vary according to the geographic location.

```
User Confirmation Requested
```

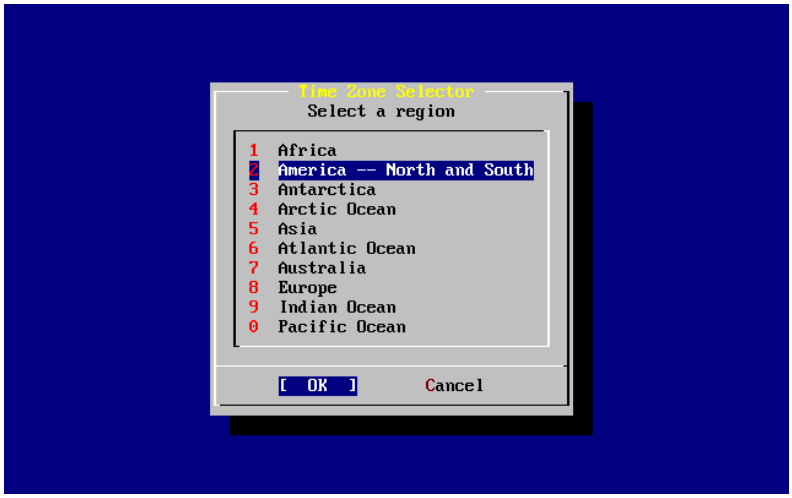
```
          Would you like to set this machine's time zone now?

                                [ Yes ]   No
```

Select [Yes] and press **Enter** to set the time zone.

```
                          User Confirmation Requested
 Is this machine's CMOS clock set to UTC? If it is set to local time
 or you don't know, please choose NO here!

                                  Yes   [ No ]
```

Select [Yes] or [No] according to how the machine's clock is configured, then press **Enter**.

Figure 3-38. Select the Region

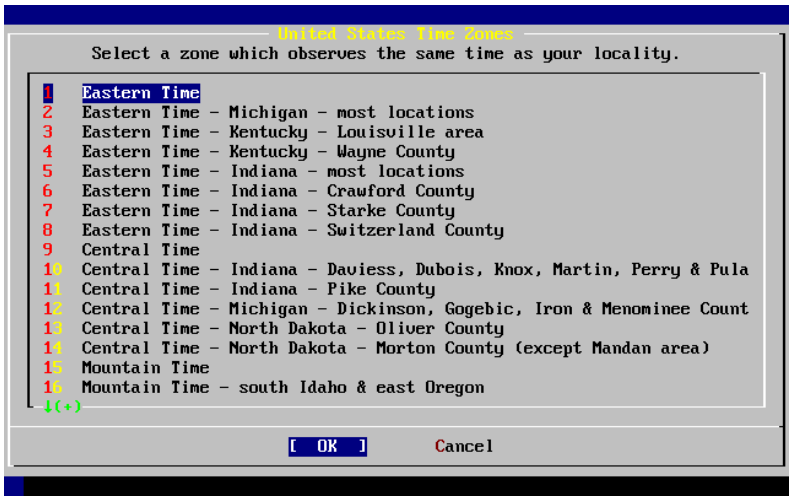


The appropriate region is selected using the arrow keys and then pressing **Enter**.

Figure 3-39. Select the Country

Select the appropriate country using the arrow keys and press **Enter**.

Figure 3-40. Select the Time Zone



The appropriate time zone is selected using the arrow keys and pressing **Enter**.

```
                    Confirmation
Does the abbreviation 'EDT' look reasonable?

                    [ Yes ]   No
```

Confirm that the abbreviation for the time zone is correct. If it looks okay, press **Enter** to continue with the post-installation configuration.

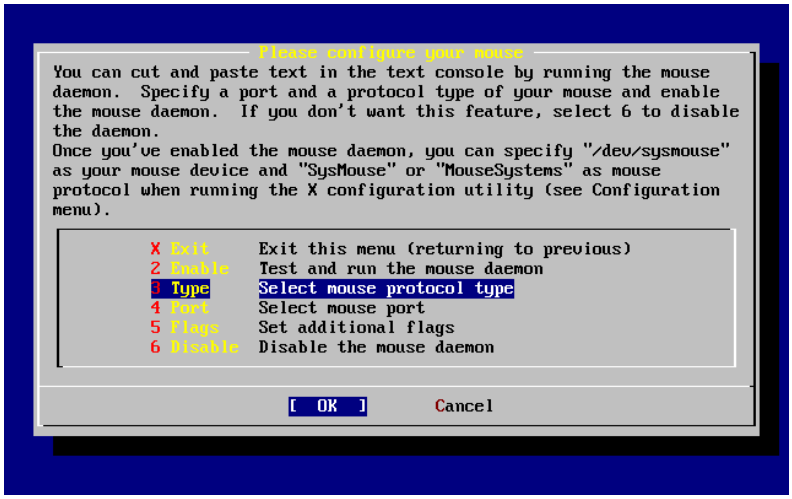
3.10.9 Mouse Settings

This option allows cut and paste in the console and user programs using a 3-button mouse. If using a 2-button mouse, refer to moused(8) for details on emulating the 3-button style. This example depicts a non-USB mouse configuration:

```
                    User Confirmation Requested
        Does this system have a PS/2, serial, or bus mouse?

                              [ Yes ]    No
```

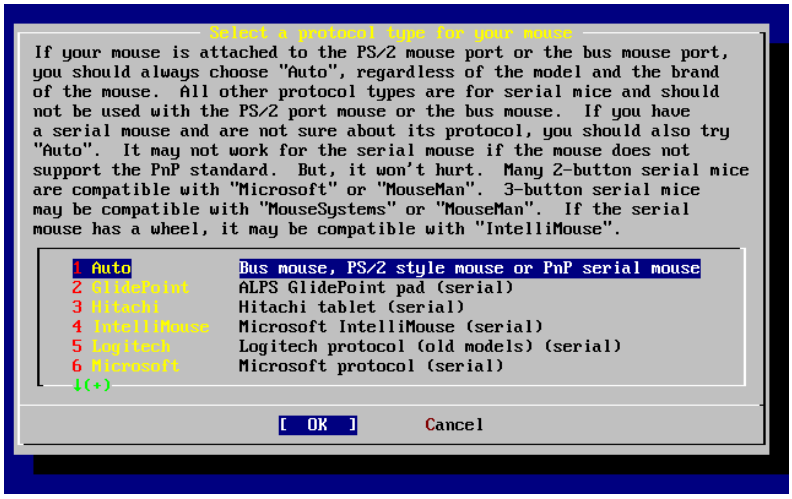
Select [Yes] for a PS/2, serial, or bus mouse, or [No] for a USB mouse, then press **Enter**.

Figure 3-41. Select Mouse Protocol Type



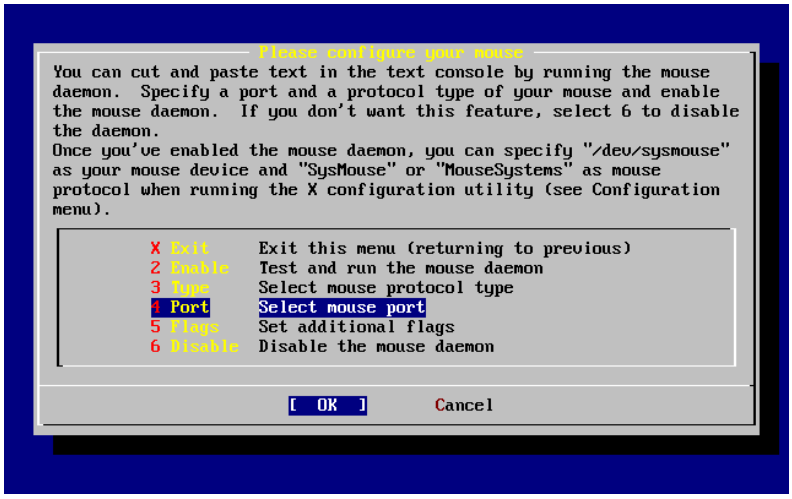
Use the arrow keys to select Type and press **Enter**.

Figure 3-42. Set Mouse Protocol



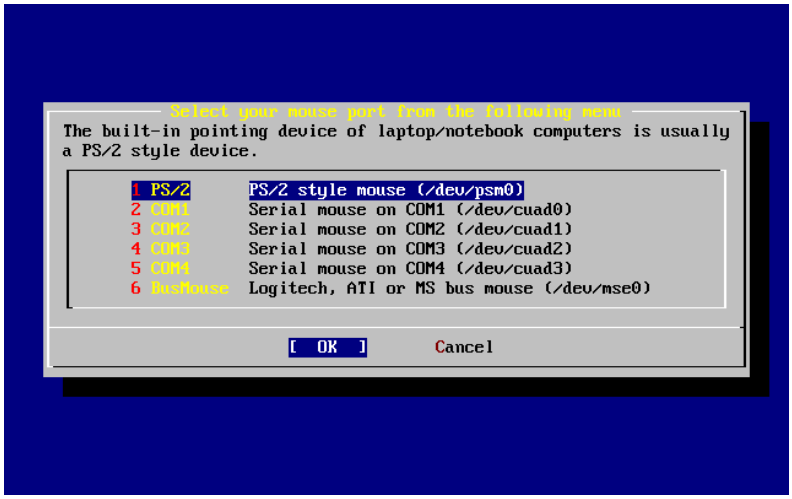
The mouse used in this example is a PS/2 type, so the default Auto is appropriate. To change the mouse protocol, use the arrow keys to select another option. Ensure that [OK] is highlighted and press **Enter** to exit this menu.

Figure 3-43. Configure Mouse Port



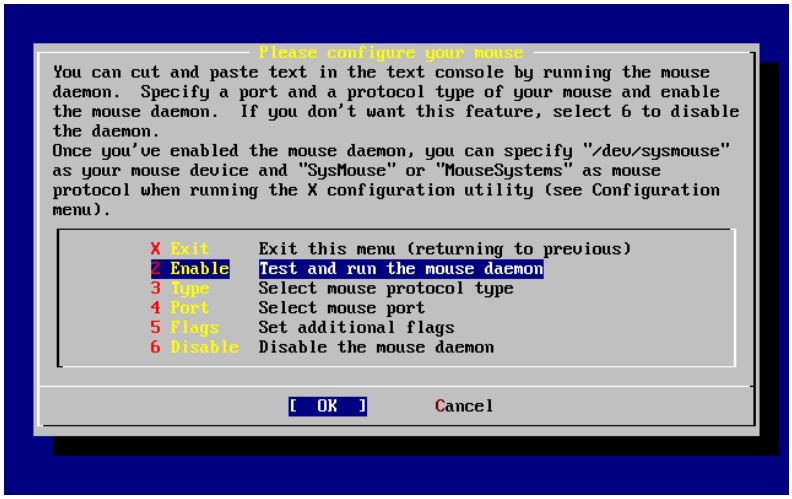
Use the arrow keys to select Port and press **Enter**.

Figure 3-44. Setting the Mouse Port



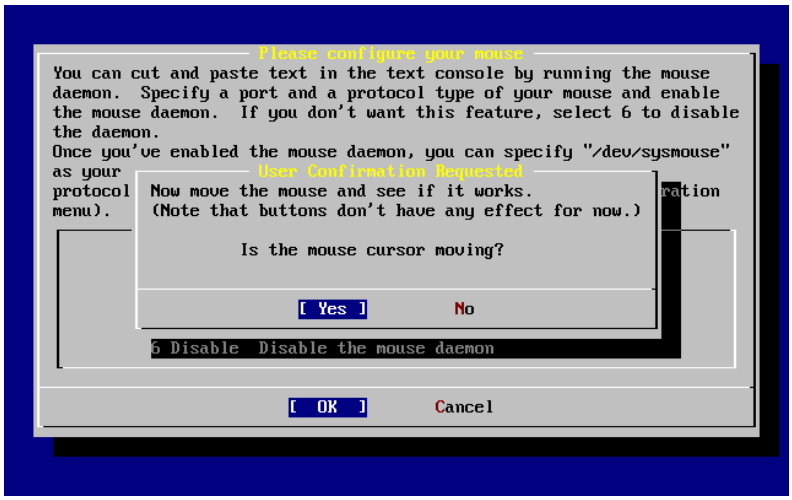
This system had a PS/2 mouse, so the default PS/2 is appropriate. To change the port, use the arrow keys and then press **Enter**.

Figure 3-45. Enable the Mouse Daemon



Last, use the arrow keys to select Enable, and press **Enter** to enable and test the mouse daemon.

Figure 3-46. Test the Mouse Daemon



Move the mouse around the screen to verify that the cursor responds properly. If it does, select [Yes] and press **Enter**. If not, the mouse has not been configured correctly. Select [No] and try using different configuration options.

Select Exit with the arrow keys and press **Enter** to continue with the post-installation configuration.

3.10.10 Install Packages

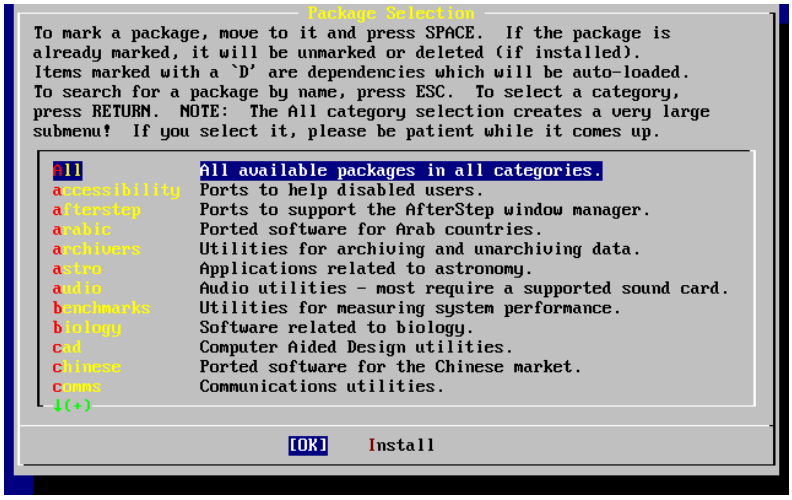
Packages are pre-compiled binaries and are a convenient way to install software.

Installation of one package is shown for purposes of illustration. Additional packages can also be added at this time if desired. After installation, sysinstall(8) can be used to add additional packages.

```
                    User Confirmation Requested
The FreeBSD package collection is a collection of hundreds of
ready-to-run applications, from text editors to games to WEB servers
and more. Would you like to browse the collection now?

                              [ Yes ]   No
```

Select [Yes] and press **Enter** to be presented with the Package Selection screens:

Figure 3-47. Select Package Category

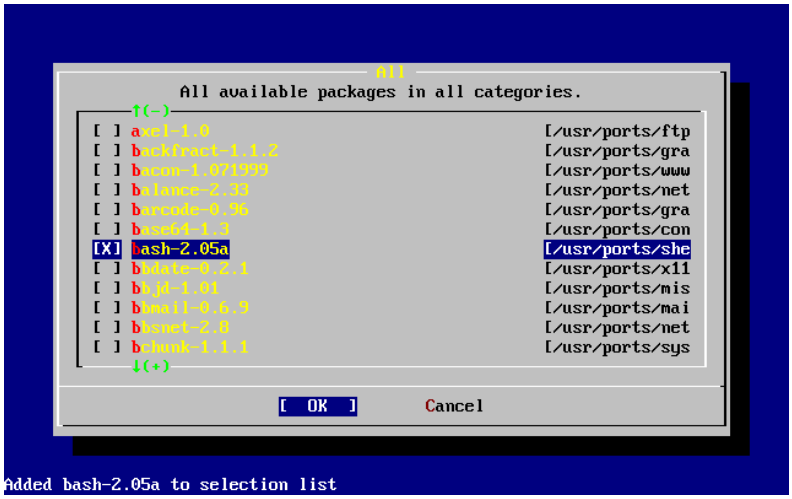


Only packages on the current installation media are available for installation at any given time.

All packages available will be displayed if All is selected. Otherwise, select a particular category. Highlight the selection with the arrow keys and press **Enter**.

A menu will display showing all the packages available for the selection made:

Figure 3-48. Select Packages



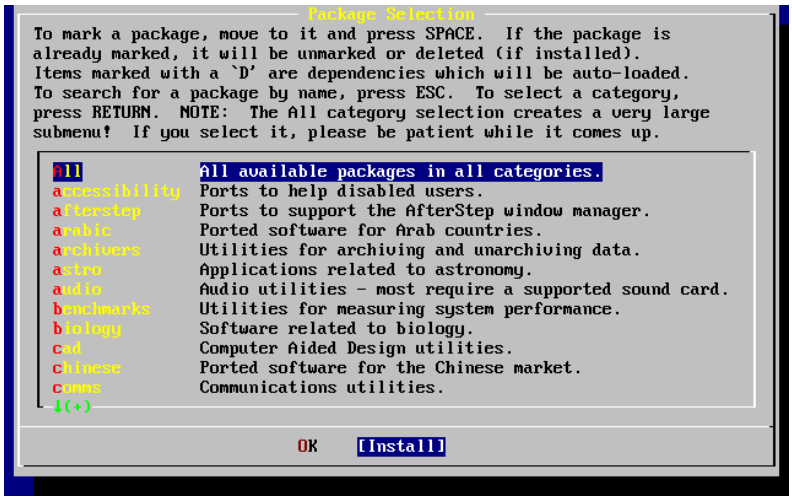
The **bash** shell is shown as selected. Select as many packages as desired by highlighting the package and pressing **Space**. A short description of each package will appear in the lower left corner of the screen.

Press **Tab** to toggle between the last selected package, [OK], and [Cancel].

Once finished marking the packages for installation, press **Tab** once to toggle to [OK] and press **Enter** to return to the Package Selection menu.

The left and right arrow keys will also toggle between [OK] and [Cancel]. This method can also be used to select [OK] and press **Enter** to return to the Package Selection menu.

Figure 3-49. Install Packages



Use the **Tab** and arrow keys to select [Install] and press **Enter** to see the installation confirmation message:

Figure 3-50. Confirm Package Installation

Select [OK] and press **Enter** to start the package installation. Installation messages will appear until all of the installations have completed. Make note if there are any error messages.

The final configuration continues after packages are installed. If no packages are selected, select Install to return to the final configuration.

3.10.11 Add Users/Groups

Add at least one user during the installation so that the system can be used without logging in as `root`. The root partition is generally small and running applications as `root` can quickly fill it. A bigger danger is noted below:

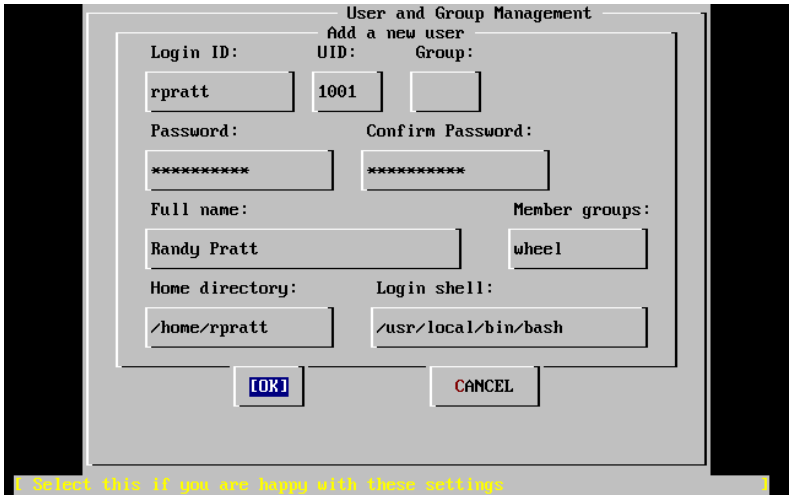
```
                    User Confirmation Requested
Would you like to add any initial user accounts to the system? Adding
at least one account for yourself at this stage is suggested since
working as the "root" user is dangerous (it is easy to do things which
adversely affect the entire system).

                              [ Yes ]   No
```

Select [Yes] and press **Enter** to continue with adding a user.

Figure 3-51. Select User

Select User with the arrow keys and press **Enter**.

Figure 3-52. Add User Information



The following descriptions will appear in the lower part of the screen as the items are selected with **Tab** to assist with entering the required information:

Login ID

The login name of the new user (mandatory).

UID

The numerical ID for this user (leave blank for automatic choice).

Group

The login group name for this user (leave blank for automatic choice).

Password

The password for this user (enter this field with care!).

Full name

The user’s full name (comment).

Member groups

The groups this user belongs to.

Home directory

The user’s home directory (leave blank for default).

Login shell

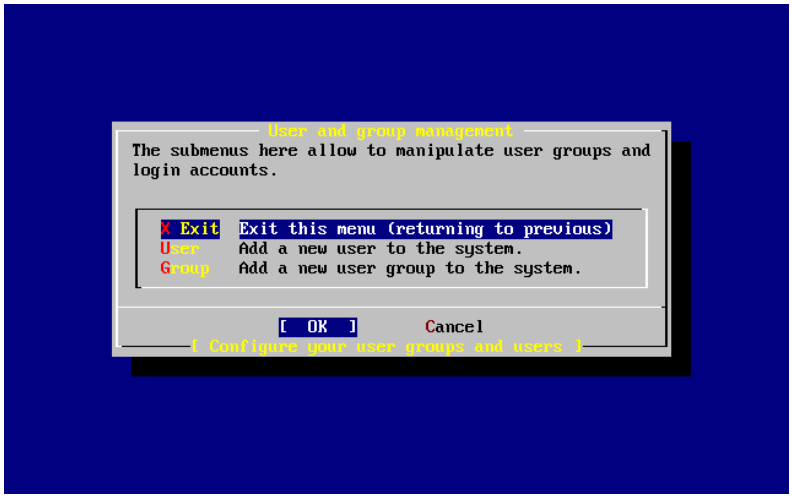
The user’s login shell (leave blank for default of `/bin/sh`).

In this example, the login shell was changed from `/bin/sh` to `/usr/local/bin/bash` to use the **bash** shell that was previously installed as a package. Do not use a shell that does not exist or the user will not be able to login. The most common shell used in FreeBSD is the C shell, `/bin/tcsh`.

The user was also added to the `wheel` group to be able to become a superuser with `root` privileges.

Once satisfied, press [OK] and the User and Group Management menu will redisplay:

Figure 3-53. Exit User and Group Management



Groups can also be added at this time. Otherwise, this menu may be accessed using sysinstall(8) at a later time.

When finished adding users, select Exit with the arrow keys and press **Enter** to continue the installation.

3.10.12 Set the `root` Password

```
                        Message
 Now you must set the system manager's password.
 This is the password you'll use to log in as "root".

                         [ OK ]

                [ Press enter or space ]
```

Press **Enter** to set the `root` password.

The password will need to be typed in twice correctly. Do not forget this password. Notice that the typed password is not echoed, nor are asterisks displayed.

```
New password:
Retype new password :
```

The installation will continue after the password is successfully entered.

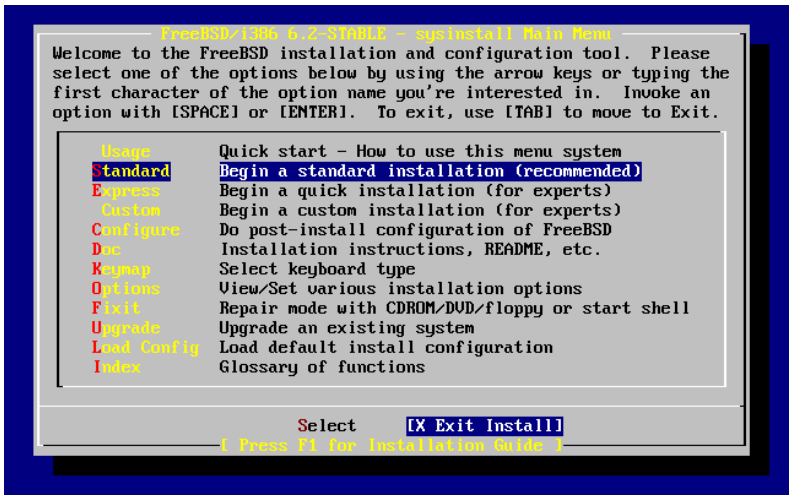
3.10.13 Exiting Install

A message will ask if configuration is complete:

```
                    User Confirmation Requested
Visit the general configuration menu for a chance to set any last
options?

                                Yes   [ No ]
```

Select [No] with the arrow keys and press **Enter** to return to the Main Installation Menu.

Figure 3-54. Exit Install



Select [X Exit Install] with the arrow keys and press **Enter**. The installer will prompt to confirm exiting the installation:

```
                    User Confirmation Requested
Are you sure you wish to exit? The system will reboot.

                              [ Yes ]   No
```

Select [Yes]. If booting from the CDROM drive, the following message will remind you to remove the disk:

```
                     Message
Be sure to remove the media from the drive.

                     [ OK ]
           [ Press enter or space ]
```

The CDROM drive is locked until the machine starts to reboot, then the disk can quickly be removed from the drive. Press [OK] to reboot.

The system will reboot so watch for any error messages that may appear, see Section 3.10.15 for more details.

3.10.14 Configure Additional Network Services

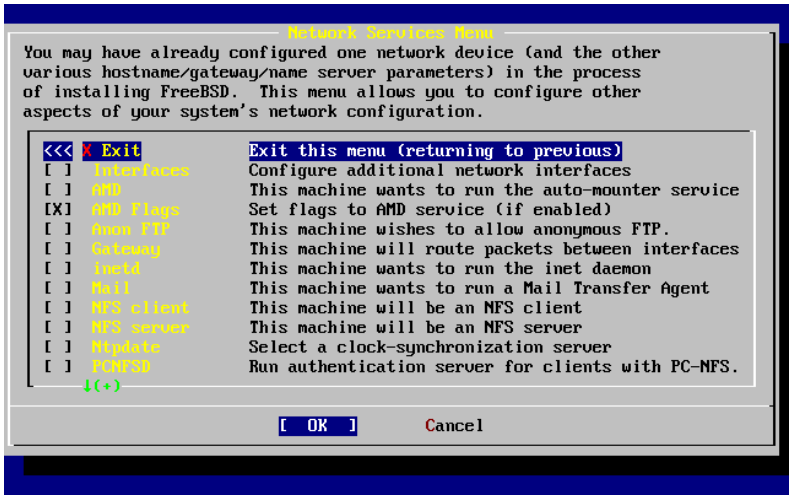
Contributed by Tom Rhodes.

Configuring network services can be a daunting task for users that lack previous knowledge in this area. Since networking and the Internet are critical to all modern operating systems, it is useful to have some understanding of FreeBSD's extensive networking capabilities.

Network services are programs that accept input from anywhere on the network. Since there have been cases where bugs in network services have been exploited by attackers, it is important to only enable needed network services. If in doubt, do not enable a network service until it is needed. Services can be enabled with sysinstall(8) or by editing `/etc/rc.conf`.

Selecting the Networking option will display a menu similar to the one below:

Figure 3-55. Network Configuration Upper-level



The first option, Interfaces, is covered in Section 3.10.1.

Selecting the AMD option adds support for amd(8). This is usually used in conjunction with NFS for automatically mounting remote filesystems.

Next is the AMD Flags option. When selected, a menu will pop up where specific AMD flags can be entered. The menu already contains a set of default options:

```
-a /.amd_mnt -l syslog /host /etc/amd.map /net /etc/amd.map
```

`-a` sets the default mount location which is specified here as `/.amd_mnt`. `-l` specifies the default `log`; however, when syslogd(8) is used, all log activity will be sent to the system log daemon. `/host` is used to mount an exported file system from a remote host, while `/net` is used to mount an exported filesystem from an IP address. The default options for AMD exports are defined in `/etc/amd.map`.

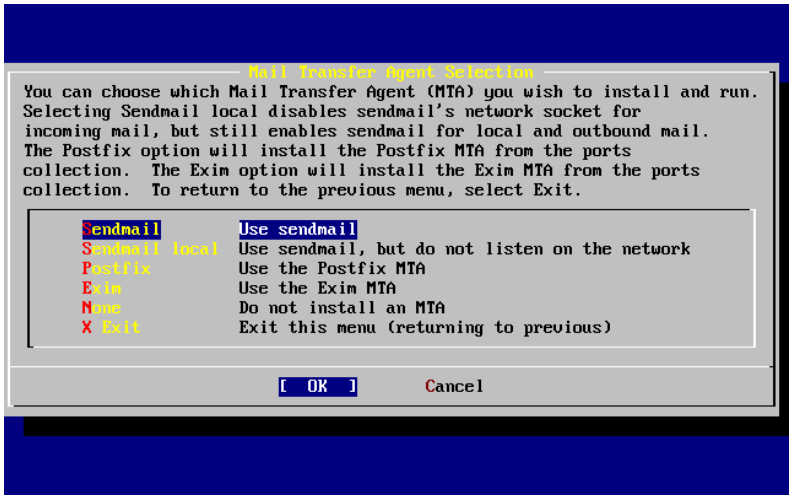
The Anon FTP option permits anonymous FTP connections. Select this option to make this machine an anonymous FTP server. Be aware of the security risks involved with this option. Another menu will be displayed to explain the security risks and configuration in depth.

The Gateway menu will configure the machine to be a gateway. This menu can also be used to unset the Gateway option if it was accidentally selected during installation.

The Inetd option can be used to configure or completely disable inetd(8).

The Mail option is used to configure the system's default Mail Transfer Agent (MTA). Selecting this option will bring up the following menu:

Figure 3-56. Select a Default MTA



This menu offers a choice as to which MTA to install and set as the default. An MTA is a mail server which delivers email to users on the system or the Internet.

Select Sendmail to install **Sendmail** as the default MTA. Select Sendmail local to set **Sendmail** as the default MTA, but disable its ability to receive incoming email from the Internet. The other options, Postfix and Exim, provide alternatives to **Sendmail**.

The next menu after the MTA menu is NFS client. This menu is used to configure the system to communicate with a NFS server which in turn is used to make filesystems available to other machines on the network over the NFS protocol. See Section 30.3 for more information about client and server configuration.

Below that option is the NFS server option, for setting the system up as an NFS server. This adds the required information to start up the Remote Procedure Call RPC services. RPC is used to coordinate connections between hosts and programs.

Next in line is the Ntpdate option, which deals with time synchronization. When selected, a menu like the one below shows up:

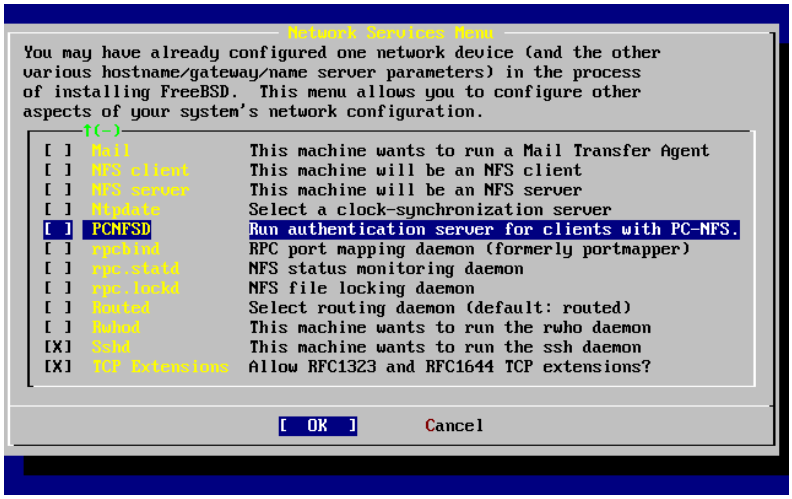
Figure 3-57. Ntpdate Configuration

From this menu, select the server which is geographically closest. This will make the time synchronization more accurate as a farther server may have more connection latency.

The next option is the PCNFSD selection. This option will install the `net/pcnfsd` package from the Ports Collection. This is a useful utility which provides NFS authentication services for systems which are unable to provide their own, such as Microsoft's MS-DOS operating system.

Now, scroll down a bit to see the other options:

Figure 3-58. Network Configuration Lower-level



RPC. communication between NFS servers and clients is managed by rpcbind(8) which is required for NFS servers to operate correctly. Status monitoring is provided by rpc.statd(8) and the reported status is usually held in `/var/db/statd.status`. The next option is for rpc.lockd(8) which provides file locking services. This is usually used with rpc.statd(8) to monitor which hosts are requesting locks and how frequently they request them. While these last two options are useful for debugging, they are not required for NFS servers and clients to operate correctly.

The next menu, Routed, configures the routing daemon. routed(8), manages network routing tables, discovers multicast routers, and supplies a copy of the routing tables to any physically connected host on the network upon request. This is mainly used for machines which act as a gateway for the local network. If selected, a menu will request the default location of the utility. To accept the default location, press **Enter**. Yet another menu will ask for the flags to pass to routed(8). The default of `-q` should appear on the screen.

The next menu, Rwhod, starts rwhod(8) during system initialization. This utility broadcasts system messages across the network periodically, or collects them when in "consumer" mode. More information can be found in ruptime(1) and rwho(1).

The next to last option in the list is for sshd(8), the secure shell server for **OpenSSH**. It is highly recommended over the standard telnetd(8) and ftpd(8) servers as it is used to create a secure, encrypted connection from one host to another.

The final option is TCP Extensions which are defined in RFC 1323 and RFC 1644. While on many hosts this can speed up connections, it can also cause some connections to be dropped. It is not recommended for servers, but may be beneficial for stand alone machines.

Once the network services are configured, scroll up to the very top item which is X Exit and continue on to the next configuration item or simply exit sysinstall(8) by selecting X Exit twice then [X Exit Install].

3.10.15 FreeBSD Bootup

3.10.15.1 FreeBSD/i386 Bootup

If everything went well, messages will scroll along the screen and a login prompt will appear. To view these messages, press **Scroll-Lock** then use **PgUp** and **PgDn**. Press **Scroll-Lock** again to return to the prompt.

All of the messages may not display due to buffer limitations, but they can be read after logging using dmesg(8).

Login using the username and password which were set during installation. Avoid logging in as `root` except when necessary.

Typical boot messages (version information omitted):

```
Copyright (c) 1992-2002 The FreeBSD Project.
Copyright (c) 1979, 1980, 1983, 1986, 1988, 1989, 1991, 1992, 1993, 1994
        The Regents of the University of California. All rights reserved.

Timecounter "i8254"  frequency 1193182 Hz
CPU: AMD-K6(tm) 3D processor (300.68-MHz 586-class CPU)
  Origin = "AuthenticAMD"  Id = 0x580  Stepping = 0
  Features=0x8001bf<FPU,VME,DE,PSE,TSC,MSR,MCE,CX8,MMX>
  AMD Features=0x80000800<SYSCALL,3DNow!>
real memory  = 268435456 (262144K bytes)
config> di sn0
config> di lnc0
config> di le0
config> di ie0
config> di fe0
config> di cs0
config> di bt0
config> di aic0
```

```
config> di aha0
config> di adv0
config> q
avail memory = 256311296 (250304K bytes)
Preloaded elf kernel "kernel" at 0xc0491000.
Preloaded userconfig_script "/boot/kernel.conf" at 0xc049109c.
md0: Malloc disk
Using $PIR table, 4 entries at 0xc00fde60
npx0: <math processor> on motherboard
npx0: INT 16 interface
pcib0: <Host to PCI bridge> on motherboard
pci0: <PCI bus> on pcib0
pcib1: <VIA 82C598MVP (Apollo MVP3) PCI-PCI (AGP) bridge> at device 1.0 on pci0
pci1: <PCI bus> on pcib1
pci1: <Matrox MGA G200 AGP graphics accelerator> at 0.0 irq 11
isab0: <VIA 82C586 PCI-ISA bridge> at device 7.0 on pci0
isa0: <ISA bus> on isab0
atapci0: <VIA 82C586 ATA33 controller> port 0xe000-0xe00f at device 7.1 on pci0
ata0: at 0x1f0 irq 14 on atapci0
ata1: at 0x170 irq 15 on atapci0
uhci0: <VIA 83C572 USB controller> port 0xe400-0xe41f irq 10 at device 7.2 on pci0
usb0: <VIA 83C572 USB controller> on uhci0
usb0: USB revision 1.0
uhub0: VIA UHCI root hub, class 9/0, rev 1.00/1.00, addr 1
uhub0: 2 ports with 2 removable, self powered
chip1: <VIA 82C586B ACPI interface> at device 7.3 on pci0
ed0: <NE2000 PCI Ethernet (RealTek 8029)> port 0xe800-0xe81f irq 9 at
device 10.0 on pci0
ed0: address 52:54:05:de:73:1b, type NE2000 (16 bit)
isa0: too many dependant configs (8)
isa0: unexpected small tag 14
fdc0: <NEC 72065B or clone> at port 0x3f0-0x3f5,0x3f7 irq 6 drq 2 on isa0
fdc0: FIFO enabled, 8 bytes threshold
fd0: <1440-KB 3.5" drive> on fdc0 drive 0
atkbdc0: <keyboard controller (i8042)> at port 0x60-0x64 on isa0
atkbd0: <AT Keyboard> flags 0x1 irq 1 on atkbdc0
kbd0 at atkbd0
psm0: <PS/2 Mouse> irq 12 on atkbdc0
psm0: model Generic PS/2 mouse, device ID 0
vga0: <Generic ISA VGA> at port 0x3c0-0x3df iomem 0xa0000-0xbffff on isa0
sc0: <System console> at flags 0x1 on isa0
sc0: VGA <16 virtual consoles, flags=0x300>
sio0 at port 0x3f8-0x3ff irq 4 flags 0x10 on isa0
sio0: type 16550A
sio1 at port 0x2f8-0x2ff irq 3 on isa0
sio1: type 16550A
ppc0: <Parallel port> at port 0x378-0x37f irq 7 on isa0
ppc0: SMC-like chipset (ECP/EPP/PS2/NIBBLE) in COMPATIBLE mode
ppc0: FIFO with 16/16/15 bytes threshold
ppbus0: IEEE1284 device found /NIBBLE
Probing for PnP devices on ppbus0:
plip0: <PLIP network interface> on ppbus0
lpt0: <Printer> on ppbus0
```

```
lpt0: Interrupt-driven port
ppi0: <Parallel I/O> on ppbus0
ad0: 8063MB <IBM-DHEA-38451> [16383/16/63] at ata0-master using UDMA33
ad2: 8063MB <IBM-DHEA-38451> [16383/16/63] at ata1-master using UDMA33
acd0: CDROM <DELTA OTC-H101/ST3 F/W by OIPD> at ata0-slave using PIO4
Mounting root from ufs:/dev/ad0s1a
swapon: adding /dev/ad0s1b as swap device
Automatic boot in progress...
/dev/ad0s1a: FILESYSTEM CLEAN; SKIPPING CHECKS
/dev/ad0s1a: clean, 48752 free (552 frags, 6025 blocks, 0.9% fragmentation)
/dev/ad0s1f: FILESYSTEM CLEAN; SKIPPING CHECKS
/dev/ad0s1f: clean, 128997 free (21 frags, 16122 blocks, 0.0% fragmentation)
/dev/ad0s1g: FILESYSTEM CLEAN; SKIPPING CHECKS
/dev/ad0s1g: clean, 3036299 free (43175 frags, 374073 blocks, 1.3% fragmentation)
/dev/ad0s1e: filesystem CLEAN; SKIPPING CHECKS
/dev/ad0s1e: clean, 128193 free (17 frags, 16022 blocks, 0.0% fragmentation)
Doing initial network setup: hostname.
ed0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        inet 192.168.0.1 netmask 0xffffff00 broadcast 192.168.0.255
        inet6 fe80::5054::5ff::fede:731b%ed0 prefixlen 64 tentative scopeid 0x1
        ether 52:54:05:de:73:1b
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
        inet6 fe80::1%lo0 prefixlen 64 scopeid 0x8
        inet6 ::1 prefixlen 128
        inet 127.0.0.1 netmask 0xff000000
Additional routing options: IP gateway=YES TCP keepalive=YES
routing daemons:.
additional daemons: syslogd.
Doing additional network setup:.
Starting final network daemons: creating ssh RSA host key
Generating public/private rsa1 key pair.
Your identification has been saved in /etc/ssh/ssh_host_key.
Your public key has been saved in /etc/ssh/ssh_host_key.pub.
The key fingerprint is:
cd:76:89:16:69:0e:d0:6e:f8:66:d0:07:26:3c:7e:2d root@k6-2.example.com
 creating ssh DSA host key
Generating public/private dsa key pair.
Your identification has been saved in /etc/ssh/ssh_host_dsa_key.
Your public key has been saved in /etc/ssh/ssh_host_dsa_key.pub.
The key fingerprint is:
f9:a1:a9:47:c4:ad:f9:8d:52:b8:b8:ff:8c:ad:2d:e6 root@k6-2.example.com.
setting ELF ldconfig path: /usr/lib /usr/lib/compat /usr/X11R6/lib
/usr/local/lib
a.out ldconfig path: /usr/lib/aout /usr/lib/compat/aout /usr/X11R6/lib/aout
starting standard daemons: inetd cron sshd usbd sendmail.
Initial rc.i386 initialization:.
rc.i386 configuring syscons: blank_time screensaver moused.
Additional ABI support: linux.
Local package initialization:.
Additional TCP options:.

FreeBSD/i386 (k6-2.example.com) (ttyv0)
```

```
login: rpratt
Password:
```

Generating the RSA and DSA keys may take some time on slower machines. This happens only on the initial boot-up of a new installation. Subsequent boots will be faster.

If **Xorg** has been configured and a default desktop chosen, it can be started by typing `startx` at the command line.

3.10.16 FreeBSD Shutdown

It is important to properly shutdown the operating system. Do not just turn off the power. First, become the superuser using su(1) and entering the `root` password. This will work only if the user is a member of `wheel`. Otherwise, login as `root`. To shutdown the system, type `shutdown -h now`.

```
The operating system has halted.
Please press any key to reboot.
```

It is safe to turn off the power after the shutdown command has been issued and the message “Please press any key to reboot” appears. If any key is pressed instead of turning off the power switch, the system will reboot.

The **Ctrl**+**Alt**+**Del** key combination can also be used to reboot the system; however, this is not recommended.

3.11 Troubleshooting

This section covers basic installation troubleshooting of common problems. There are also a few questions and answers for people wishing to dual-boot FreeBSD with Windows.

3.11.1 If Something Goes Wrong

Due to various limitations of the PC architecture, it is impossible for device probing to be 100% reliable. However, there are a few things to try if it fails.

Check the Hardware Notes (http://www.FreeBSD.org/releases/index.html) document for the version of FreeBSD to make sure the hardware is supported.

If the hardware is supported but still experiences lock-ups or other problems, build a custom kernel to add in support for devices which are not present in the `GENERIC` kernel. The default kernel assumes that most hardware devices are in their factory default configuration in terms of IRQs, I/O addresses, and DMA channels. If the hardware has been reconfigured, create a custom kernel configuration file and recompile to tell FreeBSD where to find things.

It is also possible that a probe for a device not present will cause a later probe for another device that is present to fail. In that case, the probes for the conflicting driver(s) should be disabled.

> **Note:** Some installation problems can be avoided or alleviated by updating the firmware on various hardware components, most notably the motherboard BIOS. Most motherboard and computer manufacturers have a website where upgrade information may be located.
>
> Most manufacturers strongly advise against upgrading the motherboard BIOS unless there is a good reason for doing so, such as a critical update. The upgrade process *can* go wrong, causing permanent damage to the BIOS chip.

3.11.2 Using Windows Filesystems

At this time, FreeBSD does not support file systems compressed with the **Double Space™** application. Therefore the file system will need to be uncompressed before FreeBSD can access the data. This can be done by running the **Compression Agent** located in the Start> Programs > System Tools menu.

FreeBSD can support MS-DOS file systems (sometimes called FAT file systems). The mount_msdosfs(8) command grafts such file systems onto the existing directory hierarchy, allowing the file system's contents to be accessed. The mount_msdosfs(8) program is not usually invoked directly; instead, it is called by the system through a line in `/etc/fstab` or by using mount(8) with the appropriate parameters.

A typical line in `/etc/fstab` is:

```
/dev/ad0sN  /dos  msdosfs rw  0 0
```

> **Note:** `/dos` must already exist for this to work. For details about the format of `/etc/fstab`, see fstab(5).

A typical call to mount(8) for a FAT filesystem looks like:

```
# mount -t msdosfs /dev/ad0s1 /mnt
```

In this example, the FAT filesystem is located on the first partition of the primary hard disk. The output from dmesg(8) and mount(8) should produce enough information to give an idea of the partition layout.

> **Note:** FreeBSD may number FAT partitions differently than other operating systems. In particular, extended partitions are usually given higher slice numbers than primary partitions. Use fdisk(8) to help determine which slices belong to FreeBSD and which belong to other operating systems.

NTFS partitions can also be mounted in a similar manner using mount_ntfs(8).

3.11.3 Troubleshooting Questions and Answers

1. My system hangs while probing hardware during boot or it behaves strangely during install.

FreeBSD makes extensive use of the system ACPI service on the i386, amd64, and ia64 platforms to aid in system configuration if it is detected during boot. Unfortunately, some bugs still exist in the ACPI driver and various system motherboards. The use of ACPI can be disabled by setting `hint.acpi.0.disabled` in the third stage boot loader:

```
set hint.acpi.0.disabled="1"
```

This is reset each time the system is booted, so it is necessary to add `hint.acpi.0.disabled="1"` to `/boot/loader.conf` to make this change permanent. More information about the boot loader can be found in Section 13.1.

2. When booting from the hard disk for the first time after installing FreeBSD, the kernel loads and probes hardware, but stops with messages like:

```
changing root device to ad1s1a panic: cannot mount root
```

What is wrong?

This can occur when the boot disk is not the first disk in the system. The BIOS uses a different numbering scheme to FreeBSD, and working out which numbers correspond to which is difficult to get right.

If this occurs, tell FreeBSD where the root filesystem is by specifying the BIOS disk number, the disk type, and the FreeBSD disk number for that type.

Consider two IDE disks, each configured as the master on their respective IDE bus, where FreeBSD should be booted from the second disk. The BIOS sees these as disk 0 and disk 1, while FreeBSD sees them as `ad0` and `ad2`.

If FreeBSD is on BIOS disk 1, of type `ad` and the FreeBSD disk number is 2, this is the correct value:

```
1:ad(2,a)kernel
```

Note that if there is a slave on the primary bus, the above is not necessary and is effectively wrong.

The second situation involves booting from a SCSI disk when there are one or more IDE disks in the system. In this case, the FreeBSD disk number is lower than the BIOS disk number. For two IDE disks and a SCSI disk, where the SCSI disk is BIOS disk 2, type `da`, and FreeBSD disk number 0, the correct value is:

```
2:da(0,a)kernel
```

This tells FreeBSD to boot from BIOS disk 2, which is the first SCSI disk in the system. If there is only IDE disk, use `1:` instead.

Once the correct value to use is determined, put the command in `/boot.config` using a text editor. Unless instructed otherwise, FreeBSD will use the contents of this file as the default response to the `boot:` prompt.

3. When booting from the hard disk for the first time after installing FreeBSD, the Boot Manager prompt just prints `F?` at the boot menu and the boot will not go any further.

The hard disk geometry was set incorrectly in the partition editor when FreeBSD was installed. Go back into the partition editor and specify the actual geometry of the hard disk. FreeBSD must be reinstalled again from the beginning with the correct geometry.

For a dedicated FreeBSD system that does not need future compatibility with another operating system, use the entire disk by selecting `A` in the installer's partition editor.

4. The system finds the ed(4) network card but continuously displays device timeout errors.

The card is probably on a different IRQ from what is specified in `/boot/device.hints`. The ed(4) driver does not use software configuration by default, but it will if `-1` is specified in the hints for the interface.

Either move the jumper on the card to the configuration setting or specify the IRQ as `-1` by setting the hint `hint.ed.0.irq="-1"`. This tells the kernel to use the software configuration.

Another possibility is that the card is at IRQ 9, which is shared by IRQ 2 and frequently a cause of problems, especially if a VGA card is using IRQ 2. Do not use IRQ 2 or 9 if at all possible.

5.

When sysinstall(8) is used in an **Xorg** terminal, the yellow font is difficult to read against the light gray background. Is there a way to provide higher contrast for this application?

If the default colors chosen by sysinstall(8) make text illegible while using `x11/xterm` or `x11/rxvt`, add the following to `~/.Xdefaults` to get a darker background gray: `XTerm*color7: #c0c0c0`

3.12 Advanced Installation Guide

Contributed by Valentino Vaschetto. Updated by Marc Fonvieille.

This section describes how to install FreeBSD in exceptional cases.

3.12.1 Installing FreeBSD on a System Without a Monitor or Keyboard

This type of installation is called a “headless install” because the machine to be installed does not have either an attached monitor or a VGA output. This type of installation is possible using a serial console, another machine which acts as the main display and keyboard. To do this, follow the steps to create an installation USB stick, explained in Section 3.3.7, or download the correct installation ISO image as described in Section 3.13.1.

To modify the installation media to boot into a serial console, follow these steps. If using a CD/DVD media, skip the first step):

1. Enabling the Installation USB Stick to Boot into a Serial Console

   By default, booting into the USB stick boots into the installer. To instead boot into a serial console, mount the USB disk onto a FreeBSD system using mount(8):

   ```
   # mount /dev/da0a /mnt
   ```

   Note: Adapt the device node and the mount point to the situation.

   Once the USB stick is mounted, set it to boot into a serial console. Add this line to `/boot/loader.conf` on the USB stick:

   ```
   # echo 'console="comconsole"' >> /mnt/boot/loader.conf
   ```

   Now that the USB is stick configured correctly, unmount the disk using umount(8):

   ```
   # umount /mnt
   ```

   Now, unplug the USB stick and jump directly to the third step of this procedure.

2. Enabling the Installation CD/DVD to Boot into a Serial Console

By default, when booting into the installation CD/DVD, FreeBSD boots into its normal install mode. To instead boot into a serial console, extract, modify, and regenerate the ISO image before burning it to the CD/DVD media.

From the FreeBSD system with the saved installation ISO image, use tar(1) to extract all the files:

```
# mkdir /path/to/headless-iso
# tar -C /path/to/headless-iso -pxvf FreeBSD-9.1-RELEASE-i386-disc1.iso
```

Next, set the installation media to boot into a serial console. Add this line to the `/boot/loader.conf` of the extracted ISO image:

```
# echo 'console="comconsole"' >> /path/to/headless-iso/boot/loader.conf
```

Then, create a new ISO image from the modified tree. This example uses mkisofs(8) from the `sysutils/cdrtools` package or port:

```
# mkisofs -v -b boot/cdboot -no-emul-boot -r -J -V "Headless_install" \
     -o Headless-FreeBSD-8.4-RELEASE-i386-disc1.iso/path/to/headless-iso
```

Now that the ISO image is configured correctly, burn it to a CD/DVD media using a burning application.

3. Connecting the Null-modem Cable

   Connect a null-modem cable to the serial ports of the two machines. *A normal serial cable will not work*. A null-modem cable is required.

4. Booting Up for the Install

   It is now time to go ahead and start the install. Plug in the USB stick or insert the CD/DVD media in the headless install machine and power it on.

5. Connecting to the Headless Machine

   Next, connect to that machine with cu(1):

   ```
   # cu -l /dev/cuau0
   ```

The headless machine can now be controlled using cu(1). It will load the kernel and then dispaly a selection of which type of terminal to use. Select the FreeBSD color console and proceed with the installation.

3.13 Preparing Custom Installation Media

Some situations may require a customized FreeBSD installation media and/or source. This might be physical media or a source that sysinstall(8) can use to retrieve the installation files. Some example situations include:

- A local network with many machines has a private FTP server hosting the FreeBSD installation files which the machines should use for installation.
- FreeBSD does not recognize the CD/DVD drive but Windows does. In this case, copy the FreeBSD installation files to a Windows partition on the same computer, and then install FreeBSD using those files.
- The computer to install does not have a CD/DVD drive or a network card, but can be connected using a null-printer cable to a computer that does.
- A tape will be be used to install FreeBSD.

3.13.1 Creating an Installation ISO

As part of each release, the FreeBSD Project provides ISO images for each supported architecture. These images can be written ("burned") to CD or DVD media using a burning application, and then used to install FreeBSD. If a CD/DVD writer is available, this is the easiest way to install FreeBSD.

1. Download the Correct ISO Images

   The ISO images for each release can be downloaded from `ftp://ftp.FreeBSD.org/pub/FreeBSD/ISO-IMAGES-`*`arch`*`/`*`version`* or the closest mirror. or the closest mirror. Substitute *`arch`* and *`version`* as appropriate.

   An image directory normally contains the following images:

   Table 3-4. FreeBSD ISO Image Names and Meanings

Filename	Contents
`FreeBSD-`*`version`*`-RELEASE-`*`arch`*`-bootonly.iso`	This CD image starts the installation process by booting from a CD-ROM drive but it does not contain the support for installing FreeBSD from the CD itself. Perform a network based install, such as from an FTP server, after booting from this CD.
`FreeBSD-`*`version`*`-RELEASE-`*`arch`*`-dvd1.iso.gz`	This DVD image contains everything necessary to install the base FreeBSD operating system, a collection of pre-built packages, and the documentation. It also supports booting into a "livefs" based rescue mode.
`FreeBSD-`*`version`*`-RELEASE-`*`arch`*`-memstick.img`	This image can be written to an USB memory stick in order to install machines capable of booting from USB drives. It also supports booting into a "livefs" based rescue mode. The only included package is the documentation package.
`FreeBSD-`*`version`*`-RELEASE-`*`arch`*`-disc1.iso`	This CD image contains the base FreeBSD operating system and the documentation package but no other packages.
`FreeBSD-`*`version`*`-RELEASE-`*`arch`*`-disc2.iso`	A CD image with as many third-party packages as would fit on the disc. This image is not available for FreeBSD 9.*X*.
`FreeBSD-`*`version`*`-RELEASE-`*`arch`*`-disc3.iso`	Another CD image with as many third-party packages as would fit on the disc. This image is not available for FreeBSD 9.*X*.
`FreeBSD-`*`version`*`-RELEASE-`*`arch`*`-livefs.iso`	This CD image contains support for booting into a "livefs" based rescue mode but does not support doing an install from the CD itself.

   When performing a CD installation, download either the `bootonly` ISO image or `disc1`. Do not download both, since `disc1` contains everything that the `bootonly` ISO image contains.

   Use the `bootonly` ISO to perform a network install over the Internet. Additional software can be installed as

needed using the Ports Collection as described in Chapter 5.

Use `dvd1` to install FreeBSD and a selection of third-party packages from the disc.

2. Burn the Media

   Next, write the downloaded image(s) to disc. If using another FreeBSD system, refer to Section 19.5.3 and Section 19.5.4 for instructions.

   If using another platform, use any burning utility that exists for that platform. The images are in the standard ISO format which most CD writing applications support.

   Note: To build a customized release of FreeBSD, refer to the Release Engineering Article (http://www.FreeBSD.org/doc/en_US.ISO8859-1/articles/releng).

3.13.2 Creating a Local FTP Site with a FreeBSD Disc

FreeBSD discs are laid out in the same way as the FTP site. This makes it easy to create a local FTP site that can be used by other machines on a network to install FreeBSD.

1. On the FreeBSD computer that will host the FTP site, ensure that the CD/DVD is in the drive and mounted:

   ```
   # mount /cdrom
   ```

2. Create an account for anonymous FTP. Use vipw(8) to insert this line:

   ```
   ftp:*:99:99::0:0:FTP:/cdrom:/nonexistent
   ```

3. Ensure that the FTP service is enabled in `/etc/inetd.conf`.

Anyone with network connectivity to the machine can now chose a media type of FTP and type in **`ftp://`*`your machine`*** after picking “Other” in the FTP sites menu during the install.

Note: If the boot media for the FTP clients is not precisely the same version as that provided by the local FTP site, sysinstall(8) will not complete the installation. To override this, go into the Options menu and change the distribution name to any.

Warning: This approach is acceptable for a machine on the local network which is protected by a firewall. Offering anonymous FTP services to other machines over the Internet exposes the computer to increased security risks. It is strongly recommended to follow good security practices when providing services over the Internet.

3.13.3 Installing from an Windows Partition

To prepare for an installation from a Windows partition, copy the files from the distribution into a directory in the root directory of the partition, such as `c:\freebsd`. Since the directory structure must be reproduced, it is recommended to use `robocopy` when copying from a CD/DVD. For example, to prepare for a minimal installation of FreeBSD:

```
C:\> md c:\freebsd
C:\> robocopy e:\bin c:\freebsd\bin\ /s
C:\> robocopy e:\manpages c:\freebsd\manpages\ /s
```

This example assumes that `C:` has enough free space and `E:` is where the CD/DVD is mounted.

Alternatively, download the distribution from ftp.FreeBSD.org (ftp://ftp.FreeBSD.org/pub/FreeBSD/releases/i386/8.4-RELEASE/). Each distribution is in its own directory; for example, the *base* distribution can be found in the 8.4/base/ (ftp://ftp.FreeBSD.org/pub/FreeBSD/releases/i386/8.4-RELEASE/base/) directory.

Copy the distributions to install from a Windows partition to `c:\freebsd`. Both the `base` and `kernel` distributions are needed for the most minimal installation.

3.13.4 Before Installing over a Network

There are three types of network installations available: Ethernet, PPP, and PLIP.

For the fastest possible network installation, use an Ethernet adapter. FreeBSD supports most common Ethernet cards. A list of supported cards is provided in the Hardware Notes for each release of FreeBSD. If using a supported PCMCIA Ethernet card, be sure that it is plugged in *before* the system is powered on as FreeBSD does not support hot insertion of PCMCIA cards during installation.

Make note of the system's IP address, subnet mask, hostname, default gateway address, and DNS server addresses if these values are statically assigned. If installing by FTP through a HTTP proxy, make note of the proxy's address. If you do not know these values, ask the system administrator or ISP *before* trying this type of installation.

If using a dialup modem, have the service provider's PPP information handy as it is needed early in the installation process.

If PAP or CHAP are used to connect to the ISP without using a script, type `dial` at the FreeBSD **ppp** prompt. Otherwise, know how to dial the ISP using the "AT commands" specific to the modem, as the PPP dialer provides only a simple terminal emulator. Refer to Section 28.2 and http://www.FreeBSD.org/doc/en_US.ISO8859-1/books/faq/ppp.html for further information. Logging can be directed to the screen using `set log local ...`.

If a hard-wired connection to another FreeBSD machine is available, the installation can occur over a null-modem parallel port cable. The data rate over the parallel port is higher than what is typically possible over a serial line.

3.13.4.1 Before Installing via NFS

To perform an NFS installation, copy the needed FreeBSD distribution files to an NFS server and then point the installer's NFS media selection to it.

If the server supports only a "privileged port", set the option `NFS Secure` in the Options menu so that the installation can proceed.

If using a poor quality Ethernet card which suffers from slow transfer rates, toggle the `NFS Slow` flag to on.

In order for an NFS installation to work, the server must support subdir mounts. For example, if the FreeBSD 9.1 distribution lives on: `ziggy:/usr/archive/stuff/FreeBSD`, `ziggy` will have to allow the direct mounting of `/usr/archive/stuff/FreeBSD`, not just `/usr` or `/usr/archive/stuff`.

In FreeBSD, this is controlled by using `-alldirs` in `/etc/exports`. Other NFS servers may have different conventions. If the server is displaying `permission denied` messages, it is likely that this is not enabled properly.

Chapter 4 UNIX Basics

Rewritten by Chris Shumway.

4.1 Synopsis

This chapter covers the basic commands and functionality of the FreeBSD operating system. Much of this material is relevant for any UNIX-like operating system. New FreeBSD users are encouraged to read through this chapter carefully.

After reading this chapter, you will know:

- How to use the “virtual consoles” of FreeBSD.
- How UNIX file permissions and FreeBSD file flags work.
- The default FreeBSD file system layout.
- The FreeBSD disk organization.
- How to mount and unmount file systems.
- What processes, daemons, and signals are.
- What a shell is, and how to change the default login environment.
- How to use basic text editors.
- What devices and device nodes are.
- What binary format is used under FreeBSD.
- How to read manual pages for more information.

4.2 Virtual Consoles and Terminals

FreeBSD can be used in various ways. One of them is typing commands to a text terminal. A lot of the flexibility and power of a UNIX operating system is readily available when using FreeBSD this way. This section describes what “terminals” and “consoles” are, and how to use them in FreeBSD.

4.2.1 The Console

Unless FreeBSD has been configured to automatically start a graphical environment during startup, the system will boot into a command line login prompt, as seen in this example:

```
FreeBSD/amd64 (pc3.example.org) (ttyv0)

login:
```

The first line contains some information about the system. The `amd64` indicates that the system in this example is running a 64-bit version of FreeBSD. The hostname is `pc3.example.org`, and `ttyv0` indicates that this is the system console.

The second line is the login prompt. The next section describes how to log into FreeBSD at this prompt.

4.2.2 Logging into FreeBSD

FreeBSD is a multiuser, multiprocessing system. This is the formal description that is usually given to a system that can be used by many different people, who simultaneously run a lot of programs on a single machine.

Every multiuser system needs some way to distinguish one "user" from the rest. In FreeBSD (and all the UNIX-like operating systems), this is accomplished by requiring that every user must "log into" the system before being able to run programs. Every user has a unique name (the "username") and a personal, secret key (the "password"). FreeBSD will ask for these two before allowing a user to run any programs.

When a FreeBSD system boots, startup scripts are automatically executed in order to prepare the system and to start any services which have been configured to start at system boot. Once the system finishes running its startup scripts, it will present a login prompt:

```
login:
```

Type the username that was configured during system installation, as described in Section 2.9.6, and press **Enter**. Then enter the password associated with the username and press **Enter**. The password is *not echoed* for security reasons.

Once the correct password is input, the message of the day (MOTD) will be displayed followed by a command prompt (a #, $, or % character). You are now logged into the FreeBSD console and ready to try the available commands.

4.2.3 Virtual Consoles

FreeBSD can be configured to provide many virtual consoles for inputting commands. Each virtual console has its own login prompt and output channel, and FreeBSD takes care of properly redirecting keyboard input and monitor output as switching occurs between virtual consoles.

Special key combinations have been reserved by FreeBSD for switching consoles.[1]. Use **Alt**-**F1**, **Alt**-**F2**, through **Alt**-**F8** to switch to a different virtual console in FreeBSD.

When switching from one console to the next, FreeBSD takes care of saving and restoring the screen output. The result is an "illusion" of having multiple "virtual" screens and keyboards that can be used to type commands for FreeBSD to run. The programs that are launched in one virtual console do not stop running when that console is not visible because the user has switched to a different virtual console.

4.2.4 The `/etc/ttys` File

By default, FreeBSD is configured to start eight virtual consoles. The configuration can be customized to start more or fewer virtual consoles. To change the number of and the settings of the virtual consoles, edit `/etc/ttys`.

Each uncommented line in `/etc/ttys` (lines that do not start with a # character) contains settings for a single terminal or virtual console. The default version configures nine virtual consoles, and enables eight of them. They are the lines that start with `ttyv`:

```
# name    getty                           type  status comments
#
```

```
ttyv0   "/usr/libexec/getty Pc"         cons25  on  secure
# Virtual terminals
ttyv1   "/usr/libexec/getty Pc"         cons25  on  secure
ttyv2   "/usr/libexec/getty Pc"         cons25  on  secure
ttyv3   "/usr/libexec/getty Pc"         cons25  on  secure
ttyv4   "/usr/libexec/getty Pc"         cons25  on  secure
ttyv5   "/usr/libexec/getty Pc"         cons25  on  secure
ttyv6   "/usr/libexec/getty Pc"         cons25  on  secure
ttyv7   "/usr/libexec/getty Pc"         cons25  on  secure
ttyv8   "/usr/X11R6/bin/xdm -nodaemon"  xterm   off secure
```

For a detailed description of every column in this file and the available options for the virtual consoles, refer to ttys(5).

4.2.5 Single User Mode Console

A detailed description of “single user mode” can be found in Section 13.6.2. There is only one console when FreeBSD is in single user mode as no other virtual consoles are available in this mode. The settings for single user mode are found in this section of `/etc/ttys`:

```
# name  getty                           type  status  comments
#
# If console is marked "insecure", then init will ask for the root password
# when going to single-user mode.
console none                            unknown  off  secure
```

> **Note:** As the comments above the `console` line indicate, editing `secure` to `insecure` will prompt for the `root` password when booting into single user mode. The default setting enters single user mode without prompting for a password.
>
> *Be careful when changing this setting to `insecure`.* If the `root` password is forgotten, booting into single user mode is still possible, but may be difficult for someone who is not comfortable with the FreeBSD booting process.

4.2.6 Changing Console Video Modes

The FreeBSD console default video mode may be adjusted to 1024x768, 1280x1024, or any other size supported by the graphics chip and monitor. To use a different video mode load the VESA module:

```
# kldload vesa
```

To determine which video modes are supported by the hardware, use vidcontrol(1). To get a list of supported video modes issue the following:

```
# vidcontrol -i mode
```

The output of this command lists the video modes that are supported by the hardware. To select a new video mode, specify the mode using vidcontrol(1) as the `root` user:

```
# vidcontrol MODE_279
```

If the new video mode is acceptable, it can be permanently set on boot by adding it to `/etc/rc.conf`:

```
allscreens_flags="MODE_279"
```

4.3 Permissions

FreeBSD, being a direct descendant of BSD UNIX, is based on several key UNIX concepts. The first and most pronounced is that FreeBSD is a multi-user operating system that can handle several users working simultaneously on completely unrelated tasks. The system is responsible for properly sharing and managing requests for hardware devices, peripherals, memory, and CPU time fairly to each user.

Much more information about user accounts is in the chapter about accounts. It is important to understand that each person (user) who uses the computer should be given their own username and password. The system keeps track of the people using the computer based on this username. Since it is often the case that several people are working on the same project UNIX also provides groups. Several users can be placed in the same group.

Because the system is capable of supporting multiple users, everything the system manages has a set of permissions governing who can read, write, and execute the resource. These permissions are stored as three octets broken into three pieces, one for the owner of the file, one for the group that the file belongs to, and one for everyone else. This numerical representation works like this:

> **Note:** This section will discuss the traditional UNIX permissions. For finer grained file system access control, see the File System Access Control Lists section.

Value	Permission	Directory Listing
0	No read, no write, no execute	`---`
1	No read, no write, execute	`--x`
2	No read, write, no execute	`-w-`
3	No read, write, execute	`-wx`
4	Read, no write, no execute	`r--`
5	Read, no write, execute	`r-x`
6	Read, write, no execute	`rw-`
7	Read, write, execute	`rwx`

Use the `-l` argument to ls(1) to view a long directory listing that includes a column of information about a file's permissions for the owner, group, and everyone else. For example, a `ls -l` in an arbitrary directory may show:

```
% ls -l
total 530
-rw-r--r--  1 root  wheel     512 Sep  5 12:31 myfile
-rw-r--r--  1 root  wheel     512 Sep  5 12:31 otherfile
-rw-r--r--  1 root  wheel    7680 Sep  5 12:31 email.txt
```

The first (leftmost) character in the first column indicates whether this file is a regular file, a directory, a special character device, a socket, or any other special pseudo-file device. In this example, the `-` indicates a regular file. The

next three characters, `rw-` in this example, give the permissions for the owner of the file. The next three characters, `r--`, give the permissions for the group that the file belongs to. The final three characters, `r--`, give the permissions for the rest of the world. A dash means that the permission is turned off. In this example, the permissions are set so the owner can read and write to the file, the group can read the file, and the rest of the world can only read the file. According to the table above, the permissions for this file would be `644`, where each digit represents the three parts of the file's permission.

How does the system control permissions on devices? FreeBSD treats most hardware devices as a file that programs can open, read, and write data to. These special device files are stored in `/dev/`.

Directories are also treated as files. They have read, write, and execute permissions. The executable bit for a directory has a slightly different meaning than that of files. When a directory is marked executable, it means it is possible to change into that directory using cd(1). This also means that it is possible to access the files within that directory, subject to the permissions on the files themselves.

In order to perform a directory listing, the read permission must be set on the directory. In order to delete a file that one knows the name of, it is necessary to have write *and* execute permissions to the directory containing the file.

There are more permission bits, but they are primarily used in special circumstances such as setuid binaries and sticky directories. For more information on file permissions and how to set them, refer to chmod(1).

4.3.1 Symbolic Permissions

Contributed by Tom Rhodes.

Symbolic permissions use characters instead of octal values to assign permissions to files or directories. Symbolic permissions use the syntax of (who) (action) (permissions), where the following values are available:

Option	Letter	Represents
(who)	u	User
(who)	g	Group owner
(who)	o	Other
(who)	a	All ("world")
(action)	+	Adding permissions
(action)	-	Removing permissions
(action)	=	Explicitly set permissions
(permissions)	r	Read
(permissions)	w	Write
(permissions)	x	Execute
(permissions)	t	Sticky bit
(permissions)	s	Set UID or GID

These values are used with chmod(1), but with letters instead of numbers. For example, the following command would block other users from accessing *`FILE`*:

```
% chmod go= FILE
```

A comma separated list can be provided when more than one set of changes to a file must be made. For example, the following command removes the group and "world" write permission on *`FILE`*, and adds the execute permissions for

everyone:

```
% chmod go-w,a+x FILE
```

4.3.2 FreeBSD File Flags

Contributed by Tom Rhodes.

In addition to file permissions, FreeBSD supports the use of “file flags”. These flags add an additional level of security and control over files, but not directories. With file flags, even `root` can be prevented from removing or altering files.

File flags are modified using chflags(1). For example, to enable the system undeletable flag on the file `file1`, issue the following command:

```
# chflags sunlink file1
```

To disable the system undeletable flag, put a “no” in front of the `sunlink`:

```
# chflags nosunlink file1
```

To view the flags of a file, use `-lo` with ls(1):

```
# ls -lo file1
```

```
-rw-r--r--  1 trhodes  trhodes  sunlnk 0 Mar  1 05:54 file1
```

Several file flags may only be added or removed by the `root` user. In other cases, the file owner may set its file flags. Refer to chflags(1) and chflags(2) for more information.

4.3.3 The `setuid`, `setgid`, and `sticky` Permissions

Contributed by Tom Rhodes.

Other than the permissions already discussed, there are three other specific settings that all administrators should know about. They are the `setuid`, `setgid`, and `sticky` permissions.

These settings are important for some UNIX operations as they provide functionality not normally granted to normal users. To understand them, the difference between the real user ID and effective user ID must be noted.

The real user ID is the UID who owns or starts the process. The effective UID is the user ID the process runs as. As an example, passwd(1) runs with the real user ID when a user changes their password. However, in order to update the password database, the command runs as the effective ID of the `root` user. This allows users to change their passwords without seeing a `Permission Denied` error.

The setuid permission may be set by prefixing a permission set with the number four (4) as shown in the following example:

```
# chmod 4755 suidexample.sh
```

The permissions on *`suidexample.sh`* now look like the following:

```
-rwsr-xr-x   1 trhodes  trhodes    63 Aug 29 06:36 suidexample.sh
```

Note that a `s` is now part of the permission set designated for the file owner, replacing the executable bit. This allows utilities which need elevated permissions, such as passwd(1).

> **Note:** The `nosuid` mount(8) option will cause such binaries to silently fail without alerting the user. That option is not completely reliable as a `nosuid` wrapper may be able to circumvent it.

To view this in real time, open two terminals. On one, type `passwd` as a normal user. While it waits for a new password, check the process table and look at the user information for passwd(1):

In terminal A:

```
Changing local password for trhodes
Old Password:
```

In terminal B:

```
# ps aux | grep passwd

trhodes  5232  0.0  0.2  3420  1608   0  R+    2:10AM   0:00.00 grep passwd
root     5211  0.0  0.2  3620  1724   2  I+    2:09AM   0:00.01 passwd
```

Although passwd(1) is run as a normal user, it is using the effective UID of `root`.

The `setgid` permission performs the same function as the `setuid` permission; except that it alters the group settings. When an application or utility executes with this setting, it will be granted the permissions based on the group that owns the file, not the user who started the process.

To set the `setgid` permission on a file, provide chmod(1) with a leading two (2):

```
# chmod 2755 sgidexample.sh
```

In the following listing, notice that the `s` is now in the field designated for the group permission settings:

```
-rwxr-sr-x   1 trhodes  trhodes    44 Aug 31 01:49 sgidexample.sh
```

> **Note:** In these examples, even though the shell script in question is an executable file, it will not run with a different EUID or effective user ID. This is because shell scripts may not access the setuid(2) system calls.

The `setuid` and `setgid` permission bits may lower system security, by allowing for elevated permissions. The third special permission, the `sticky bit`, can strengthen the security of a system.

When the `sticky bit` is set on a directory, it allows file deletion only by the file owner. This is useful to prevent file deletion in public directories, such as `/tmp`, by users who do not own the file. To utilize this permission, prefix the permission set with a one (1):

```
# chmod 1777 /tmp
```

The `sticky bit` permission will display as a `t` at the very end of the permission set:

```
# ls -al / | grep tmp
```

```
drwxrwxrwt  10 root  wheel         512 Aug 31 01:49 tmp
```

4.4 Directory Structure

The FreeBSD directory hierarchy is fundamental to obtaining an overall understanding of the system. The most important directory is root or, "/". This directory is the first one mounted at boot time and it contains the base system necessary to prepare the operating system for multi-user operation. The root directory also contains mount points for other file systems that are mounted during the transition to multi-user operation.

A mount point is a directory where additional file systems can be grafted onto a parent file system (usually the root file system). This is further described in Section 4.5. Standard mount points include `/usr/`, `/var/`, `/tmp/`, `/mnt/`, and `/cdrom/`. These directories are usually referenced to entries in `/etc/fstab`. This file is a table of various file systems and mount points and is read by the system. Most of the file systems in `/etc/fstab` are mounted automatically at boot time from the script rc(8) unless their entry includes `noauto`. Details can be found in Section 4.6.1.

A complete description of the file system hierarchy is available in hier(7). The following table provides a brief overview of the most common directories.

Directory	Description
`/`	Root directory of the file system.
`/bin/`	User utilities fundamental to both single-user and multi-user environments.
`/boot/`	Programs and configuration files used during operating system bootstrap.
`/boot/defaults/`	Default boot configuration files. Refer to loader.conf(5) for details.
`/dev/`	Device nodes. Refer to intro(4) for details.
`/etc/`	System configuration files and scripts.
`/etc/defaults/`	Default system configuration files. Refer to rc(8) for details.
`/etc/mail/`	Configuration files for mail transport agents such as sendmail(8).
`/etc/namedb/`	named(8) configuration files.
`/etc/periodic/`	Scripts that run daily, weekly, and monthly, via cron(8). Refer to periodic(8) for details.
`/etc/ppp/`	ppp(8) configuration files.
`/mnt/`	Empty directory commonly used by system administrators as a temporary mount point.
`/proc/`	Process file system. Refer to procfs(5), mount_procfs(8) for details.
`/rescue/`	Statically linked programs for emergency recovery as described in rescue(8).
`/root/`	Home directory for the `root` account.

Directory	Description
/sbin/	System programs and administration utilities fundamental to both single-user and multi-user environments.
/tmp/	Temporary files which are usually *not* preserved across a system reboot. A memory-based file system is often mounted at /tmp. This can be automated using the tmpmfs-related variables of rc.conf(5) or with an entry in /etc/fstab; refer to mdmfs(8) for details.
/usr/	The majority of user utilities and applications.
/usr/bin/	Common utilities, programming tools, and applications.
/usr/include/	Standard C include files.
/usr/lib/	Archive libraries.
/usr/libdata/	Miscellaneous utility data files.
/usr/libexec/	System daemons and system utilities executed by other programs.
/usr/local/	Local executables and libraries. Also used as the default destination for the FreeBSD ports framework. Within /usr/local, the general layout sketched out by hier(7) for /usr should be used. Exceptions are the man directory, which is directly under /usr/local rather than under /usr/local/share, and the ports documentation is in share/doc/*port*.
/usr/obj/	Architecture-specific target tree produced by building the /usr/src tree.
/usr/ports/	The FreeBSD Ports Collection (optional).
/usr/sbin/	System daemons and system utilities executed by users.
/usr/share/	Architecture-independent files.
/usr/src/	BSD and/or local source files.
/var/	Multi-purpose log, temporary, transient, and spool files. A memory-based file system is sometimes mounted at /var. This can be automated using the varmfs-related variables in rc.conf(5) or with an entry in /etc/fstab; refer to mdmfs(8) for details.
/var/log/	Miscellaneous system log files.
/var/mail/	User mailbox files.
/var/spool/	Miscellaneous printer and mail system spooling directories.
/var/tmp/	Temporary files which are usually preserved across a system reboot, unless /var is a memory-based file system.
/var/yp/	NIS maps.

4.5 Disk Organization

The smallest unit of organization that FreeBSD uses to find files is the filename. Filenames are case-sensitive, which means that `readme.txt` and `README.TXT` are two separate files. FreeBSD does not use the extension of a file to determine whether the file is a program, document, or some other form of data.

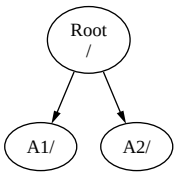
Files are stored in directories. A directory may contain no files, or it may contain many hundreds of files. A directory can also contain other directories, allowing a hierarchy of directories within one another in order to organize data.

Files and directories are referenced by giving the file or directory name, followed by a forward slash, `/`, followed by any other directory names that are necessary. For example, if the directory `foo` contains a directory `bar` which contains the file `readme.txt`, the full name, or *path*, to the file is `foo/bar/readme.txt`. Note that this is different from Windows which uses `\` to separate file and directory names. FreeBSD does not use drive letters, or other drive names in the path. For example, one would not type `c:/foo/bar/readme.txt` on FreeBSD.

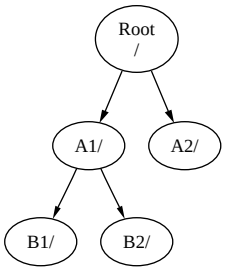
Directories and files are stored in a file system. Each file system contains exactly one directory at the very top level, called the *root directory* for that file system. This root directory can contain other directories. One file system is designated the *root file system* or `/`. Every other file system is *mounted* under the root file system. No matter how many disks are on the FreeBSD system, every directory appears to be part of the same disk.

Consider three file systems, called `A`, `B`, and `C`. Each file system has one root directory, which contains two other directories, called `A1`, `A2` (and likewise `B1`, `B2` and `C1`, `C2`).

Call `A` the root file system. If ls(1) is used to view the contents of this directory, it will show two subdirectories, `A1` and `A2`. The directory tree looks like this:

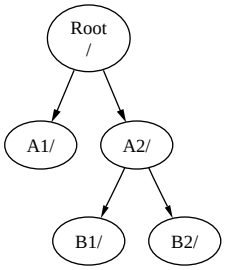


A file system must be mounted on to a directory in another file system. When mounting file system `B` on to the directory `A1`, the root directory of `B` replaces `A1`, and the directories in `B` appear accordingly:



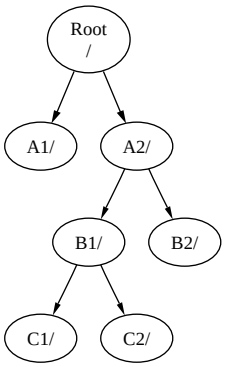
Any files that are in the `B1` or `B2` directories can be reached with the path `/A1/B1` or `/A1/B2` as necessary. Any files that were in `/A1` have been temporarily hidden. They will reappear if `B` is *unmounted* from `A`.

If `B` had been mounted on `A2` then the diagram would look like this:

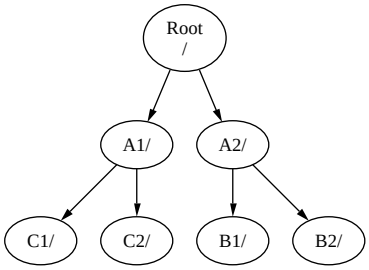


and the paths would be `/A2/B1` and `/A2/B2` respectively.

File systems can be mounted on top of one another. Continuing the last example, the `C` file system could be mounted on top of the `B1` directory in the `B` file system, leading to this arrangement:



Or `C` could be mounted directly on to the `A` file system, under the `A1` directory:



It is entirely possible to have one large root file system, and not need to create any others. There are some drawbacks to this approach, and one advantage.

Benefits of Multiple File Systems

- Different file systems can have different *mount options*. For example, the root file system can be mounted read-only, making it impossible for users to inadvertently delete or edit a critical file. Separating user-writable file systems, such as `/home`, from other file systems allows them to be mounted *nosuid*. This option prevents the *suid*/*guid* bits on executables stored on the file system from taking effect, possibly improving security.
- FreeBSD automatically optimizes the layout of files on a file system, depending on how the file system is being used. So a file system that contains many small files that are written frequently will have a different optimization to one that contains fewer, larger files. By having one big file system this optimization breaks down.

- FreeBSD's file systems are robust if power is lost. However, a power loss at a critical point could still damage the structure of the file system. By splitting data over multiple file systems it is more likely that the system will still come up, making it easier to restore from backup as necessary.

Benefit of a Single File System

- File systems are a fixed size. If you create a file system when you install FreeBSD and give it a specific size, you may later discover that you need to make the partition bigger. This is not easily accomplished without backing up, recreating the file system with the new size, and then restoring the backed up data.

> **Important:** FreeBSD features the growfs(8) command, which makes it possible to increase the size of file system on the fly, removing this limitation.

File systems are contained in partitions. This does not have the same meaning as the common usage of the term partition (for example, MS-DOS partition), because of FreeBSD's UNIX heritage. Each partition is identified by a letter from a through to h. Each partition can contain only one file system, which means that file systems are often described by either their typical mount point in the file system hierarchy, or the letter of the partition they are contained in.

FreeBSD also uses disk space for *swap space* to provide *virtual memory*. This allows your computer to behave as though it has much more memory than it actually does. When FreeBSD runs out of memory, it moves some of the data that is not currently being used to the swap space, and moves it back in (moving something else out) when it needs it.

Some partitions have certain conventions associated with them.

Partition	Convention
a	Normally contains the root file system.
b	Normally contains swap space.
c	Normally the same size as the enclosing slice. This allows utilities that need to work on the entire slice, such as a bad block scanner, to work on the c partition. A file system would not normally be created on this partition.
d	Partition d used to have a special meaning associated with it, although that is now gone and d may work as any normal partition.

Disks in FreeBSD are divided into slices, referred to in Windows as partitions, which are numbered from 1 to 4. These are then then divided into partitions, which contain file systems, and are labeled using letters.

Slice numbers follow the device name, prefixed with an s, starting at 1. So "da0s*1*" is the first slice on the first SCSI drive. There can only be four physical slices on a disk, but there can be logical slices inside physical slices of the appropriate type. These extended slices are numbered starting at 5, so "ad0s*5*" is the first extended slice on the first IDE disk. These devices are used by file systems that expect to occupy a slice.

Slices, "dangerously dedicated" physical drives, and other drives contain *partitions*, which are represented as letters from a to h. This letter is appended to the device name, so "da0*a*" is the a partition on the first da drive, which is "dangerously dedicated". "ad1s3*e*" is the fifth partition in the third slice of the second IDE disk drive.

Finally, each disk on the system is identified. A disk name starts with a code that indicates the type of disk, and then a number, indicating which disk it is. Unlike slices, disk numbering starts at 0. Common codes are listed in Table 4-1.

When referring to a partition, include the disk name, `s`, the slice number, and then the partition letter. Examples are shown in Example 4-1.

Example 4-2 shows a conceptual model of a disk layout.

When installing FreeBSD, configure the disk slices, create partitions within the slice to be used for FreeBSD, create a file system or swap space in each partition, and decide where each file system will be mounted.

Table 4-1. Disk Device Codes

Code	Meaning
`ad`	ATAPI (IDE) disk
`da`	SCSI direct access disk
`acd`	ATAPI (IDE) CDROM
`cd`	SCSI CDROM
`fd`	Floppy disk

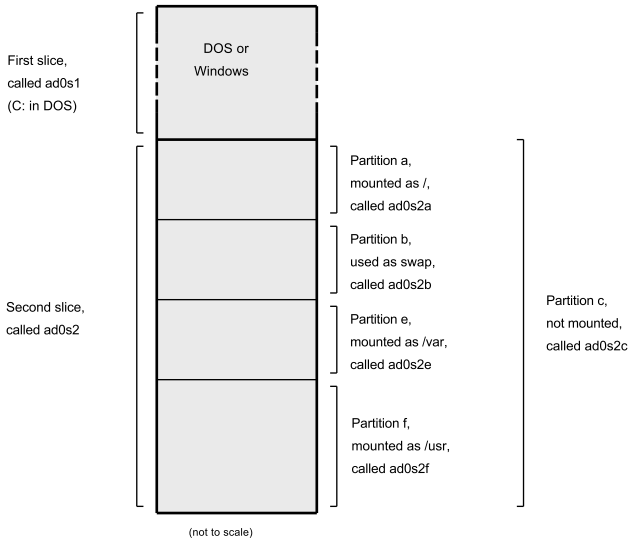
Example 4-1. Sample Disk, Slice, and Partition Names

Name	Meaning
`ad0s1a`	The first partition (`a`) on the first slice (`s1`) on the first IDE disk (`ad0`).
`da1s2e`	The fifth partition (`e`) on the second slice (`s2`) on the second SCSI disk (`da1`).

Example 4-2. Conceptual Model of a Disk

This diagram shows FreeBSD's view of the first IDE disk attached to the system. Assume that the disk is 4 GB in size, and contains two 2 GB slices (MS-DOS partitions). The first slice contains a MS-DOS disk, `C:`, and the second slice contains a FreeBSD installation. This example FreeBSD installation has three data partitions, and a swap partition.

The three partitions will each hold a file system. Partition `a` will be used for the root file system, `e` for the `/var/` directory hierarchy, and `f` for the `/usr/` directory hierarchy.



4.6 Mounting and Unmounting File Systems

The file system is best visualized as a tree, rooted, as it were, at `/`. `/dev`, `/usr`, and the other directories in the root directory are branches, which may have their own branches, such as `/usr/local`, and so on.

There are various reasons to house some of these directories on separate file systems. `/var` contains the directories `log/`, `spool/`, and various types of temporary files, and as such, may get filled up. Filling up the root file system is not a good idea, so splitting `/var` from `/` is often favorable.

Another common reason to contain certain directory trees on other file systems is if they are to be housed on separate physical disks, or are separate virtual disks, such as Network File System mounts, described in Section 30.3, or CDROM drives.

4.6.1 The `fstab` File

During the boot process (Chapter 13), file systems listed in `/etc/fstab` are automatically mounted except for the entries containing `noauto`. This file contains entries in the following format:

```
device        /mount-point  fstype      options       dumpfreq      passno
```

`device`

An existing device name as explained in Section 19.2.

`mount-point`

An existing directory on which to mount the file system.

`fstype`

The file system type to pass to mount(8). The default FreeBSD file system is `ufs`.

`options`

Either `rw` for read-write file systems, or `ro` for read-only file systems, followed by any other options that may be needed. A common option is `noauto` for file systems not normally mounted during the boot sequence. Other options are listed in mount(8).

`dumpfreq`

Used by dump(8) to determine which file systems require dumping. If the field is missing, a value of zero is assumed.

`passno`

Determines the order in which file systems should be checked. File systems that should be skipped should have their `passno` set to zero. The root file system needs to be checked before everything else and should have its `passno` set to one. The other file systems should be set to values greater than one. If more than one file system has the same `passno`, fsck(8) will attempt to check file systems in parallel if possible.

Refer to fstab(5) for more information on the format of `/etc/fstab` and its options.

4.6.2 Using mount(8)

File systems are mounted using mount(8). The most basic syntax is as follows:

```
# mount device mountpoint
```

This command provides many options which are described in mount(8), The most commonly used options include:

Mount Options

`-a`

Mount all the file systems listed in `/etc/fstab`, except those marked as “noauto”, excluded by the `-t` flag, or those that are already mounted.

`-d`

Do everything except for the actual mount system call. This option is useful in conjunction with the `-v` flag to determine what mount(8) is actually trying to do.

`-f`

Force the mount of an unclean file system (dangerous), or the revocation of write access when downgrading a file system’s mount status from read-write to read-only.

`-r`

Mount the file system read-only. This is identical to using `-o ro`.

`-t` *`fstype`*

Mount the specified file system type or mount only file systems of the given type, if `-a` is included. “ufs” is the default file system type.

`-u`

Update mount options on the file system.

`-v`

Be verbose.

`-w`

Mount the file system read-write.

The following options can be passed to `-o` as a comma-separated list:

nosuid

Do not interpret setuid or setgid flags on the file system. This is also a useful security option.

4.6.3 Using umount(8)

To unmount a file system use umount(8). This command takes one parameter which can be a mountpoint, device name, `-a` or `-A`.

All forms take `-f` to force unmounting, and `-v` for verbosity. Be warned that `-f` is not generally a good idea as it might crash the computer or damage data on the file system.

To unmount all mounted file systems, or just the file system types listed after `-t`, use `-a` or `-A`. Note that `-A` does not attempt to unmount the root file system.

4.7 Processes

FreeBSD is a multi-tasking operating system. Each program running at any one time is called a *process*. Every running command starts at least one new process and there are a number of system processes that are run by FreeBSD.

Each process is uniquely identified by a number called a *process ID* (PID). Similar to files, each process has one owner and group, and the owner and group permissions are used to determine which files and devices the process can open. Most processes also have a parent process that started them. For example, the shell is a process, and any command started in the shell is a process which has the shell as its parent process. The exception is a special process called init(8) which is always the first process to start at boot time and which always has a PID of 1.

To see the processes on the system, use ps(1) and top(1). To display a static list of the currently running processes, their PIDs, how much memory they are using, and the command they were started with, use ps(1). To display all the running processes and update the display every few seconds in order to interactively see what the computer is doing, use top(1).

By default, ps(1) only shows the commands that are running and owned by the user. For example:

```
% ps
  PID  TT  STAT      TIME COMMAND
  298  p0  Ss     0:01.10 tcsh
 7078  p0  S      2:40.88 xemacs mdoc.xsl (xemacs-21.1.14)
37393  p0  I      0:03.11 xemacs freebsd.dsl (xemacs-21.1.14)
72210  p0  R+     0:00.00 ps
  390  p1  Is     0:01.14 tcsh
 7059  p2  Is+    1:36.18 /usr/local/bin/mutt -y
 6688  p3  IWs    0:00.00 tcsh
10735  p4  IWs    0:00.00 tcsh
20256  p5  IWs    0:00.00 tcsh
  262  v0  IWs    0:00.00 -tcsh (tcsh)
  270  v0  IW+    0:00.00 /bin/sh /usr/X11R6/bin/startx -- -bpp 16
  280  v0  IW+    0:00.00 xinit /home/nik/.xinitrc -- -bpp 16
  284  v0  IW     0:00.00 /bin/sh /home/nik/.xinitrc
  285  v0  S      0:38.45 /usr/X11R6/bin/sawfish
```

The output from ps(1) is organized into a number of columns. The `PID` column displays the process ID. PIDs are assigned starting at 1, go up to 99999, then wrap around back to the beginning. However, a PID is not reassigned if it is already in use. The `TT` column shows the tty the program is running on and `STAT` shows the program's state. `TIME` is the amount of time the program has been running on the CPU. This is usually not the elapsed time since the program was started, as most programs spend a lot of time waiting for things to happen before they need to spend time on the CPU. Finally, `COMMAND` is the command that was used to start the program.

ps(1) supports a number of different options to change the information that is displayed. One of the most useful sets is `auxww`. `a` displays information about all the running processes of all users. `u` displays the username of the process' owner, as well as memory usage. `x` displays information about daemon processes, and `ww` causes ps(1) to display the full command line for each process, rather than truncating it once it gets too long to fit on the screen.

The output from top(1) is similar. A sample session looks like this:

```
% top
last pid: 72257;  load averages:  0.13,  0.09,  0.03    up 0+13:38:33  22:39:10
47 processes:  1 running, 46 sleeping
CPU states: 12.6% user,  0.0% nice,  7.8% system,  0.0% interrupt, 79.7% idle
Mem: 36M Active, 5256K Inact, 13M Wired, 6312K Cache, 15M Buf, 408K Free
Swap: 256M Total, 38M Used, 217M Free, 15% Inuse

  PID USERNAME PRI NICE  SIZE    RES STATE    TIME   WCPU    CPU COMMAND
72257 nik       28   0  1960K  1044K RUN      0:00 14.86%  1.42% top
 7078 nik        2   0 15280K 10960K select   2:54  0.88%  0.88% xemacs-21.1.14
  281 nik        2   0 18636K  7112K select   5:36  0.73%  0.73% XF86_SVGA
  296 nik        2   0  3240K  1644K select   0:12  0.05%  0.05% xterm
  175 root       2   0   924K   252K select   1:41  0.00%  0.00% syslogd
 7059 nik        2   0  7260K  4644K poll     1:38  0.00%  0.00% mutt
...
```

The output is split into two sections. The header (the first five lines) shows the PID of the last process to run, the system load averages (which are a measure of how busy the system is), the system uptime (time since the last reboot) and the current time. The other figures in the header relate to how many processes are running (47 in this case), how much memory and swap space has been used, and how much time the system is spending in different CPU states.

Below the header is a series of columns containing similar information to the output from ps(1), such as the PID, username, amount of CPU time, and the command that started the process. By default, top(1) also displays the amount of memory space taken by the process. This is split into two columns: one for total size and one for resident size. Total size is how much memory the application has needed and the resident size is how much it is actually using at the moment. In this example, **mutt** has required almost 8 MB of RAM, but is currently only using 5 MB.

top(1) automatically updates the display every two seconds. A different interval can be specified with `-s`.

4.8 Daemons, Signals, and Killing Processes

When using an editor, it is easy to control the editor and load files because the editor provides facilities to do so, and because the editor is attached to a *terminal*. Some programs are not designed to be run with continuous user input and disconnect from the terminal at the first opportunity. For example, a web server responds to web requests, rather than user input. Mail servers are another example of this type of application.

These programs are known as *daemons*. The term daemon comes from Greek mythology and represents an entity that is neither good or evil, and which invisibly performs useful tasks. This is why the BSD mascot is the cheerful-looking daemon with sneakers and a pitchfork.

There is a convention to name programs that normally run as daemons with a trailing "d". **BIND** is the Berkeley Internet Name Domain, but the actual program that executes is named(8). The **Apache** web server program is `httpd` and the line printer spooling daemon is lpd(8). This is only a naming convention. For example, the main mail daemon for the **Sendmail** application is sendmail(8), and not `maild`.

One way to communicate with a daemon, or any running process, is to send a *signal* using kill(1). There are a number of different signals; some have a specific meaning while others are described in the application's documentation. A user can only send a signal to a process they own and sending a signal to someone else's process will result in a permission denied error. The exception is the `root` user, who can send signals to anyone's processes.

FreeBSD can also send a signal to a process. If an application is badly written and tries to access memory that it is not supposed to, FreeBSD will send the process the *Segmentation Violation* signal (`SIGSEGV`). If an application has used the alarm(3) system call to be alerted after a period of time has elapsed, it will be sent the Alarm signal (`SIGALRM`).

Two signals can be used to stop a process: `SIGTERM` and `SIGKILL`. `SIGTERM` is the polite way to kill a process as the process can read the signal, close any log files it may have open, and attempt to finish what it is doing before shutting down. In some cases, a process may ignore `SIGTERM` if it is in the middle of some task that can not be interrupted.

`SIGKILL` can not be ignored by a process. This is the "I do not care what you are doing, stop right now" signal. Sending a `SIGKILL` to a process will usually stop that process there and then.[2].

Other commonly used signals are `SIGHUP`, `SIGUSR1`, and `SIGUSR2`. These are general purpose signals and different applications will respond differently.

For example, after changing a web server's configuration file, the web server needs to be told to re-read its configuration. Restarting `httpd` would result in a brief outage period on the web server. Instead, send the daemon the `SIGHUP` signal. Be aware that different daemons will have different behavior, so refer to the documentation for the daemon to determine if `SIGHUP` will achieve the desired results.

Sending a Signal to a Process

This example shows how to send a signal to inetd(8). The inetd(8) configuration file is `/etc/inetd.conf`, and inetd(8) will re-read this configuration file when it is sent a `SIGHUP`.

1. Find the PID of the process to send the signal to using pgrep(1). In this example, the PID for inetd(8) is 198:

```
% pgrep -l inetd
198  inetd -wW
```

2. Use kill(1) to send the signal. Because inetd(8) is owned by `root`, use su(1) to become `root` first.

```
% su
Password:
# /bin/kill -s HUP 198
```

Like most UNIX commands, kill(1) will not print any output if it is successful. If a signal is sent to a process not owned by that user, the message `kill: PID: Operation not permitted` will be displayed. Mistyping the PID will either send the signal to the wrong process, which could have negative results, or will send the signal to a PID that is not currently in use, resulting in the error `kill: PID: No such process`.

> **Why Use `/bin/kill`?:** Many shells provide `kill` as a built in command, meaning that the shell will send the signal directly, rather than running `/bin/kill`. Be aware that different shells have a different syntax for specifying the name of the signal to send. Rather than try to learn all of them, it can be simpler to use `/bin/kill ...` directly.

When sending other signals, substitute `TERM` or `KILL` in the command line as necessary.

> **Important:** Killing a random process on the system can be a bad idea. In particular, init(8), PID 1, is special. Running `/bin/kill -s KILL 1` is a quick, and unrecommended, way to shutdown the system. *Always* double check the arguments to kill(1) *before* pressing **Return**.

4.9 Shells

FreeBSD provides a command line interface called a shell. A shell receives commands from the input channel and executes them. Many shells provide built in functions to help with everyday tasks such as file management, file globbing, command line editing, command macros, and environment variables. FreeBSD comes with several shells, including the Bourne shell (sh(1)) and the extended C shell (tcsh(1)). Other shells are available from the FreeBSD Ports Collection, such as `zsh` and `bash`.

The shell that is used is really a matter of taste. A C programmer might feel more comfortable with a C-like shell such as tcsh(1). A Linux user might prefer `bash`. Each shell has unique properties that may or may not work with a user's preferred working environment, which is why there is a choice of which shell to use.

One common shell feature is filename completion. After a user types the first few letters of a command or filename and presses **Tab**, the shell will automatically complete the rest of the command or filename. Consider two files called `foobar` and `foo.bar`. To delete `foo.bar`, type `rm fo[Tab].[Tab]`.

The shell should print out `rm foo[BEEP].bar`.

The [BEEP] is the console bell, which the shell used to indicate it was unable to complete the filename because there is more than one match. Both `foobar` and `foo.bar` start with `fo`. By typing `.`, then pressing **Tab** again, the shell would be able to fill in the rest of the filename.

Another feature of the shell is the use of environment variables. Environment variables are a variable/key pair stored in the shell's environment. This environment can be read by any program invoked by the shell, and thus contains a lot of program configuration. Here is a list of common environment variables and their meanings:

Variable	Description
USER	Current logged in user's name.
PATH	Colon-separated list of directories to search for binaries.
DISPLAY	Network name of the **Xorg** display to connect to, if available.
SHELL	The current shell.
TERM	The name of the user's type of terminal. Used to determine the capabilities of the terminal.
TERMCAP	Database entry of the terminal escape codes to perform various terminal functions.
OSTYPE	Type of operating system.
MACHTYPE	The system's CPU architecture.
EDITOR	The user's preferred text editor.
PAGER	The user's preferred text pager.
MANPATH	Colon-separated list of directories to search for manual pages.

How to set an environment variable differs between shells. In tcsh(1) and csh(1), use `setenv` to set environment variables. In sh(1) and `bash`, use `export` to set the current environment variables. This example sets the default `EDITOR` to `/usr/local/bin/emacs` for the tcsh(1) shell:

```
% setenv EDITOR /usr/local/bin/emacs
```

The equivalent command for `bash` would be:

```
% export EDITOR="/usr/local/bin/emacs"
```

To expand an environment variable in order to see its current setting, type a `$` character in front of its name on the command line. For example, `echo $TERM` displays the current `$TERM` setting.

Shells treat special characters, known as meta-characters, as special representations of data. The most common meta-character is `*`, which represents any number of characters in a filename. Meta-characters can be used to perform filename globbing. For example, `echo *` is equivalent to ls(1) because the shell takes all the files that match `*` and echo(1) lists them on the command line.

To prevent the shell from interpreting a special character, escape it from the shell by starting it with a backslash (`\`). For example, `echo $TERM` prints the terminal setting whereas `echo \$TERM` literally prints the string `$TERM`.

4.9.1 Changing Your Shell

The easiest way to permanently change the default shell is to use `chsh`. Running this command will open the editor that is configured in the `EDITOR` environment variable, which by default is set to vi(1). Change the "Shell:" line to the full path of the new shell.

Alternately, use `chsh -s` which will set the specified shell without opening an editor. For example, to change the shell to `bash`:

```
% chsh -s /usr/local/bin/bash
```

> **Note:** The new shell *must* be present in `/etc/shells`. If the shell was installed from the FreeBSD Ports Collection as described in Chapter 5, it should be automatically added to this file. If it is missing, add it using this command, replacing the path with the path of the shell:
>
> ```
> # echo /usr/local/bin/bash >> /etc/shells
> ```
>
> Then rerun chsh(1).

4.10 Text Editors

Most FreeBSD configuration is done by editing text files. Because of this, it is a good idea to become familiar with a text editor. FreeBSD comes with a few as part of the base system, and many more are available in the Ports Collection.

A simple editor to learn is ee(1), which stands for easy editor. To start this editor, type `ee` *`filename`* where *`filename`* is the name of the file to be edited. Once inside the editor, all of the commands for manipulating the editor's functions are listed at the top of the display. The caret `^` represents **Ctrl**, so `^e` expands to **Ctrl**+**e**. To leave ee(1), press **Esc**, then choose the "leave editor" option from the main menu. The editor will prompt to save any changes if the file has been modified.

FreeBSD also comes with more powerful text editors, such as vi(1), as part of the base system. Other editors, like `editors/emacs` and `editors/vim`, are part of the FreeBSD Ports Collection. These editors offer more functionality at the expense of being a more complicated to learn. Learning a more powerful editor such as **vim** or **Emacs** can save more time in the long run.

Many applications which modify files or require typed input will automatically open a text editor. To alter the default editor used, set the `EDITOR` environment variable as described in Section 4.9.

4.11 Devices and Device Nodes

A device is a term used mostly for hardware-related activities in a system, including disks, printers, graphics cards, and keyboards. When FreeBSD boots, the majority of the boot messages refer to devices being detected. A copy of the boot messages are saved to `/var/run/dmesg.boot`.

Each device has a device name and number. For example, `acd0` is the first IDE CD-ROM drive, while `kbd0` represents the keyboard.

Most devices in a FreeBSD must be accessed through special files called device nodes, which are located in `/dev`.

4.12 Binary Formats

Typically when a command is passed to the shell, the shell will arrange for an executable file to be loaded into memory and a new process is created. Executable files can either be a binary file (usually created by the linker as part of compiling a program) or a shell script (text file to be interpreted by a binary file, like sh(1) or perl(1)). The file(1) command can usually determine what is inside a file.

Binary files need to have a well defined format for the system to be able to use them properly. Part of the file will be the executable machine code (the instructions that tell the CPU what to do), part of it will be data space with pre-defined values, part will be data space with no pre-defined values, etc. Through time, different binary file formats have evolved.

To understand why FreeBSD uses the elf(5) format, the three currently "dominant", executable formats for UNIX must be described:

- a.out(5)

  The oldest and "classic" UNIX object format. It uses a short and compact header with a magic(5) number at the beginning that is often used to characterize the format. It contains three loaded segments: .text, .data, and .bss, plus a symbol table and a string table.

- COFF

  The SVR3 object format. The header comprises a section table which can contain more than just .text, .data, and .bss sections.

- elf(5)

  The successor to COFF, featuring multiple sections and 32-bit or 64-bit possible values. One major drawback is that ELF was designed with the assumption that there would be only one ABI per system architecture. That assumption is actually incorrect, and not even in the commercial SYSV world (which has at least three ABIs: SVR4, Solaris, SCO) does it hold true.

  FreeBSD tries to work around this problem somewhat by providing a utility for *branding* a known ELF executable with information about its compliant ABI. Refer to brandelf(1) for more information.

FreeBSD comes from the "classic" camp and used the a.out(5) format, a technology tried and proven through many generations of BSD releases, until the beginning of the 3.X branch. Though it was possible to build and run native ELF binaries and kernels on a FreeBSD system for some time before that, FreeBSD initially resisted the "push" to switch to ELF as the default format. Why? When Linux made its painful transition to ELF, it was due to their inflexible jump-table based shared library mechanism, which made the construction of shared libraries difficult for vendors and developers. Since ELF tools offered a solution to the shared library problem and were generally seen as "the way forward", the migration cost was accepted as necessary and the transition made. FreeBSD's shared library mechanism is based more closely on the SunOS™ style shared library mechanism and is easy to use.

So, why are there so many different formats? Back in the PDP-11 days when simple hardware supported a simple, small system, `a.out` was adequate for the job of representing binaries. As UNIX was ported, the `a.out` format was retained because it was sufficient for the early ports of UNIX to architectures like the Motorola 68k or VAXen.

Then some hardware engineer decided that if he could force software to do some sleazy tricks, a few gates could be shaved off the design and the CPU core could run faster. `a.out` was ill-suited for this new kind of hardware, known as RISC. Many formats were developed to get better performance from this hardware than the limited, simple `a.out` format could offer. COFF, ECOFF, and a few others were invented and their limitations explored before settling on ELF.

In addition, program sizes were getting huge while disks and physical memory were still relatively small, so the concept of a shared library was born. The virtual memory system became more sophisticated. While each advancement was done using the `a.out` format, its usefulness was stretched with each new feature. In addition, people wanted to dynamically load things at run time, or to junk parts of their program after the init code had run to save in core memory and swap space. Languages became more sophisticated and people wanted code called before the main() function automatically. Lots of hacks were done to the `a.out` format to allow all of these things to happen, and they basically worked for a time. In time, `a.out` was not up to handling all these problems without an ever increasing overhead in code and complexity. While ELF solved many of these problems, it would be painful to switch from the system that basically worked. So ELF had to wait until it was more painful to remain with `a.out` than it was to migrate to ELF.

As time passed, the build tools that FreeBSD derived their build tools from, especially the assembler and loader, evolved in two parallel trees. The FreeBSD tree added shared libraries and fixed some bugs. The GNU folks that originally wrote these programs rewrote them and added simpler support for building cross compilers and plugging in different formats. Those who wanted to build cross compilers targeting FreeBSD were out of luck since the older sources that FreeBSD had for as(1) and ld(1) were not up to the task. The new GNU tools chain (**binutils**) supports cross compiling, ELF, shared libraries, and C++ extensions. In addition, many vendors release ELF binaries, and FreeBSD should be able to run them.

ELF is more expressive than `a.out` and allows more extensibility in the base system. The ELF tools are better maintained and offer cross compilation support. ELF may be a little slower than `a.out`, but trying to measure it can be difficult. There are also numerous details that are different between the two such as how they map pages and handle init code.

4.13 For More Information

4.13.1 Manual Pages

The most comprehensive documentation on FreeBSD is in the form of manual pages. Nearly every program on the system comes with a short reference manual explaining the basic operation and available arguments. These manuals can be viewed using `man`:

```
% man command
```

where *command* is the name of the command to learn about. For example, to learn more about ls(1), type:

```
% man ls
```

The online manual is divided into numbered sections:

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 developers.

In some cases, the same topic may appear in more than one section of the online manual. For example, there is a chmod(1) user command and a `chmod()` system call. To tell man(1) which section to display, specify the section number:

```
% man 1 chmod
```

This will display the manual page for the user command chmod(1). References to a particular section of the online manual are traditionally placed in parenthesis in written documentation, so chmod(1) refers to the user command and chmod(2) refers to the system call.

If the command name is unknown, use `man -k` to search for keywords in the command descriptions:

```
% man -k mail
```

This command displays a list of commands that have the keyword "mail" in their descriptions. This is equivalent to using apropos(1).

To determine what the commands in `/usr/bin` do, type:

```
% cd /usr/bin
% man -f *
```

or

```
% cd /usr/bin
% whatis *
```

4.13.2 GNU Info Files

FreeBSD includes many applications and utilities produced by the Free Software Foundation (FSF). In addition to manual pages, these programs may include hypertext documents called `info` files. These can be viewed using info(1) or, if `editors/emacs` is installed, the info mode of **emacs**.

To use info(1), type:

```
% info
```

For a brief introduction, type `h`. For a quick command reference, type `?`.

Notes

1. Refer to syscons(4), atkbd(4), vidcontrol(1) and kbdcontrol(1) for a more technical description of the FreeBSD console and its keyboard drivers.
2. There are a few tasks that can not be interrupted. For example, if the process is trying to read from a file that is on another computer on the network, and the other computer is unavailable, the process is said to be

"uninterruptible". Eventually the process will time out, typically after two minutes. As soon as this time out occurs the process will be killed.

Chapter 5 Installing Applications: Packages and Ports

5.1 Synopsis

FreeBSD is bundled with a rich collection of system tools as part of the base system. However, there is only so much one can do before needing to install an additional third-party application to get real work done. FreeBSD provides two complementary technologies for installing third-party software: the FreeBSD Ports Collection (for installing from source), and packages (for installing from pre-built binaries). Either method may be used to install software from local media or from the network.

After reading this chapter, you will know how to:

- Install third-party binary software packages.
- Build third-party software from source by using the Ports Collection.
- Remove previously installed packages or ports.
- Override the default values used by the Ports Collection.
- Find the appropriate software package.
- Upgrade installed software.

5.2 Overview of Software Installation

The typical steps for installing third-party software on a UNIX system include:

1. Download the software, which might be distributed in source code format, or as a binary.
2. Unpack the software from its distribution format (typically a tarball compressed with compress(1), gzip(1), or bzip2(1)).
3. Locate the documentation in `INSTALL`, `README` or some file in a `doc/` subdirectory and read up on how to install the software.
4. If the software was distributed in source format, compile it. This may involve editing a `Makefile`, or running a `configure` script, and other work.
5. Test and install the software.

If you are installing a software package that was not deliberately ported to FreeBSD you may even have to go in and edit the code to make it work properly.

FreeBSD provides two technologies which perform these steps for you. At the time of writing, over 24,000 third-party applications are available.

A FreeBSD package contains pre-compiled copies of all the commands for an application, as well as any configuration files and documentation. A package can be manipulated with FreeBSD package management commands, such as pkg_add(1), pkg_delete(1), and pkg_info(1).

A FreeBSD port is a collection of files designed to automate the process of compiling an application from source code. The files that comprise a port contain all the necessary information to automatically download, extract, patch, compile, and install the application.

The ports system can also be used to generate packages which can be manipulated with the FreeBSD package management commands.

Both packages and ports understand *dependencies*. If pkg_add(1) or the Ports Collection is used to install an application and a dependent library is not already installed, the library will automatically be installed first.

While the two technologies are quite similar, packages and ports each have their own strengths. Select the technology that meets your requirements for installing a particular application.

Package Benefits

- A compressed package tarball is typically smaller than the compressed tarball containing the source code for the application.
- Packages do not require compilation time. For large applications, such as **Mozilla**, **KDE**, or **GNOME** this can be important, on a slow system.
- Packages do not require any understanding of the process involved in compiling software on FreeBSD.

Ports Benefits

- Packages are normally compiled with conservative options because they have to run on the maximum number of systems. By compiling from the port, one can change the compilation options.
- Some applications have compile-time options relating to which features are installed. For example, **Apache** can be configured with a wide variety of different built-in options.

  In some cases, multiple packages will exist for the same application to specify certain settings. For example, **Ghostscript** is available as a `ghostscript` package and a `ghostscript-nox11` package, depending on whether or not **Xorg** is installed. Creating multiple packages rapidly becomes impossible if an application has more than one or two different compile-time options.
- The licensing conditions of some software forbid binary distribution. These must be distributed as source code which must be compiled by the end-user.
- Some people do not trust binary distributions or prefer to read through source code in order to look for potential problems.
- If you have local patches, you will need the source in order to apply them.

To keep track of updated ports, subscribe to the FreeBSD ports mailing list (http://lists.FreeBSD.org/mailman/listinfo/freebsd-ports) and the FreeBSD ports bugs mailing list (http://lists.FreeBSD.org/mailman/listinfo/freebsd-ports-bugs).

> **Warning:** Before installing any application, check http://vuxml.freebsd.org/ for security issues related to the application or install `ports-mgmt/portaudit`. Once installed, type `portaudit -F -a` to check all installed applications for known vulnerabilities.

The remainder of this chapter explains how to use packages and ports to install and manage third-party software on FreeBSD.

5.3 Finding Software

FreeBSD's list of available applications is growing all the time. There are a number of ways to find software to install:

- The FreeBSD web site maintains an up-to-date searchable list of all the available applications, at http://www.FreeBSD.org/ports/ (http://www.FreeBSD.org/ports/index.html). The ports can be searched by application name or by software category.
- Dan Langille maintains FreshPorts (http://www.FreshPorts.org/) which provides a comprehensive search utility and also tracks changes to the applications in the Ports Collection. Registered users can create a customized watch list in order to receive an automated email when their watched ports are updated.
- If you do not know the name of the application you want, try using a site like Freecode (http://www.freecode.com/) to find an application, then check back at the FreeBSD site to see if the application has been ported yet.
- To find out which category a port is in, type `whereis` *`file`*, where *`file`* is the program to be installed:

  ```
  # whereis lsof
  lsof: /usr/ports/sysutils/lsof
  ```

  Alternately, an echo(1) statement can be used:

  ```
  # echo /usr/ports/*/*lsof*
  /usr/ports/sysutils/lsof
  ```

  Note that this will return any matched files downloaded into the `/usr/ports/distfiles` directory.
- Another way to find software is by using the Ports Collection's built-in search mechanism. To use the search feature, **cd** to `/usr/ports` then run `make search name=`*`program-name`* where *`program-name`* is the name of the software. For example, to search for `lsof`:

  ```
  # cd /usr/ports
  # make search name=lsof
  Port:   lsof-4.56.4
  Path:   /usr/ports/sysutils/lsof
  Info:   Lists information about open files (similar to fstat(1))
  Maint:  obrien@FreeBSD.org
  Index:  sysutils
  B-deps:
  R-deps:
  ```

  > **Tip:** `make search` searches through a file of index information. If a message indicates the `INDEX` is required, run `make fetchindex` to download the current index file. With the `INDEX` present, `make search` will be able to perform the requested search.

  The "Path:" line indicates where to find the port.

  To receive less information, use the `quicksearch` feature:

  ```
  # cd /usr/ports
  # make quicksearch name=lsof
  Port:   lsof-4.87.a,7
  ```

```
Path:   /usr/ports/sysutils/lsof
Info:   Lists information about open files (similar to fstat(1))
```

For more in-depth searching, use `make search key=`*`string`* or `make quicksearch key=`*`string`*, where *`string`* is some text to search for. The text can be in comments, descriptions, or dependencies in order to find ports which relate to a particular subject when the name of the program is unknown.

When using (`search` and `quicksearch`), the search string is case-insensitive. Searching for “LSOF” will yield the same results as searching for “lsof”.

5.4 Using Binary Packages

Contributed by Chern Lee.

At the present time, FreeBSD is transitioning toward a new method of package management. Users of the latest releases may wish to investigate the benefits of using PKGng to manage third party software on FreeBSD. For those not yet migrated to the **pkgng** tool, the tools discussed here may be used for managing the package database. For simplicity, the `sysinstall` utility is also available post-install for package management.

All package installation files are stored in the package database directory, `/var/db/pkg`.

5.4.1 Installing a Package

Use pkg_add(1) to install a FreeBSD binary package from a local file or from a server on the network.

Example 5-1. Downloading a Package Manually and Installing It Locally

```
# ftp -a ftp2.FreeBSD.org
Connected to ftp2.FreeBSD.org.
220 ftp2.FreeBSD.org FTP server (Version 6.00LS) ready.
331 Guest login ok, send your email address as password.
230-
230-     This machine is in Vienna, VA, USA, hosted by Verio.
230-         Questions? E-mail freebsd@vienna.verio.net.
230-
230-
230 Guest login ok, access restrictions apply.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd /pub/FreeBSD/ports/packages/sysutils/
250 CWD command successful.
ftp> get lsof-4.56.4.tgz
local: lsof-4.56.4.tgz remote: lsof-4.56.4.tgz
200 PORT command successful.
150 Opening BINARY mode data connection for 'lsof-4.56.4.tgz' (92375 bytes).
100% |***********************************************| 92375       00:00 ETA
226 Transfer complete.
92375 bytes received in 5.60 seconds (16.11 KB/s)
ftp> exit
# pkg_add lsof-4.56.4.tgz
```

If you do not have a source of local packages, such as a FreeBSD CD-ROM set, include `-r` with pkg_add(1). This automatically determines the correct object format and release, and then fetches and installs the package from an FTP site without any further user intervention.

```
# pkg_add -r lsof
```

To specify an alternative FreeBSD FTP mirror, specify the mirror in the `PACKAGESITE` environment variable. pkg_add(1) uses fetch(3) to download files, which uses various environment variables, including `FTP_PASSIVE_MODE`, `FTP_PROXY`, and `FTP_PASSWORD`. You may need to set one or more of these if you are behind a firewall, or need to use an FTP/HTTP proxy. See fetch(3) for the complete list of variables. Note that in the example above `lsof` is used instead of `lsof-4.56.4`. When the remote fetching feature is used, the version number of the package must be removed.

> **Note:** pkg_add(1) will automatically download the latest version of the application if you are using FreeBSD-CURRENT or FreeBSD-STABLE. If you run a -RELEASE version, it instead installs the version of the package that was built with that release. It is possible to change this behavior by overriding `PACKAGESITE`. For example, on a FreeBSD 8.1-RELEASE system, by default pkg_add(1) will try to fetch packages from `ftp://ftp.freebsd.org/pub/FreeBSD/ports/i386/packages-8.1-release/Latest/`. To force pkg_add(1) to download FreeBSD 8-STABLE packages, set `PACKAGESITE` to `ftp://ftp.freebsd.org/pub/FreeBSD/ports/i386/packages-8-stable/Latest/`.

Package files are distributed in `.tgz` and `.tbz` formats. Packages are available from ftp://ftp.FreeBSD.org/pub/FreeBSD/ports/packages/, or the `/packages` directory of the FreeBSD DVD distribution. The layout of the packages directory is similar to that of the `/usr/ports` tree. Each category has its own directory, and every package can be found within the `All` directory.

5.4.2 Managing Packages

pkg_info(1) can be used to list and describe installed packages:

```
# pkg_info
colordiff-1.0.13    A tool to colorize diff output
docbook-1.2         Meta-port for the different versions of the DocBook DTD
...
```

pkg_version(1) summarizes the versions of all installed packages and compares the package version to the current version found in the ports tree.

```
# pkg_version
colordiff                   =
docbook                     =
...
```

The symbols in the second column indicate the relative age of the installed version and the version available in the local ports tree.

Symbol	Meaning

Symbol	Meaning
=	The version of the installed package matches the one in the local ports tree.
<	The version of the installed package is older than the one in the local ports tree.
>	The version of the installed package is newer than the one in the local ports tree, meaning that the local ports tree is probably out of date.
?	The installed package cannot be found in the ports index. This can happen when an installed port is removed from the Ports Collection or is renamed.
*	There are multiple versions of the package.
!	The installed package exists in the index but for some reason, `pkg_version` was unable to compare the version number of the installed package with the corresponding entry in the index.

5.4.3 Deleting a Package

To remove a previously installed software package, use pkg_delete(1):

```
# pkg_delete xchat-1.7.1
```

Note that pkg_delete(1) requires the full package name and number; the above command would not work if *`xchat`* was given instead of *`xchat-1.7.1`*. Use pkg_version(1) to find the version of the installed package, or use a wildcard:

```
# pkg_delete xchat\*
```

in this case, all packages whose names start with `xchat` will be deleted.

5.5 Using pkgng for Binary Package Management

pkgng is an improved replacement for the traditional FreeBSD package management tools, offering many features that make dealing with binary packages faster and easier. The first release of **pkgng** was in August, 2012.

pkgng is not a replacement for port management tools like `ports-mgmt/portmaster` or `ports-mgmt/portupgrade`. While `ports-mgmt/portmaster` and `ports-mgmt/portupgrade` can install third-party software from both binary packages and the Ports Collection, **pkgng** installs only binary packages.

5.5.1 Getting Started with pkgng

FreeBSD 9.1 and later includes a "bootstrap" utility for **pkgng**. The bootstrap utility will download and install **pkgng**.

To bootstrap the system, run:

```
# /usr/sbin/pkg
```

For earlier FreeBSD versions, **pkgng** must be installed from the Ports Collection, or as a binary package.

To install the **pkgng** port, run:

```
# cd /usr/ports/ports-mgmt/pkg
# make
# make install clean
```

To install the binary package, run:

```
# pkg_add -r pkg
```

Existing FreeBSD installations require conversion of the **pkg_install** package database to the new format. To convert the package database, run:

```
# pkg2ng
```

This step is not required for new installations that do not have third-party software installed.

> **Important:** This step is not reversible. Once the package database has been converted to the **pkgng** format, the **pkg_install** tools should not be used.

> **Note:** The package database conversion may emit errors as the contents are converted to the new version. Generally, these errors can be safely ignored, however a list of third-party software that was not successfully converted will be listed after `pkg2ng` has finished. These must be fixed by hand.

To ensure the FreeBSD Ports Collection registers new software with **pkgng**, and not **pkg_install**, FreeBSD versions earlier than 10.*X* require this line in `/etc/make.conf`:

```
WITH_PKGNG=     yes
```

5.5.2 Configuring the pkgng Environment

The **pkgng** package management system uses a package repository for most operations. The default package repository location is defined in `/usr/local/etc/pkg.conf` or the `PACKAGESITE` environment variable, which overrides the configuration file.

Additional **pkgng** configuration options are described in pkg.conf(5).

5.5.3 Basic pkgng Operations

Usage information for **pkgng** is available in the pkg(8) manual page, or by running `pkg` without additional arguments.

Each **pkgng** command argument is documented in a command-specific manual page. To read the manual page for `pkg install`, for example, run either:

```
# pkg help install

# man pkg-install
```

5.5.3.1 Obtaining Information About Installed Packages with pkgng

Information about the packages installed on a system can be viewed by running `pkg info`. Similar to pkg_info(1), the package version and description for all packages will be listed.

Information about a specific package is available by running:

```
# pkg info packagename
```

For example, to see which version of **pkgng** is installed on the system, run:

```
# pkg info pkg
pkg-1.0.2                      New generation package manager
```

5.5.3.2 Installing and Removing Packages with pkgng

In general, most FreeBSD users will install binary packages by running:

```
# pkg install packagename
```

`pkg install` uses repository data, as mentioned in Section 5.5.2. Conversely, pkg-add(8) does not use repository data, nor does it use the defined `PACKAGESITE`, so dependencies may not be properly tracked, and missing dependencies will not be fetched from a remote source. This section covers usage of `pkg install`. For information on usage of `pkg add`, see pkg-add(8).

Additional binary packages can be installed with `pkg install`. For example, to install **curl**:

```
# pkg install curl
Updating repository catalogue
Repository catalogue is up-to-date, no need to fetch fresh copy
The following packages will be installed:

       Installing ca_root_nss: 3.13.5
       Installing curl: 7.24.0

The installation will require 4 MB more space

1 MB to be downloaded

Proceed with installing packages [y/N]: y
ca_root_nss-3.13.5.txz         100%    255KB   255.1KB/s 255.1KB/s     00:00
curl-7.24.0.txz                        100%    1108KB  1.1MB/s 1.1MB/s         00:00
Checking integrity... done
Installing ca_root_nss-3.13.5... done
Installing curl-7.24.0... done
```

The new package and any additional packages that were installed as dependencies can be seen in the installed packages list:

```
# pkg info
ca_root_nss-3.13.5     The root certificate bundle from the Mozilla Project
curl-7.24.0    Non-interactive tool to get files from FTP, GOPHER, HTTP(S) servers
pkg-1.0.2      New generation package manager
```

Packages that are no longer needed can be removed with `pkg delete`. For example, if it turns out that **curl** is not needed after all:

```
# pkg delete curl
The following packages will be deleted:

       curl-7.24.0_1

The deletion will free 3 MB

Proceed with deleting packages [y/N]: y
Deleting curl-7.24.0_1... done
```

5.5.3.3 Upgrading Installed Packages with pkgng

Packages that are outdated can be found with `pkg version`. If a local ports tree does not exist, pkg-version(8) will use the remote repository catalogue, otherwise the local ports tree will be used to identify package versions.

Packages can be upgraded to newer versions with **pkgng**. Suppose a new version of **curl** has been released. The local package can be upgraded to the new version:

```
# pkg upgrade
Updating repository catalogue
repo.txz               100%    297KB 296.5KB/s 296.5KB/s       00:00
The following packages will be upgraded:

       Upgrading curl: 7.24.0 -> 7.24.0_1

1 MB to be downloaded

Proceed with upgrading packages [y/N]: y
curl-7.24.0_1.txz      100% 1108KB     1.1MB/s 1.1MB/s         00:00
Checking integrity... done
Upgrading curl from 7.24.0 to 7.24.0_1... done
```

5.5.3.4 Auditing Installed Packages with pkgng

Occasionally, software vulnerabilities may be discovered in software within the Ports Collection. **pkgng** includes built-in auditing, similar to the `ports-mgmt/portaudit` package. To audit the software installed on the system, run:

```
# pkg audit -F
```

5.5.4 Advanced pkgng Operations

5.5.4.1 Automatically Removing Leaf Dependencies with pkgng

Removing a package may leave behind unnecessary dependencies, like `security/ca_root_nss` in the example above. Such packages are still installed, but nothing depends on them any more. Unneeded packages that were installed as dependencies can be automatically detected and removed:

```
# pkg autoremove
Packages to be autoremoved:
       ca_root_nss-3.13.5

The autoremoval will free 723 kB

Proceed with autoremoval of packages [y/N]: y
Deinstalling ca_root_nss-3.13.5... done
```

5.5.4.2 Backing Up the pkgng Package Database

Unlike the traditional package management system, **pkgng** includes its own package database backup mechanism. To manually back up the package database contents, run:

```
# pkg backup -d pkgng.db
```

> **Note:** Replace the file name *pkgng.db* to a suitable file name.

Additionally, **pkgng** includes a periodic(8) script to automatically back up the package database daily if `daily_backup_pkgng_enable` is set to `YES` in periodic.conf(5).

> **Tip:** To prevent the **pkg_install** periodic script from also backing up the package database, set `daily_backup_pkgdb_enable` to `NO` in periodic.conf(5).

To restore the contents of a previous package database backup, run:

```
# pkg backup -r /path/to/pkgng.db
```

5.5.4.3 Removing Stale pkgng Packages

By default, **pkgng** stores binary packages in a cache directory as defined by `PKG_CACHEDIR` in pkg.conf(5). When upgrading packages with `pkg upgrade`, old versions of the upgraded packages are not automatically removed.

To remove the outdated binary packages, run:

```
# pkg clean
```

5.5.4.4 Modifying pkgng Package Metadata

Historically, software within the FreeBSD Ports Collection can undergo major version number changes. Unlike **pkg_install**, **pkgng** has a built-in command to update package origins. For example, if `lang/php5` was originally at version `5.3`, but has been renamed to `lang/php53` for the inclusion of version `5.4`, **pkg_install** would require the use of additional software such as `ports-mgmt/portmaster` to update the package database, reflecting from which port the installation originated.

Unlike the `ports-mgmt/portmaster` and `ports-mgmt/portupgrade` ports, the order in which the new and old versions are listed differ. For **pkgng**, the syntax is:

```
# pkg set -o category/oldport:category/newport
```

For example, to change the package origin for the above example, run:

```
# pkg set -o lang/php5:lang/php53
```

As another example, to update `lang/ruby18` to `lang/ruby19`, run:

```
# pkg set -o lang/ruby18:lang/ruby19
```

As a final example, to change the origin of the `libglut` shared libraries from `graphics/libglut` to `graphics/freeglut`, run:

```
# pkg set -o graphics/libglut:graphics/freeglut
```

> **Note:** When changing package origins, in most cases it is important to reinstall packages that are dependent on the package that has had the origin changed. To force a reinstallation of dependent packages, run:
>
> ```
> # pkg install -Rf graphics/freeglut
> ```

5.6 Using the Ports Collection

This section provides basic instructions on using the Ports Collection to install or remove software. The detailed description of available `make` targets and environment variables is available in ports(7).

> **Warning:** As of mid 2012, the FreeBSD Ports Project has migrated revision control systems from CVS to Subversion. The preferred method for obtaining and maintaining the ports tree is **Portsnap**. Users requiring local customization of ports (that is, maintaining additional local patches) will probably prefer to use Subversion directly. The **CVSup** service was phased out as of February 28, 2013.

5.6.1 Obtaining the Ports Collection

The Ports Collection is a set of `Makefiles`, patches, and description files stored in `/usr/ports`. This set of files is used to compile and install applications on FreeBSD. The instructions below show several methods of obtaining the Ports Collection if it was not installed during initial FreeBSD setup.

Portsnap Method

Portsnap is a fast and user-friendly tool for retrieving the Ports Collection, the preferred choice for most users. See Using Portsnap for a detailed description of **Portsnap**.

1. Download a compressed snapshot of the Ports Collection into `/var/db/portsnap`.

   ```
   # portsnap fetch
   ```

2. When running **Portsnap** for the first time, extract the snapshot into `/usr/ports`:

   ```
   # portsnap extract
   ```

3. After the first use of **Portsnap** has been completed as shown above, `/usr/ports` can be updated with:

   ```
   # portsnap fetch
   # portsnap update
   ```

Subversion Method

If more control over the ports tree is needed (for example, for maintaining local changes), **Subversion** can be used to obtain the Ports Collection. Refer to the Subversion Primer (http://www.FreeBSD.org/doc/en_US.ISO8859-1/articles/committers-guide/subversion-primer.html) for a detailed description of **Subversion**.

1. **Subversion** must be installed before it can be used to check out the ports tree. If a copy of the ports tree is already present, install **Subversion** like this:

   ```
   # cd /usr/ports/devel/subversion
   # make install clean
   ```

   If the ports tree is not available, **Subversion** can be installed as a package:

   ```
   # pkg_add -r subversion
   ```

   If **pkgng** is being used to manage packages, **Subversion** can be installed with it instead:

   ```
   # pkg install subversion
   ```

2. Check out a copy of the ports tree. Use a specific Subversion mirror (http://www.FreeBSD.org/doc/en_US.ISO8859-1/books/handbook/svn-mirrors.html) close to your geographic location instead of *`svn0.us-east.FreeBSD.org`* in the command below for better performance. Committers should read the Subversion Primer (http://www.FreeBSD.org/doc/en_US.ISO8859-1/articles/committers-guide/subversion-primer.html) first to be sure the correct protocol is chosen.

   ```
   # svn checkout https://svn0.us-east.FreeBSD.org/ports/head /usr/ports
   ```

3. To update `/usr/ports` after the initial **Subversion** checkout:

   ```
   # svn update /usr/ports
   ```

Sysinstall Method

This method involves using **sysinstall** to install the Ports Collection from the installation media. Note that the old copy of Ports Collection from the date of the release will be installed. If you have Internet access, you should always use one of the methods mentioned above.

1. As `root`, run `sysinstall` as shown below:

   ```
   # sysinstall
   ```

2. Scroll down and select Configure, press **Enter**.
3. Scroll down and select Distributions, press **Enter**.
4. Scroll down to ports, press **Space**.
5. Scroll up to Exit, press **Enter**.
6. Select your desired installation media, such as CDROM, FTP, and so on.
7. Scroll up to Exit and press **Enter**.
8. Press **X** to exit **sysinstall**.

5.6.2 Migrating from CVSup/csup to portsnap

> **Warning:** By February 28, 2013, the ports tree will no longer be exported to **CVS** and therefore **CVSup** and **csup** will no longer provide updates for the ports tree.

Migration to Portsnap

The migration will require about 1 GB of disk space on `/usr`, plus **Portsnap** requires about 150 MB disk space on `/var`.

1. Disable any automated ports updates you may use, such as a cron(8) job calling **CVSup** or **csup**.
2. Move the existing ports tree to a temporary location:

   ```
   # mv /usr/ports /usr/ports.old
   ```

3. Fetch the new ports tree with **Portsnap** and extract it to `/usr/ports`:

   ```
   # portsnap fetch extract
   ```

4. Move distfiles and saved packages to the new ports tree:

   ```
   # mv /usr/ports.old/distfiles /usr/ports
   # mv /usr/ports.old/packages /usr/ports
   ```

5. Delete the old ports tree:

   ```
   # rm -rf /usr/ports.old
   ```

6. If **CVSup** was used before, it can now be uninstalled:

   ```
   # pkg_delete -r -v cvsup-without-gui-\*
   ```

   Users of **pkgng** can use the following command:

```
# pkg delete cvsup-without-gui
```

See Using Portsnap for a detailed description of **Portsnap** and how to update the ports tree with **Portsnap**.

5.6.3 Installing Ports

A port skeleton is a set of files that tell FreeBSD system how to compile and install a program. Each port skeleton includes:

- `Makefile`: The `Makefile` contains statements that specify how the application should be compiled and where its components should be installed.
- `distinfo`: This file contains information about the files that must be downloaded to build the port, and their checksums (using sha256(1)), to verify that files have not been corrupted during the download.
- `files/`: This directory contains any patches needed for the program to compile and install on FreeBSD. This directory may also contain other files used to build the port.
- `pkg-descr`: This file provides a more detailed description of the program.
- `pkg-plist`: This is a list of all the files that will be installed by the port. It also tells the ports system what files to remove upon deinstallation.

Some ports include other files, such as `pkg-message`. The ports system uses these files to handle special situations. If you want more details on these files, and on ports in general, refer to the FreeBSD Porter's Handbook (http://www.FreeBSD.org/doc/en_US.ISO8859-1/books/porters-handbook/index.html).

The port does not include the actual source code, also known as a "distfile". Source code is distributed in whatever manner the software author desires. The two methods for installing a FreeBSD port are described below.

> **Note:** You must be logged in as `root` to install ports.

> **Warning:** Before compiling any port, be sure to have an up-to-date Ports Collection and check http://vuxml.freebsd.org/ for security issues related to your port. If `ports-mgmt/portaudit` is installed, run `portaudit -F` before installing a new port, to fetch the current vulnerabilities database. A security audit and an update of the database will be performed during the daily security system check. For more information read the portaudit(1) and periodic(8) manual pages.

Using the Ports Collection assumes a working Internet connection. Otherwise, manually obtain and place a copy of the distfile into `/usr/ports/distfiles`.

To begin, change to the directory of the port to be installed:

```
# cd /usr/ports/sysutils/lsof
```

To compile, or "build", the port, type `make` at the prompt. You should see messages similar to the ones in this example:

```
# make
>> lsof_4.57D.freebsd.tar.gz doesn't seem to exist in /usr/ports/distfiles/.
```

```
>> Attempting to fetch from ftp://lsof.itap.purdue.edu/pub/tools/unix/lsof/.
===>  Extracting for lsof-4.57
...
[extraction output snipped]
...
>> Checksum OK for lsof_4.57D.freebsd.tar.gz.
===>  Patching for lsof-4.57
===>  Applying FreeBSD patches for lsof-4.57
===>  Configuring for lsof-4.57
...
[configure output snipped]
...
===>  Building for lsof-4.57
...
[compilation output snipped]
...
#
```

Once the compile is complete, you are returned to the prompt. The next step is to install the port using `make install`:

```
# make install
===>  Installing for lsof-4.57
...
[installation output snipped]
...
===>   Generating temporary packing list
===>   Compressing manual pages for lsof-4.57
===>   Registering installation for lsof-4.57
===>  SECURITY NOTE:
      This port has installed the following binaries which execute with
      increased privileges.
#
```

Once you are returned to the prompt, you should be able to run the installed application. Since `lsof` is a program that runs with increased privileges, a security warning is shown. During the building and installation of ports, take heed of any other warnings that may appear.

It is a good idea to delete the working subdirectory, which contains all the temporary files used during compilation. Doing so saves disk space and minimizes the chance of problems later when upgrading to the newer version of the port.

```
# make clean
===>  Cleaning for lsof-4.57
#
```

> **Note:** You can save two extra steps by just running `make install clean` instead of `make`, `make install` and `make clean` as three separate steps.

Note: Using only `make install` means there will potentially be many waiting periods between user interaction as the default behaviour is to prompt the user for options. To avoid this when there are many dependencies, first run `make config-recursive` to do the configuration in one batch. Then run `make install [clean]` afterwards.

Tip: When using `config-recursive`, the list of ports to configure are gathered by the `all-depends-list` make(1) target. It is often recommended to run `make config-recursive` until all dependent ports options have been defined, and ports options dialog(1) screens no longer appear, to be certain all ports options have been configured as intended.

Note: Some shells keep a cache of the commands that are available in the directories listed in the `PATH` environment variable, to speed up lookup operations for the executable file of these commands. If you are using `tcsh`, you might have to type `rehash` so that a newly installed command can be used without specifying its full path. Use `hash -r` instead for the `sh` shell. Refer to the documentation for the shell for more information.

Some third-party DVD products such as the FreeBSD Toolkit from the FreeBSD Mall (http://www.freebsdmall.com/) contain distfiles. They can be used with the Ports Collection. Mount the DVD on `/cdrom`. If you use a different mount point, set `CD_MOUNTPTS` make variable. The needed distfiles will be automatically used if they are present on the disk.

Note: The licenses of a few ports do not allow their inclusion on the DVD. This could be because a registration form needs to be filled out before downloading or redistribution is not allowed. If you wish to install a port not included on the DVD, you will need to be connected to the Internet.

The ports system uses fetch(1) to download the files, which honors various environment variables, including `FTP_PASSIVE_MODE`, `FTP_PROXY`, and `FTP_PASSWORD`. You may need to set one or more of these if you are behind a firewall, or need to use an FTP/HTTP proxy. See fetch(3) for the complete list.

For users which cannot be connected all the time, the `make fetch` option is provided. Run this command within `/usr/ports` and the required files will be downloaded. This command also works in the lower level categories, such as `/usr/ports/net`. Note that if a port depends on libraries or other ports, this will *not* fetch the distfiles of ports from another category. Use `make fetch-recursive` to fetch all the dependencies of a port.

Note: You can build all the ports in a category or as a whole by running `make` in the top level directory. This is dangerous, however, as some ports cannot co-exist. In other cases, some ports can install two different files with the same filename.

In some rare cases, users may need to acquire the tarballs from a site other than the default `MASTER_SITES`. You can override the `MASTER_SITES` option with the following command:

```
# cd /usr/ports/directory
# make MASTER_SITE_OVERRIDE= \
ftp://ftp.FreeBSD.org/pub/FreeBSD/ports/distfiles/ fetch
```

In this example, `MASTER_SITES` is changed to `ftp.FreeBSD.org/pub/FreeBSD/ports/distfiles/`.

> **Note:** Some ports provide build options which can be used to enable/disable parts of the application which are unneeded, provide security options, or allow for other customizations. Examples include `www/firefox`, `security/gpgme`, and `mail/sylpheed-claws`. A menu will be displayed at the beginning of a port compile when compile options are available.

5.6.3.1 Overriding the Default Ports Directories

The `WRKDIRPREFIX` and `PREFIX` variables can override the default working and target directories. For example:

```
# make WRKDIRPREFIX=/usr/home/example/ports install
```

will compile the port in `/usr/home/example/ports` and install everything under `/usr/local`.

```
# make PREFIX=/usr/home/example/local install
```

will compile the port in `/usr/ports` and install it in `/usr/home/example/local`.

And

```
# make WRKDIRPREFIX=../ports PREFIX=../local install
```

will combine the two.

Alternatively, these can be set as environmental variables. Refer to the manual page for your shell for instructions on how to set an environmental variable.

5.6.3.2 Reconfiguring Ports

Certain ports provide an ncurses-based menu containing build options. There are several ways to revisit this menu in order to add, remove, or change these options after a port has been built. One method is to `cd` into the directory containing the port and type `make config`. Another option is to use `make showconfig`. Another option is to execute `make rmconfig` which will remove all selected options and allow you to start over. All of these options, and others, are explained in great detail in the manual page for ports(7).

5.6.4 Removing Installed Ports

Installed ports and packages are uninstalled using the pkg_delete(1) command:

```
# pkg_delete lsof-4.57
```

5.6.5 Upgrading Ports

First, list outdated ports that have a newer version available in the Ports Collection with the pkg_version(1) command:

```
# pkg_version -v
```

5.6.5.1 Read `/usr/ports/UPDATING`

Once you have updated your Ports Collection, before attempting a port upgrade, you should check `/usr/ports/UPDATING`. This file describes various issues and additional steps users may encounter and need to perform when updating a port, including such things as file format changes, changes in locations of configuration files, or other such incompatibilities with previous versions.

If `UPDATING` contradicts something you read here, `UPDATING` takes precedence.

5.6.5.2 Upgrading Ports Using Portupgrade

The **portupgrade** utility is designed to easily upgrade installed ports. It is available from the `ports-mgmt/portupgrade` port. Install it like any other port, using `make install clean`:

```
# cd /usr/ports/ports-mgmt/portupgrade
# make install clean
```

Scan the list of installed ports using `pkgdb -F` and fix all the inconsistencies it reports. It is a good idea to do this regularly, before every upgrade.

Use `portupgrade -a` to upgrade all the outdated ports installed on the system. Include `-i` to be asked for confirmation of every individual upgrade.

```
# portupgrade -ai
```

To upgrade only a specified application instead of all available ports, use `portupgrade` *`pkgname`*. Include `-R` to first upgrade all the ports required by the given application.

```
# portupgrade -R firefox
```

To use packages instead of ports, include the `-P` flag. With this option, **portupgrade** searches the local directories listed in `PKG_PATH`, then fetches packages from a remote site if not found locally. If packages can not be found locally or fetched remotely, **portupgrade** will use ports. To avoid using ports, specify `-PP`.

```
# portupgrade -PP gnome2
```

To just fetch distfiles (or packages, if `-P` is specified) without building or installing anything, use `-F`. For further information see portupgrade(1).

5.6.5.3 Upgrading Ports Using portmaster

`ports-mgmt/portmaster` is another utility for upgrading installed ports. **portmaster** was designed to use the tools found in the “base” system without depending upon other ports. It uses the information in `/var/db/pkg/` to determine which ports to upgrade. To install the port:

```
# cd /usr/ports/ports-mgmt/portmaster
# make install clean
```

Portmaster groups ports into four categories:

- Root ports: no dependencies and is not depended on by other ports

- Trunk ports: no dependencies, but other ports depend upon it
- Branch ports: have dependencies and are depended upon by other ports
- Leaf ports: have dependencies but are not depended upon by other ports

To list all installed software and search for updates, use `-L`:

```
# portmaster -L
===>>> Root ports (No dependencies, not depended on)
===>>> ispell-3.2.06_18
===>>> screen-4.0.3
        ===>>> New version available: screen-4.0.3_1
===>>> tcpflow-0.21_1
===>>> 7 root ports
...
===>>> Branch ports (Have dependencies, are depended on)
===>>> apache22-2.2.3
        ===>>> New version available: apache22-2.2.8
...
===>>> Leaf ports (Have dependencies, not depended on)
===>>> automake-1.9.6_2
===>>> bash-3.1.17
        ===>>> New version available: bash-3.2.33
...
===>>> 32 leaf ports

===>>> 137 total installed ports
        ===>>> 83 have new versions available
```

All the installed ports can be upgraded using this command:

```
# portmaster -a
```

> **Note:** By default, **portmaster** will make a backup package before deleting the existing port. If the installation of the new version is successful, **portmaster** will delete the backup. Using `-b` will instruct **portmaster** not to automatically delete the backup. Adding `-i` will start **portmaster** in interactive mode, prompting for confirmation before upgrading each port.

If you encounter errors during the upgrade process, use `-f` to upgrade/rebuild all ports:

```
# portmaster -af
```

You can also use **portmaster** to install new ports on the system, upgrading all dependencies before building and installing the new port:

```
# portmaster shells/bash
```

Refer to portmaster(8) for more information.

5.6.6 Ports and Disk Space

Using the Ports Collection will use up disk space over time. After building and installing a port, `make clean` will clean up the temporary `work` directory. To sweep the whole Ports Collection:

```
# portsclean -C
```

A lot of out-dated source distribution files will collect in `distfiles` over time. The following command will delete all the distfiles that are no longer referenced by any ports:

```
# portsclean -D
```

To remove all distfiles not referenced by any port currently installed on the system:

```
# portsclean -DD
```

> **Note:** The `portsclean` utility is part of the `ports-mgmt/portupgrade` suite.

`ports-mgmt/pkg_cutleaves` automates the task of removing installed ports that are no longer needed.

5.7 Working With Installed Ports

Most third party applications will need some level of configuration after they were installed. This may be a simple configuration file alteration, or perhaps the application will just generate a configuration file. Most applications will have documentation installed into `/usr/local/share/doc` and manual pages. This documentation should be consulted before continuing. Some applications run services which must be added to the `/etc/rc.conf` file before starting.

The following list contains useful information for post-install port management. In several cases, finding the location of binaries if they were installed outside of the `PATH`. Users of csh(1) should run `rehash` to rebuild the known binary list in the shells `PATH`.

- The pkg_info(1) command will print all installed files and their location. For example, if the FooPackage version 1.0.0 was just installed, then the following command will show all the files installed with the package.

  ```
  # pkg_info -L foopackage-1.0.0 | less
  ```

  Configuration files are always installed in `/usr/local/etc` and should definitely be consulted before attempting to use the new application.

  To determine which version of the application was installed:

  ```
  # pkg_info | grep -i foopackage
  ```

  will find all the installed packages that have *`foopackage`* in the package name. Replace *`foopackage`* as necessary.

- These commands will also show the names of any manual pages installed with the application. This additional documentation will now be available to the man(1) command.

- If the application has a web site, consult it for additional documentation or a frequently asked questions page. If the website is unknown, the following command will be useful to print out this information if it's available.

  ```
  # pkg_info foopackage-1.0.0
  ```

  A `WWW:` line, if present, should provide a URL for the application's web site.

- Ports that should start at boot time usually install a startup script in `/usr/local/etc/rc.d`. Review this script for correctness and edit or rename it if needed. See Starting Services for more information.

5.8 Dealing with Broken Ports

When coming across a port that does not build or install:

1. Find out if there is a fix pending for the port in the Problem Report database (http://www.FreeBSD.org/support.html#gnats). If so, the proposed fix may work.
2. Ask the maintainer of the port for help. Type `make maintainer` or read the `Makefile` to find the maintainer's email address. Remember to include the name and version of the port (send the `$FreeBSD:` line from the `Makefile`) and the output leading up to the error when you email the maintainer.

   > **Note:** Some ports are not maintained by an individual but instead by a mailing list (http://www.FreeBSD.org/doc/en_US.ISO8859-1/articles/mailing-list-faq/article.html). Many, but not all, of these addresses look like `<freebsd-listname@FreeBSD.org>`. Please take this into account when phrasing your questions.
   >
   > In particular, ports shown as maintained by `<ports@FreeBSD.org>` are actually not maintained by anyone. Fixes and support, if any, come from the general community who subscribe to that mailing list. More volunteers are always needed!

   If you do not get a response, use send-pr(1) to submit a bug report (see Writing FreeBSD Problem Reports (http://www.FreeBSD.org/doc/en_US.ISO8859-1/articles/problem-reports/article.html)).
3. Fix it! The Porter's Handbook (http://www.FreeBSD.org/doc/en_US.ISO8859-1/books/porters-handbook/index.html) includes detailed information on the "Ports" infrastructure so that you can fix the occasional broken port or even submit your own!
4. Use pkg_add(1) to instead install the package.

Chapter 6 The X Window System

Updated for X.Org's X11 server by Ken Tom and Marc Fonvieille.

6.1 Synopsis

FreeBSD uses X11 to provide users with a powerful graphical user interface. X11 is a freely available version of the X Window System that is implemented in **Xorg** (and other software packages not discussed here). The default and official flavor of X11 in FreeBSD is **Xorg**, the X11 server developed by the X.Org Foundation under a license very similar to the one used by FreeBSD.

For more information on the video hardware that X11 supports, check the Xorg (http://www.x.org/) web site.

After reading this chapter, you will know:

- The various components of the X Window System, and how they interoperate.
- How to install and configure X11.
- How to install and use different window managers.
- How to use TrueType® fonts in X11.
- How to set up your system for graphical logins (**XDM**).

Before reading this chapter, you should:

- Know how to install additional third-party software (Chapter 5).

6.2 Understanding X

Using X for the first time can be somewhat of a shock to someone familiar with other graphical environments, such as Microsoft Windows or Mac OS®.

While it is not necessary to understand all of the details of various X components and how they interact, some basic knowledge makes it possible to take advantage of X's strengths.

6.2.1 Why X?

X is not the first window system written for UNIX, but it is the most popular of them. X's original development team had worked on another window system prior to writing X. That system's name was "W" (for "Window"). X was just the next letter in the Roman alphabet.

X can be called "X", "X Window System", "X11", and a number of other terms. You may find that using the term "X Windows" to describe X11 can be offensive to some people; for a bit more insight on this, see X(7).

6.2.2 The X Client/Server Model

X was designed from the beginning to be network-centric, and adopts a "client-server" model.

In the X model, the "X server" runs on the computer that has the keyboard, monitor, and mouse attached. The server's responsibility includes tasks such as managing the display, handling input from the keyboard and mouse, and other input or output devices (i.e., a "tablet" can be used as an input device, and a video projector may be an alternative output device). Each X application (such as **XTerm** or **Firefox**) is a "client". A client sends messages to the server such as "Please draw a window at these coordinates", and the server sends back messages such as "The user just clicked on the OK button".

In a home or small office environment, the X server and the X clients commonly run on the same computer. However, it is perfectly possible to run the X server on a less powerful desktop computer, and run X applications (the clients) on, say, the powerful and expensive machine that serves the office. In this scenario the communication between the X client and server takes place over the network.

This confuses some people, because the X terminology is exactly backward to what they expect. They expect the "X server" to be the big powerful machine down the hall, and the "X client" to be the machine on their desk.

It is important to remember that the X server is the machine with the monitor and keyboard, and the X clients are the programs that display the windows.

There is nothing in the protocol that forces the client and server machines to be running the same operating system, or even to be running on the same type of computer. It is certainly possible to run an X server on Microsoft Windows or Apple's Mac OS, and there are various free and commercial applications available that do exactly that.

6.2.3 The Window Manager

The X design philosophy is much like the UNIX design philosophy, "tools, not policy". This means that X does not try to dictate how a task is to be accomplished. Instead, tools are provided to the user, and it is the user's responsibility to decide how to use those tools.

This philosophy extends to X not dictating what windows should look like on screen, how to move them around with the mouse, what keystrokes should be used to move between windows (i.e., **Alt**+**Tab**, in the case of Microsoft Windows), what the title bars on each window should look like, whether or not they have close buttons on them, and so on.

Instead, X delegates this responsibility to an application called a "Window Manager". There are dozens of window managers (http://xwinman.org/) available for X. Each of these window managers provides a different look and feel; some of them support "virtual desktops"; some of them allow customized keystrokes to manage the desktop; some have a "Start" button or similar device; some are "themeable", allowing a complete change of look-and-feel by applying a new theme. Window managers are available in the `x11-wm` category of the Ports Collection.

In addition, the **KDE** and **GNOME** desktop environments both have their own window managers which integrate with the desktop.

Each window manager also has a different configuration mechanism; some expect configuration file written by hand, others feature GUI tools for most of the configuration tasks; at least one (**Sawfish**) has a configuration file written in a dialect of the Lisp language.

> **Focus Policy:** Another feature the window manager is responsible for is the mouse "focus policy". Every windowing system needs some means of choosing a window to be actively receiving keystrokes, and should visibly indicate which window is active as well.
>
> A familiar focus policy is called "click-to-focus". This is the model utilized by Microsoft Windows, in which a window becomes active upon receiving a mouse click.

X does not support any particular focus policy. Instead, the window manager controls which window has the focus at any one time. Different window managers will support different focus methods. All of them support click to focus, and the majority of them support several others.

The most popular focus policies are:

focus-follows-mouse

The window that is under the mouse pointer is the window that has the focus. This may not necessarily be the window that is on top of all the other windows. The focus is changed by pointing at another window, there is no need to click in it as well.

sloppy-focus

This policy is a small extension to focus-follows-mouse. With focus-follows-mouse, if the mouse is moved over the root window (or background) then no window has the focus, and keystrokes are simply lost. With sloppy-focus, focus is only changed when the cursor enters a new window, and not when exiting the current window.

click-to-focus

The active window is selected by mouse click. The window may then be “raised”, and appear in front of all other windows. All keystrokes will now be directed to this window, even if the cursor is moved to another window.

Many window managers support other policies, as well as variations on these. Be sure to consult the documentation for the window manager itself.

6.2.4 Widgets

The X approach of providing tools and not policy extends to the widgets seen on screen in each application.

“Widget” is a term for all the items in the user interface that can be clicked or manipulated in some way; buttons, check boxes, radio buttons, icons, lists, and so on. Microsoft Windows calls these “controls”.

Microsoft Windows and Apple’s Mac OS both have a very rigid widget policy. Application developers are supposed to ensure that their applications share a common look and feel. With X, it was not considered sensible to mandate a particular graphical style, or set of widgets to adhere to.

As a result, do not expect X applications to have a common look and feel. There are several popular widget sets and variations, including Qt, used by **KDE**, and GTK+, used by the **GNOME** project. In this respect, there is some convergence in look-and-feel of the UNIX desktop, which certainly makes things easier for the novice user.

6.3 Installing X11

Xorg is the X11 implementation for FreeBSD. **Xorg** is the X server of the open source X Window System implementation released by the X.Org Foundation. **Xorg** is based on the code of **XFree86 4.4RC2** and X11R6.6. The version of **Xorg** currently available in the FreeBSD Ports Collection is 7.7.

To build and install **Xorg** from the Ports Collection:

```
# cd /usr/ports/x11/xorg
```

```
# make install clean
```

Note: To build **Xorg** in its entirety, be sure to have at least 4 GB of free space available.

Alternatively, X11 can be installed directly from packages. Binary packages to use with pkg_add(1) tool are also available for X11. When the remote fetching feature of pkg_add(1) is used, the version number of the package must be removed. pkg_add(1) will automatically fetch the latest version of the application.

So to fetch and install the package of **Xorg**, simply type:

```
# pkg_add -r xorg
```

Note: The examples above will install the complete X11 distribution including the servers, clients, fonts etc. Separate packages and ports of X11 are also available.

To install a minimal X11 distribution you can alternatively install `x11/xorg-minimal`.

The rest of this chapter will explain how to configure X11, and how to set up a productive desktop environment.

6.4 X11 Configuration

Contributed by Christopher Shumway.

6.4.1 Before Starting

In most cases, X11 is self-configuring. Those with older or unusual equipment may find it helpful to gather some hardware information before beginning configuration.

- Monitor sync frequencies
- Video card chipset
- Video card memory

Screen resolution and refresh rate are determined by the monitor's horizontal and vertical sync frequencies. Almost all monitors support electronic autodetection of these values. A few monitors do not provide these values, and the specifications must be determined from the printed manual or manufacturer web site.

The video card chipset is also autodetected, and used to select the proper video driver. It is beneficial for the user to be aware of which chipset is installed for when autodetection does not provide the desired result.

Video card memory determines the maximum resolution and color depth which can be displayed.

6.4.2 Configuring X11

Xorg uses HAL to autodetect keyboards and mice. The `sysutils/hal` and `devel/dbus` ports are installed as dependencies of `x11/xorg`, but must be enabled by the following entries in the `/etc/rc.conf` file:

```
hald_enable="YES"
dbus_enable="YES"
```

These services should be started (either manually or by rebooting) before further **Xorg** configuration or use is attempted.

Xorg can often work without any further configuration steps by simply typing at prompt:

```
% startx
```

The automatic configuration may fail to work with some hardware, or may not set things up quite as desired. In these cases, manual configuration will be necessary.

> **Note:** Desktop environments like **GNOME**, **KDE** or **Xfce** have tools allowing the user to easily set the screen parameters such as the resolution. So if the default configuration is not acceptable and you planned to install a desktop environment then just continue with the installation of the desktop environment and use the appropriate screen settings tool.

Configuration of X11 is a multi-step process. The first step is to build an initial configuration file. As the super user, simply run:

```
# Xorg -configure
```

This will generate an X11 configuration skeleton file in the `/root` directory called `xorg.conf.new` (whether you su(1) or do a direct login affects the inherited supervisor `$HOME` directory variable). The X11 program will attempt to probe the graphics hardware on the system and write a configuration file to load the proper drivers for the detected hardware on the target system.

The next step is to test the existing configuration to verify that **Xorg** can work with the graphics hardware on the target system. Type:

```
# Xorg -config xorg.conf.new -retro
```

If a black and grey grid and an X mouse cursor appear, the configuration was successful. To exit the test, switch to the virtual console used to start it by pressing **Ctrl**+**Alt**+**F*n*** (**F1** for the first virtual console) and press **Ctrl**+**C**.

> **Note:** The **Ctrl**+**Alt**+**Backspace** key combination may also be used to break out of **Xorg**. To enable it, you can either type the following command from any X terminal emulator:
>
> ```
> % setxkbmap -option terminate:ctrl_alt_bksp
> ```
>
> or create a keyboard configuration file for **hald** called `x11-input.fdi` and saved in the `/usr/local/etc/hal/fdi/policy` directory. This file should contain the following lines:
>
> ```
> <?xml version="1.0" encoding="iso-8859-1"?>
> <deviceinfo version="0.2">
>   <device>
>     <match key="info.capabilities" contains="input.keyboard">
>          <merge key="input.x11_options.XkbOptions" type="string">terminate:ctrl_alt_bksp</merge>
>     </match>
>   </device>
> </deviceinfo>
> ```

You will have to reboot your machine to force **hald** to read this file.

The following line will also have to be added to `xorg.conf.new`, in the `ServerLayout` or `ServerFlags` section:

```
Option  "DontZap"       "off"
```

If the mouse does not work, you will need to first configure it before proceeding. See Section 3.10.9 in the FreeBSD install chapter. In recent **Xorg** versions, the `InputDevice` sections in `xorg.conf` are ignored in favor of the autodetected devices. To restore the old behavior, add the following line to the `ServerLayout` or `ServerFlags` section of this file:

```
Option "AutoAddDevices" "false"
```

Input devices may then be configured as in previous versions, along with any other options needed (e.g., keyboard layout switching).

Note: As previously explained the **hald** daemon will, by default, automatically detect your keyboard. There are chances that your keyboard layout or model will not be correct, desktop environments like **GNOME**, **KDE** or **Xfce** provide tools to configure the keyboard. However, it is possible to set the keyboard properties directly either with the help of the setxkbmap(1) utility or with a **hald**'s configuration rule.

For example if one wants to use a PC 102 keys keyboard coming with a french layout, we have to create a keyboard configuration file for **hald** called `x11-input.fdi` and saved in the `/usr/local/etc/hal/fdi/policy` directory. This file should contain the following lines:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<deviceinfo version="0.2">
  <device>
    <match key="info.capabilities" contains="input.keyboard">
         <merge key="input.x11_options.XkbModel" type="string">pc102</merge>
         <merge key="input.x11_options.XkbLayout" type="string">fr</merge>
    </match>
  </device>
</deviceinfo>
```

If this file already exists, just copy and add to your file the lines regarding the keyboard configuration.

You will have to reboot your machine to force **hald** to read this file.

It is possible to do the same configuration from an X terminal or a script with this command line:

```
% setxkbmap -model pc102 -layout fr
```

The `/usr/local/share/X11/xkb/rules/base.lst` file lists the various keyboard, layouts and options available.

The `xorg.conf.new` configuration file may now be tuned to taste. Open the file in a text editor such as emacs(1) or ee(1). If the monitor is an older or unusual model that does not support autodetection of sync frequencies, those settings can be added to `xorg.conf.new` under the `"Monitor"` section:

```
Section "Monitor"
       Identifier   "Monitor0"
       VendorName   "Monitor Vendor"
       ModelName    "Monitor Model"
```

```
        HorizSync    30-107
        VertRefresh  48-120
EndSection
```

Most monitors support sync frequency autodetection, making manual entry of these values unnecessary. For the few monitors that do not support autodetection, avoid potential damage by only entering values provided by the manufacturer.

X allows DPMS (Energy Star) features to be used with capable monitors. The xset(1) program controls the time-outs and can force standby, suspend, or off modes. If you wish to enable DPMS features for your monitor, you must add the following line to the monitor section:

```
Option       "DPMS"
```

While the `xorg.conf.new` configuration file is still open in an editor, select the default resolution and color depth desired. This is defined in the `"Screen"` section:

```
Section "Screen"
        Identifier "Screen0"
        Device     "Card0"
        Monitor    "Monitor0"
        DefaultDepth 24
        SubSection "Display"
                Viewport  0 0
                Depth     24
                Modes     "1024x768"
        EndSubSection
EndSection
```

The `DefaultDepth` keyword describes the color depth to run at by default. This can be overridden with the `-depth` command line switch to Xorg(1). The `Modes` keyword describes the resolution to run at for the given color depth. Note that only VESA standard modes are supported as defined by the target system's graphics hardware. In the example above, the default color depth is twenty-four bits per pixel. At this color depth, the accepted resolution is 1024 by 768 pixels.

Finally, write the configuration file and test it using the test mode given above.

> **Note:** One of the tools available to assist you during troubleshooting process are the X11 log files, which contain information on each device that the X11 server attaches to. **Xorg** log file names are in the format of `/var/log/Xorg.0.log`. The exact name of the log can vary from `Xorg.0.log` to `Xorg.8.log` and so forth.

If all is well, the configuration file needs to be installed in a common location where Xorg(1) can find it. This is typically `/etc/X11/xorg.conf` or `/usr/local/etc/X11/xorg.conf`.

```
# cp xorg.conf.new /etc/X11/xorg.conf
```

The X11 configuration process is now complete. **Xorg** may be now started with the startx(1) utility. The X11 server may also be started with the use of xdm(1).

6.4.3 Advanced Configuration Topics

6.4.3.1 Configuration with Intel® `i810` Graphics Chipsets

Configuration with Intel i810 integrated chipsets requires the `agpgart` AGP programming interface for X11 to drive the card. See the agp(4) driver manual page for more information.

This will allow configuration of the hardware as any other graphics board. Note on systems without the agp(4) driver compiled in the kernel, trying to load the module with kldload(8) will not work. This driver has to be in the kernel at boot time through being compiled in or using `/boot/loader.conf`.

6.4.3.2 Adding a Widescreen Flatpanel to the Mix

This section assumes a bit of advanced configuration knowledge. If attempts to use the standard configuration tools above have not resulted in a working configuration, there is information enough in the log files to be of use in getting the setup working. Use of a text editor will be necessary.

Current widescreen (WSXGA, WSXGA+, WUXGA, WXGA, WXGA+, et.al.) formats support 16:10 and 10:9 formats or aspect ratios that can be problematic. Examples of some common screen resolutions for 16:10 aspect ratios are:

- 2560x1600
- 1920x1200
- 1680x1050
- 1440x900
- 1280x800

At some point, it will be as easy as adding one of these resolutions as a possible `Mode` in the `Section "Screen"` as such:

```
Section "Screen"
Identifier "Screen0"
Device     "Card0"
Monitor    "Monitor0"
DefaultDepth 24
SubSection "Display"
       Viewport  0 0
       Depth     24
       Modes     "1680x1050"
EndSubSection
EndSection
```

Xorg is smart enough to pull the resolution information from the widescreen via I2C/DDC information so it knows what the monitor can handle as far as frequencies and resolutions.

If those `ModeLines` do not exist in the drivers, one might need to give **Xorg** a little hint. Using `/var/log/Xorg.0.log` one can extract enough information to manually create a `ModeLine` that will work. Simply look for information resembling this:

```
(II) MGA(0): Supported additional Video Mode:
```

```
(II) MGA(0): clock: 146.2 MHz   Image Size:  433 x 271 mm
(II) MGA(0): h_active: 1680  h_sync: 1784  h_sync_end 1960 h_blank_end 2240 h_border: 0
(II) MGA(0): v_active: 1050  v_sync: 1053  v_sync_end 1059 v_blanking: 1089 v_border: 0
(II) MGA(0): Ranges: V min: 48  V max: 85 Hz, H min: 30  H max: 94 kHz, PixClock max 170 MHz
```

This information is called EDID information. Creating a `ModeLine` from this is just a matter of putting the numbers in the correct order:

```
ModeLine <name> <clock> <4 horiz. timings> <4 vert. timings>
```

So that the `ModeLine` in `Section "Monitor"` for this example would look like this:

```
Section "Monitor"
Identifier      "Monitor1"
VendorName      "Bigname"
ModelName       "BestModel"
ModeLine        "1680x1050" 146.2 1680 1784 1960 2240 1050 1053 1059 1089
Option          "DPMS"
EndSection
```

Now having completed these simple editing steps, X should start on your new widescreen monitor.

6.5 Using Fonts in X11

Contributed by Murray Stokely.

6.5.1 Type1 Fonts

The default fonts that ship with X11 are less than ideal for typical desktop publishing applications. Large presentation fonts show up jagged and unprofessional looking, and small fonts are almost completely unintelligible. However, there are several free, high quality Type1 (PostScript®) fonts available which can be readily used with X11. For instance, the URW font collection (`x11-fonts/urwfonts`) includes high quality versions of standard type1 fonts (Times Roman®, Helvetica®, Palatino® and others). The Freefonts collection (`x11-fonts/freefonts`) includes many more fonts, but most of them are intended for use in graphics software such as the **Gimp**, and are not complete enough to serve as screen fonts. In addition, X11 can be configured to use TrueType fonts with a minimum of effort. For more details on this, see the X(7) manual page or the section on TrueType fonts.

To install the above Type1 font collections from the Ports Collection, run the following commands:

```
# cd /usr/ports/x11-fonts/urwfonts
# make install clean
```

And likewise with the freefont or other collections. To have the X server detect these fonts, add an appropriate line to the X server configuration file (`/etc/X11/xorg.conf`), which reads:

```
FontPath "/usr/local/lib/X11/fonts/URW/"
```

Alternatively, at the command line in the X session run:

```
% xset fp+ /usr/local/lib/X11/fonts/URW
```

```
% xset fp rehash
```

This will work but will be lost when the X session is closed, unless it is added to the startup file (`~/.xinitrc` for a normal `startx` session, or `~/.xsession` when logging in through a graphical login manager like **XDM**). A third way is to use the new `/usr/local/etc/fonts/local.conf` file: see the section on anti-aliasing.

6.5.2 TrueType® Fonts

Xorg has built in support for rendering TrueType fonts. There are two different modules that can enable this functionality. The freetype module is used in this example because it is more consistent with the other font rendering back-ends. To enable the freetype module just add the following line to the `"Module"` section of the `/etc/X11/xorg.conf` file.

```
Load  "freetype"
```

Now make a directory for the TrueType fonts (for example, `/usr/local/lib/X11/fonts/TrueType`) and copy all of the TrueType fonts into this directory. Keep in mind that TrueType fonts cannot be directly taken from a Macintosh; they must be in UNIX/MS-DOS/Windows format for use by X11. Once the files have been copied into this directory, use **ttmkfdir** to create a `fonts.dir` file, so that the X font renderer knows that these new files have been installed. `ttmkfdir` is available from the FreeBSD Ports Collection as `x11-fonts/ttmkfdir`.

```
# cd /usr/local/lib/X11/fonts/TrueType
# ttmkfdir -o fonts.dir
```

Now add the TrueType directory to the font path. This is just the same as described above for Type1 fonts, that is, use

```
% xset fp+ /usr/local/lib/X11/fonts/TrueType
% xset fp rehash
```

or add a `FontPath` line to the `xorg.conf` file.

That's it. Now **Gimp**, **Apache OpenOffice**, and all of the other X applications should now recognize the installed TrueType fonts. Extremely small fonts (as with text in a high resolution display on a web page) and extremely large fonts (within **StarOffice™**) will look much better now.

6.5.3 Anti-Aliased Fonts

Updated by Joe Marcus Clarke.

All fonts in X11 that are found in `/usr/local/lib/X11/fonts/` and `~/.fonts/` are automatically made available for anti-aliasing to Xft-aware applications. Most recent applications are Xft-aware, including **KDE**, **GNOME**, and **Firefox**.

In order to control which fonts are anti-aliased, or to configure anti-aliasing properties, create (or edit, if it already exists) the file `/usr/local/etc/fonts/local.conf`. Several advanced features of the Xft font system can be tuned using this file; this section describes only some simple possibilities. For more details, please see fonts-conf(5).

This file must be in XML format. Pay careful attention to case, and make sure all tags are properly closed. The file begins with the usual XML header followed by a DOCTYPE definition, and then the `<fontconfig>` tag:

```
<?xml version="1.0"?>
```

```
<!DOCTYPE fontconfig SYSTEM "fonts.dtd">
<fontconfig>
```

As previously stated, all fonts in `/usr/local/lib/X11/fonts/` as well as `~/.fonts/` are already made available to Xft-aware applications. If you wish to add another directory outside of these two directory trees, add a line similar to the following to `/usr/local/etc/fonts/local.conf`:

```
<dir>/path/to/my/fonts</dir>
```

After adding new fonts, and especially new font directories, you should run the following command to rebuild the font caches:

```
# fc-cache -f
```

Anti-aliasing makes borders slightly fuzzy, which makes very small text more readable and removes “staircases” from large text, but can cause eyestrain if applied to normal text. To exclude font sizes smaller than 14 point from anti-aliasing, include these lines:

```
 <match target="font">
    <test name="size" compare="less">
        <double>14</double>
    </test>
    <edit name="antialias" mode="assign">
        <bool>false</bool>
    </edit>
</match>
<match target="font">
    <test name="pixelsize" compare="less" qual="any">
        <double>14</double>
    </test>
    <edit mode="assign" name="antialias">
        <bool>false</bool>
    </edit>
</match>
```

Spacing for some monospaced fonts may also be inappropriate with anti-aliasing. This seems to be an issue with **KDE**, in particular. One possible fix for this is to force the spacing for such fonts to be 100. Add the following lines:

```
<match target="pattern" name="family">
   <test qual="any" name="family">
       <string>fixed</string>
   </test>
   <edit name="family" mode="assign">
       <string>mono</string>
   </edit>
</match>
<match target="pattern" name="family">
    <test qual="any" name="family">
        <string>console</string>
    </test>
    <edit name="family" mode="assign">
        <string>mono</string>
    </edit>
```

```
</match>
```

(this aliases the other common names for fixed fonts as `"mono"`), and then add:

```
<match target="pattern" name="family">
   <test qual="any" name="family">
       <string>mono</string>
   </test>
   <edit name="spacing" mode="assign">
       <int>100</int>
   </edit>
</match>
```

Certain fonts, such as Helvetica, may have a problem when anti-aliased. Usually this manifests itself as a font that seems cut in half vertically. At worst, it may cause applications to crash. To avoid this, consider adding the following to `local.conf`:

```
<match target="pattern" name="family">
   <test qual="any" name="family">
       <string>Helvetica</string>
   </test>
   <edit name="family" mode="assign">
       <string>sans-serif</string>
   </edit>
</match>
```

Once you have finished editing `local.conf` make sure you end the file with the `</fontconfig>` tag. Not doing this will cause your changes to be ignored.

Finally, users can add their own settings via their personal `.fonts.conf` files. To do this, each user should simply create a `~/.fonts.conf`. This file must also be in XML format.

One last point: with an LCD screen, sub-pixel sampling may be desired. This basically treats the (horizontally separated) red, green and blue components separately to improve the horizontal resolution; the results can be dramatic. To enable this, add the line somewhere in the `local.conf` file:

```
<match target="font">
   <test qual="all" name="rgba">
       <const>unknown</const>
   </test>
   <edit name="rgba" mode="assign">
       <const>rgb</const>
   </edit>
</match>
```

Note: Depending on the sort of display, `rgb` may need to be changed to `bgr`, `vrgb` or `vbgr`: experiment and see which works best.

6.6 The X Display Manager

Contributed by Seth Kingsley.

6.6.1 Overview

The X Display Manager (**XDM**) is an optional part of the X Window System that is used for login session management. This is useful for several types of situations, including minimal "X Terminals", desktops, and large network display servers. Since the X Window System is network and protocol independent, there are a wide variety of possible configurations for running X clients and servers on different machines connected by a network. **XDM** provides a graphical interface for choosing which display server to connect to, and entering authorization information such as a login and password combination.

Think of **XDM** as providing the same functionality to the user as the getty(8) utility (see Section 27.3.2 for details). That is, it performs system logins to the display being connected to and then runs a session manager on behalf of the user (usually an X window manager). **XDM** then waits for this program to exit, signaling that the user is done and should be logged out of the display. At this point, **XDM** can display the login and display chooser screens for the next user to login.

6.6.2 Using XDM

To start using **XDM**, install the `x11/xdm` port (it is not installed by default in recent versions of **Xorg**). The **XDM** daemon program may then be found in `/usr/local/bin/xdm`. This program can be run at any time as `root` and it will start managing the X display on the local machine. If **XDM** is to be run every time the machine boots up, a convenient way to do this is by adding an entry to `/etc/ttys`. For more information about the format and usage of this file, see Section 27.3.2.1. There is a line in the default `/etc/ttys` file for running the **XDM** daemon on a virtual terminal:

```
ttyv8   "/usr/local/bin/xdm -nodaemon"  xterm   off secure
```

By default this entry is disabled; in order to enable it change field 5 from `off` to `on` and restart init(8) using the directions in Section 27.3.2.2. The first field, the name of the terminal this program will manage, is `ttyv8`. This means that **XDM** will start running on the 9th virtual terminal.

6.6.3 Configuring XDM

The **XDM** configuration directory is located in `/usr/local/lib/X11/xdm`. In this directory there are several files used to change the behavior and appearance of **XDM**. Typically these files will be found:

File	Description
`Xaccess`	Client authorization ruleset.
`Xresources`	Default X resource values.
`Xservers`	List of remote and local displays to manage.
`Xsession`	Default session script for logins.
`Xsetup_*`	Script to launch applications before the login interface.
`xdm-config`	Global configuration for all displays running on this machine.

File	Description
`xdm-errors`	Errors generated by the server program.
`xdm-pid`	The process ID of the currently running XDM.

Also in this directory are a few scripts and programs used to set up the desktop when **XDM** is running. The purpose of each of these files will be briefly described. The exact syntax and usage of all of these files is described in xdm(1).

The default configuration is a simple rectangular login window with the hostname of the machine displayed at the top in a large font and "Login:" and "Password:" prompts below. This is a good starting point for changing the look and feel of **XDM** screens.

6.6.3.1 Xaccess

The protocol for connecting to **XDM**-controlled displays is called the X Display Manager Connection Protocol (XDMCP). This file is a ruleset for controlling XDMCP connections from remote machines. It is ignored unless the `xdm-config` is changed to listen for remote connections. By default, it does not allow any clients to connect.

6.6.3.2 Xresources

This is an application-defaults file for the display chooser and login screens. In it, the appearance of the login program can be modified. The format is identical to the app-defaults file described in the X11 documentation.

6.6.3.3 Xservers

This is a list of the remote displays the chooser should provide as choices.

6.6.3.4 Xsession

This is the default session script for **XDM** to run after a user has logged in. Normally each user will have a customized session script in `~/.xsession` that overrides this script.

6.6.3.5 Xsetup_*

These will be run automatically before displaying the chooser or login interfaces. There is a script for each display being used, named `Xsetup_` followed by the local display number (for instance `Xsetup_0`). Typically these scripts will run one or two programs in the background such as `xconsole`.

6.6.3.6 xdm-config

This contains settings in the form of app-defaults that are applicable to every display that this installation manages.

6.6.3.7 xdm-errors

This contains the output of the X servers that **XDM** is trying to run. If a display that **XDM** is trying to start hangs for some reason, this is a good place to look for error messages. These messages are also written to the user's `~/.xsession-errors` file on a per-session basis.

6.6.4 Running a Network Display Server

In order for other clients to connect to the display server, you must edit the access control rules and enable the connection listener. By default these are set to conservative values. To make **XDM** listen for connections, first comment out a line in the `xdm-config` file:

```
! SECURITY: do not listen for XDMCP or Chooser requests
! Comment out this line if you want to manage X terminals with xdm
DisplayManager.requestPort:     0
```

and then restart **XDM**. Remember that comments in app-defaults files begin with a "!" character, not the usual "#". More strict access controls may be desired — look at the example entries in `Xaccess`, and refer to the xdm(1) manual page for further information.

6.6.5 Replacements for XDM

Several replacements for the default **XDM** program exist. One of them, **KDM** (bundled with **KDE**) is described later in this chapter. The **KDM** display manager offers many visual improvements and cosmetic frills, as well as the functionality to allow users to choose their window manager of choice at login time.

6.7 Desktop Environments

Contributed by Valentino Vaschetto.

This section describes the different desktop environments available for X on FreeBSD. A "desktop environment" can mean anything ranging from a simple window manager to a complete suite of desktop applications, such as **KDE** or **GNOME**.

6.7.1 GNOME

6.7.1.1 About GNOME

GNOME is a user-friendly desktop environment that enables users to easily use and configure their computers. **GNOME** includes a panel (for starting applications and displaying status), a desktop (where data and applications can be placed), a set of standard desktop tools and applications, anda set of conventions that make it easy for applications to cooperate and be consistent with each other. Users of other operating systems or environments should feel right at home using the powerful graphics-driven environment that **GNOME** provides. More information regarding **GNOME** on FreeBSD can be found on the FreeBSD GNOME Project (http://www.FreeBSD.org/gnome)'s web site. The web site also contains fairly comprehensive FAQs about installing, configuring, and managing **GNOME**.

6.7.1.2 Installing GNOME

The software can be easily installed from a package or the Ports Collection:

To install the **GNOME** package from the network, simply type:

```
# pkg_add -r gnome2
```

To build **GNOME** from source, use the ports tree:

```
# cd /usr/ports/x11/gnome2
# make install clean
```

For proper operation, **GNOME** requires the `/proc` filesystem to be mounted. Add

```
proc            /proc       procfs  rw  0   0
```

to `/etc/fstab` to mount procfs(5) automatically during startup.

Once **GNOME** is installed, the X server must be told to start **GNOME** instead of a default window manager.

The easiest way to start **GNOME** is with **GDM**, the GNOME Display Manager. **GDM** is installed as part of the **GNOME** desktop, although it is disabled by default. It can be enabled by adding this line to `/etc/rc.conf`:

```
gdm_enable="YES"
```

Once you have rebooted, **GDM** will start automatically.

It is often desirable to start all **GNOME** services together with **GDM**. To achieve this, add the following line to `/etc/rc.conf`:

```
gnome_enable="YES"
```

GNOME may also be started from the command-line by properly configuring a file named `.xinitrc`. If a custom `.xinitrc` is already in place, simply replace the line that starts the current window manager with one that starts **/usr/local/bin/gnome-session** instead. If nothing special has been done to the configuration file, then it is enough simply to type:

```
% echo "/usr/local/bin/gnome-session" > ~/.xinitrc
```

Next, type `startx`, and the **GNOME** desktop environment will be started.

> **Note:** If an older display manager, like **XDM**, is being used, this will not work. Instead, create an executable `.xsession` file with the same command in it. To do this, edit the file and replace the existing window manager command with **/usr/local/bin/gnome-session**:

```
% echo "#!/bin/sh" > ~/.xsession
% echo "/usr/local/bin/gnome-session" >> ~/.xsession
% chmod +x ~/.xsession
```

Yet another option is to configure the display manager to allow choosing the window manager at login time; the section on KDE details explains how to do this for **KDM**, the display manager of **KDE**.

6.7.2 KDE

6.7.2.1 About KDE

KDE is an easy to use contemporary desktop environment. Some of the things that **KDE** brings to the user are:

- A beautiful contemporary desktop
- A desktop exhibiting complete network transparency
- An integrated help system allowing for convenient, consistent access to help on the use of the **KDE** desktop and its applications
- Consistent look and feel of all **KDE** applications
- Standardized menu and toolbars, keybindings, color-schemes, etc.
- Internationalization: **KDE** is available in more than 55 languages
- Centralized, consistent, dialog-driven desktop configuration
- A great number of useful **KDE** applications

KDE comes with a web browser called **Konqueror**, which is a solid competitor to other existing web browsers on UNIX systems. More information on **KDE** can be found on the KDE website (http://www.kde.org/). For FreeBSD specific information and resources on **KDE**, consult the KDE/FreeBSD initiative (http://freebsd.kde.org/)'s website.

There are two versions of **KDE** available on FreeBSD. Version 3 has been around for a long time, and is still available in the Ports Collection though it's now unmaintained and partially broken. Version 4 is punctually updated and is the default choice for **KDE** users. They can even be installed side by side.

6.7.2.2 Installing KDE

Just as with **GNOME** or any other desktop environment, the software can be easily installed from a package or the Ports Collection:

To install the **KDE 3** package from the network, type:

```
# pkg_add -r kde
```

To install the **KDE 4** package from the network, type:

```
# pkg_add -r kde4
```

pkg_add(1) will automatically fetch the latest version of the application.

To build **KDE 3** from source, use the ports tree:

```
# cd /usr/ports/x11/kde3
# make install clean
```

To build **KDE 4** from source, use the ports tree:

```
# cd /usr/ports/x11/kde4
# make install clean
```

After **KDE** has been installed, the X server must be told to launch this application instead of the default window manager. This is accomplished by editing the `.xinitrc` file:

For **KDE 3**:

```
% echo "exec startkde" > ~/.xinitrc
```

For **KDE 4**:

```
% echo "exec /usr/local/kde4/bin/startkde" > ~/.xinitrc
```

Now, whenever the X Window System is invoked with `startx`, **KDE** will be the desktop.

If a display manager such as **XDM** is being used, the configuration is slightly different. Edit the `.xsession` file instead. Instructions for **KDM** are described later in this chapter.

6.7.3 More Details on KDE

Now that **KDE** is installed on the system, most things can be discovered through the help pages, or just by pointing and clicking at various menus. Windows or Mac® users will feel quite at home.

The best reference for **KDE** is the on-line documentation. **KDE** comes with its own web browser, **Konqueror**, dozens of useful applications, and extensive documentation. The remainder of this section discusses the technical items that are difficult to learn by random exploration.

6.7.3.1 The KDE Display Manager

An administrator of a multi-user system may wish to have a graphical login screen to welcome users. XDM can be used, as described earlier. However, **KDE** includes an alternative, **KDM**, which is designed to look more attractive and include more login-time options. In particular, users can easily choose (via a menu) which desktop environment (**KDE**, **GNOME**, or something else) to run after logging on.

To enable **KDM**, different files need to be edited depending on the version of **KDE**.

For **KDE 3**, the `ttyv8` entry in `/etc/ttys` has to be adapted as follows:

```
ttyv8 "/usr/local/bin/kdm -nodaemon" xterm on secure
```

For **KDE 4**, you have to mount procfs(5) and add the following line to `/etc/rc.conf`:

```
kdm4_enable="YES"
```

6.7.4 Xfce

6.7.4.1 About Xfce

Xfce is a desktop environment based on the GTK+ toolkit used by **GNOME**, but is much more lightweight and meant for those who want a simple, efficient desktop which is nevertheless easy to use and configure. Visually, it looks very much like **CDE**, found on commercial UNIX systems. Some of **Xfce**'s features are:

- A simple, easy-to-handle desktop
- Fully configurable via mouse, with drag and drop, etc.
- Main panel similar to **CDE**, with menus, applets and applications launchers
- Integrated window manager, file manager, sound manager, **GNOME** compliance module, and more
- Themeable (since it uses GTK+)
- Fast, light and efficient: ideal for older/slower machines or machines with memory limitations

More information on **Xfce** can be found on the Xfce website (http://www.xfce.org/).

6.7.4.2 Installing Xfce

A binary package for **Xfce** exists (at the time of writing). To install, simply type:

```
# pkg_add -r xfce4
```

Alternatively, to build from source, use the Ports Collection:

```
# cd /usr/ports/x11-wm/xfce4
# make install clean
```

Now, tell the X server to launch **Xfce** the next time X is started. Simply type this:

```
% echo "/usr/local/bin/startxfce4" > ~/.xinitrc
```

The next time X is started, **Xfce** will be the desktop. As before, if a display manager like **XDM** is being used, create an `.xsession`, as described in the section on GNOME, but with the `/usr/local/bin/startxfce4` command; or, configure the display manager to allow choosing a desktop at login time, as explained in the section on kdm.

II. Common Tasks

Now that the basics have been covered, this part of the FreeBSD Handbook will discuss some frequently used features of FreeBSD. These chapters:

- Introduce you to popular and useful desktop applications: browsers, productivity tools, document viewers, etc.
- Introduce you to a number of multimedia tools available for FreeBSD.
- Explain the process of building a customized FreeBSD kernel, to enable extra functionality on your system.
- Describe the print system in detail, both for desktop and network-connected printer setups.
- Show you how to run Linux applications on your FreeBSD system.

Some of these chapters recommend that you do some prior reading, and this is noted in the synopsis at the beginning of each chapter.

Chapter 7 Desktop Applications

Contributed by Christophe Juniet.

7.1 Synopsis

While FreeBSD is popular as a server for its performance and stability, it is also suited for day-to-day use as a desktop. With over 24,000 applications available as packages or ports, it is easy to build a customized desktop that runs a wide variety of desktop applications. This chapter demonstrates how to install some popular desktop applications effortlessly using packages or the FreeBSD Ports Collection.

As FreeBSD features Linux binary compatibility, many applications developed for Linux can be installed on a FreeBSD desktop. Many of the ports using Linux binary compatibility start with “linux-”. This chapter assumes that Linux binary compatibility has been enabled before any Linux applications are installed.

This chapter demonstrates how to install the following desktop applications:

Type of Application	Application Name	Package Name	Ports Name
Browser	**Firefox**	`firefox`	`www/firefo`
Browser	**Opera**	`opera`	`www/opera`
Browser	**Konqueror**	`kde4-baseapps`	`x11/kde4-b`
Browser	**Chromium**	`chromium`	`www/chromi`
Productivity	**Calligra**	`calligra`	`editors/ca`
Productivity	**AbiWord**	`abiword`	`editors/ab`
Productivity	**The GIMP**	`gimp`	`graphics/g`
Productivity	**Apache OpenOffice**	`openoffice`	`editors/op`
Productivity	**LibreOffice**	`libreoffice`	`editors/li`
Document Viewer	**Acrobat Reader®**	`no package due to license restriction`	`print/acro`
Document Viewer	**gv**	`gv`	`print/gv`
Document Viewer	**Xpdf**	`xpdf`	`graphics/x`
Document Viewer	**GQview**	`gqview`	`graphics/g`
Finance	**GnuCash**	`gnucash`	`finance/gn`
Finance	**Gnumeric**	`gnumeric`	`math/gnume`
Finance	**KMyMoney**	`kmymoney-kde4`	`finance/km`

Before reading this chapter, you should know how to:

- Install additional software using packages or ports.
- Enable Linux binary compatibility.

For information on how to configure a multimedia environment, refer to Chapter 8. For information on how to set up and use electronic mail, refer to Chapter 29.

7.2 Browsers

FreeBSD does not come with a pre-installed web browser. Instead, the www (http://www.FreeBSD.org/ports/www.html) category of the Ports Collection contains many browsers which can be installed as a package or compiled from the Ports Collection.

The **KDE** and **GNOME** desktop environments include their own HTML browser. Refer to Section 6.7 for more information on how to set up these complete desktops.

Some light-weight browsers include `www/dillo2`, `www/links`, and `www/w3m`.

This section demonstrates how to install the following popular web browsers and indicates if the application is resource-heavy, takes time to compile from ports, or has any major dependencies.

Application Name	Resources Needed	Installation from Ports	Notes
Firefox	medium	heavy	FreeBSD and Linux versions are available
Opera	light	light	FreeBSD and Linux versions are available
Konqueror	medium	heavy	Requires **KDE** libraries
Chromium	medium	heavy	Requires **Gtk+**

7.2.1 Firefox

Firefox is a modern, free, open source browser that is fully ported to FreeBSD. It features a standards-compliant HTML display engine, tabbed browsing, popup blocking, extensions, improved security, and more. **Firefox** is based on the **Mozilla** codebase.

Install the package of the latest release version of **Firefox** by typing:

```
# pkg_add -r firefox
```

To instead install **Firefox** Extended Support Release (ESR) version, use:

```
# pkg_add -r firefox-esr
```

Localized versions are available in `www/firefox-i18n` and `www/firefox-esr-i18n`.

The Ports Collection can instead be used to compile the desired version of **firefox** from source code. This example builds `www/firefox`, where `firefox` can be replaced with the ESR or localized version to install.

```
# cd /usr/ports/www/firefox
# make install clean
```

7.2.1.1 Firefox and Java™ Plugin

> **Note:** The following sections assume that **Firefox** is already installed.

`java/icedtea-web` provides a free software web browser plugin for running Java applets. It can be installed as a package. To alternately compile the port:

```
# cd /usr/ports/java/icedtea-web
# make install clean
```

Keep the default configuration options when compiling the port.

Once installed, start **firefox**, enter `about:plugins` in the location bar and press **Enter**. A page listing the installed plugins will be displayed. The **Java™** plugin should be listed.

If the browser is unable to find the plugin, each user will have to run the following command and relaunch the browser:

```
% ln -s /usr/local/lib/IcedTeaPlugin.so \
  $HOME/.mozilla/plugins/
```

7.2.1.2 Firefox and Adobe® Flash™ Plugin

A native Adobe® Flash™ plugin is not available for FreeBSD. However, a software layer (wrapper) for running the Linux version of the plugin exists. This wrapper also provides support for other browser plugins such as RealPlayer®.

To install and enable this plugin:

1. Install the `www/nspluginwrapper` port. Due to licensing restrictions, a package is not available. This port requires `emulators/linux_base-f10` which is a large port.
2. Install the `www/linux-f10-flashplugin11` port. Due to licensing restrictions, a package is not available.
3. # **`ln -s /usr/local/lib/npapi/linux-f10-flashplugin/libflashplayer.so \ /usr/local/lib/browser_plugins/`**

   Create the `/usr/local/lib/browser_plugins` directory if it is not already present.
4. Before the plugin is first used, each user must run:

   ```
   % nspluginwrapper -v -a -i
   ```

   When the plugin port has been updated and reinstalled, each user must run:

   ```
   % nspluginwrapper -v -a -u
   ```

   Start the browser, enter `about:plugins` in the location bar and press **Enter**. A list of all the currently available plugins will be shown.

7.2.1.3 Firefox and Swfdec Flash Plugin

Swfdec is the library for decoding and rendering Flash animations. Swfdec-Mozilla is a plugin for **Firefox** browsers that uses the Swfdec library for playing SWF files. It is still in heavy development.

To install the package:

```
# pkg_add -r swfdec-plugin
```

If the package is not available, compile and install it from the Ports Collection:

```
# cd /usr/ports/www/swfdec-plugin
# make install clean
```

Restart the browser for this plugin to take effect.

7.2.2 Opera

Opera is a full-featured and standards-compliant browser which is still lightweight and fast. It comes with a built-in mail and news reader, an IRC client, an RSS/Atom feeds reader, and more. It is available as a native FreeBSD version and as a version that runs under Linux emulation.

This command installs the package of the FreeBSD version of **Opera**. Replace `opera` with `linux-opera` to instead install the Linux version.

```
# pkg_add -r opera
```

Alternately, install either version through the Ports Collection. This example compiles the native version:

```
# cd /usr/ports/www/opera
# make install clean
```

To install the Linux version, substitute `linux-opera` in place of `opera`.

To install Adobe Flash plugin support, first compile the `www/linux-f10-flashplugin11` port, as a package is not available due to licensing restrictions. Then install either the `www/opera-linuxplugins` port or package. This example compiles both from ports:

```
# cd /usr/ports/www/linux-f10-flashplugin11
# make install clean
# cd /usr/ports/www/opera-linuxplugins
# make install clean
```

Once installed, check the presence of the plugin by starting the browser, entering `opera:plugins` in the location bar and pressing **Enter**. A list should appear with all the currently available plugins.

To add the **Java** plugin, follow the instructions for Firefox.

7.2.3 Konqueror

Konqueror is part of `x11/kde4-baseapps`. **Konqueror** is more than a web browser as it is also a file manager and a multimedia viewer.

Konqueror supports WebKit as well as its own KHTML. WebKit is a rendering engine used by many modern browsers including Chromium. To use WebKit with **Konqueror** on FreeBSD, install the `www/kwebkitpart` package or port. This example compiles the port:

```
# cd /usr/ports/www/kwebkitpart
# make install clean
```

To enable WebKit within **Konqueror**, click “Settings”, “Configure Konqueror”. In the “General” settings page, click the drop-down menu next to “Default web browser engine” and change “KHTML” to “WebKit”.

Konqueror also supports **Flash**. A “How To” guide for getting **Flash** support on **Konqueror** is available at http://freebsd.kde.org/howtos/konqueror-flash.php.

7.2.4 Chromium

Chromium is an open source browser project that aims to build a safer, faster, and more stable web browsing experience. **Chromium** features tabbed browsing, popup blocking, extensions, and much more. **Chromium** is the open source project upon which the Google Chrome web browser is based.

Chromium can be installed as a package by typing:

```
# pkg_add -r chromium
```

Alternatively, **Chromium** can be compiled from source using the Ports Collection:

```
# cd /usr/ports/www/chromium
# make install clean
```

Note: The executable for **Chromium** is `/usr/local/bin/chrome`, not `/usr/local/bin/chromium`.

7.2.4.1 Chromium and Java Plugin

Note: The following sections assume that **Chromium** is already installed.

To install Java plugin support, follow the instructions in Section 7.2.1.1.

Once Java support is installed, start **Chromium**, and enter `about:plugins` in the address bar. IcedTea-Web should be listed as one of the installed plugins.

If **Chromium** does not display the IcedTea-Web plugin, run the following commands, and restart the web browser:

```
# mkdir -p /usr/local/share/chromium/plugins
# ln -s /usr/local/lib/IcedTeaPlugin.so \
  /usr/local/share/chromium/plugins/
```

7.2.4.2 Chromium and Adobe Flash Plugin

Configuring **Chromium** and Adobe Flash is similar to the instructions for Firefox. No additional configuration should be necessary, since **Chromium** is able to use some plugins from other browsers.

7.3 Productivity

When it comes to productivity, new users often look for a good office suite or a friendly word processor. While some desktop environments like **KDE** already provide an office suite, there is no default productivity package. Several office suites and word processors are available for FreeBSD, regardless of the installed desktop environment.

This section demonstrates how to install the following popular productivity software and indicates if the application is resource-heavy, takes time to compile from ports, or has any major dependencies.

Application Name	Resources Needed	Installation from Ports	Major Dependencies
Calligra	light	heavy	**KDE**
AbiWord	light	light	**Gtk+** or **GNOME**
The Gimp	light	heavy	**Gtk+**
Apache OpenOffice	heavy	huge	**JDK™** and **Mozilla**
LibreOffice	somewhat heavy	huge	**Gtk+**, or **KDE**/ **GNOME**, or **JDK**

7.3.1 Calligra

The KDE community provides its desktop environment with an office suite which can be used outside of **KDE**. **Calligra** includes standard components that can be found in other office suites. **Words** is the word processor, **Sheets** is the spreadsheet program, **Stage** manages slide presentations, and **Karbon** is used to draw graphical documents.

`editors/calligra` can be installed as a package or a port. To install the package:

```
# pkg_add -r calligra
```

If the package is not available, use the Ports Collection instead:

```
# cd /usr/ports/editors/calligra
# make install clean
```

7.3.2 AbiWord

AbiWord is a free word processing program similar in look and feel to **Microsoft Word**. It is suitable for typing papers, letters, reports, memos, and so forth. It is fast, contains many features, and is user-friendly.

AbiWord can import or export many file formats, including some proprietary ones like Microsoft `.doc`.

To install the **AbiWord** package:

```
# pkg_add -r abiword
```

If the package is not available, it can be compiled from the Ports Collection:

```
# cd /usr/ports/editors/abiword
# make install clean
```

7.3.3 The GIMP

For image authoring or picture retouching, **The GIMP** provides a sophisticated image manipulation program. It can be used as a simple paint program or as a quality photo retouching suite. It supports a large number of plugins and features a scripting interface. **The GIMP** can read and write a wide range of file formats and supports interfaces with scanners and tablets.

To install the package:

```
# pkg_add -r gimp
```

Alternately, use the Ports Collection:

```
# cd /usr/ports/graphics/gimp
# make install clean
```

The graphics (http://www.FreeBSD.org/ports/graphics.html) category of the Ports Collection contains several **GIMP**-related plugins, help files, and user manuals.

7.3.4 Apache OpenOffice

On 1 June 2011, Oracle donated the **OpenOffice.org** code base to the Apache Software Foundation. **OpenOffice.org** is now known as **Apache OpenOffice** and is developed under the wing of the Apache Software Foundation's Incubator.

Apache OpenOffice includes all of the mandatory applications in a complete office productivity suite: a word processor, spreadsheet, presentation manager, and drawing program. Its user interface is very similar to other office suites, and it can import and export in various popular file formats. It is available in a number of different languages and internationalization has been extended to interfaces, spell checkers, and dictionaries.

The word processor of **Apache OpenOffice** uses a native XML file format for increased portability and flexibility. The spreadsheet program features a macro language which can be interfaced with external databases. **Apache OpenOffice** is stable and runs natively on Windows, Solaris™, Linux, FreeBSD, and Mac OS X. More information about **Apache OpenOffice** can be found on the Apache OpenOffice web site (http://incubator.apache.org/openofficeorg/). For FreeBSD specific information, and to directly download packages, refer to the web site of the FreeBSD Apache OpenOffice Porting Team (http://porting.openoffice.org/freebsd/).

To install the **Apache OpenOffice** package:

```
# pkg_add -r apache-openoffice
```

> **Note:** When running a -RELEASE version of FreeBSD, this should work. Otherwise, download the latest package from the website of the FreeBSD **Apache OpenOffice** Porting Team and install it using pkg_add(1). Both the current release and development versions are available for download at this web site.

Once the package is installed, type the following command to launch **Apache OpenOffice**:

```
% openoffice-X.Y.Z
```

where *X.Y.Z* is the version number of the installed version of **Apache OpenOffice**.

> **Note:** During the first launch, some questions will be asked and a `.openoffice.org` folder will be created in the user's home directory.

If the desired **Apache OpenOffice** package is not available, compiling the port is still an opton. However, this requires a lot of disk space and a fairly long time to compile:

```
# cd /usr/ports/editors/openoffice-3
# make install clean
```

Note: To build a localized version, replace the previous command with:

```
# make LOCALIZED_LANG=your_language install clean
```

Replace *your_language* with the correct language ISO-code. A list of supported language codes is available in `files/Makefile.localized`, located in the port's directory.

7.3.5 LibreOffice

LibreOffice is a free software office suite developed by The Document Foundation (http://www.documentfoundation.org/). It is compatible with other major office suites and available on a variety of platforms. It is a rebranded fork of **OpenOffice.org** which includes all of the mandatory applications in a complete office productivity suite: a word processor, spreadsheet, presentation manager, drawing program, database management program, and a tool for creating and editing mathematical formula. It is available in a number of different languages and internationalization has been extended to interfaces, spell checkers, and dictionaries.

The word processor of **LibreOffice** uses a native XML file format for increased portability and flexibility. The spreadsheet program features a macro language which can be interfaced with external databases. **LibreOffice** is stable and runs natively on Windows, Linux, FreeBSD, and Mac OS X. More information about **LibreOffice** can be found on the LibreOffice web site (http://www.libreoffice.org/).

To install the English version of the **LibreOffice** package:

```
# pkg_add -r libreoffice
```

The editors (http://www.FreeBSD.org/ports/editors.html) category of the Ports Collection contains several localizations for **LibreOffice**. When installing a localized package, replace `libreoffice` with the name of the localized package.

Once the package is installed, type the following command to run **LibreOffice**:

```
% libreoffice
```

Note: During the first launch, some questions will be asked and a `.libreoffice` folder will be created in the user's home directory.

If the desired **LibreOffice** package is not available, compiling the port is still an option. However, this requires a lot of disk space and a fairly long time to compile. This example compiles the English version:

```
# cd /usr/ports/editors/libreoffice
# make install clean
```

Note: To build a localized version, **cd** into the port directory of the desired language. Supported languages can be found in the editors (http://www.FreeBSD.org/ports/editors.html) category of the Ports Collection.

7.4 Document Viewers

Some new document formats have gained popularity since the advent of UNIX and the viewers they require may not be available in the base system. This section demonstrates how to install the following viewers:

Application Name	Resources Needed	Installation from Ports	Major Dependencies
Acrobat Reader	light	light	Linux binary compatibility
gv	light	light	**Xaw3d**
Xpdf	light	light	**FreeType**
GQview	light	light	**Gtk+** or **GNOME**

7.4.1 Acrobat Reader®

Many documents are now distributed as Portable Document Format (PDF) files. One popular viewer for PDFs is **Acrobat Reader**, released by Adobe for Linux. As FreeBSD can run Linux binaries, it is also available for FreeBSD. Due to licensing restrictions, a package is not available so it must be compiled from ports. Several localizations are available from the print (http://www.FreeBSD.org/ports/print.html) category of the Ports Collection.

This command installs the English version of **Acrobat Reader 9** from the Ports Collection. To instead install a localized version, **cd** into the desired port's directory.

```
# cd /usr/ports/print/acroread9
# make install clean
```

7.4.2 gv

gv (http://www.gnu.org/software/gv/) is a PostScript and PDF viewer. It is based on **ghostview**, but has a nicer look due to the **Xaw3d** library. It is fast with a clean interface. **gv** has many configurable features, such as orientation, paper size, scale, and anti-aliasing. Almost any operation can be performed with either the keyboard or the mouse.

To install **gv** as a package:

```
# pkg_add -r gv
```

If a package is unavailable, use the Ports Collection:

```
# cd /usr/ports/print/gv
# make install clean
```

7.4.3 Xpdf

For users that prefer a small FreeBSD PDF viewer, xpdf (http://www.foolabs.com/xpdf/) provides a light-weight and efficient viewer which requires few resources. It uses the standard X fonts and does not require **Motif®** or any other X toolkit.

To install the **Xpdf** package:

```
# pkg_add -r xpdf
```

If the package is not available, use the Ports Collection:

```
# cd /usr/ports/graphics/xpdf
# make install clean
```

Once the installation is complete, launch `xpdf` and use the right mouse button to activate the menu.

7.4.4 GQview

GQview (http://gqview.sourceforge.net/) is an image manager which supports viewing a file with a single click, launching an external editor, and thumbnail previews. It also features a slideshow mode and some basic file operations, making it easy to manage image collections and to find duplicate files. **GQview** supports full screen viewing and internationalization.

To install the **GQview** package:

```
# pkg_add -r gqview
```

If the package is not available, use the Ports Collection:

```
# cd /usr/ports/graphics/gqview
# make install clean
```

7.5 Finance

For managing personal finances on a FreeBSD desktop, some powerful and easy-to-use applications can be installed. Some are compatible with widespread file formats, such as the formats used by **Quicken®** and **Excel**.

This section covers these programs:

Application Name	Resources Needed	Installation from Ports	Major Dependencies
GnuCash	light	heavy	**GNOME**
Gnumeric	light	heavy	**GNOME**
KMyMoney	light	heavy	**KDE**

7.5.1 GnuCash

GnuCash (http://www.gnucash.org/) is part of the **GNOME** effort to provide user-friendly, yet powerful, applications to end-users. **GnuCash** can be used to keep track of income and expenses, bank accounts, and stocks. It features an intuitive interface while remaining professional.

GnuCash provides a smart register, a hierarchical system of accounts, and many keyboard accelerators and auto-completion methods. It can split a single transaction into several more detailed pieces. **GnuCash** can import and merge **Quicken** QIF files. It also handles most international date and currency formats.

To install the **GnuCash** package:

```
# pkg_add -r gnucash
```

If the package is not available, use the Ports Collection:

```
# cd /usr/ports/finance/gnucash
# make install clean
```

7.5.2 Gnumeric

Gnumeric (http://projects.gnome.org/gnumeric/index.shtml) is a spreadsheet program developed by the **GNOME** community. It features convenient automatic "guessing" of user input according to the cell format with an autofill system for many sequences. It can import files in a number of popular formats, including **Excel**, **Lotus 1-2-3**, and **Quattro Pro**. It has a large number of built-in functions and allows all of the usual cell formats such as number, currency, date, time, and much more.

To install **Gnumeric** as a package:

```
# pkg_add -r gnumeric
```

If the package is not available, use the Ports Collection:

```
# cd /usr/ports/math/gnumeric
# make install clean
```

7.5.3 KMyMoney

KMyMoney (http://kmymoney2.sourceforge.net) is a personal finance created by the **KDE** community. **KMyMoney** intends to provide and incorporate all the important features found in commercial personal finance manager applications. It also highlights ease-of-use and proper double-entry accounting among its features. **KMyMoney** imports from standard Quicken Interchange Format (QIF) files, tracks investments, handles multiple currencies, and provides a wealth of reports.

To install **KMyMoney** as a package:

```
# pkg_add -r kmymoney-kde4
```

If the package is not available, use the Ports Collection:

```
# cd /usr/ports/finance/kmymoney-kde4
# make install clean
```

Chapter 8 Multimedia

Edited by Ross Lippert.

8.1 Synopsis

FreeBSD supports a wide variety of sound cards, allowing users to enjoy high fidelity output from a FreeBSD system. This includes the ability to record and playback audio in the MPEG Audio Layer 3 (MP3), Waveform Audio File (WAV), Ogg Vorbis, and other formats. The FreeBSD Ports Collection contains many applications for editing recorded audio, adding sound effects, and controlling attached MIDI devices.

FreeBSD also supports the playback of video files and DVDs. The FreeBSD Ports Collection contains applications to encode, convert, and playback various video media.

This chapter describes how to configure sound cards, video playback, TV tuner cards, and scanners on FreeBSD. It also describes some of the applications which are available for using these devices.

After reading this chapter, you will know how to:

- Configure a sound card on os;.
- Troubleshoot the sound setup.
- Playback and encode MP3s and other audio.
- Prepare a FreeBSD system for video playback.
- Playback DVDs, `.mpg`, and `.avi` files.
- Rip CD and DVD content into files.
- Configure a TV card.
- Install and setup MythTV on FreeBSD
- Configure an image scanner.
- How to configure an image scanner.

Before reading this chapter, you should:

- Know how to configure and install a new kernel (Chapter 9).

> **Warning:** Audio CDs have specialized encodings which differ from the usual ISO-filesystem. This means that they should not be mounted using mount(8).

8.2 Setting Up the Sound Card

Contributed by Moses Moore. Enhanced by Marc Fonvieille.

8.2.1 Configuring the System

Before beginning the configuration, determine the model of the sound card and the chip it uses. FreeBSD supports a wide variety of sound cards. Check the supported audio devices list of the Hardware Notes (http://www.FreeBSD.org/releases/9.1R/hardware.html) to see if the card is supported and which FreeBSD driver it uses.

In order to use the sound device, the proper device driver must be loaded. This may be accomplished in one of two ways. The easiest way is to load a kernel module for the sound card with kldload(8). This example loads the driver for a Creative SoundBlaster® Live! sound card:

```
# kldload snd_emu10k1
```

To automate the loading of this driver at boot time, add the driver to `/boot/loader.conf`. The line for this driver is:

```
snd_emu10k1_load="YES"
```

Other available sound modules are listed in `/boot/defaults/loader.conf`. When unsure which driver to use, load the `snd_driver` module:

```
# kldload snd_driver
```

This is a metadriver which loads all of the most common sound drivers and can be used to speed up the search for the correct driver. It is also possible to load all sound drivers by adding the metadriver to `/boot/loader.conf`.

To determine which driver was selected for the sound card after loading the `snd_driver` metadriver, type `cat /dev/sndstat`.

Users who prefer to statically compile in support for the sound card in a custom kernel should refer to the instructions in the next section. For more information about recompiling a kernel, refer to Chapter 9.

8.2.1.1 Configuring a Custom Kernel with Sound Support

When using a custom kernel to provide sound support, make sure that the audio framework driver exists in the custom kernel configuration file:

```
device sound
```

Next, add support for the sound card. Therefore, you need to know which driver supports the card. To continue the example of the Creative SoundBlaster Live! sound card from the previous section, use the following line in the custom kernel configuration file:

```
device snd_emu10k1
```

Be sure to read the manual page of the driver for the syntax to use. The explicit syntax for the kernel configuration of every supported sound driver can also be found in `/usr/src/sys/conf/NOTES`.

Non-PnP ISA sound cards may require the IRQ and I/O port settings of the card to be added to `/boot/device.hints`. During the boot process, loader(8) reads this file and passes the settings to the kernel. For

example, an old Creative SoundBlaster 16 ISA non-PnP card will use the snd_sbc(4) driver in conjunction with `snd_sb16`. For this card, the following lines must be added to the kernel configuration file:

```
device snd_sbc
device snd_sb16
```

If the card uses the `0x220` I/O port and IRQ `5`, these lines must also be added to `/boot/device.hints`:

```
hint.sbc.0.at="isa"
hint.sbc.0.port="0x220"
hint.sbc.0.irq="5"
hint.sbc.0.drq="1"
hint.sbc.0.flags="0x15"
```

In this case, the card uses the `0x220` I/O port and the IRQ `5`.

The syntax used in `/boot/device.hints` is described in sound(4) and the manual page for the driver of the sound card.

The settings shown above are the defaults. In some cases, the IRQ or other settings may need to be changed to match the card. Refer to snd_sbc(4) for more information about this card.

8.2.2 Testing the Sound Card

After rebooting into the custom kernel, or after loading the required module, the sound card should appear in the system message buffer. Run dmesg(8) and look for a message like:

```
pcm0: <Intel ICH3 (82801CA)> port 0xdc80-0xdcbf,0xd800-0xd8ff irq 5 at device 31.5 on pci0
pcm0: [GIANT-LOCKED]
pcm0: <Cirrus Logic CS4205 AC97 Codec>
```

The status of the sound card may also be checked using this command:

```
# cat /dev/sndstat
FreeBSD Audio Driver (newpcm)
Installed devices:
pcm0: <Intel ICH3 (82801CA)> at io 0xd800, 0xdc80 irq 5 bufsz 16384
kld snd_ich (1p/2r/0v channels duplex default)
```

The output may vary between systems. If no `pcm` devices are listed, go back and review the kernel configuration file and make sure the correct device driver was chosen. Common problems are listed in Section 8.2.2.1.

If all goes well, the sound card should now work in os;. If the CD-ROM or DVD-ROM drive's audio-out pins are properly connected to the sound card, one can insert an audio CD in the drive and play it with cdcontrol(1):

```
% cdcontrol -f /dev/acd0 play 1
```

Various applications, such as `audio/workman` provide a friendlier interface. The `audio/mpg123` port can be installed to listen to MP3 audio files.

Another quick way to test the card is to send data to `/dev/dsp`:

```
% cat filename > /dev/dsp
```

where *filename* can be any file. This command should produce some noise, confirming that the sound card is actually working.

> **Note:** The /dev/dsp* device nodes will be created automatically as needed. When not in use, they do not exist and will not appear in the output of ls(1).

Sound card mixer levels can be changed using mixer(8). More details can be found in mixer(8).

8.2.2.1 Common Problems

Error	Solution
sb_dspwr(XX) timed out	The I/O port is not set correctly.
bad irq XX	The IRQ is set incorrectly. Make sure that the set IRQ and the sound IRQ are the same.
xxx: gus pcm not attached, out of memory	There is not enough available memory to use the device.
xxx: can't open /dev/dsp!	Check with fstat | grep dsp if another application is holding the device open. Noteworthy troublemakers are **esound** and **KDE**'s sound support.

Another issue is that modern graphics cards often come with their own sound driver, for use with HDMI and similar. This sound device will sometimes be enumerated before the sound card and the sound card will subsequently not be used as the default playback device. To check if this is the case, run **dmesg** and look for pcm. The output looks something like this:

```
...
hdac0: HDA Driver Revision: 20100226_0142
hdac1: HDA Driver Revision: 20100226_0142
hdac0: HDA Codec #0: NVidia (Unknown)
hdac0: HDA Codec #1: NVidia (Unknown)
hdac0: HDA Codec #2: NVidia (Unknown)
hdac0: HDA Codec #3: NVidia (Unknown)
pcm0: <HDA NVidia (Unknown) PCM #0 DisplayPort> at cad 0 nid 1 on hdac0
pcm1: <HDA NVidia (Unknown) PCM #0 DisplayPort> at cad 1 nid 1 on hdac0
pcm2: <HDA NVidia (Unknown) PCM #0 DisplayPort> at cad 2 nid 1 on hdac0
pcm3: <HDA NVidia (Unknown) PCM #0 DisplayPort> at cad 3 nid 1 on hdac0
hdac1: HDA Codec #2: Realtek ALC889
pcm4: <HDA Realtek ALC889 PCM #0 Analog> at cad 2 nid 1 on hdac1
pcm5: <HDA Realtek ALC889 PCM #1 Analog> at cad 2 nid 1 on hdac1
pcm6: <HDA Realtek ALC889 PCM #2 Digital> at cad 2 nid 1 on hdac1
pcm7: <HDA Realtek ALC889 PCM #3 Digital> at cad 2 nid 1 on hdac1
...
```

Here the graphics card (NVidia) has been enumerated before the sound card (Realtek ALC889). To use the sound card as the default playback device, change hw.snd.default_unit to the unit that should be used for playback:

```
# sysctl hw.snd.default_unit=n
```

Here, n is the number of the sound device to use. In this example, it should be 4. Make this change permanent by adding the following line to /etc/sysctl.conf:

```
hw.snd.default_unit=4
```

8.2.3 Utilizing Multiple Sound Sources

Contributed by Munish Chopra.

It is often desirable to have multiple sources of sound that are able to play simultaneously. FreeBSD uses *Virtual Sound Channels*, which can be enabled using sysctl(8). Virtual channels allow one to multiplex the sound card's playback by mixing sound in the kernel.

To set the number of virtual channels, three sysctl(8) knobs are available:

```
# sysctl dev.pcm.0.play.vchans=4
# sysctl dev.pcm.0.rec.vchans=4
# sysctl hw.snd.maxautovchans=4
```

The above example allocates four virtual channels, which is a practical number for everyday use. Both dev.pcm.0.play.vchans=4 and dev.pcm.0.rec.vchans=4 are the number of virtual channels pcm0 has for playback and recording, and are configurable after a device has been attached. hw.snd.maxautovchans is the number of virtual channels a new audio device is given when it is attached using kldload(8). Since the pcm module can be loaded independently of the hardware drivers, hw.snd.maxautovchans indicates how many virtual channels will be given to devices when they are attached. Refer to pcm(4) for more information.

> **Note:** The number of virtual channels for a device cannot be changed while it is in use. First, close any programs using the device, such as music players or sound daemons.

The correct pcm device will automatically be allocated transparently to a program that requests /dev/dsp0.

8.2.4 Setting Default Values for Mixer Channels

Contributed by Josef El-Rayes.

The default values for the different mixer channels are hardcoded in the source code of the pcm(4) driver. There are many different applications and daemons that allow values to be set for the mixer that are remembered between invocations, but this is not a clean solution. It is possible to set default mixer values at the driver level. This is accomplished by defining the appropriate values in /boot/device.hints, as seen in this example:

```
hint.pcm.0.vol="50"
```

This will set the volume channel to a default value of 50 when the pcm(4) module is loaded.

8.3 MP3 Audio

Contributed by Chern Lee.

This section describes some MP3 players available for FreeBSD, how to rip audio CD tracks, and how to encode and decode MP3s.

8.3.1 MP3 Players

A popular graphical MP3 player is **XMMS**. **Winamp** skins can be used with **XMMS** since the interface is almost identical to that of Nullsoft's **Winamp**. **XMMS** also has native plug-in support.

XMMS can be installed from the `multimedia/xmms` port or package.

XMMS's interface is intuitive, with a playlist, graphic equalizer, and more. Those familiar with **Winamp** will find **XMMS** simple to use.

The `audio/mpg123` port provides an alternative, command-line MP3 player.

mpg123 can be run by specifying the sound device and the MP3 file on the command line. Assuming the audio device is `/dev/dsp1.0` and the MP3 file is *`Foobar-GreatestHits.mp3`*, enter the following to play the file:

```
# mpg123 -a /dev/dsp1.0 Foobar-GreatestHits.mp3
High Performance MPEG 1.0/2.0/2.5 Audio Player for Layer 1, 2 and 3.
Version 0.59r (1999/Jun/15). Written and copyrights by Michael Hipp.
Uses code from various people. See 'README' for more!
THIS SOFTWARE COMES WITH ABSOLUTELY NO WARRANTY! USE AT YOUR OWN RISK!




Playing MPEG stream from Foobar-GreatestHits.mp3 ...
MPEG 1.0 layer III, 128 kbit/s, 44100 Hz joint-stereo
```

8.3.2 Ripping CD Audio Tracks

Before encoding a CD or CD track to MP3, the audio data on the CD must be ripped to the hard drive. This is done by copying the raw CD Digital Audio (CDDA) data to WAV files.

The `cdda2wav` tool, which is installed with the `sysutils/cdrtools` suite, is used for ripping audio information from CDs and the information associated with them.

With the audio CD in the drive, the following command can be issued as `root` to rip an entire CD into individual (per track) WAV files:

```
# cdda2wav -D 0,1,0 -B
```

The `-D 0,1,0` indicates the SCSI device `0,1,0`, which corresponds to the output of `cdrecord -scanbus`.

cdda2wav will support ATAPI (IDE) CDROM drives. To rip from an IDE drive, specify the device name in place of the SCSI unit numbers. For example, to rip track 7 from an IDE drive:

```
# cdda2wav -D /dev/acd0 -t 7
```

To rip individual tracks, make use of the `-t` as shown:

```
# cdda2wav -D 0,1,0 -t 7
```

This example rips track seven of the audio CDROM. To rip a range of tracks, such as track one to seven, specify a range:

```
# cdda2wav -D 0,1,0 -t 1+7
```

dd(1) can also be used to extract audio tracks on ATAPI drives, as described in Section 19.5.5.

8.3.3 Encoding MP3s

Lame is a popular MP3 encoder which can be installed from the `audio/lame` port. Due to licensing restrictions, a package is not available.

The following command will convert the ripped WAV files *`audio01.wav`* to *`audio01.mp3`*:

```
# lame -h -b 128 \
--tt "Foo Song Title" \
--ta "FooBar Artist" \
--tl "FooBar Album" \
--ty "2001" \
--tc "Ripped and encoded by Foo" \
--tg "Genre" \
audio01.wav audio01.mp3
```

128 kbits is a standard MP3 bitrate. The 160 and 192 bitrates provide higher quality. The higher the bitrate, the larger the size of the resulting MP3. `-h` turns on the “higher quality but a little slower” mode. The options beginning with `--t` indicate ID3 tags, which usually contain song information, to be embedded within the MP3 file. Additional encoding options can be found in the **lame** manual page.

8.3.4 Decoding MP3s

In order to burn an audio CD from MP3s, they must first be converted to a non-compressed WAV format. Both **XMMS** and **mpg123** support the output of MP3 to an uncompressed file format.

Writing to Disk in **XMMS**:

1. Launch **XMMS**.
2. Right-click the window to bring up the **XMMS** menu.
3. Select `Preferences` under `Options`.
4. Change the Output Plugin to “Disk Writer Plugin”.
5. Press `Configure`.
6. Enter or browse to a directory to write the uncompressed files to.
7. Load the MP3 file into **XMMS** as usual, with volume at 100% and EQ settings turned off.

8. Press `Play`. The **XMMS** will appear as if it is playing the MP3, but no music will be heard. It is actually playing the MP3 to a file.
9. When finished, be sure to set the default Output Plugin back to what it was before in order to listen to MP3s again.

Writing to stdout in **mpg123**:

1. Run `mpg123 -s` *`audio01.mp3`* `>` *`audio01.pcm`*

XMMS writes a file in the WAV format, while **mpg123** converts the MP3 into raw PCM audio data. Both of these formats can be used with **cdrecord** to create audio CDs, whereas burncd(8) requires a raw Pulse-Code Modulation (PCM. When using WAV files, there will be a small tick sound at the beginning of each track. This sound is the header of the WAV file. One can remove the header with **SoX**, which can be installed from the `audio/sox` port or package:

```
% sox -t wav -r 44100 -s -w -c 2 track.wav track.raw
```

Refer to Section 19.5 for more information on using a CD burner in FreeBSD.

8.4 Video Playback

Contributed by Ross Lippert.

Before configuring video playback, determine the model of the video card and the chip it uses. While **Xorg** supports a wide variety of video cards, fewer give good playback performance. To obtain a list of extensions supported by the **Xorg** server using the card, run xdpyinfo(1) while **Xorg** is running.

It is a good idea to have a short MPEG test file for evaluating various players and options. Since some DVD applications look for DVD media in `/dev/dvd` by default, or have this device name hardcoded in them, it might be useful to make symbolic links to the proper devices:

```
# ln -sf /dev/acd0 /dev/dvd
# ln -sf /dev/acd0 /dev/rdvd
```

Due to the nature of devfs(5), manually created links will not persist after a system reboot. In order to create the symbolic links automatically when the system boots, add the following lines to `/etc/devfs.conf`:

```
link acd0 dvd
link acd0 rdvd
```

DVD decryption invokes special DVD-ROM functions and requires write permission on the DVD devices.

To enhance the shared memory **Xorg** interface, it is recommended to increase the values of these sysctl(8) variables:

```
kern.ipc.shmmax=67108864
kern.ipc.shmall=32768
```

8.4.1 Determining Video Capabilities

There are several possible ways to display video under **Xorg**. What works is largely hardware dependent. Each method described below will have varying quality across different hardware.

Common video interfaces include:

1. **Xorg**: normal output using shared memory.
2. XVideo: an extension to the **Xorg** interface which supports video in any drawable object.
3. SDL: the Simple Directmedia Layer.
4. DGA: the Direct Graphics Access.
5. SVGAlib: low level console graphics layer.

8.4.1.1 XVideo

Xorg has an extension called *XVideo*, also known as Xvideo, Xv, and xv. It allows video to be directly displayed in drawable objects through a special acceleration. This extension provides good quality playback even on low-end machines.

To check whether the extension is running, use `xvinfo`:

```
% xvinfo
```

XVideo is supported for the card if the result looks like:

```
X-Video Extension version 2.2
  screen #0
  Adaptor #0: "Savage Streams Engine"
    number of ports: 1
    port base: 43
    operations supported: PutImage
    supported visuals:
      depth 16, visualID 0x22
      depth 16, visualID 0x23
    number of attributes: 5
      "XV_COLORKEY" (range 0 to 16777215)
              client settable attribute
              client gettable attribute (current value is 2110)
      "XV_BRIGHTNESS" (range -128 to 127)
              client settable attribute
              client gettable attribute (current value is 0)
      "XV_CONTRAST" (range 0 to 255)
              client settable attribute
              client gettable attribute (current value is 128)
      "XV_SATURATION" (range 0 to 255)
              client settable attribute
              client gettable attribute (current value is 128)
      "XV_HUE" (range -180 to 180)
              client settable attribute
              client gettable attribute (current value is 0)
    maximum XvImage size: 1024 x 1024
    Number of image formats: 7
```

```
    id: 0x32595559 (YUY2)
      guid: 59555932-0000-0010-8000-00aa00389b71
      bits per pixel: 16
      number of planes: 1
      type: YUV (packed)
    id: 0x32315659 (YV12)
      guid: 59563132-0000-0010-8000-00aa00389b71
      bits per pixel: 12
      number of planes: 3
      type: YUV (planar)
    id: 0x30323449 (I420)
      guid: 49343230-0000-0010-8000-00aa00389b71
      bits per pixel: 12
      number of planes: 3
      type: YUV (planar)
    id: 0x36315652 (RV16)
      guid: 52563135-0000-0000-0000-000000000000
      bits per pixel: 16
      number of planes: 1
      type: RGB (packed)
      depth: 0
      red, green, blue masks: 0x1f, 0x3e0, 0x7c00
    id: 0x35315652 (RV15)
      guid: 52563136-0000-0000-0000-000000000000
      bits per pixel: 16
      number of planes: 1
      type: RGB (packed)
      depth: 0
      red, green, blue masks: 0x1f, 0x7e0, 0xf800
    id: 0x31313259 (Y211)
      guid: 59323131-0000-0010-8000-00aa00389b71
      bits per pixel: 6
      number of planes: 3
      type: YUV (packed)
    id: 0x0
      guid: 00000000-0000-0000-0000-000000000000
      bits per pixel: 0
      number of planes: 0
      type: RGB (packed)
      depth: 1
      red, green, blue masks: 0x0, 0x0, 0x0
```

The formats listed, such as YUV2 and YUV12, are not present with every implementation of XVideo and their absence may hinder some players.

If the result looks like:

```
X-Video Extension version 2.2
screen #0
no adaptors present
```

XVideo is probably not supported for the card. This means that it will be more difficult for the display to meet the computational demands of rendering video. Depending on the video card and processor, one might still be able to have a satisfying experience.

8.4.1.2 Simple Directmedia Layer

The Simple Directmedia Layer, SDL, is a porting layer for many operating systems allowing cross-platform applications to be developed which make efficient use of sound and graphics. The SDL layer provides a low-level abstraction to the hardware which can sometimes be more efficient than the **Xorg** interface.

SDL can be installed using the `devel/sdl12` package or port.

8.4.1.3 Direct Graphics Access

DGA is an **Xorg** extension which allows a program to bypass the **Xorg** server and directly alter the framebuffer. Because it relies on a low level memory mapping, programs using it must be run as `root`.

The DGA extension can be tested and benchmarked using dga(1). When `dga` is running, it changes the colors of the display whenever a key is pressed. To quit, press **q**.

8.4.2 Ports and Packages Dealing with Video

This section introduces some of the software available from the FreeBSD Ports Collection which can be used for video playback.

Many of the video applications which run on FreeBSD were developed as Linux applications. Many of these applications are still beta-quality. Some of the problems commonly encountered with video packages on FreeBSD include:

1. An application cannot playback a file which another application produced.
2. An application cannot playback a file which the application itself produced.
3. The same application on two different machines, rebuilt on each machine for that machine, plays back the same file differently.
4. A seemingly trivial filter, like rescaling of the image size, results in bad artifacts from a buggy rescaling routine.
5. An application frequently dumps core.
6. Documentation is not installed with the port and can be found either on the web or under the port's `work` directory.

Many applications may also exhibit "Linux-isms". There may be issues resulting from the way some standard libraries are implemented in the Linux distributions, or some features of the Linux kernel which have been assumed by the authors of the applications. These issues are not always noticed and worked around by the port maintainers, which can lead to problems like these:

1. The use of `/proc/cpuinfo` to detect processor characteristics.
2. A misuse of threads which causes a program to hang upon completion instead of truly terminating.

3. Relies on software which is not yet available in the FreeBSD Ports Collection.

8.4.2.1 MPlayer

MPlayer is a command-line video player with an optional graphical interface which aims to provide speed and flexibility. This application, as well as other graphical front-ends, is available from the FreeBSD Ports Collection.

8.4.2.1.1 Building MPlayer

MPlayer is available as a package or port in `multimedia/mplayer`. Several compile options are available and a variety of hardware checks occur during the build process. For these reasons, some users prefer to build the port rather than install the package. The available options will be displayed in a menu after these commands are input:

```
# cd /usr/ports/multimedia/mplayer
# make
```

The menu options should be reviewed to determine the type of support to compile into the port. If an option is not selected, **MPlayer** will not be able to display that type of video format. Use the arrow keys and spacebar to select the required formats. When finished, press **Enter** to continue the port compile and installation.

By default, this package or port will build the `mplayer` command line utility and the `gmplayer` graphical utility. To encode videos, install the `multimedia/mencoder` port. Due to licensing restrictions, a package is not available for `MEncoder`.

8.4.2.1.2 Using MPlayer

The first time **MPlayer** is run, it will create `~/.mplayer` in the user's home directory. This subdirectory contains default versions of the user-specific configuration files.

This section describes only a few common uses. Refer to the `mplayer` manual page for a complete description of its numerous options.

To play the file *`testfile.avi`*, specify the video interfaces with `-vo`:

```
% mplayer -vo xv testfile.avi

% mplayer -vo sdl testfile.avi

% mplayer -vo x11 testfile.avi

# mplayer -vo dga testfile.avi

# mplayer -vo 'sdl:dga' testfile.avi
```

It is worth trying all of these options, as their relative performance depends on many factors and will vary significantly with hardware.

To play a DVD, replace the *`testfile.avi`* with `dvd://`*`N`* `-dvd-device` *`DEVICE`*, where *`N`* is the title number to play and *`DEVICE`* is the device node for the DVD-ROM. For example, to play title 3 from `/dev/dvd`:

```
# mplayer -vo xv dvd://3 -dvd-device /dev/dvd
```

> **Note:** The default DVD device can be defined during the build of the **MPlayer** port by including the `WITH_DVD_DEVICE=/path/to/desired/device` option. By default, the device is `/dev/acd0`. More details can be found in the port's `Makefile.options`.

To stop, pause, advance, and so on, consult the keybindings, which are displayed by running `mplayer -h`, or read the manual page.

Additional playback options include `-fs -zoom`, which engages fullscreen mode, and `-framedrop`, which helps performance.

Each user can add commonly used options to their `~/.mplayer/config` like so:

```
vo=xv
fs=yes
zoom=yes
```

`mplayer` can be used to rip a DVD title to a `.vob`. To dump the second title from a DVD:

```
# mplayer -dumpstream -dumpfile out.vob dvd://2 -dvd-device /dev/dvd
```

The output file, `out.vob`, will be MPEG and can be manipulated by the other packages described in this section.

The MPlayer documentation (http://www.mplayerhq.hu/DOCS/) is technically informative and should be consulted by anyone wishing to obtain a high level of expertise with UNIX video. The **MPlayer** mailing list is hostile to anyone who has not bothered to read the documentation, so before making a bug report, read the documentation first.

8.4.2.1.3 MEncoder

Before using `mencoder`, it is a good idea to become familiar with the options described in the HTML documentation (http://www.mplayerhq.hu/DOCS/HTML/en/mencoder.html). There are innumerable ways to improve quality, lower bitrate, and change formats, and some of these options may make the difference between good or bad performance. Improper combinations of command line options can yield output files that are unplayable even by `mplayer`.

Here is an example of a simple copy:

```
% mencoder input.avi -oac copy -ovc copy -o output.avi
```

To rip to a file, use `-dumpfile` with `mplayer`.

To convert *`input.avi`* to the MPEG4 codec with MPEG3 audio encoding, first install the `audio/lame` port. Due to licensing restrictions, a package is not available. Once installed, type:

```
% mencoder input.avi -oac mp3lame -lameopts br=192 \
  -ovc lavc -lavcopts vcodec=mpeg4:vhq -o output.avi
```

This will produce output playable by applications such as `mplayer` and `xine`.

`input.avi` can be replaced with `dvd://1 -dvd-device /dev/dvd` and run as `root` to re-encode a DVD title directly. Since it may take a few tries to get the desired result, it is recommended to dump the title to a file and to work on the file.

8.4.2.2 The xine Video Player

xine is a video player with a reusable base library and a modular executable which can be extended with plugins. It can be installed using the `multimedia/xine` package or port.

In practice, **xine** requires either a fast CPU with a fast video card, or support for the XVideo extension. The **xine** video player performs best on XVideo interfaces.

By default, the **xine** player starts a graphical user interface. The menus can then be used to open a specific file.

Alternatively, **xine** may be invoked to play a file immediately without the graphical interface:

```
% xine
```

Alternatively, it may be invoked to play a file immediately without the GUI with the command:

```
% xine -g -p mymovie.avi
```

The xine HOWTO (http://dvd.sourceforge.net/xine-howto/en_GB/html/howto.html) contains a chapter on performance improvement which is general to all players.

8.4.2.3 The transcode Utilities

transcode provides a suite of tools for re-encoding video and audio files. **transcode** can be used to merge video files or repair broken files using command line tools with `stdin/stdout` stream interfaces.

transcode can be installed using the `multimedia/transcode` package or port. Many users prefer to compile the port as it provides a menu of compile options for specifying the support and codecs to compile in. If an option is not selected, **transcode** will not be able to encode that format. Use the arrow keys and spacebar to select the required formats. When finished, press **Enter** to continue the port compile and installation.

This example demonstrates how to convert a DivX file into a PAL MPEG-1 file (PAL VCD):

```
% transcode -i
input.avi -V --export_prof vcd-pal -o output_vcd
% mplex -f 1 -o output_vcd.mpg output_vcd.m1v output_vcd.mpa
```

The resulting MPEG file, *`output_vcd.mpg`*, is ready to be played with **MPlayer**. The file can be burned on a CD-R media to create a Video CD. In this, install and use the `multimedia/vcdimager` and `sysutils/cdrdao` programs.

In addition to the manual page for `transcode`, refer to the transcode wiki (http://www.transcoding.org/cgi-bin/transcode) for further information and examples.

8.5 Setting Up TV Cards

Original contribution by Josef El-Rayes. Enhanced and adapted by Marc Fonvieille.

8.5.1 Introduction

TV cards allow can be used to watch broadcast or cable TV on a computer. Most cards accept composite video via an

RCA or S-video input and some cards include a FM radio tuner.

FreeBSD provides support for PCI-based TV cards using a Brooktree Bt848/849/878/879 or a Conexant CN-878/Fusion 878a video capture chip with the bktr(4) driver. Ensure the board comes with a supported tuner. Consult bktr(4) for a list of supported tuners.

8.5.2 Loading the Driver

In order to use the card, the bktr(4) driver must be loaded. To automate this at boot time, add the following line to `/boot/loader.conf`:

```
bktr_load="YES"
```

Alternatively, one can statically compile support for the TV card into a custom kernel. In that case, add the following lines to the custom kernel configuration file:

```
device   bktr
device iicbus
device iicbb
device smbus
```

These additional devices are necessary as the card components are interconnected via an I2C bus. Then, build and install a new kernel.

To test the driver, reboot the system. The TV card should appear in the boot messages, as seen in this example:

```
bktr0: <BrookTree 848A> mem 0xd7000000-0xd7000fff irq 10 at device 10.0 on pci0
iicbb0: <I2C bit-banging driver> on bti2c0
iicbus0: <Philips I2C bus> on iicbb0 master-only
iicbus1: <Philips I2C bus> on iicbb0 master-only
smbus0: <System Management Bus> on bti2c0
bktr0: Pinnacle/Miro TV, Philips SECAM tuner.
```

The messages will differ according to the hardware. Check the messages to determine if the tuner is correctly detected. It is still possible to override some of the detected parameters with sysctl(8) MIBs and kernel configuration file options. For example, to force the tuner to a Philips SECAM tuner, add the following line to a custom kernel configuration file:

```
options OVERRIDE_TUNER=6
```

or, use sysctl(8):

```
# sysctl hw.bt848.tuner=6
```

Refer to bktr(4) and `/usr/src/sys/conf/NOTES` for more details on the available options.

8.5.3 Useful Applications

To use the TV card, install one of the following applications:

- `multimedia/fxtv` provides TV-in-a-window and image/audio/video capture capabilities.

- `multimedia/xawtv` is another TV application with similar features.
- `audio/xmradio` provides an application for using the FM radio tuner of a TV card.

More applications are available in the FreeBSD Ports Collection.

8.5.4 Troubleshooting

If any problems are encountered with the TV card, check that the video capture chip and the tuner are supported by bktr(4) and that the right configuration options were used. For more support and various questions about TV cards, refer to the archives of the freebsd-multimedia (http://lists.FreeBSD.org/mailman/listinfo/freebsd-multimedia) mailing list.

8.6 MythTV

MythTV is a popular, open source PVR application. This section demonstrates how to install and setup MythTV on FreeBSD. Refer to the MythTV wiki (http://www.mythtv.org/wiki/) for more information on how to use MythTV.

MythTV requires a frontend and a backend; however, it allows the user to have the frontend and backend on different machines.

For the frontend, `multimedia/mythtv-frontend` is required, as well as an X server, which can be found in `x11/xorg`. Ideally, the frontend computer also has a video card that supports XvMC and, optionally, a LIRC-compatible remote.

For the backend, `multimedia/mythtv` is required, along with the MySQL™ database server. Optionally a tuner and storage for any recorded data. The MySQL package should be automatically installed as a dependency when installing `multimedia/mythtv`.

8.6.1 Hardware

MythTV is designed to utilize V4L to access video input devices such as encoders and tuners. At this time, MythTV works best with USB DVB-S/C/T cards supported by `multimedia/webcamd`, as it provides a V4L userland application. Any DVB card supported by **webcamd** should work with MythTV. A list of known working cards can be found here (http://wiki.freebsd.org/WebcamCompat). Drivers are also available for Hauppauge cards in the following ports: `multimedia/pvr250` and `multimedia/pvrxxx`, but they provide a non-standard driver interface that does not work with versions of MythTV greater than 0.23. Due to licensing restrictions, no packages are available and these two ports must be compiled.

The HTPC wiki page (http://wiki.freebsd.org/HTPC) contains a list of all available DVB drivers.

8.6.2 Setting up MythTV

To install the MythTV port:

```
# cd /usr/ports/multimedia/mythtv
# make install
```

Once installed, set up the MythTV database:

```
# mysql -uroot -p < /usr/local/share/mythtv/database/mc.sql
```

Configure the backend:

```
# mythtv-setup
```

Start the backend:

```
# echo 'mythbackend_enable="YES"' >> /etc/rc.conf
# service mythbackend start
```

8.7 Image Scanners

Written by Marc Fonvieille.

In FreeBSD, access to image scanners is provided by the **SANE** (Scanner Access Now Easy) API available through the FreeBSD Ports Collection. **SANE** will also use some FreeBSD device drivers to provide access to the scanner hardware.

FreeBSD supports both SCSI and USB scanners. Be sure the scanner is supported by **SANE** prior to performing any configuration. Refer to the supported devices list (http://www.sane-project.org/sane-supported-devices.html) for more information about supported scanners.

8.7.1 Kernel Configuration

Both SCSI and USB interfaces are supported. Depending upon the scanner interface, different device drivers are required.

8.7.1.1 USB Interface

The `GENERIC` kernel by default includes the device drivers needed to support USB scanners. Users with a custom kernel should ensure that the following lines are present in the custom kernel configuration file:

```
device usb
device uhci
device ohci
device ehci
```

Plug in the USB scanner. Use dmesg(8) to determine whether the scanner appears in the system message buffer:

```
ugen0.2: <EPSON> at usbus0
```

These messages indicate that the scanner is using either `/dev/ugen0.2` or `/dev/uscanner0`, depending on the FreeBSD version. For this example, a EPSON Perfection® 1650 USB scanner was used.

8.7.1.2 SCSI Interface

If the scanner uses a SCSI interface, it is important to know which SCSI controller board it will use. Depending upon the SCSI chipset, a custom kernel configuration file may be needed. The `GENERIC` kernel supports the most common

SCSI controllers. Refer to `/usr/src/sys/conf/NOTES` to determine the correct line to add to a custom kernel configuration file. In addition to the SCSI adapter driver, the following lines are needed in the kernel configuration file:

```
device scbus
device pass
```

Verify that the device is displayed in the system message buffer:

```
pass2 at aic0 bus 0 target 2 lun 0
pass2: <AGFA SNAPSCAN 600 1.10> Fixed Scanner SCSI-2 device
pass2: 3.300MB/s transfers
```

If the scanner was not powered-on at system boot, it is still possible to manually force the detection by performing a SCSI bus scan with the camcontrol(8) command:

```
# camcontrol rescan all
Re-scan of bus 0 was successful
Re-scan of bus 1 was successful
Re-scan of bus 2 was successful
Re-scan of bus 3 was successful
```

The scanner should now appear in the SCSI devices list:

```
# camcontrol devlist
<IBM DDRS-34560 S97B>              at scbus0 target 5 lun 0 (pass0,da0)
<IBM DDRS-34560 S97B>              at scbus0 target 6 lun 0 (pass1,da1)
<AGFA SNAPSCAN 600 1.10>           at scbus1 target 2 lun 0 (pass3)
<PHILIPS CDD3610 CD-R/RW 1.00>     at scbus2 target 0 lun 0 (pass2,cd0)
```

Refer to scsi(4) and camcontrol(8) for more details about SCSI devices on FreeBSD.

8.7.2 SANE Configuration

The **SANE** system is split in two parts: the backends (`graphics/sane-backends`) and the frontends (`graphics/sane-frontends`). The backends provide access to the scanner. The **SANE**'s supported devices (http://www.sane-project.org/sane-supported-devices.html) list specifies which backend will support the image scanner. The correct backend is needed in order to use the scanner. The frontends provide the graphical scanning interface, **xscanimage**.

After installing the `graphics/sane-backends` port or package, use `sane-find-scanner` to check the scanner detection by the **SANE** system:

```
# sane-find-scanner -q
found SCSI scanner "AGFA SNAPSCAN 600 1.10" at /dev/pass3
```

The output should show the interface type of the scanner and the device node used to attach the scanner to the system. The vendor and the product model may or may not appear.

> **Note:** Some USB scanners require firmware to be loaded. Refer to sane-find-scanner(1) and sane(7) for details.

Next, check if the scanner will be identified by a scanning frontend. By default, the **SANE** backends come with a command line tool called scanimage(1). This command can be used to list the devices and perform an image acquisition. Use `-L` to list the scanner devices:

```
# scanimage -L
device ‘snapscan:/dev/pass3’ is a AGFA SNAPSCAN 600 flatbed scanner
```

Here is the output for the USB scanner used in Section 8.7.1.1:

```
# scanimage -L
device ’epson2:libusb:/dev/usb:/dev/ugen0.2’ is a Epson GT-8200 flatbed scanner
```

In this output, `’epson2:libusb:/dev/usb:/dev/ugen0.2’` is the backend name (`epson2`) and the device node (`/dev/ugen0.2`) used by the scanner.

> **Note:** No output or a message saying that no scanners were identified indicates that scanimage(1) is unable to identify the scanner. If this happens, edit the backend configuration file in `/usr/local/etc/sane.d/` and define the scanner device used.
>
> In the above example, the USB scanner is perfectly detected and working.
>
> To determine if the scanner is correctly identified:
>
> ```
> # scanimage -L
>
> No scanners were identified. If you were expecting something different,
> check that the scanner is plugged in, turned on and detected by the
> sane-find-scanner tool (if appropriate). Please read the documentation
> which came with this software (README, FAQ, manpages).
> ```
>
> Since the scanner is not identified, edit `/usr/local/etc/sane.d/epson2.conf`. In this example, the scanner model is EPSON Perfection 1650 and it uses the `epson2` backend. When editing, read the help comments in the backend configuration file. Line changes are simple: comment out all lines that have the wrong interface for the scanner. In this example, comment out all lines starting with the word `scsi` as the scanner uses the USB interface. Then, at the end of the file, add a line specifying the interface and the device node used. In this case, add the following line:
>
> ```
> usb /dev/uscanner0
> ```
>
> Save the edits and verify that the scanner is identified:
>
> ```
> # scanimage -L
> device ‘epson:/dev/uscanner0’ is a Epson GT-8200 flatbed scanner
> ```
>
> The `‘epson:/dev/uscanner0’` field now gives the right backend name and the device node.

Once `scanimage -L` sees the scanner, the configuration is complete and the device is now ready to scan.

While scanimage(1) can be used to perform an image acquisition from the command line, it is often preferable to use a graphical interface to perform image scanning. The `graphics/sane-frontends` package or port installs a simple but efficient graphical interface, **xscanimage**.

Xsane, which is installed with the `graphics/xsane` package or port, is another popular graphical scanning frontend. It offers advanced features such as various scanning modes, color correction, and batch scans. Both of these applications are usable as a **GIMP** plugin.

8.7.3 Giving Other Users Access to the Scanner

In order to have access to the scanner, a user needs read and write permissions to the device node used by the scanner. In the previous example, the USB scanner uses the device node `/dev/ugen0.2` which is really a symlink to the real device node `/dev/usb/0.2.0`. The symlink and the device node are owned, respectively, by the `wheel` and `operator` groups. Adding the user to these groups will allow access to the scanner. However, for security reasons, always think twice before adding a user to any group, especially `wheel`. A better solution is to create a group to make the scanner device accessible to members of this group.

This example creates a group called *`usb`* using pw(8):

```
# pw groupadd usb
```

Then, make the `/dev/ugen0.2` symlink and the `/dev/usb/0.2.0` device node accessible to the `usb` group with write permissions of (`0660` or `0664`. All of this is done by adding the following lines to `/etc/devfs.rules`:

```
[system=5]
add path ugen0.2 mode 0660 group usb
add path usb/0.2.0 mode 0666 group usb
```

Finally, add the users to *`usb`* in order to allow access to the scanner:

```
# pw groupmod usb -m joe
```

For more details refer to pw(8).

Chapter 9 Configuring the FreeBSD Kernel

Updated and restructured by Jim Mock. Originally contributed by Jake Hamby.

9.1 Synopsis

The kernel is the core of the FreeBSD operating system. It is responsible for managing memory, enforcing security controls, networking, disk access, and much more. While much of FreeBSD is dynamically configurable, it is still occasionally necessary to configure and compile a custom kernel.

After reading this chapter, you will know:

- When to build a custom kernel.
- How to customize a kernel configuration file.
- How to use the kernel configuration file to create and build a new kernel.
- How to install the new kernel.
- How to troubleshoot if things go wrong.

All of the commands listed in the examples in this chapter should be executed as `root`.

9.2 Why Build a Custom Kernel?

Traditionally, FreeBSD used a "monolithic" kernel. The kernel was one large program, supported a fixed list of devices, and in order to change the kernel's behavior, one had to compile a new kernel, and then reboot into the new kernel.

Today, most of the functionality in the FreeBSD kernel is contained in modules which can be dynamically loaded and unloaded from the kernel as necessary. This allows the running kernel to adapt immediately to new hardware or for new functionality to be brought into the kernel. This is known as a modular kernel.

Occasionally, it is still necessary to perform static kernel configuration. This may be because the functionality is so tied to the kernel that it can not be made dynamically loadable. Some security environments prevent the loading and unloading of kernel modules, and require that only needed functionality is statically compiled into the kernel.

Building a custom kernel is often a rite of passage for advanced BSD users. This process, while time consuming, can provide benefits to the FreeBSD system. Unlike the `GENERIC` kernel, which must support a wide range of hardware, a custom kernel can be stripped down to only provide support for that computer's hardware. This has a number of benefits, such as:

- Faster boot time. Since the kernel will only probe the hardware on the system, the time it takes the system to boot can decrease.
- Lower memory usage. A custom kernel often uses less memory than the `GENERIC` kernel by omitting unused features and device drivers. This is important because the kernel code remains resident in physical memory at all times, preventing that memory from being used by applications. For this reason, a custom kernel is useful on a system with a small amount of RAM.

- Additional hardware support. A custom kernel can add in support for devices which are not present in the `GENERIC` kernel.

9.3 Finding the System Hardware

Written by Tom Rhodes.

Before venturing into kernel configuration, it would be wise to get an inventory of the machine's hardware. In cases where FreeBSD is not the primary operating system, the inventory list can be created by viewing the current operating system configuration. For example, Microsoft's **Device Manager** contains information about installed devices.

> **Note:** Some versions of Microsoft Windows have a **System** icon which will display a screen where **Device Manager** may be accessed.

If another operating system does not exist on the machine, the administrator must find this information out manually. One method is using dmesg(8) and man(1). Most device drivers on FreeBSD have a manual page, listing supported hardware. During the boot probe, found hardware will be listed. For example, the following lines indicate that the psm(4) driver found a mouse:

```
psm0: <PS/2 Mouse> irq 12 on atkbdc0
psm0: [GIANT-LOCKED]
psm0: [ITHREAD]
psm0: model Generic PS/2 mouse, device ID 0
```

This driver will need to be included in the custom kernel configuration file or loaded using loader.conf(5).

On occasion, the data from `dmesg` will only show system messages instead of the boot probe output. In these situations, the output may be obtained by reading `/var/run/dmesg.boot`.

Another method for finding hardware is to use pciconf(8) which provides more verbose output. For example:

```
ath0@pci0:3:0:0:        class=0x020000 card=0x058a1014 chip=0x1014168c rev=0x01 hdr=0x00
    vendor     = 'Atheros Communications Inc.'
    device     = 'AR5212 Atheros AR5212 802.11abg wireless'
    class      = network
    subclass   = ethernet
```

This output, obtained by using `pciconf -lv`, shows that the `ath` driver located a wireless Ethernet device. Type `man ath` to read ath(4).

The `-k` flag, when passed to man(1) can be used to provide useful information. For example, to display a list of manual pages which contain the specified word:

```
# man -k Atheros

ath(4)                  - Atheros IEEE 802.11 wireless network driver
ath_hal(4)              - Atheros Hardware Access Layer (HAL)
```

Armed with a hardware inventory list, the process of building a custom kernel should appear less daunting.

9.4 Kernel Drivers, Subsystems, and Modules

Before building a custom kernel, consider the reason for doing so. If there is a need for specific hardware support, it may already exist as a module.

Kernel modules exist in `/boot/kernel` and may be dynamically loaded into the running kernel using kldload(8). Most, if not all kernel drivers have a loadable module and manual page. For example, the ath(4) wireless Ethernet driver has the following information in its manual page:

```
Alternatively, to load the driver as a module at boot time, place the
following line in loader.conf(5):

    if_ath_load="YES"
```

Adding `if_ath_load="YES"` to `/boot/loader.conf` will enable loading this module dynamically at boot time.

In some cases, there is no associated module. This is mostly true for certain subsystems. One way to tell if a driver is available is to check for the module itself.

> **Warning:** It is easy to remove support for a device or option and end up with a broken kernel. For example, if the ata(4) driver is removed from the kernel configuration file, a system using ATA disk drivers may not boot. When in doubt, just leave support in the kernel.

9.5 Building and Installing a Custom Kernel

> **Note:** It is required to have the full FreeBSD source tree installed to build the kernel.

The kernel build is located at `/usr/src/sys`. It contains a number of subdirectories representing different parts of the kernel. These include *`arch`*`/conf`, which contains the kernel configuration file, and `compile`, which is the staging area where the kernel will be built. *`arch`* contains subdirectories for each supported architecture: `i386`, `amd64`, `ia64`, `powerpc`, `sparc64`, and `pc98`. Everything inside a particular architecture's directory deals with that architecture only and the rest of the code is machine independent code common to all platforms. Notice the logical organization of the directory structure, with each supported device, file system, and option in its own subdirectory.

The examples in this chapter assume the i386 architecture. If the system has a different architecture, change the path names accordingly.

> **Note:** If `/usr/src/` does not exist or it is empty, source has not been installed. The easiest way to install source is to use **svn** as described in Section A.5. One should also create a symlink to `/usr/src/sys/`:
>
> ```
> # ln -s /usr/src/sys /sys
> ```

Next, **cd** to *`arch`*`/conf` and copy the `GENERIC` configuration file to the name of the custom kernel. For example:

```
# cd /usr/src/sys/i386/conf
# cp GENERIC MYKERNEL
```

Traditionally, this name is in all capital letters. When maintaining multiple FreeBSD machines with different hardware, it is a good idea to name it after the machine's hostname. This example uses *MYKERNEL*.

> **Tip:** When finished customizing the kernel configuration file, save a backup copy to a location outside of `/usr/src`. Do not edit `GENERIC` directly.
>
> Alternately, keep the kernel configuration file elsewhere and create a symbolic link to the file in *i386*.
>
> For example:

```
# cd /usr/src/sys/i386/conf
# mkdir /root/kernels
# cp GENERIC /root/kernels/MYKERNEL
# ln -s /root/kernels/MYKERNEL
```

Edit *MYKERNEL* with a text editor. The default editor is **vi**, whose usage is covered well in many books in the bibliography. An easier editor for beginners, called **ee**, is also available. Feel free to change the comment lines at the top to reflect the configuration or the changes made to differentiate it from `GENERIC`.

If the `GENERIC` configuration file seems overwhelming, follow the descriptions in the Configuration File section slowly and carefully.

> **Note:** After syncing the source tree with the latest sources, *always* read `/usr/src/UPDATING` before performing any update steps. This file describes any important issues or areas requiring special attention within the updated source code. `/usr/src/UPDATING` always matches the version of the FreeBSD source and contains more up-to-date information than this Handbook.

After saving the edits, compile the source code for the kernel.

Building a Kernel

> **Note:** It is required to have the full FreeBSD source tree installed to build the kernel.

1. `cd` to `/usr/src`:

   ```
   # cd /usr/src
   ```

2. Compile the new kernel by specifying the name of the custom kernel configuration file:

   ```
   # make buildkernel KERNCONF=MYKERNEL
   ```

3. Install the new kernel:

   ```
   # make installkernel KERNCONF=MYKERNEL
   ```

   > **Tip:** By default, when a custom kernel is compiled, *all* kernel modules are rebuilt as well. To update a kernel faster or to build only custom modules, edit `/etc/make.conf` before starting to build the kernel:
   >
   > ```
   > MODULES_OVERRIDE = linux acpi sound/sound sound/driver/ds1 ntfs
   > ```

This variable specifies the list of modules to build instead the default of building of all of them.

```
WITHOUT_MODULES = linux acpi sound ntfs
```

This variable sets up a list of top level modules to exclude from the build process. For other available variables, refer to make.conf(5).

The new kernel will be copied to `/boot/kernel` as `/boot/kernel/kernel` and the old kernel will be moved to `/boot/kernel.old/kernel`. Now, shutdown the system and reboot into the new kernel. If something goes wrong, refer to the troubleshooting instructions and the section which explains how to recover when the new kernel does not boot.

Note: Other files relating to the boot process, such as the boot loader(8) and configuration, are stored in `/boot`. Third party or custom modules can be placed in `/boot/kernel`, although users should be aware that keeping modules in sync with the compiled kernel is very important. Modules not intended to run with the compiled kernel may result in instability.

9.6 The Configuration File

Updated by Joel Dahl.

The general format of a configuration file is quite simple. Each line contains a keyword and one or more arguments. For simplicity, most lines only contain one argument. Anything following a # is considered a comment and ignored. The following sections describe each keyword, in the order they are listed in `GENERIC`. For an exhaustive list of architecture dependent options and devices, refer to `NOTES` in the same directory as `GENERIC` for that architecture. For architecture independent options, refer to `/usr/src/sys/conf/NOTES`.

An `include` directive is available for use in configuration files. This allows another configuration file to be included in the current one, making it easy to maintain small changes relative to an existing file. For example, if only a small number of additional options or drivers are required, this allows a delta to be maintained with respect to GENERIC:

```
include GENERIC
ident MYKERNEL

options         IPFIREWALL
options         DUMMYNET
options         IPFIREWALL_DEFAULT_TO_ACCEPT
options         IPDIVERT
```

Using this method, the local configuration file expresses local differences from a `GENERIC` kernel. As upgrades are performed, new features added to `GENERIC` will be also be added to the local kernel unless they are specifically prevented using `nooptions` or `nodevice`. A comprehensive list of configuration directives and their descriptions may be found in config(5).

The remainder of this chapter addresses the contents of a typical configuration file and the role various options and devices play.

Note: To build a file which contains all available options, run the following command as `root`:

```
# cd /usr/src/sys/i386/conf && make LINT
```

The following is an example of the `GENERIC` kernel configuration file with various additional comments where needed for clarity. This example should match the copy in `/usr/src/sys/i386/conf/GENERIC` fairly closely.

```
machine         i386
```

This is the machine architecture. It must be either `amd64`, `i386`, `ia64`, `pc98`, `powerpc`, or `sparc64`.

```
cpu          I486_CPU
cpu          I586_CPU
cpu          I686_CPU
```

This option specifies the type of CPU. It is fine to have multiple instances of the CPU entries, but for a custom kernel it is best to specify the CPU. To determine the CPU type, review the boot messages in `/var/run/dmesg.boot`.

```
ident          GENERIC
```

This is the identification of the kernel. Change this to the new kernel name, such as *`MYKERNEL`*. The value in the `ident` string will print when the kernel boots.

```
#To statically compile in device wiring instead of /boot/device.hints
#hints          "GENERIC.hints"         # Default places to look for devices.
```

device.hints(5) is used to configure options for device drivers. The default location is `/boot/device.hints`. The `hints` option compiles these hints statically into the kernel so that there is no need to create `/boot/device.hints`.

```
makeoptions     DEBUG=-g          # Build kernel with gdb(1) debug symbols
```

This option enables debugging information when passed to gcc(1).

```
options          SCHED_ULE         # ULE scheduler
```

The default system scheduler for FreeBSD. Keep this.

```
options          PREEMPTION         # Enable kernel thread preemption
```

Allows kernel threads to be preempted by higher priority threads. This helps with interactivity and allows interrupt threads to run sooner rather than waiting.

```
options          INET              # InterNETworking
```

Networking support. This is mandatory as most programs require at least loopback networking.

```
options          INET6             # IPv6 communications protocols
```

This enables the IPv6 communication protocols.

```
options          FFS               # Berkeley Fast Filesystem
```

This is the basic hard drive file system. Leave it in if the system boots from the hard disk.

```
options         SOFTUPDATES       # Enable FFS Soft Updates support
```

This option enables Soft Updates in the kernel which helps to speed up write access on the disks. Even when this functionality is provided by the kernel, it must be turned on for specific disks. Review the output of mount(8) to determine if Soft Updates is enabled. If the `soft-updates` option is not in the output, it can be activated using tunefs(8) for existing file systems or newfs(8) for new file systems.

```
options         UFS_ACL           # Support for access control lists
```

This option enables kernel support for access control lists (ACLs). This relies on the use of extended attributes and UFS2, and the feature is described in detail in Section 15.11. ACLs are enabled by default and should not be disabled in the kernel if they have been used previously on a file system, as this will remove the ACLs, changing the way files are protected in unpredictable ways.

```
options         UFS_DIRHASH       # Improve performance on big directories
```

This option includes functionality to speed up disk operations on large directories, at the expense of using additional memory. Keep this for a large server or interactive workstation, and remove it from smaller systems where memory is at a premium and disk access speed is less important, such as a firewall.

```
options         MD_ROOT           # MD is a potential root device
```

This option enables support for a memory backed virtual disk used as a root device.

```
options         NFSCLIENT         # Network Filesystem Client
options         NFSSERVER         # Network Filesystem Server
options         NFS_ROOT          # NFS usable as /, requires NFSCLIENT
```

The network file system (NFS). These lines can be commented unless the system needs to mount partitions from a NFS file server over TCP/IP.

```
options         MSDOSFS           # MSDOS Filesystem
```

The MS-DOS file system. Unless the system needs to mount a DOS formatted hard drive partition at boot time, comment this out. It will be automatically loaded the first time a DOS partition is mounted. The `emulators/mtools` package allows access to DOS floppies without having to mount and unmount them and does not require `MSDOSFS`.

```
options         CD9660            # ISO 9660 Filesystem
```

The ISO 9660 file system for CDROMs. Comment it out if the system does not have a CDROM drive or only mounts data CDs occasionally since it will be dynamically loaded the first time a data CD is mounted. Audio CDs do not need this file system.

```
options         PROCFS            # Process filesystem (requires PSEUDOFS)
```

The process file system. This is a “pretend” file system mounted on `/proc` which allows some programs to provide more information on what processes are running. Use of `PROCFS` is not required under most circumstances, as most debugging and monitoring tools have been adapted to run without `PROCFS`. The default installation will not mount this file system by default.

```
options         PSEUDOFS          # Pseudo-filesystem framework
```

Kernels making use of `PROCFS` must also include support for `PSEUDOFS`.

```
options         GEOM_PART_GPT     # GUID Partition Tables.
```

Adds support for GUID Partition Tables (http://en.wikipedia.org/wiki/GUID_Partition_Table) (GPT). GPT provides the ability to have a large number of partitions per disk, 128 in the standard configuration.

```
options         COMPAT_43         # Compatible with BSD 4.3 [KEEP THIS!]
```

Compatibility with 4.3BSD. Leave this in as some programs will act strangely if this is commented out.

```
options         COMPAT_FREEBSD4   # Compatible with FreeBSD4
```

This option is required to support applications compiled on older versions of FreeBSD that use older system call interfaces. It is recommended that this option be used on all i386 systems that may run older applications. Platforms that gained support after FreeBSD 4.X, such as ia64 and SPARC64, do not require this option.

```
options         COMPAT_FREEBSD5   # Compatible with FreeBSD5
```

This option is required to support applications compiled on FreeBSD 5.X versions that use FreeBSD 5.X system call interfaces.

```
options         COMPAT_FREEBSD6   # Compatible with FreeBSD6
```

This option is required to support applications compiled on FreeBSD 6.X versions that use FreeBSD 6.X system call interfaces.

```
options         COMPAT_FREEBSD7   # Compatible with FreeBSD7
```

This option is required on FreeBSD 8 and above to support applications compiled on FreeBSD 7.X versions that use FreeBSD 7.X system call interfaces.

```
options         SCSI_DELAY=5000  # Delay (in ms) before probing SCSI
```

This causes the kernel to pause for 5 seconds before probing each SCSI device in the system. If the system only has IDE hard drives, ignore this or lower the number to speed up booting. However, if FreeBSD has trouble recognizing the SCSI devices, the number will have to be raised again.

```
options         KTRACE            # ktrace(1) support
```

This enables kernel process tracing, which is useful in debugging.

```
options         SYSVSHM           # SYSV-style shared memory
```

This option provides for System V shared memory. The most common use of this is the XSHM extension in X, which many graphics-intensive programs will automatically take advantage of for extra speed. If **Xorg** is installed, include this.

```
options         SYSVMSG           # SYSV-style message queues
```

Support for System V messages. This option only adds a few hundred bytes to the kernel.

```
options         SYSVSEM           # SYSV-style semaphores
```

Support for System V semaphores. Less commonly used, but only adds a few hundred bytes to the kernel.

> **Note:** Using `-p` with ipcs(1) will list any processes using each of these System V facilities.

```
options             _KPOSIX_PRIORITY_SCHEDULING # POSIX P1003_1B real-time extensions
```

Real-time extensions added in the 1993 POSIX®. Certain applications in the Ports Collection use these.

```
options          KBD_INSTALL_CDEV  # install a CDEV entry in /dev
```

This option is required to allow the creation of keyboard device nodes in `/dev`.

```
device          apic               # I/O APIC
```

This device enables the use of the I/O APIC for interrupt delivery. It can be used in both uni-processor and SMP kernels, but is required for SMP kernels. Add `options SMP` to include support for multiple processors.

> **Note:** This device exists only on the i386 architecture and this configuration line should not be used on other architectures.

```
device          eisa
```

Include this for systems with an EISA motherboard. This enables auto-detection and configuration support for all devices on the EISA bus.

```
device          pci
```

Include this for systems with a PCI motherboard. This enables auto-detection of PCI cards and gatewaying from the PCI to ISA bus.

```
# Floppy drives
device          fdc
```

This is the floppy drive controller.

```
# ATA and ATAPI devices
device          ata
```

This driver supports all ATA and ATAPI devices. Only one `device ata` line is needed for the kernel to detect all PCI ATA/ATAPI devices on modern machines.

```
device          atadisk                 # ATA disk drives
```

This is needed along with `device ata` for ATA disk drives.

```
device          ataraid                 # ATA RAID drives
```

This is needed along with `device ata` for ATA RAID drives.

```
device          atapicd                 # ATAPI CDROM drives
```

This is needed along with `device ata` for ATAPI CDROM drives.

```
device          atapifd                 # ATAPI floppy drives
```

This is needed along with `device ata` for ATAPI floppy drives.

```
device          atapist                 # ATAPI tape drives
```

This is needed along with `device ata` for ATAPI tape drives.

```
options         ATA_STATIC_ID           # Static device numbering
```

This makes the controller number static. Without this, the device numbers are dynamically allocated.

```
# SCSI Controllers
device          ahb        # EISA AHA1742 family
device          ahc        # AHA2940 and onboard AIC7xxx devices
options         AHC_REG_PRETTY_PRINT    # Print register bitfields in debug
                                        # output.  Adds ~128k to driver.
device          ahd        # AHA39320/29320 and onboard AIC79xx devices
options         AHD_REG_PRETTY_PRINT    # Print register bitfields in debug
                                        # output.  Adds ~215k to driver.
device          amd        # AMD 53C974 (Teckram DC-390(T))
device          isp        # Qlogic family
#device         ispfw      # Firmware for QLogic HBAs- normally a module
device          mpt        # LSI-Logic MPT-Fusion
#device         ncr        # NCR/Symbios Logic
device          sym        # NCR/Symbios Logic (newer chipsets + those of 'ncr')
device          trm        # Tekram DC395U/UW/F DC315U adapters

device          adv        # Advansys SCSI adapters
device          adw        # Advansys wide SCSI adapters
device          aha        # Adaptec 154x SCSI adapters
device          aic        # Adaptec 15[012]x SCSI adapters, AIC-6[23]60.
device          bt         # Buslogic/Mylex MultiMaster SCSI adapters

device          ncv        # NCR 53C500
device          nsp        # Workbit Ninja SCSI-3
device          stg        # TMC 18C30/18C50
```

In this section, comment out any SCSI controllers not on the system. For an IDE only system, these lines can be removed. The `*_REG_PRETTY_PRINT` lines are debugging options for their respective drivers.

```
# SCSI peripherals
device          scbus      # SCSI bus (required for SCSI)
device          ch         # SCSI media changers
device          da         # Direct Access (disks)
device          sa         # Sequential Access (tape etc)
device          cd         # CD
device          pass       # Passthrough device (direct SCSI access)
device          ses        # SCSI Environmental Services (and SAF-TE)
```

Comment out any SCSI peripherals not on the system. If the system only has IDE hardware, these lines can be removed completely.

Note: The USB umass(4) driver and a few other drivers use the SCSI subsystem even though they are not real SCSI devices. Do not remove SCSI support if any such drivers are included in the kernel configuration.

```
# RAID controllers interfaced to the SCSI subsystem
device          amr         # AMI MegaRAID
device          arcmsr      # Areca SATA II RAID
device          asr         # DPT SmartRAID V, VI and Adaptec SCSI RAID
device          ciss        # Compaq Smart RAID 5*
device          dpt         # DPT Smartcache III, IV - See NOTES for options
device          hptmv       # Highpoint RocketRAID 182x
device          hptrr       # Highpoint RocketRAID 17xx, 22xx, 23xx, 25xx
device          iir         # Intel Integrated RAID
device          ips         # IBM (Adaptec) ServeRAID
device          mly         # Mylex AcceleRAID/eXtremeRAID
device          twa         # 3ware 9000 series PATA/SATA RAID

# RAID controllers
device          aac         # Adaptec FSA RAID
device          aacp        # SCSI passthrough for aac (requires CAM)
device          ida         # Compaq Smart RAID
device          mfi         # LSI MegaRAID SAS
device          mlx         # Mylex DAC960 family
device          pst         # Promise Supertrak SX6000
device          twe         # 3ware ATA RAID
```

Supported RAID controllers. If the system does not have any of these, comment them out or remove them.

```
# atkbdc0 controls both the keyboard and the PS/2 mouse
device          atkbdc      # AT keyboard controller
```

The `atkbdc` keyboard controller provides I/O services for the AT keyboard and PS/2 style pointing devices. This controller is required by atkbd(4) and psm(4).

```
device          atkbd       # AT keyboard
```

The atkbd(4) driver, together with the atkbdc(4) controller, provides access to the AT 84 keyboard or the AT enhanced keyboard which is connected to the AT keyboard controller.

```
device          psm         # PS/2 mouse
```

Use this device if the mouse plugs into the PS/2 mouse port.

```
device          kbdmux         # keyboard multiplexer
```

Basic support for keyboard multiplexing. If the system does not use more than one keyboard, this line can be safely removed.

```
device          vga         # VGA video card driver
```

The vga(4) video card driver.

```
device          splash      # Splash screen and screen saver support
```

Required by the boot splash screen and screen savers.

```
# syscons is the default console driver, resembling an SCO console
device          sc
```

sc(4) is the default console driver and resembles a SCO console. Since most full-screen programs access the console through a terminal database library like `termcap`, it should not matter whether this or `vt`, the `VT220` compatible console driver, is used. When a user logs in, the `TERM` variable can be set to `scoansi` if full-screen programs have trouble running under this console.

```
# Enable this for the pcvt (VT220 compatible) console driver
#device          vt
#options         XSERVER          # support for X server on a vt console
#options         FAT_CURSOR       # start with block cursor
```

This is a VT220-compatible console driver, backward compatible to VT100/102. It works well on some laptops which have hardware incompatibilities with `sc`. Users may need to set `TERM` to `vt100` or `vt220` after login. This driver is useful when connecting to a large number of different machines over the network, where `termcap` or `terminfo` entries for the `sc` device are not available as `vt100` should be available on virtually any platform.

```
device          agp
```

Include this if the system has an AGP card. This will enable support for AGP and AGP GART for boards which have these features.

```
# Add suspend/resume support for the i8254.
device          pmtimer
```

Timer device driver for power management events, such as APM and ACPI.

```
# PCCARD (PCMCIA) support
# PCMCIA and cardbus bridge support
device          cbb              # cardbus (yenta) bridge
device          pccard           # PC Card (16-bit) bus
device          cardbus          # CardBus (32-bit) bus
```

PCMCIA support. Keep this on laptop systems.

```
# Serial (COM) ports
device          sio              # 8250, 16[45]50 based serial ports
```

These are the serial ports referred to as `COM` ports in Windows.

> **Note:** If the system has an internal modem on `COM4` and a serial port at `COM2`, change the IRQ of the modem to 2. For a multiport serial card, refer to sio(4) for more information on the proper values to add to `/boot/device.hints`. Some video cards, notably those based on S3 chips, use I/O addresses in the form of `0x*2e8`. Since many cheap serial cards do not fully decode the 16-bit I/O address space, they clash with these cards, making the `COM4` port practically unavailable.
>
> Each serial port is required to have a unique IRQ and the default IRQs for `COM3` and `COM4` cannot be used. The exception is multiport cards where shared interrupts are supported.

```
# Parallel port
device          ppc
```

This is the ISA bus parallel port interface.

```
device          ppbus      # Parallel port bus (required)
```

Provides support for the parallel port bus.

```
device          lpt        # Printer
```

Adds support for parallel port printers.

> **Note:** All three of the above are required to enable parallel printer support.

```
device          ppi        # Parallel port interface device
```

The general-purpose I/O (“geek port”) + IEEE1284 I/O.

```
#device         vpo        # Requires scbus and da
```

This is for an Iomega Zip drive. It requires `scbus` and `da` support. Best performance is achieved with ports in EPP 1.9 mode.

```
#device         puc
```

Uncomment this device if the system has a “dumb” serial or parallel PCI card that is supported by the puc(4) glue driver.

```
# PCI Ethernet NICs.
device          de         # DEC/Intel DC21x4x (“Tulip”)
device          em         # Intel PRO/1000 adapter Gigabit Ethernet Card
device          ixgb       # Intel PRO/10GbE Ethernet Card
device          txp        # 3Com 3cR990 (“Typhoon”)
device          vx         # 3Com 3c590, 3c595 (“Vortex”)
```

Various PCI network card drivers. Comment out or remove any of these which are not present in the system.

```
# PCI Ethernet NICs that use the common MII bus controller code.
# NOTE: Be sure to keep the ’device miibus’ line in order to use these NICs!
device          miibus     # MII bus support
```

MII bus support is required for some PCI 10/100 Ethernet NICs, namely those which use MII-compliant transceivers or implement transceiver control interfaces that operate like an MII. Adding `device miibus` to the kernel config pulls in support for the generic miibus API and all of the PHY drivers, including a generic one for PHYs that are not specifically handled by an individual driver.

```
device          bce        # Broadcom BCM5706/BCM5708 Gigabit Ethernet
device          bfe        # Broadcom BCM440x 10/100 Ethernet
device          bge        # Broadcom BCM570xx Gigabit Ethernet
device          dc         # DEC/Intel 21143 and various workalikes
device          fxp        # Intel EtherExpress PRO/100B (82557, 82558)
```

```
device          lge        # Level 1 LXT1001 gigabit ethernet
device          msk        # Marvell/SysKonnect Yukon II Gigabit Ethernet
device          nge        # NatSemi DP83820 gigabit ethernet
device          nve        # nVidia nForce MCP on-board Ethernet Networking
device          pcn        # AMD Am79C97x PCI 10/100 (precedence over 'lnc')
device          re         # RealTek 8139C+/8169/8169S/8110S
device          rl         # RealTek 8129/8139
device          sf         # Adaptec AIC-6915 ("Starfire")
device          sis        # Silicon Integrated Systems SiS 900/SiS 7016
device          sk         # SysKonnect SK-984x & SK-982x gigabit Ethernet
device          ste        # Sundance ST201 (D-Link DFE-550TX)
device          stge       # Sundance/Tamarack TC9021 gigabit Ethernet
device          ti         # Alteon Networks Tigon I/II gigabit Ethernet
device          tl         # Texas Instruments ThunderLAN
device          tx         # SMC EtherPower II (83c170 "EPIC")
device          vge        # VIA VT612x gigabit ethernet
device          vr         # VIA Rhine, Rhine II
device          wb         # Winbond W89C840F
device          xl         # 3Com 3c90x ("Boomerang", "Cyclone")
```

Drivers that use the MII bus controller code.

```
# ISA Ethernet NICs.  pccard NICs included.
device          cs         # Crystal Semiconductor CS89x0 NIC
# 'device ed' requires 'device miibus'
device          ed         # NE[12]000, SMC Ultra, 3c503, DS8390 cards
device          ex         # Intel EtherExpress Pro/10 and Pro/10+
device          ep         # Etherlink III based cards
device          fe         # Fujitsu MB8696x based cards
device          ie         # EtherExpress 8/16, 3C507, StarLAN 10 etc.
device          lnc        # NE2100, NE32-VL Lance Ethernet cards
device          sn         # SMC's 9000 series of Ethernet chips
device          xe         # Xircom pccard Ethernet

# ISA devices that use the old ISA shims
#device         le
```

ISA Ethernet drivers. See `/usr/src/sys/`*`i386`*`/conf/NOTES` for details of which cards are supported by which driver.

```
# Wireless NIC cards
device          wlan            # 802.11 support
```

Generic 802.11 support. This line is required for wireless networking.

```
device          wlan_wep        # 802.11 WEP support
device          wlan_ccmp       # 802.11 CCMP support
device          wlan_tkip       # 802.11 TKIP support
```

Crypto support for 802.11 devices. These lines are needed on systems which use encryption and 802.11i security protocols.

```
device          an         # Aironet 4500/4800 802.11 wireless NICs.
```

```
device          ath             # Atheros pci/cardbus NIC's
device          ath_hal         # Atheros HAL (Hardware Access Layer)
device          ath_rate_sample # SampleRate tx rate control for ath
device          awi         # BayStack 660 and others
device          ral         # Ralink Technology RT2500 wireless NICs.
device          wi          # WaveLAN/Intersil/Symbol 802.11 wireless NICs.
#device         wl          # Older non 802.11 Wavelan wireless NIC.
```

Support for various wireless cards.

```
# Pseudo devices
device   loop           # Network loopback
```

This is the generic loopback device for TCP/IP. This is *mandatory*.

```
device   random         # Entropy device
```

Cryptographically secure random number generator.

```
device   ether          # Ethernet support
```

`ether` is only needed if the system has an Ethernet card. It includes generic Ethernet protocol code.

```
device   sl             # Kernel SLIP
```

`sl` provides SLIP support. This has been almost entirely supplanted by PPP, which is easier to set up, better suited for modem-to-modem connection, and more powerful.

```
device   ppp            # Kernel PPP
```

This is for kernel PPP support for dial-up connections. There is also a version of PPP implemented as a userland application that uses `tun` and offers more flexibility and features such as demand dialing.

```
device   tun            # Packet tunnel.
```

This is used by the userland PPP software. See the **PPP** section of the Handbook for more information.

```
device   pty            # Pseudo-ttys (telnet etc)
```

This is a "pseudo-terminal" or simulated login port. It is used by incoming `telnet` and `rlogin` sessions, **xterm**, and some other applications such as **Emacs**.

```
device   md             # Memory "disks"
```

Memory disk pseudo-devices.

```
device   gif            # IPv6 and IPv4 tunneling
```

This implements IPv6 over IPv4 tunneling, IPv4 over IPv6 tunneling, IPv4 over IPv4 tunneling, and IPv6 over IPv6 tunneling. The `gif` device is "auto-cloning", and will create device nodes as needed.

```
device   faith          # IPv6-to-IPv4 relaying (translation)
```

This pseudo-device captures packets that are sent to it and diverts them to the IPv4/IPv6 translation daemon.

```
# The ‘bpf’ device enables the Berkeley Packet Filter.
# Be aware of the administrative consequences of enabling this!
# Note that ’bpf’ is required for DHCP.
device   bpf           # Berkeley packet filter
```

The Berkeley Packet Filter pseudo-device allows network interfaces to be placed in promiscuous mode, capturing every packet on a broadcast network such as an Ethernet network. These packets can be captured to disk and or examined using tcpdump(1).

Note: The bpf(4) device is also used by dhclient(8). If DHCP is used, leave this uncommented.

```
# USB support
device          uhci          # UHCI PCI->USB interface
device          ohci          # OHCI PCI->USB interface
device          ehci          # EHCI PCI->USB interface (USB 2.0)
device          usb           # USB Bus (required)
#device         udbp          # USB Double Bulk Pipe devices
device          ugen          # Generic
device          uhid          # “Human Interface Devices”
device          ukbd          # Keyboard
device          ulpt          # Printer
device          umass         # Disks/Mass storage - Requires scbus and da
device          ums           # Mouse
device          ural          # Ralink Technology RT2500USB wireless NICs
device          urio          # Diamond Rio 500 MP3 player
device          uscanner      # Scanners
# USB Ethernet, requires mii
device          aue           # ADMtek USB Ethernet
device          axe           # ASIX Electronics USB Ethernet
device          cdce          # Generic USB over Ethernet
device          cue           # CATC USB Ethernet
device          kue           # Kawasaki LSI USB Ethernet
device          rue           # RealTek RTL8150 USB Ethernet
```

Support for various USB devices.

```
# FireWire support
device          firewire      # FireWire bus code
device          sbp           # SCSI over FireWire (Requires scbus and da)
device          fwe           # Ethernet over FireWire (non-standard!)
```

Support for various Firewire devices.

For more information and additional devices supported by FreeBSD, see `/usr/src/sys/`*`i386`*`/conf/NOTES`.

9.6.1 Large Memory Configurations (PAE)

Large memory configuration machines require access to more than the 4 gigabyte limit on User+Kernel Virtual Address (KVA) space. Due to this limitation, Intel added support for 36-bit physical address space access in the Pentium® Pro and later line of CPUs.

The Physical Address Extension (PAE) capability of the Intel Pentium Pro and later CPUs allows memory configurations of up to 64 gigabytes. FreeBSD provides support for this capability via the `PAE` kernel configuration option, available in all current release versions of FreeBSD. Due to the limitations of the Intel memory architecture, no distinction is made for memory above or below 4 gigabytes. Memory allocated above 4 gigabytes is simply added to the pool of available memory.

To enable PAE support in the kernel, add the following line to the kernel configuration file:

```
options             PAE
```

> **Note:** The PAE support in FreeBSD is only available for Intel IA-32 processors. It should also be noted that the PAE support in FreeBSD has not received wide testing, and should be considered beta quality compared to other stable features of FreeBSD.

PAE support in FreeBSD has a few limitations:

- A process is not able to access more than 4 gigabytes of virtual memory space.
- Device drivers that do not use the bus_dma(9) interface will cause data corruption in a PAE enabled kernel and are not recommended for use. For this reason, a `PAE` kernel configuration file is provided in FreeBSD which excludes all drivers not known to work in a PAE enabled kernel.
- Some system tunables determine memory resource usage by the amount of available physical memory. Such tunables can unnecessarily over-allocate due to the large memory nature of a PAE system. One such example is the `kern.maxvnodes` sysctl, which controls the maximum number of vnodes allowed in the kernel. It is advised to adjust this and other such tunables to a reasonable value.
- It might be necessary to increase the kernel virtual address (KVA) space or to reduce the amount of specific kernel resource that is heavily used in order to avoid KVA exhaustion. The `KVA_PAGES` kernel option can be used for increasing the KVA space.

For performance and stability concerns, it is advised to consult tuning(7). pae(4) contains up-to-date information on FreeBSD's PAE support.

9.7 If Something Goes Wrong

There are four categories of trouble that can occur when building a custom kernel. They are:

`config` fails:

If config(8) fails, it is probably a simple error. Fortunately, config(8) will print the line number that it had trouble with. For example, for this message:

```
config: line 17: syntax error
```

Make sure the keyword on line 17 is typed correctly by comparing it to the `GENERIC` kernel or another reference.

`make` fails:

If `make` fails, it usually signals an error in the kernel description which is not severe enough for config(8) to catch. Review the configuration, and if you still cannot resolve the problem, send an email to the FreeBSD general questions mailing list (http://lists.FreeBSD.org/mailman/listinfo/freebsd-questions) with the kernel configuration.

The kernel does not boot:

If the new kernel does not boot, or fails to recognize devices, do not panic! Fortunately, FreeBSD has an excellent mechanism for recovering from incompatible kernels. Simply choose the kernel to boot from at the FreeBSD boot loader. This can be accessed when the system boot menu appears by selecting the "Escape to a loader prompt" option. At the prompt, type `boot` *`kernel.old`*, or the name of any other kernel that will boot properly. When reconfiguring a kernel, it is always a good idea to keep a kernel that is known to work on hand.

After booting with a good kernel, check over the configuration file and try to build it again. One helpful resource is `/var/log/messages` which records the kernel messages from every successful boot. Also, dmesg(8) will print the kernel messages from the current boot.

> **Note:** When troubleshooting a kernel, make sure to keep `GENERIC`, or some other kernel that is known to work, on hand as a different name that will not get erased on the next build. Do not rely on `kernel.old` because when installing a new kernel, `kernel.old` is overwritten with the last installed kernel which may be non-functional. As soon as possible, move the working kernel to the proper `/boot/kernel` location or commands such as ps(1) may not work properly. To do this, simply rename the directory containing the good kernel:
>
> ```
> # mv /boot/kernel /boot/kernel.bad
> # mv /boot/kernel.good /boot/kernel
> ```

The kernel works, but ps(1) does not work any more:

If the kernel version differs from the one that the system utilities have been built with, for example, a -CURRENT kernel on a -RELEASE, many system status commands like ps(1) and vmstat(8) will not work. To fix this, recompile and install a world built with the same version of the source tree as the kernel. This is one reason why it is not a good idea to use a different version of the kernel than the rest of the operating system.

Chapter 10 Printing

Contributed by Sean Kelly. Restructured and updated by Jim Mock.

10.1 Synopsis

FreeBSD can be used to print with a wide variety of printers, from the oldest impact printer to the latest laser printers, and everything in between, allowing you to produce high-quality printed output from the applications you run.

FreeBSD can also be configured to act as a print server on a network; in this capacity FreeBSD can receive print jobs from a variety of other computers, including other FreeBSD computers, Windows and Mac OS hosts. FreeBSD will ensure that one job at a time is printed, and can keep statistics on which users and machines are doing the most printing, produce “banner” pages showing whose printout is whose, and more.

After reading this chapter, you will know:

- How to configure the FreeBSD print spooler.
- How to install print filters, to handle special print jobs differently, including converting incoming documents to print formats that your printers understand.
- How to enable header, or banner pages on your printout.
- How to print with printers connected to other computers.
- How to print with printers connected directly to the network.
- How to control printer restrictions, including limiting the size of print jobs, and preventing certain users from printing.
- How to keep printer statistics, and account for printer usage.
- How to troubleshoot printing problems.

Before reading this chapter, you should:

- Know how to configure and install a new kernel (Chapter 9).

10.2 Introduction

In order to use printers with FreeBSD you may set them up to work with the Berkeley line printer spooling system, also known as the **LPD** spooling system, or just **LPD**. It is the standard printer control system in FreeBSD. This chapter introduces **LPD** and will guide you through its configuration.

If you are already familiar with **LPD** or another printer spooling system, you may wish to skip to section Basic Setup.

LPD controls everything about a host’s printers. It is responsible for a number of things:

- It controls access to attached printers and printers attached to other hosts on the network.
- It enables users to submit files to be printed; these submissions are known as *jobs*.

- It prevents multiple users from accessing a printer at the same time by maintaining a *queue* for each printer.
- It can print *header pages* (also known as *banner* or *burst* pages) so users can easily find jobs they have printed in a stack of printouts.
- It takes care of communications parameters for printers connected on serial ports.
- It can send jobs over the network to a **LPD** spooler on another host.
- It can run special filters to format jobs to be printed for various printer languages or printer capabilities.
- It can account for printer usage.

Through a configuration file (`/etc/printcap`), and by providing the special filter programs, you can enable the **LPD** system to do all or some subset of the above for a great variety of printer hardware.

10.2.1 Why You Should Use the Spooler

The spooler still provides benefit on a single-user system and should be used because:

- **LPD** prints jobs in the background; you do not have to wait for data to be copied to the printer.
- **LPD** can conveniently run a job to be printed through filters to add date/time headers or convert a special file format (such as a TEX DVI file) into a format the printer will understand. You will not have to do these steps manually.
- Many free and commercial programs that provide a print feature usually expect to talk to the spooler on your system. By setting up the spooling system, you will more easily support other software you may later add or already have.

10.3 Basic Setup

To use printers with the **LPD** spooling system, you will need to set up both your printer hardware and the **LPD** software. This document describes two levels of setup:

- See section Simple Printer Setup to learn how to connect a printer, tell **LPD** how to communicate with it, and print plain text files to the printer.
- See section Advanced Printer Setup to learn how to print a variety of special file formats, to print header pages, to print across a network, to control access to printers, and to do printer accounting.

10.3.1 Simple Printer Setup

This section tells how to configure printer hardware and the **LPD** software to use the printer. It teaches the basics:

- Section Hardware Setup gives some hints on connecting the printer to a port on your computer.
- Section Software Setup shows how to set up the **LPD** spooler configuration file (`/etc/printcap`).

If you are setting up a printer that uses a network protocol to accept data to print instead of a computer's local interfaces, see Printers With Networked Data Stream Interfaces.

Although this section is called “Simple Printer Setup”, it is actually fairly complex. Getting the printer to work with your computer and the **LPD** spooler is the hardest part. The advanced options like header pages and accounting are fairly easy once you get the printer working.

10.3.1.1 Hardware Setup

This section tells about the various ways you can connect a printer to your PC. It talks about the kinds of ports and cables, and also the kernel configuration you may need to enable FreeBSD to speak to the printer.

If you have already connected your printer and have successfully printed with it under another operating system, you can probably skip to section Software Setup.

10.3.1.1.1 Ports and Cables

Printers sold for use on PC’s today generally come with one or more of the following three interfaces:

- *Serial* interfaces, also known as RS-232 or COM ports, use a serial port on your computer to send data to the printer. Serial interfaces are common in the computer industry and cables are readily available and also easy to construct. Serial interfaces sometimes need special cables and might require you to configure somewhat complex communications options. Most PC serial ports have a maximum transmission rate of 115200 bps, which makes printing large graphic print jobs with them impractical.
- *Parallel* interfaces use a parallel port on your computer to send data to the printer. Parallel interfaces are common in the PC market and are faster than RS-232 serial. Cables are readily available but more difficult to construct by hand. There are usually no communications options with parallel interfaces, making their configuration exceedingly simple.

  Parallel interfaces are sometimes known as “Centronics” interfaces, named after the connector type on the printer.
- USB interfaces, named for the Universal Serial Bus, can run at even faster speeds than parallel or RS-232 serial interfaces. Cables are simple and cheap. USB is superior to RS-232 Serial and to Parallel for printing, but it is not as well supported under UNIX systems. A way to avoid this problem is to purchase a printer that has both a USB interface and a Parallel interface, as many printers do.

In general, Parallel interfaces usually offer just one-way communication (computer to printer) while serial and USB gives you two-way. Newer parallel ports (EPP and ECP) and printers can communicate in both directions under FreeBSD when a IEEE-1284-compliant cable is used.

Two-way communication to the printer over a parallel port is generally done in one of two ways. The first method uses a custom-built printer driver for FreeBSD that speaks the proprietary language used by the printer. This is common with inkjet printers and can be used for reporting ink levels and other status information. The second method is used when the printer supports PostScript.

PostScript jobs are actually programs sent to the printer; they need not produce paper at all and may return results directly to the computer. PostScript also uses two-way communication to tell the computer about problems, such as errors in the PostScript program or paper jams. Your users may be appreciative of such information. Furthermore, the best way to do effective accounting with a PostScript printer requires two-way communication: you ask the printer

for its page count (how many pages it has printed in its lifetime), then send the user's job, then ask again for its page count. Subtract the two values and you know how much paper to charge to the user.

10.3.1.1.2 Parallel Ports

To hook up a printer using a parallel interface, connect the Centronics cable between the printer and the computer. The instructions that came with the printer, the computer, or both should give you complete guidance.

Remember which parallel port you used on the computer. The first parallel port is `ppc0` to FreeBSD; the second is `ppc1`, and so on. The printer device name uses the same scheme: `/dev/lpt0` for the printer on the first parallel ports etc.

10.3.1.1.3 Serial Ports

To hook up a printer using a serial interface, connect the proper serial cable between the printer and the computer. The instructions that came with the printer, the computer, or both should give you complete guidance.

If you are unsure what the "proper serial cable" is, you may wish to try one of the following alternatives:

- A *modem* cable connects each pin of the connector on one end of the cable straight through to its corresponding pin of the connector on the other end. This type of cable is also known as a "DTE-to-DCE" cable.
- A *null-modem* cable connects some pins straight through, swaps others (send data to receive data, for example), and shorts some internally in each connector hood. This type of cable is also known as a "DTE-to-DTE" cable.
- A *serial printer* cable, required for some unusual printers, is like the null-modem cable, but sends some signals to their counterparts instead of being internally shorted.

You should also set up the communications parameters for the printer, usually through front-panel controls or DIP switches on the printer. Choose the highest `bps` (bits per second, sometimes *baud rate*) that both your computer and the printer can support. Choose 7 or 8 data bits; none, even, or odd parity; and 1 or 2 stop bits. Also choose a flow control protocol: either none, or XON/XOFF (also known as "in-band" or "software") flow control. Remember these settings for the software configuration that follows.

10.3.1.2 Software Setup

This section describes the software setup necessary to print with the **LPD** spooling system in FreeBSD.

Here is an outline of the steps involved:

1. Configure your kernel, if necessary, for the port you are using for the printer; section Kernel Configuration tells you what you need to do.
2. Set the communications mode for the parallel port, if you are using a parallel port; section Setting the Communication Mode for the Parallel Port gives details.
3. Test if the operating system can send data to the printer. Section Checking Printer Communications gives some suggestions on how to do this.

4. Set up **LPD** for the printer by modifying the file `/etc/printcap`. You will find out how to do this later in this chapter.

10.3.1.2.1 Kernel Configuration

The operating system kernel is compiled to work with a specific set of devices. The serial or parallel interface for your printer is a part of that set. Therefore, it might be necessary to add support for an additional serial or parallel port if your kernel is not already configured for one.

To find out if the kernel you are currently using supports a serial interface, type:

```
# grep sioN /var/run/dmesg.boot
```

Where *N* is the number of the serial port, starting from zero. If you see output similar to the following:

```
sio2 at port 0x3e8-0x3ef irq 5 on isa
sio2: type 16550A
```

then the kernel supports the port.

To find out if the kernel supports a parallel interface, type:

```
# grep ppcN /var/run/dmesg.boot
```

Where *N* is the number of the parallel port, starting from zero. If you see output similar to the following:

```
ppc0: <Parallel port> at port 0x378-0x37f irq 7 on isa0
ppc0: SMC-like chipset (ECP/EPP/PS2/NIBBLE) in COMPATIBLE mode
ppc0: FIFO with 16/16/8 bytes threshold
```

then the kernel supports the port.

You might have to reconfigure your kernel in order for the operating system to recognize and use the parallel or serial port you are using for the printer.

To add support for a serial port, see the section on kernel configuration. To add support for a parallel port, see that section *and* the section that follows.

10.3.1.3 Setting the Communication Mode for the Parallel Port

When you are using the parallel interface, you can choose whether FreeBSD should use interrupt-driven or polled communication with the printer. The generic printer device driver (lpt(4)) on FreeBSD uses the ppbus(4) system, which controls the port chipset with the ppc(4) driver.

- The *interrupt-driven* method is the default with the GENERIC kernel. With this method, the operating system uses an IRQ line to determine when the printer is ready for data.
- The *polled* method directs the operating system to repeatedly ask the printer if it is ready for more data. When it responds ready, the kernel sends more data.

The interrupt-driven method is usually somewhat faster but uses up a precious IRQ line. Some newer HP printers are claimed not to work correctly in interrupt mode, apparently due to some (not yet exactly understood) timing

problem. These printers need polled mode. You should use whichever one works. Some printers will work in both modes, but are painfully slow in interrupt mode.

You can set the communications mode in two ways: by configuring the kernel or by using the lptcontrol(8) program.

To set the communications mode by configuring the kernel:

1. Edit your kernel configuration file. Look for an `ppc0` entry. If you are setting up the second parallel port, use `ppc1` instead. Use `ppc2` for the third port, and so on.
   - If you want interrupt-driven mode, edit the following line:

     `hint.ppc.0.irq="N"`

     in the `/boot/device.hints` file and replace *N* with the right IRQ number. The kernel configuration file must also contain the ppc(4) driver:

     `device ppc`
   - If you want polled mode, remove in your `/boot/device.hints` file, the following line:

     `hint.ppc.0.irq="N"`

     In some cases, this is not enough to put the port in polled mode under FreeBSD. Most of time it comes from acpi(4) driver, this latter is able to probe and attach devices, and therefore, control the access mode to the printer port. You should check your acpi(4) configuration to correct this problem.
2. Save the file. Then configure, build, and install the kernel, then reboot. See kernel configuration for more details.

To set the communications mode with lptcontrol(8):

1. Type:

   `# lptcontrol -i -d /dev/lptN`

   to set interrupt-driven mode for `lptN`.
2. Type:

   `# lptcontrol -p -d /dev/lptN`

   to set polled-mode for `lptN`.

You could put these commands in your `/etc/rc.local` file to set the mode each time your system boots. See lptcontrol(8) for more information.

10.3.1.4 Checking Printer Communications

Before proceeding to configure the spooling system, you should make sure the operating system can successfully send data to your printer. It is a lot easier to debug printer communication and the spooling system separately.

To test the printer, we will send some text to it. For printers that can immediately print characters sent to them, the program lptest(1) is perfect: it generates all 96 printable ASCII characters in 96 lines.

For a PostScript (or other language-based) printer, we will need a more sophisticated test. A small PostScript program, such as the following, will suffice:

```
%!PS
100 100 moveto 300 300 lineto stroke
```

```
310 310 moveto /Helvetica findfont 12 scalefont setfont
(Is this thing working?) show
showpage
```

The above PostScript code can be placed into a file and used as shown in the examples appearing in the following sections.

> **Note:** When this document refers to a printer language, it is assuming a language like PostScript, and not Hewlett Packard's PCL. Although PCL has great functionality, you can intermingle plain text with its escape sequences. PostScript cannot directly print plain text, and that is the kind of printer language for which we must make special accommodations.

10.3.1.4.1 Checking a Parallel Printer

This section tells you how to check if FreeBSD can communicate with a printer connected to a parallel port.

To test a printer on a parallel port:

1. Become `root` with su(1).
2. Send data to the printer.
   - If the printer can print plain text, then use lptest(1). Type:

     ```
     # lptest > /dev/lptN
     ```

     Where *`N`* is the number of the parallel port, starting from zero.
   - If the printer understands PostScript or other printer language, then send a small program to the printer. Type:

     ```
     # cat > /dev/lptN
     ```

     Then, line by line, type the program *carefully* as you cannot edit a line once you have pressed `RETURN` or `ENTER`. When you have finished entering the program, press `CONTROL+D`, or whatever your end of file key is.

     Alternatively, you can put the program in a file and type:

     ```
     # cat file > /dev/lptN
     ```

     Where *`file`* is the name of the file containing the program you want to send to the printer.

You should see something print. Do not worry if the text does not look right; we will fix such things later.

10.3.1.4.2 Checking a Serial Printer

This section tells you how to check if FreeBSD can communicate with a printer on a serial port.

To test a printer on a serial port:

1. Become `root` with su(1).
2. Edit the file `/etc/remote`. Add the following entry:

   `printer:dv=/dev/`*`port`*`:br#`*`bps-rate`*`:pa=`*`parity`*

Where *port* is the device entry for the serial port (ttyu0, ttyu1, etc.), *bps-rate* is the bits-per-second rate at which the printer communicates, and *parity* is the parity required by the printer (either even, odd, none, or zero).

Here is a sample entry for a printer connected via a serial line to the third serial port at 19200 bps with no parity:

```
printer:dv=/dev/ttyu2:br#19200:pa=none
```

3. Connect to the printer with tip(1). Type:

```
# tip printer
```

If this step does not work, edit the file /etc/remote again and try using /dev/cuaa*N* instead of /dev/ttyu*N*.

4. Send data to the printer.

- If the printer can print plain text, then use lptest(1). Type:

```
% $lptest
```

- If the printer understands PostScript or other printer language, then send a small program to the printer. Type the program, line by line, *very carefully* as backspacing or other editing keys may be significant to the printer. You may also need to type a special end-of-file key for the printer so it knows it received the whole program. For PostScript printers, press CONTROL+D.

  Alternatively, you can put the program in a file and type:

```
% >file
```

  Where *file* is the name of the file containing the program. After tip(1) sends the file, press any required end-of-file key.

You should see something print. Do not worry if the text does not look right; we will fix that later.

10.3.1.5 Enabling the Spooler: the /etc/printcap File

At this point, your printer should be hooked up, your kernel configured to communicate with it (if necessary), and you have been able to send some simple data to the printer. Now, we are ready to configure **LPD** to control access to your printer.

You configure **LPD** by editing the file /etc/printcap. The **LPD** spooling system reads this file each time the spooler is used, so updates to the file take immediate effect.

The format of the printcap(5) file is straightforward. Use your favorite text editor to make changes to /etc/printcap. The format is identical to other capability files like /usr/share/misc/termcap and /etc/remote. For complete information about the format, see the cgetent(3).

The simple spooler configuration consists of the following steps:

1. Pick a name (and a few convenient aliases) for the printer, and put them in the /etc/printcap file; see the Naming the Printer section for more information on naming.
2.

   Turn off header pages (which are on by default) by inserting the sh capability; see the Suppressing Header Pages section for more information.

3. Make a spooling directory, and specify its location with the `sd` capability; see the Making the Spooling Directory section for more information.
4. Set the `/dev` entry to use for the printer, and note it in `/etc/printcap` with the `lp` capability; see the Identifying the Printer Device for more information. Also, if the printer is on a serial port, set up the communication parameters with the `ms#` capability which is discussed in the Configuring Spooler Communications Parameters section.
5. Install a plain text input filter; see the Installing the Text Filter section for details.
6. Test the setup by printing something with the lpr(1) command. More details are available in the Trying It Out and Troubleshooting sections.

> **Note:** Language-based printers, such as PostScript printers, cannot directly print plain text. The simple setup outlined above and described in the following sections assumes that if you are installing such a printer you will print only files that the printer can understand.

Users often expect that they can print plain text to any of the printers installed on your system. Programs that interface to **LPD** to do their printing usually make the same assumption. If you are installing such a printer and want to be able to print jobs in the printer language *and* print plain text jobs, you are strongly urged to add an additional step to the simple setup outlined above: install an automatic plain-text-to-PostScript (or other printer language) conversion program. The section entitled Accommodating Plain Text Jobs on PostScript Printers tells how to do this.

10.3.1.5.1 Naming the Printer

The first (easy) step is to pick a name for your printer. It really does not matter whether you choose functional or whimsical names since you can also provide a number of aliases for the printer.

At least one of the printers specified in the `/etc/printcap` should have the alias `lp`. This is the default printer's name. If users do not have the `PRINTER` environment variable nor specify a printer name on the command line of any of the **LPD** commands, then `lp` will be the default printer they get to use.

Also, it is common practice to make the last alias for a printer be a full description of the printer, including make and model.

Once you have picked a name and some common aliases, put them in the `/etc/printcap` file. The name of the printer should start in the leftmost column. Separate each alias with a vertical bar and put a colon after the last alias.

In the following example, we start with a skeletal `/etc/printcap` that defines two printers (a Diablo 630 line printer and a Panasonic KX-P4455 PostScript laser printer):

```
#
#  /etc/printcap for host rose
#
rattan|line|diablo|lp|Diablo 630 Line Printer:

bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:
```

In this example, the first printer is named `rattan` and has as aliases `line`, `diablo`, `lp`, and `Diablo 630 Line Printer`. Since it has the alias `lp`, it is also the default printer. The second is named `bamboo`, and has as aliases `ps`, `PS`, `S`, `panasonic`, and `Panasonic KX-P4455 PostScript v51.4`.

10.3.1.5.2 Suppressing Header Pages

The **LPD** spooling system will by default print a *header page* for each job. The header page contains the user name who requested the job, the host from which the job came, and the name of the job, in nice large letters. Unfortunately, all this extra text gets in the way of debugging the simple printer setup, so we will suppress header pages.

To suppress header pages, add the `sh` capability to the entry for the printer in `/etc/printcap`. Here is an example `/etc/printcap` with `sh` added:

```
#
#  /etc/printcap for host rose - no header pages anywhere
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
        :sh:

bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
        :sh:
```

Note how we used the correct format: the first line starts in the leftmost column, and subsequent lines are indented. Every line in an entry except the last ends in a backslash character.

10.3.1.5.3 Making the Spooling Directory

The next step in the simple spooler setup is to make a *spooling directory*, a directory where print jobs reside until they are printed, and where a number of other spooler support files live.

Because of the variable nature of spooling directories, it is customary to put these directories under `/var/spool`. It is not necessary to backup the contents of spooling directories, either. Recreating them is as simple as running mkdir(1).

It is also customary to make the directory with a name that is identical to the name of the printer, as shown below:

```
# mkdir /var/spool/printer-name
```

However, if you have a lot of printers on your network, you might want to put the spooling directories under a single directory that you reserve just for printing with **LPD**. We will do this for our two example printers `rattan` and `bamboo`:

```
# mkdir /var/spool/lpd
# mkdir /var/spool/lpd/rattan
# mkdir /var/spool/lpd/bamboo
```

> **Note:** If you are concerned about the privacy of jobs that users print, you might want to protect the spooling directory so it is not publicly accessible. Spooling directories should be owned and be readable, writable, and searchable by user `daemon` and group `daemon`, and no one else. We will do this for our example printers:
>
> ```
> # chown daemon:daemon /var/spool/lpd/rattan
> # chown daemon:daemon /var/spool/lpd/bamboo
> # chmod 770 /var/spool/lpd/rattan
> # chmod 770 /var/spool/lpd/bamboo
> ```

Finally, you need to tell **LPD** about these directories using the `/etc/printcap` file. You specify the pathname of the spooling directory with the `sd` capability:

```
#
#  /etc/printcap for host rose - added spooling directories
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
        :sh:sd=/var/spool/lpd/rattan:

bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
        :sh:sd=/var/spool/lpd/bamboo:
```

Note that the name of the printer starts in the first column but all other entries describing the printer should be indented and each line end escaped with a backslash.

If you do not specify a spooling directory with `sd`, the spooling system will use `/var/spool/lpd` as a default.

10.3.1.5.4 Identifying the Printer Device

In the Hardware Setup section, we identified the port and the relevant `/dev` directory entry that FreeBSD will use to communicate with the printer. Now, we tell **LPD** that information. When the spooling system has a job to print, it will open the specified device on behalf of the filter program (which is responsible for passing data to the printer).

List the `/dev` entry pathname in the `/etc/printcap` file using the `lp` capability.

In our running example, let us assume that `rattan` is on the first parallel port, and `bamboo` is on a sixth serial port; here are the additions to `/etc/printcap`:

```
#
#  /etc/printcap for host rose - identified what devices to use
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
        :sh:sd=/var/spool/lpd/rattan:\
        :lp=/dev/lpt0:

bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
        :sh:sd=/var/spool/lpd/bamboo:\
        :lp=/dev/ttyu5:
```

If you do not specify the `lp` capability for a printer in your `/etc/printcap` file, **LPD** uses `/dev/lp` as a default. `/dev/lp` currently does not exist in FreeBSD.

If the printer you are installing is connected to a parallel port, skip to the section entitled, Installing the Text Filter. Otherwise, be sure to follow the instructions in the next section.

10.3.1.5.5 Configuring Spooler Communication Parameters

For printers on serial ports, **LPD** can set up the bps rate, parity, and other serial communication parameters on behalf of the filter program that sends data to the printer. This is advantageous since:

- It lets you try different communication parameters by simply editing the `/etc/printcap` file; you do not have to recompile the filter program.

- It enables the spooling system to use the same filter program for multiple printers which may have different serial communication settings.

The following `/etc/printcap` capabilities control serial communication parameters of the device listed in the `lp` capability:

`br#`*`bps-rate`*

Sets the communications speed of the device to *`bps-rate`*, where *`bps-rate`* can be 50, 75, 110, 134, 150, 200, 300, 600, 1200, 1800, 2400, 4800, 9600, 19200, 38400, 57600, or 115200 bits-per-second.

`ms#`*`stty-mode`*

Sets the options for the terminal device after opening the device. stty(1) explains the available options.

When **LPD** opens the device specified by the `lp` capability, it sets the characteristics of the device to those specified with the `ms#` capability. Of particular interest will be the `parenb`, `parodd`, `cs5`, `cs6`, `cs7`, `cs8`, `cstopb`, `crtscts`, and `ixon` modes, which are explained in the stty(1) manual page.

Let us add to our example printer on the sixth serial port. We will set the bps rate to 38400. For the mode, we will set no parity with `-parenb`, 8-bit characters with `cs8`, no modem control with `clocal` and hardware flow control with `crtscts`:

```
bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
        :sh:sd=/var/spool/lpd/bamboo:\
        :lp=/dev/ttyu5:ms#-parenb cs8 clocal crtscts:
```

10.3.1.5.6 Installing the Text Filter

We are now ready to tell **LPD** what text filter to use to send jobs to the printer. A *text filter*, also known as an *input filter*, is a program that **LPD** runs when it has a job to print. When **LPD** runs the text filter for a printer, it sets the filter's standard input to the job to print, and its standard output to the printer device specified with the `lp` capability. The filter is expected to read the job from standard input, perform any necessary translation for the printer, and write the results to standard output, which will get printed. For more information on the text filter, see the Filters section.

For our simple printer setup, the text filter can be a small shell script that just executes `/bin/cat` to send the job to the printer. FreeBSD comes with another filter called `lpf` that handles backspacing and underlining for printers that might not deal with such character streams well. And, of course, you can use any other filter program you want. The filter `lpf` is described in detail in section entitled lpf: a Text Filter.

First, let us make the shell script `/usr/local/libexec/if-simple` be a simple text filter. Put the following text into that file with your favorite text editor:

```
#!/bin/sh
#
# if-simple - Simple text input filter for lpd
# Installed in /usr/local/libexec/if-simple
#
# Simply copies stdin to stdout.  Ignores all filter arguments.

/bin/cat && exit 0
exit 2
```

Make the file executable:

```
# chmod 555 /usr/local/libexec/if-simple
```

And then tell LPD to use it by specifying it with the `if` capability in `/etc/printcap`. We will add it to the two printers we have so far in the example `/etc/printcap`:

```
#
#  /etc/printcap for host rose - added text filter
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
        :sh:sd=/var/spool/lpd/rattan:\
        :lp=/dev/lpt0:\
        :if=/usr/local/libexec/if-simple:

bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
        :sh:sd=/var/spool/lpd/bamboo:\
        :lp=/dev/ttyu5:ms#-parenb cs8 clocal crtscts:\
        :if=/usr/local/libexec/if-simple:
```

Note: A copy of the `if-simple` script can be found in the `/usr/share/examples/printing` directory.

*10.3.1.5.7 Turn on **LPD***

lpd(8) is run from `/etc/rc`, controlled by the `lpd_enable` variable. This variable defaults to `NO`. If you have not done so already, add the line:

```
lpd_enable="YES"
```

to `/etc/rc.conf`, and then either restart your machine, or just run lpd(8).

```
# lpd
```

10.3.1.5.8 Trying It Out

You have reached the end of the simple **LPD** setup. Unfortunately, congratulations are not quite yet in order, since we still have to test the setup and correct any problems. To test the setup, try printing something. To print with the **LPD** system, you use the command lpr(1), which submits a job for printing.

You can combine lpr(1) with the lptest(1) program, introduced in section Checking Printer Communications to generate some test text.

*To test the simple **LPD** setup:*

Type:

```
# lptest 20 5 | lpr -Pprinter-name
```

Where *`printer-name`* is a the name of a printer (or an alias) specified in `/etc/printcap`. To test the default printer, type lpr(1) without any `-P` argument. Again, if you are testing a printer that expects PostScript, send a

PostScript program in that language instead of using lptest(1). You can do so by putting the program in a file and typing `lpr` *`file`*.

For a PostScript printer, you should get the results of the program. If you are using lptest(1), then your results should look like the following:

```
!"#$%&'()*+,-./01234
"#$%&'()*+,-./012345
#$%&'()*+,-./0123456
$%&'()*+,-./01234567
%&'()*+,-./012345678
```

To further test the printer, try downloading larger programs (for language-based printers) or running lptest(1) with different arguments. For example, `lptest 80 60` will produce 60 lines of 80 characters each.

If the printer did not work, see the Troubleshooting section.

10.4 Advanced Printer Setup

This section describes filters for printing specially formatted files, header pages, printing across networks, and restricting and accounting for printer usage.

10.4.1 Filters

Although **LPD** handles network protocols, queuing, access control, and other aspects of printing, most of the *real* work happens in the *filters*. Filters are programs that communicate with the printer and handle its device dependencies and special requirements. In the simple printer setup, we installed a plain text filter—an extremely simple one that should work with most printers (section Installing the Text Filter).

However, in order to take advantage of format conversion, printer accounting, specific printer quirks, and so on, you should understand how filters work. It will ultimately be the filter's responsibility to handle these aspects. And the bad news is that most of the time *you* have to provide filters yourself. The good news is that many are generally available; when they are not, they are usually easy to write.

Also, FreeBSD comes with one, `/usr/libexec/lpr/lpf`, that works with many printers that can print plain text. (It handles backspacing and tabs in the file, and does accounting, but that is about all it does.) There are also several filters and filter components in the FreeBSD Ports Collection.

Here is what you will find in this section:

- Section How Filters Work, tries to give an overview of a filter's role in the printing process. You should read this section to get an understanding of what is happening "under the hood" when **LPD** uses filters. This knowledge could help you anticipate and debug problems you might encounter as you install more and more filters for each of your printers.
- **LPD** expects every printer to be able to print plain text by default. This presents a problem for PostScript printers (or other language-based printers) which cannot directly print plain text. Section Accommodating Plain Text Jobs on PostScript Printers tells you what you should do to overcome this problem. You should read this section if you have a PostScript printer.

- PostScript is a popular output format for many programs. Some people even write PostScript code directly. Unfortunately, PostScript printers are expensive. Section Simulating PostScript on Non PostScript Printers tells how you can further modify a printer's text filter to accept and print PostScript data on a *non PostScript* printer. You should read this section if you do not have a PostScript printer.
- Section Conversion Filters tells about a way you can automate the conversion of specific file formats, such as graphic or typesetting data, into formats your printer can understand. After reading this section, you should be able to set up your printers such that users can type `lpr -t` to print troff data, or `lpr -d` to print TEX DVI data, or `lpr -v` to print raster image data, and so forth. The reading of this section is recommended.
- Section Output Filters tells all about a not often used feature of **LPD**: output filters. Unless you are printing header pages (see Header Pages), you can probably skip that section altogether.
- Section lpf: a Text Filter describes `lpf`, a fairly complete if simple text filter for line printers (and laser printers that act like line printers) that comes with FreeBSD. If you need a quick way to get printer accounting working for plain text, or if you have a printer which emits smoke when it sees backspace characters, you should definitely consider `lpf`.

> **Note:** A copy of the various scripts described below can be found in the `/usr/share/examples/printing` directory.

10.4.1.1 How Filters Work

As mentioned before, a filter is an executable program started by **LPD** to handle the device-dependent part of communicating with the printer.

When **LPD** wants to print a file in a job, it starts a filter program. It sets the filter's standard input to the file to print, its standard output to the printer, and its standard error to the error logging file (specified in the `lf` capability in `/etc/printcap`, or `/dev/console` by default).

Which filter **LPD** starts and the filter's arguments depend on what is listed in the `/etc/printcap` file and what arguments the user specified for the job on the lpr(1) command line. For example, if the user typed `lpr -t`, **LPD** would start the troff filter, listed in the `tf` capability for the destination printer. If the user wanted to print plain text, it would start the `if` filter (this is mostly true: see Output Filters for details).

There are three kinds of filters you can specify in `/etc/printcap`:

- The *text filter*, confusingly called the *input filter* in **LPD** documentation, handles regular text printing. Think of it as the default filter. **LPD** expects every printer to be able to print plain text by default, and it is the text filter's job to make sure backspaces, tabs, or other special characters do not confuse the printer. If you are in an environment where you have to account for printer usage, the text filter must also account for pages printed, usually by counting the number of lines printed and comparing that to the number of lines per page the printer supports. The text filter is started with the following argument list:

  `filter-name` [-c] -w *`width`* -l *`length`* -i *`indent`* -n *`login`* -h *`host acct-file`*

  where

  `-c`

  appears if the job is submitted with `lpr -l`

width

is the value from the pw (page width) capability specified in /etc/printcap, default 132

length

is the value from the pl (page length) capability, default 66

indent

is the amount of the indentation from lpr -i, default 0

login

is the account name of the user printing the file

host

is the host name from which the job was submitted

acct-file

is the name of the accounting file from the af capability.

- A *conversion filter* converts a specific file format into one the printer can render onto paper. For example, ditroff typesetting data cannot be directly printed, but you can install a conversion filter for ditroff files to convert the ditroff data into a form the printer can digest and print. Section Conversion Filters tells all about them. Conversion filters also need to do accounting, if you need printer accounting. Conversion filters are started with the following arguments:

  filter-name -x *pixel-width* -y *pixel-height* -n *login* -h *host acct-file*

  where *pixel-width* is the value from the px capability (default 0) and *pixel-height* is the value from the py capability (default 0).

- The *output filter* is used only if there is no text filter, or if header pages are enabled. In our experience, output filters are rarely used. Section Output Filters describes them. There are only two arguments to an output filter:

  filter-name -w *width* -l *length*

  which are identical to the text filters -w and -l arguments.

Filters should also *exit* with the following exit status:

exit 0

If the filter printed the file successfully.

exit 1

If the filter failed to print the file but wants **LPD** to try to print the file again. **LPD** will restart a filter if it exits with this status.

exit 2

If the filter failed to print the file and does not want **LPD** to try again. **LPD** will throw out the file.

The text filter that comes with the FreeBSD release, `/usr/libexec/lpr/lpf`, takes advantage of the page width and length arguments to determine when to send a form feed and how to account for printer usage. It uses the login, host, and accounting file arguments to make the accounting entries.

If you are shopping for filters, see if they are LPD-compatible. If they are, they must support the argument lists described above. If you plan on writing filters for general use, then have them support the same argument lists and exit codes.

10.4.1.2 Accommodating Plain Text Jobs on PostScript® Printers

If you are the only user of your computer and PostScript (or other language-based) printer, and you promise to never send plain text to your printer and to never use features of various programs that will want to send plain text to your printer, then you do not need to worry about this section at all.

But, if you would like to send both PostScript and plain text jobs to the printer, then you are urged to augment your printer setup. To do so, we have the text filter detect if the arriving job is plain text or PostScript. All PostScript jobs must start with `%!` (for other printer languages, see your printer documentation). If those are the first two characters in the job, we have PostScript, and can pass the rest of the job directly. If those are not the first two characters in the file, then the filter will convert the text into PostScript and print the result.

How do we do this?

If you have got a serial printer, a great way to do it is to install `lprps`. `lprps` is a PostScript printer filter which performs two-way communication with the printer. It updates the printer's status file with verbose information from the printer, so users and administrators can see exactly what the state of the printer is (such as `toner low` or `paper jam`). But more importantly, it includes a program called `psif` which detects whether the incoming job is plain text and calls `textps` (another program that comes with `lprps`) to convert it to PostScript. It then uses `lprps` to send the job to the printer.

`lprps` is part of the FreeBSD Ports Collection (see The Ports Collection). You can install one of the both `print/lprps-a4` and `print/lprps-letter` ports according to the paper size used. After installing `lprps`, just specify the pathname to the `psif` program that is part of `lprps`. If you installed `lprps` from the Ports Collection, use the following in the serial PostScript printer's entry in `/etc/printcap`:

```
:if=/usr/local/libexec/psif:
```

The `rw` capability should be also included in order to let **LPD** to open the printer in the read-write mode.

If you have a parallel PostScript printer (and therefore cannot use two-way communication with the printer, which `lprps` needs), you can use the following shell script as the text filter:

```
#!/bin/sh
#
#  psif - Print PostScript or plain text on a PostScript printer
#  Script version; NOT the version that comes with lprps
#  Installed in /usr/local/libexec/psif
#

IFS="" read -r first_line
first_two_chars=‘expr "$first_line" : ’\(..\)’‘
```

```
if [ "$first_two_chars" = "%!" ]; then
    #
    #  PostScript job, print it.
    #
    echo "$first_line" && cat && printf "\004" && exit 0
    exit 2
else
    #
    #  Plain text, convert it, then print it.
    #
    ( echo "$first_line"; cat ) | /usr/local/bin/textps && printf "\004" && exit 0
    exit 2
fi
```

In the above script, `textps` is a program we installed separately to convert plain text to PostScript. You can use any text-to-PostScript program you wish. The FreeBSD Ports Collection (see The Ports Collection) includes a full featured text-to-PostScript program called `a2ps` that you might want to investigate.

10.4.1.3 Simulating PostScript on Non PostScript Printers

PostScript is the *de facto* standard for high quality typesetting and printing. PostScript is, however, an *expensive* standard. Thankfully, Aladdin Enterprises has a free PostScript work-alike called **Ghostscript** that runs with FreeBSD. **Ghostscript** can read most PostScript files and can render their pages onto a variety of devices, including many brands of non-PostScript printers. By installing **Ghostscript** and using a special text filter for your printer, you can make your non PostScript printer act like a real PostScript printer.

Ghostscript is in the FreeBSD Ports Collection, many versions are available, the most commonly used version is `print/ghostscript-gpl`.

To simulate PostScript, we have the text filter detect if it is printing a PostScript file. If it is not, then the filter will pass the file directly to the printer; otherwise, it will use **Ghostscript** to first convert the file into a format the printer will understand.

Here is an example: the following script is a text filter for Hewlett Packard DeskJet 500 printers. For other printers, substitute the `-sDEVICE` argument to the `gs` (**Ghostscript**) command. (Type `gs -h` to get a list of devices the current installation of **Ghostscript** supports.)

```
#!/bin/sh
#
#  ifhp - Print Ghostscript-simulated PostScript on a DeskJet 500
#  Installed in /usr/local/libexec/ifhp

#
#  Treat LF as CR+LF (to avoid the "staircase effect" on HP/PCL
#  printers):
#
printf "\033&k2G" || exit 2

#
#  Read first two characters of the file
#
IFS="" read -r first_line
```

```
first_two_chars=‘expr "$first_line" : ’\(..\)’‘

if [ "$first_two_chars" = "%!" ]; then
    #
    #  It is PostScript; use Ghostscript to scan-convert and print it.
    #
    /usr/local/bin/gs -dSAFER -dNOPAUSE -q -sDEVICE=djet500 \
      -sOutputFile=- - && exit 0
else
    #
    #  Plain text or HP/PCL, so just print it directly; print a form feed
    #  at the end to eject the last page.
    #
    echo "$first_line" && cat && printf "\033&l0H" &&
exit 0
fi

exit 2
```

Finally, you need to notify **LPD** of the filter via the `if` capability:

```
:if=/usr/local/libexec/ifhp:
```

That is it. You can type `lpr` *`plain.text`* and `lpr` *`whatever.ps`* and both should print successfully.

10.4.1.4 Conversion Filters

After completing the simple setup described in Simple Printer Setup, the first thing you will probably want to do is install conversion filters for your favorite file formats (besides plain ASCII text).

10.4.1.4.1 Why Install Conversion Filters?

Conversion filters make printing various kinds of files easy. As an example, suppose we do a lot of work with the TEX typesetting system, and we have a PostScript printer. Every time we generate a DVI file from TEX, we cannot print it directly until we convert the DVI file into PostScript. The command sequence goes like this:

```
% dvips seaweed-analysis.dvi
% lpr seaweed-analysis.ps
```

By installing a conversion filter for DVI files, we can skip the hand conversion step each time by having **LPD** do it for us. Now, each time we get a DVI file, we are just one step away from printing it:

```
% lpr -d seaweed-analysis.dvi
```

We got **LPD** to do the DVI file conversion for us by specifying the `-d` option. Section Formatting and Conversion Options lists the conversion options.

For each of the conversion options you want a printer to support, install a *conversion filter* and specify its pathname in `/etc/printcap`. A conversion filter is like the text filter for the simple printer setup (see section Installing the Text Filter) except that instead of printing plain text, the filter converts the file into a format the printer can understand.

10.4.1.4.2 Which Conversion Filters Should I Install?

You should install the conversion filters you expect to use. If you print a lot of DVI data, then a DVI conversion filter is in order. If you have got plenty of troff to print out, then you probably want a troff filter.

The following table summarizes the filters that **LPD** works with, their capability entries for the `/etc/printcap` file, and how to invoke them with the `lpr` command:

File type	`/etc/printcap` capability	`lpr` option
cifplot	`cf`	`-c`
DVI	`df`	`-d`
plot	`gf`	`-g`
ditroff	`nf`	`-n`
FORTRAN text	`rf`	`-f`
troff	`tf`	`-f`
raster	`vf`	`-v`
plain text	`if`	none, `-p`, or `-l`

In our example, using `lpr -d` means the printer needs a `df` capability in its entry in `/etc/printcap`.

Despite what others might contend, formats like FORTRAN text and plot are probably obsolete. At your site, you can give new meanings to these or any of the formatting options just by installing custom filters. For example, suppose you would like to directly print Printerleaf files (files from the Interleaf desktop publishing program), but will never print plot files. You could install a Printerleaf conversion filter under the `gf` capability and then educate your users that `lpr -g` mean “print Printerleaf files.”

10.4.1.4.3 Installing Conversion Filters

Since conversion filters are programs you install outside of the base FreeBSD installation, they should probably go under `/usr/local`. The directory `/usr/local/libexec` is a popular location, since they are specialized programs that only **LPD** will run; regular users should not ever need to run them.

To enable a conversion filter, specify its pathname under the appropriate capability for the destination printer in `/etc/printcap`.

In our example, we will add the DVI conversion filter to the entry for the printer named `bamboo`. Here is the example `/etc/printcap` file again, with the new `df` capability for the printer `bamboo`:

```
#
#  /etc/printcap for host rose - added df filter for bamboo
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
        :sh:sd=/var/spool/lpd/rattan:\
        :lp=/dev/lpt0:\
        :if=/usr/local/libexec/if-simple:

bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
        :sh:sd=/var/spool/lpd/bamboo:\
        :lp=/dev/ttyu5:ms#-parenb cs8 clocal crtscts:rw:\
        :if=/usr/local/libexec/psif:\
        :df=/usr/local/libexec/psdf:
```

The DVI filter is a shell script named /usr/local/libexec/psdf. Here is that script:

```
#!/bin/sh
#
#  psdf - DVI to PostScript printer filter
#  Installed in /usr/local/libexec/psdf
#
# Invoked by lpd when user runs lpr -d
#
exec /usr/local/bin/dvips -f | /usr/local/libexec/lprps "$@"
```

This script runs dvips in filter mode (the -f argument) on standard input, which is the job to print. It then starts the PostScript printer filter lprps (see section Accommodating Plain Text Jobs on PostScript Printers) with the arguments **LPD** passed to this script. The lprps utility will use those arguments to account for the pages printed.

10.4.1.4.4 More Conversion Filter Examples

There is no fixed set of steps to install conversion filters, some working examples are described in this section. Use these as guidance to making your own filters. Use them directly, if appropriate.

This example script is a raster (well, GIF file, actually) conversion filter for a Hewlett Packard LaserJet III-Si printer:

```
#!/bin/sh
#
#  hpvf - Convert GIF files into HP/PCL, then print
#  Installed in /usr/local/libexec/hpvf

PATH=/usr/X11R6/bin:$PATH; export PATH
giftopnm | ppmtopgm | pgmtopbm | pbmtolj -resolution 300 \
    && exit 0 \
    || exit 2
```

It works by converting the GIF file into a portable anymap, converting that into a portable graymap, converting that into a portable bitmap, and converting that into LaserJet/PCL-compatible data.

Here is the /etc/printcap file with an entry for a printer using the above filter:

```
#
#  /etc/printcap for host orchid
#
teak|hp|laserjet|Hewlett Packard LaserJet 3Si:\
        :lp=/dev/lpt0:sh:sd=/var/spool/lpd/teak:mx#0:\
        :if=/usr/local/libexec/hpif:\
        :vf=/usr/local/libexec/hpvf:
```

The following script is a conversion filter for troff data from the groff typesetting system for the PostScript printer named bamboo:

```
#!/bin/sh
#
#  pstf - Convert groff's troff data into PS, then print.
#  Installed in /usr/local/libexec/pstf
#
```

```
exec grops | /usr/local/libexec/lprps "$@"
```

The above script makes use of `lprps` again to handle the communication with the printer. If the printer were on a parallel port, we would use this script instead:

```
#!/bin/sh
#
#  pstf - Convert groff's troff data into PS, then print.
#  Installed in /usr/local/libexec/pstf
#
exec grops
```

That is it. Here is the entry we need to add to `/etc/printcap` to enable the filter:

```
:tf=/usr/local/libexec/pstf:
```

Here is an example that might make old hands at FORTRAN blush. It is a FORTRAN-text filter for any printer that can directly print plain text. We will install it for the printer `teak`:

```
#!/bin/sh
#
# hprf - FORTRAN text filter for LaserJet 3si:
# Installed in /usr/local/libexec/hprf
#

printf "\033&k2G" && fpr && printf "\033&l0H" &&
 exit 0
exit 2
```

And we will add this line to the `/etc/printcap` for the printer `teak` to enable this filter:

```
:rf=/usr/local/libexec/hprf:
```

Here is one final, somewhat complex example. We will add a DVI filter to the LaserJet printer `teak` introduced earlier. First, the easy part: updating `/etc/printcap` with the location of the DVI filter:

```
:df=/usr/local/libexec/hpdf:
```

Now, for the hard part: making the filter. For that, we need a DVI-to-LaserJet/PCL conversion program. The FreeBSD Ports Collection (see The Ports Collection) has one: `print/dvi2xx`. Installing this port gives us the program we need, `dvilj2p`, which converts DVI into LaserJet IIp, LaserJet III, and LaserJet 2000 compatible codes.

The `dvilj2p` utility makes the filter `hpdf` quite complex since `dvilj2p` cannot read from standard input. It wants to work with a filename. What is worse, the filename has to end in `.dvi` so using `/dev/fd/0` for standard input is problematic. We can get around that problem by linking (symbolically) a temporary file name (one that ends in `.dvi`) to `/dev/fd/0`, thereby forcing `dvilj2p` to read from standard input.

The only other fly in the ointment is the fact that we cannot use `/tmp` for the temporary link. Symbolic links are owned by user and group `bin`. The filter runs as user `daemon`. And the `/tmp` directory has the sticky bit set. The filter can create the link, but it will not be able clean up when done and remove it since the link will belong to a different user.

Instead, the filter will make the symbolic link in the current working directory, which is the spooling directory (specified by the `sd` capability in `/etc/printcap`). This is a perfect place for filters to do their work, especially since there is (sometimes) more free disk space in the spooling directory than under `/tmp`.

Here, finally, is the filter:

```
#!/bin/sh
#
#  hpdf - Print DVI data on HP/PCL printer
#  Installed in /usr/local/libexec/hpdf

PATH=/usr/local/bin:$PATH; export PATH

#
#  Define a function to clean up our temporary files.  These exist
#  in the current directory, which will be the spooling directory
#  for the printer.
#
cleanup() {
   rm -f hpdf$$.dvi
}

#
#  Define a function to handle fatal errors: print the given message
#  and exit 2.  Exiting with 2 tells LPD to do not try to reprint the
#  job.
#
fatal() {
    echo "$@" 1>&2
    cleanup
    exit 2
}

#
#  If user removes the job, LPD will send SIGINT, so trap SIGINT
#  (and a few other signals) to clean up after ourselves.
#
trap cleanup 1 2 15

#
#  Make sure we are not colliding with any existing files.
#
cleanup

#
#  Link the DVI input file to standard input (the file to print).
#
ln -s /dev/fd/0 hpdf$$.dvi || fatal "Cannot symlink /dev/fd/0"

#
#  Make LF = CR+LF
#
printf "\033&k2G" || fatal "Cannot initialize printer"
```

```
#
#  Convert and print.  Return value from dvilj2p does not seem to be
#  reliable, so we ignore it.
#
dvilj2p -M1 -q -e- dfhp$$.dvi

#
#  Clean up and exit
#
cleanup
exit 0
```

10.4.1.4.5 Automated Conversion: an Alternative to Conversion Filters

All these conversion filters accomplish a lot for your printing environment, but at the cost forcing the user to specify (on the lpr(1) command line) which one to use. If your users are not particularly computer literate, having to specify a filter option will become annoying. What is worse, though, is that an incorrectly specified filter option may run a filter on the wrong type of file and cause your printer to spew out hundreds of sheets of paper.

Rather than install conversion filters at all, you might want to try having the text filter (since it is the default filter) detect the type of file it has been asked to print and then automatically run the right conversion filter. Tools such as `file` can be of help here. Of course, it will be hard to determine the differences between *some* file types—and, of course, you can still provide conversion filters just for them.

The FreeBSD Ports Collection has a text filter that performs automatic conversion called `apsfilter` (`print/apsfilter`). It can detect plain text, PostScript, DVI and almost any kind of files, run the proper conversions, and print.

10.4.1.5 Output Filters

The **LPD** spooling system supports one other type of filter that we have not yet explored: an output filter. An output filter is intended for printing plain text only, like the text filter, but with many simplifications. If you are using an output filter but no text filter, then:

- **LPD** starts an output filter once for the entire job instead of once for each file in the job.
- **LPD** does not make any provision to identify the start or the end of files within the job for the output filter.
- **LPD** does not pass the user's login or host to the filter, so it is not intended to do accounting. In fact, it gets only two arguments:

  `filter-name -w`*`width`* `-l`*`length`*

  Where *`width`* is from the `pw` capability and *`length`* is from the `pl` capability for the printer in question.

Do not be seduced by an output filter's simplicity. If you would like each file in a job to start on a different page an output filter *will not work*. Use a text filter (also known as an input filter); see section Installing the Text Filter. Furthermore, an output filter is actually *more complex* in that it has to examine the byte stream being sent to it for special flag characters and must send signals to itself on behalf of **LPD**.

However, an output filter is *necessary* if you want header pages and need to send escape sequences or other initialization strings to be able to print the header page. (But it is also *futile* if you want to charge header pages to the requesting user's account, since **LPD** does not give any user or host information to the output filter.)

On a single printer, **LPD** allows both an output filter and text or other filters. In such cases, **LPD** will start the output filter to print the header page (see section Header Pages) only. **LPD** then expects the output filter to *stop itself* by sending two bytes to the filter: ASCII 031 followed by ASCII 001. When an output filter sees these two bytes (031, 001), it should stop by sending `SIGSTOP` to itself. When **LPD**'s done running other filters, it will restart the output filter by sending `SIGCONT` to it.

If there is an output filter but *no* text filter and **LPD** is working on a plain text job, **LPD** uses the output filter to do the job. As stated before, the output filter will print each file of the job in sequence with no intervening form feeds or other paper advancement, and this is probably *not* what you want. In almost all cases, you need a text filter.

The program `lpf`, which we introduced earlier as a text filter, can also run as an output filter. If you need a quick-and-dirty output filter but do not want to write the byte detection and signal sending code, try `lpf`. You can also wrap `lpf` in a shell script to handle any initialization codes the printer might require.

10.4.1.6 `lpf`: a Text Filter

The program `/usr/libexec/lpr/lpf` that comes with FreeBSD binary distribution is a text filter (input filter) that can indent output (job submitted with `lpr -i`), allow literal characters to pass (job submitted with `lpr -l`), adjust the printing position for backspaces and tabs in the job, and account for pages printed. It can also act like an output filter.

The `lpf` filter is suitable for many printing environments. And although it has no capability to send initialization sequences to a printer, it is easy to write a shell script to do the needed initialization and then execute `lpf`.

In order for `lpf` to do page accounting correctly, it needs correct values filled in for the `pw` and `pl` capabilities in the `/etc/printcap` file. It uses these values to determine how much text can fit on a page and how many pages were in a user's job. For more information on printer accounting, see Accounting for Printer Usage.

10.4.2 Header Pages

If you have *lots* of users, all of them using various printers, then you probably want to consider *header pages* as a necessary evil.

Header pages, also known as *banner* or *burst pages* identify to whom jobs belong after they are printed. They are usually printed in large, bold letters, perhaps with decorative borders, so that in a stack of printouts they stand out from the real documents that comprise users' jobs. They enable users to locate their jobs quickly. The obvious drawback to a header page is that it is yet one more sheet that has to be printed for every job, their ephemeral usefulness lasting not more than a few minutes, ultimately finding themselves in a recycling bin or rubbish heap. (Note that header pages go with each job, not each file in a job, so the paper waste might not be that bad.)

The **LPD** system can provide header pages automatically for your printouts *if* your printer can directly print plain text. If you have a PostScript printer, you will need an external program to generate the header page; see Header Pages on PostScript Printers.

10.4.2.1 Enabling Header Pages

In the Simple Printer Setup section, we turned off header pages by specifying `sh` (meaning "suppress header") in the `/etc/printcap` file. To enable header pages for a printer, just remove the `sh` capability.

Sounds too easy, right?

You are right. You *might* have to provide an output filter to send initialization strings to the printer. Here is an example output filter for Hewlett Packard PCL-compatible printers:

```
#!/bin/sh
#
#  hpof - Output filter for Hewlett Packard PCL-compatible printers
#  Installed in /usr/local/libexec/hpof

printf "\033&k2G" || exit 2
exec /usr/libexec/lpr/lpf
```

Specify the path to the output filter in the `of` capability. See the Output Filters section for more information.

Here is an example `/etc/printcap` file for the printer `teak` that we introduced earlier; we enabled header pages and added the above output filter:

```
#
#  /etc/printcap for host orchid
#
teak|hp|laserjet|Hewlett Packard LaserJet 3Si:\
        :lp=/dev/lpt0:sd=/var/spool/lpd/teak:mx#0:\
        :if=/usr/local/libexec/hpif:\
        :vf=/usr/local/libexec/hpvf:\
        :of=/usr/local/libexec/hpof:
```

Now, when users print jobs to `teak`, they get a header page with each job. If users want to spend time searching for their printouts, they can suppress header pages by submitting the job with `lpr -h`; see the Header Page Options section for more lpr(1) options.

> **Note: LPD** prints a form feed character after the header page. If your printer uses a different character or sequence of characters to eject a page, specify them with the `ff` capability in `/etc/printcap`.

10.4.2.2 Controlling Header Pages

By enabling header pages, **LPD** will produce a *long header*, a full page of large letters identifying the user, host, and job. Here is an example (`kelly` printed the job named "outline" from host `rose`):

```
      k                   ll       ll
      k                    l        l
      k                    l        l
      k   k     eeee       l        l     y    y
      k  k     e    e      l        l     y    y
      k k      eeeeee      l        l     y    y
      kk k     e           l        l     y    y
      k   k    e    e      l        l     y   yy
```

```
k    k    eeee      lll      lll     yyy y
                                        y
                                      y    y
                                       yyyy


                              ll
                    t          l        i
                    t          l
 oooo    u    u   ttttt       l       ii     n nnn     eeee
o    o   u    u     t         l        i     nn   n   e    e
o    o   u    u     t         l        i     n    n   eeeeee
o    o   u    u     t         l        i     n    n   e
o    o   u   uu     t  t      l        i     n    n   e    e
 oooo     uuu u      tt      lll      iii    n    n    eeee
```

```
r rrr     oooo     ssss     eeee
rr   r   o    o   s    s   e    e
r        o    o    ss      eeeeee
r        o    o      ss    e
r        o    o   s    s   e    e
r         oooo     ssss     eeee
```

```
                                    Job:  outline
                                    Date: Sun Sep 17 11:04:58 1995
```

LPD appends a form feed after this text so the job starts on a new page (unless you have `sf` (suppress form feeds) in the destination printer's entry in `/etc/printcap`).

If you prefer, **LPD** can make a *short header*; specify `sb` (short banner) in the `/etc/printcap` file. The header page will look like this:

```
rose:kelly  Job: outline  Date: Sun Sep 17 11:07:51 1995
```

Also by default, **LPD** prints the header page first, then the job. To reverse that, specify `hl` (header last) in `/etc/printcap`.

10.4.2.3 Accounting for Header Pages

Using **LPD**'s built-in header pages enforces a particular paradigm when it comes to printer accounting: header pages must be *free of charge*.

Why?

Because the output filter is the only external program that will have control when the header page is printed that could do accounting, and it is not provided with any *user or host* information or an accounting file, so it has no idea whom to charge for printer use. It is also not enough to just "increase the page count by one" by modifying the text filter or any of the conversion filters (which do have user and host information) since users can suppress header pages with `lpr -h`. They could still be charged for header pages they did not print. Basically, `lpr -h` will be the preferred option of environmentally-minded users, but you cannot offer any incentive to use it.

It is *still not enough* to have each of the filters generate their own header pages (thereby being able to charge for them). If users wanted the option of suppressing the header pages with `lpr -h`, they will still get them and be charged for them since **LPD** does not pass any knowledge of the `-h` option to any of the filters.

So, what are your options?

You can:

- Accept **LPD**'s paradigm and make header pages free.
- Install an alternative to **LPD**, such as **LPRng**. Section Alternatives to the Standard Spooler tells more about other spooling software you can substitute for **LPD**.
- Write a *smart* output filter. Normally, an output filter is not meant to do anything more than initialize a printer or do some simple character conversion. It is suited for header pages and plain text jobs (when there is no text (input) filter). But, if there is a text filter for the plain text jobs, then **LPD** will start the output filter only for the header pages. And the output filter can parse the header page text that **LPD** generates to determine what user and host to charge for the header page. The only other problem with this method is that the output filter still does not know what accounting file to use (it is not passed the name of the file from the `af` capability), but if you have a well-known accounting file, you can hard-code that into the output filter. To facilitate the parsing step, use the `sh` (short header) capability in `/etc/printcap`. Then again, all that might be too much trouble, and users will certainly appreciate the more generous system administrator who makes header pages free.

10.4.2.4 Header Pages on PostScript Printers

As described above, **LPD** can generate a plain text header page suitable for many printers. Of course, PostScript cannot directly print plain text, so the header page feature of **LPD** is useless—or mostly so.

One obvious way to get header pages is to have every conversion filter and the text filter generate the header page. The filters should use the user and host arguments to generate a suitable header page. The drawback of this method is that users will always get a header page, even if they submit jobs with `lpr -h`.

Let us explore this method. The following script takes three arguments (user login name, host name, and job name) and makes a simple PostScript header page:

```
#!/bin/sh
#
#  make-ps-header - make a PostScript header page on stdout
#  Installed in /usr/local/libexec/make-ps-header
#
```

```
#
#  These are PostScript units (72 to the inch).  Modify for A4 or
#  whatever size paper you are using:
#
page_width=612
page_height=792
border=72

#
#  Check arguments
#
if [ $# -ne 3 ]; then
    echo "Usage: ‘basename $0‘ <user> <host> <job>" 1>&2
    exit 1
fi

#
#  Save these, mostly for readability in the PostScript, below.
#
user=$1
host=$2
job=$3
date=‘date‘

#
#  Send the PostScript code to stdout.
#
exec cat <<EOF
%!PS

%
%  Make sure we do not interfere with user’s job that will follow
%
save

%
%  Make a thick, unpleasant border around the edge of the paper.
%
$border $border moveto
$page_width $border 2 mul sub 0 rlineto
0 $page_height $border 2 mul sub rlineto
currentscreen 3 -1 roll pop 100 3 1 roll setscreen
$border 2 mul $page_width sub 0 rlineto closepath
0.8 setgray 10 setlinewidth stroke 0 setgray

%
%  Display user’s login name, nice and large and prominent
%
/Helvetica-Bold findfont 64 scalefont setfont
$page_width ($user) stringwidth pop sub 2 div $page_height 200 sub moveto
($user) show
```

```
%
%  Now show the boring particulars
%
/Helvetica findfont 14 scalefont setfont
/y 200 def
[ (Job:) (Host:) (Date:) ] {
200 y moveto show /y y 18 sub def }
forall

/Helvetica-Bold findfont 14 scalefont setfont
/y 200 def
[ ($job) ($host) ($date) ] {
        270 y moveto show /y y 18 sub def
} forall

%
% That is it
%
restore
showpage
EOF
```

Now, each of the conversion filters and the text filter can call this script to first generate the header page, and then print the user's job. Here is the DVI conversion filter from earlier in this document, modified to make a header page:

```
#!/bin/sh
#
#  psdf - DVI to PostScript printer filter
#  Installed in /usr/local/libexec/psdf
#
#  Invoked by lpd when user runs lpr -d
#

orig_args="$@"

fail() {
    echo "$@" 1>&2
    exit 2
}

while getopts "x:y:n:h:" option; do
    case $option in
        x|y)  ;; # Ignore
        n)    login=$OPTARG ;;
        h)    host=$OPTARG ;;
        *)    echo "LPD started ‘basename $0‘ wrong." 1>&2
              exit 2
              ;;
    esac
done

[ "$login" ] || fail "No login name"
[ "$host" ] || fail "No host name"
```

```
( /usr/local/libexec/make-ps-header $login $host "DVI File"
  /usr/local/bin/dvips -f ) | eval /usr/local/libexec/lprps $orig_args
```

Notice how the filter has to parse the argument list in order to determine the user and host name. The parsing for the other conversion filters is identical. The text filter takes a slightly different set of arguments, though (see section How Filters Work).

As we have mentioned before, the above scheme, though fairly simple, disables the “suppress header page” option (the `-h` option) to `lpr`. If users wanted to save a tree (or a few pennies, if you charge for header pages), they would not be able to do so, since every filter’s going to print a header page with every job.

To allow users to shut off header pages on a per-job basis, you will need to use the trick introduced in section Accounting for Header Pages: write an output filter that parses the LPD-generated header page and produces a PostScript version. If the user submits the job with `lpr -h`, then **LPD** will not generate a header page, and neither will your output filter. Otherwise, your output filter will read the text from **LPD** and send the appropriate header page PostScript code to the printer.

If you have a PostScript printer on a serial line, you can make use of `lprps`, which comes with an output filter, `psof`, which does the above. Note that `psof` does not charge for header pages.

10.4.3 Networked Printing

FreeBSD supports networked printing: sending jobs to remote printers. Networked printing generally refers to two different things:

- Accessing a printer attached to a remote host. You install a printer that has a conventional serial or parallel interface on one host. Then, you set up **LPD** to enable access to the printer from other hosts on the network. Section Printers Installed on Remote Hosts tells how to do this.
- Accessing a printer attached directly to a network. The printer has a network interface in addition to (or in place of) a more conventional serial or parallel interface. Such a printer might work as follows:
  - It might understand the **LPD** protocol and can even queue jobs from remote hosts. In this case, it acts just like a regular host running **LPD**. Follow the same procedure in section Printers Installed on Remote Hosts to set up such a printer.
  - It might support a data stream network connection. In this case, you “attach” the printer to one host on the network by making that host responsible for spooling jobs and sending them to the printer. Section Printers with Networked Data Stream Interfaces gives some suggestions on installing such printers.

10.4.3.1 Printers Installed on Remote Hosts

The **LPD** spooling system has built-in support for sending jobs to other hosts also running **LPD** (or are compatible with **LPD**). This feature enables you to install a printer on one host and make it accessible from other hosts. It also works with printers that have network interfaces that understand the **LPD** protocol.

To enable this kind of remote printing, first install a printer on one host, the *printer host*, using the simple printer setup described in the Simple Printer Setup section. Do any advanced setup in Advanced Printer Setup that you need.

Make sure to test the printer and see if it works with the features of **LPD** you have enabled. Also ensure that the *local host* has authorization to use the **LPD** service in the *remote host* (see Restricting Jobs from Remote Hosts).

If you are using a printer with a network interface that is compatible with **LPD**, then the *printer host* in the discussion below is the printer itself, and the *printer name* is the name you configured for the printer. See the documentation that accompanied your printer and/or printer-network interface.

> **Tip:** If you are using a Hewlett Packard Laserjet then the printer name `text` will automatically perform the LF to CRLF conversion for you, so you will not require the `hpif` script.

Then, on the other hosts you want to have access to the printer, make an entry in their `/etc/printcap` files with the following:

1. Name the entry anything you want. For simplicity, though, you probably want to use the same name and aliases as on the printer host.
2. Leave the `lp` capability blank, explicitly (`:lp=:`).
3. Make a spooling directory and specify its location in the `sd` capability. **LPD** will store jobs here before they get sent to the printer host.
4. Place the name of the printer host in the `rm` capability.
5. Place the printer name on the *printer host* in the `rp` capability.

That is it. You do not need to list conversion filters, page dimensions, or anything else in the `/etc/printcap` file.

Here is an example. The host `rose` has two printers, `bamboo` and `rattan`. We will enable users on the host `orchid` to print to those printers. Here is the `/etc/printcap` file for `orchid` (back from section Enabling Header Pages). It already had the entry for the printer `teak`; we have added entries for the two printers on the host `rose`:

```
#
#  /etc/printcap for host orchid - added (remote) printers on rose
#

#
#  teak is local; it is connected directly to orchid:
#
teak|hp|laserjet|Hewlett Packard LaserJet 3Si:\
        :lp=/dev/lpt0:sd=/var/spool/lpd/teak:mx#0:\
        :if=/usr/local/libexec/ifhp:\
        :vf=/usr/local/libexec/vfhp:\
        :of=/usr/local/libexec/ofhp:

#
#  rattan is connected to rose; send jobs for rattan to rose:
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
        :lp=:rm=rose:rp=rattan:sd=/var/spool/lpd/rattan:

#
#  bamboo is connected to rose as well:
#
```

```
bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
        :lp=:rm=rose:rp=bamboo:sd=/var/spool/lpd/bamboo:
```

Then, we just need to make spooling directories on `orchid`:

```
# mkdir -p /var/spool/lpd/rattan /var/spool/lpd/bamboo
# chmod 770 /var/spool/lpd/rattan /var/spool/lpd/bamboo
# chown daemon:daemon /var/spool/lpd/rattan /var/spool/lpd/bamboo
```

Now, users on `orchid` can print to `rattan` and `bamboo`. If, for example, a user on `orchid` typed:

```
% lpr -P bamboo -d sushi-review.dvi
```

the **LPD** system on `orchid` would copy the job to the spooling directory `/var/spool/lpd/bamboo` and note that it was a DVI job. As soon as the host `rose` has room in its `bamboo` spooling directory, the two **LPD**s would transfer the file to `rose`. The file would wait in `rose`'s queue until it was finally printed. It would be converted from DVI to PostScript (since `bamboo` is a PostScript printer) on `rose`.

10.4.3.2 Printers with Networked Data Stream Interfaces

Often, when you buy a network interface card for a printer, you can get two versions: one which emulates a spooler (the more expensive version), or one which just lets you send data to it as if you were using a serial or parallel port (the cheaper version). This section tells how to use the cheaper version. For the more expensive one, see the previous section Printers Installed on Remote Hosts.

The format of the `/etc/printcap` file lets you specify what serial or parallel interface to use, and (if you are using a serial interface), what baud rate, whether to use flow control, delays for tabs, conversion of newlines, and more. But there is no way to specify a connection to a printer that is listening on a TCP/IP or other network port.

To send data to a networked printer, you need to develop a communications program that can be called by the text and conversion filters. Here is one such example: the script `netprint` takes all data on standard input and sends it to a network-attached printer. We specify the hostname of the printer as the first argument and the port number to which to connect as the second argument to `netprint`. Note that this supports one-way communication only (FreeBSD to printer); many network printers support two-way communication, and you might want to take advantage of that (to get printer status, perform accounting, etc.).

```
#!/usr/bin/perl
#
#  netprint - Text filter for printer attached to network
#  Installed in /usr/local/libexec/netprint
#
$#ARGV eq 1 || die "Usage: $0 <printer-hostname> <port-number>";

$printer_host = $ARGV[0];
$printer_port = $ARGV[1];

require 'sys/socket.ph';

($ignore, $ignore, $protocol) = getprotobyname('tcp');
($ignore, $ignore, $ignore, $ignore, $address)
    = gethostbyname($printer_host);
```

```
$sockaddr = pack('S n a4 x8', &AF_INET, $printer_port, $address);

socket(PRINTER, &PF_INET, &SOCK_STREAM, $protocol)
    || die "Can't create TCP/IP stream socket: $!";
connect(PRINTER, $sockaddr) || die "Can't contact $printer_host: $!";
while (<STDIN>) { print PRINTER; }
exit 0;
```

We can then use this script in various filters. Suppose we had a Diablo 750-N line printer connected to the network. The printer accepts data to print on port number 5100. The host name of the printer is `scrivener`. Here is the text filter for the printer:

```
#!/bin/sh
#
#  diablo-if-net - Text filter for Diablo printer 'scrivener' listening
#  on port 5100.   Installed in /usr/local/libexec/diablo-if-net
#
exec /usr/libexec/lpr/lpf "$@" | /usr/local/libexec/netprint scrivener 5100
```

10.4.4 Restricting Printer Usage

This section gives information on restricting printer usage. The **LPD** system lets you control who can access a printer, both locally or remotely, whether they can print multiple copies, how large their jobs can be, and how large the printer queues can get.

10.4.4.1 Restricting Multiple Copies

The **LPD** system makes it easy for users to print multiple copies of a file. Users can print jobs with `lpr -#5` (for example) and get five copies of each file in the job. Whether this is a good thing is up to you.

If you feel multiple copies cause unnecessary wear and tear on your printers, you can disable the `-#` option to lpr(1) by adding the `sc` capability to the `/etc/printcap` file. When users submit jobs with the `-#` option, they will see:

```
lpr: multiple copies are not allowed
```

Note that if you have set up access to a printer remotely (see section Printers Installed on Remote Hosts), you need the `sc` capability on the remote `/etc/printcap` files as well, or else users will still be able to submit multiple-copy jobs by using another host.

Here is an example. This is the `/etc/printcap` file for the host `rose`. The printer `rattan` is quite hearty, so we will allow multiple copies, but the laser printer `bamboo` is a bit more delicate, so we will disable multiple copies by adding the `sc` capability:

```
#
#  /etc/printcap for host rose - restrict multiple copies on bamboo
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
        :sh:sd=/var/spool/lpd/rattan:\
        :lp=/dev/lpt0:\
        :if=/usr/local/libexec/if-simple:
```

```
bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
        :sh:sd=/var/spool/lpd/bamboo:sc:\
        :lp=/dev/ttyu5:ms#-parenb cs8 clocal crtscts:rw:\
        :if=/usr/local/libexec/psif:\
        :df=/usr/local/libexec/psdf:
```

Now, we also need to add the `sc` capability on the host `orchid`'s `/etc/printcap` (and while we are at it, let us disable multiple copies for the printer `teak`):

```
#
#  /etc/printcap for host orchid - no multiple copies for local
#  printer teak or remote printer bamboo
teak|hp|laserjet|Hewlett Packard LaserJet 3Si:\
        :lp=/dev/lpt0:sd=/var/spool/lpd/teak:mx#0:sc:\
        :if=/usr/local/libexec/ifhp:\
        :vf=/usr/local/libexec/vfhp:\
        :of=/usr/local/libexec/ofhp:

rattan|line|diablo|lp|Diablo 630 Line Printer:\
        :lp=:rm=rose:rp=rattan:sd=/var/spool/lpd/rattan:

bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
        :lp=:rm=rose:rp=bamboo:sd=/var/spool/lpd/bamboo:sc:
```

By using the `sc` capability, we prevent the use of `lpr -#`, but that still does not prevent users from running lpr(1) multiple times, or from submitting the same file multiple times in one job like this:

```
% lpr forsale.sign forsale.sign forsale.sign forsale.sign forsale.sign
```

There are many ways to prevent this abuse (including ignoring it) which you are free to explore.

10.4.4.2 Restricting Access to Printers

You can control who can print to what printers by using the UNIX group mechanism and the `rg` capability in `/etc/printcap`. Just place the users you want to have access to a printer in a certain group, and then name that group in the `rg` capability.

If users outside the group (including `root`) try to print to the controlled printer then they will be greeted with the following message:

```
lpr: Not a member of the restricted group
```

As with the `sc` (suppress multiple copies) capability, you need to specify `rg` on remote hosts that also have access to your printers, if you feel it is appropriate (see section Printers Installed on Remote Hosts).

For example, we will let anyone access the printer `rattan`, but only those in group `artists` can use `bamboo`. Here is the familiar `/etc/printcap` for host `rose`:

```
#
#  /etc/printcap for host rose - restricted group for bamboo
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
        :sh:sd=/var/spool/lpd/rattan:\
```

```
        :lp=/dev/lpt0:\
        :if=/usr/local/libexec/if-simple:

bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
        :sh:sd=/var/spool/lpd/bamboo:sc:rg=artists:\
        :lp=/dev/ttyu5:ms#-parenb cs8 clocal crtscts:rw:\
        :if=/usr/local/libexec/psif:\
        :df=/usr/local/libexec/psdf:
```

Let us leave the other example `/etc/printcap` file (for the host `orchid`) alone. Of course, anyone on `orchid` can print to `bamboo`. It might be the case that we only allow certain logins on `orchid` anyway, and want them to have access to the printer. Or not.

> **Note:** There can be only one restricted group per printer.

10.4.4.3 Controlling Sizes of Jobs Submitted

If you have many users accessing the printers, you probably need to put an upper limit on the sizes of the files users can submit to print. After all, there is only so much free space on the filesystem that houses the spooling directories, and you also need to make sure there is room for the jobs of other users.

LPD enables you to limit the maximum byte size a file in a job can be with the `mx` capability. The units are in `BUFSIZ` blocks, which are 1024 bytes. If you put a zero for this capability, there will be no limit on file size; however, if no `mx` capability is specified, then a default limit of 1000 blocks will be used.

> **Note:** The limit applies to *files* in a job, and *not* the total job size.

LPD will not refuse a file that is larger than the limit you place on a printer. Instead, it will queue as much of the file up to the limit, which will then get printed. The rest will be discarded. Whether this is correct behavior is up for debate.

Let us add limits to our example printers `rattan` and `bamboo`. Since those `artists`' PostScript files tend to be large, we will limit them to five megabytes. We will put no limit on the plain text line printer:

```
#
#  /etc/printcap for host rose
#

#
#  No limit on job size:
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
        :sh:mx#0:sd=/var/spool/lpd/rattan:\
        :lp=/dev/lpt0:\
        :if=/usr/local/libexec/if-simple:

#
#  Limit of five megabytes:
```

```
#
bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
        :sh:sd=/var/spool/lpd/bamboo:sc:rg=artists:mx#5000:\
        :lp=/dev/ttyu5:ms#-parenb cs8 clocal crtscts:rw:\
        :if=/usr/local/libexec/psif:\
        :df=/usr/local/libexec/psdf:
```

Again, the limits apply to the local users only. If you have set up access to your printers remotely, remote users will not get those limits. You will need to specify the `mx` capability in the remote `/etc/printcap` files as well. See section Printers Installed on Remote Hosts for more information on remote printing.

There is another specialized way to limit job sizes from remote printers; see section Restricting Jobs from Remote Hosts.

10.4.4.4 Restricting Jobs from Remote Hosts

The **LPD** spooling system provides several ways to restrict print jobs submitted from remote hosts:

Host restrictions

You can control from which remote hosts a local **LPD** accepts requests with the files `/etc/hosts.equiv` and `/etc/hosts.lpd`. **LPD** checks to see if an incoming request is from a host listed in either one of these files. If not, **LPD** refuses the request.

The format of these files is simple: one host name per line. Note that the file `/etc/hosts.equiv` is also used by the ruserok(3) protocol, and affects programs like rsh(1) and rcp(1), so be careful.

For example, here is the `/etc/hosts.lpd` file on the host `rose`:

```
orchid
violet
madrigal.fishbaum.de
```

This means `rose` will accept requests from the hosts `orchid`, `violet`, and `madrigal.fishbaum.de`. If any other host tries to access `rose`'s **LPD**, the job will be refused.

Size restrictions

You can control how much free space there needs to remain on the filesystem where a spooling directory resides. Make a file called `minfree` in the spooling directory for the local printer. Insert in that file a number representing how many disk blocks (512 bytes) of free space there has to be for a remote job to be accepted.

This lets you insure that remote users will not fill your filesystem. You can also use it to give a certain priority to local users: they will be able to queue jobs long after the free disk space has fallen below the amount specified in the `minfree` file.

For example, let us add a `minfree` file for the printer `bamboo`. We examine `/etc/printcap` to find the spooling directory for this printer; here is `bamboo`'s entry:

```
bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
        :sh:sd=/var/spool/lpd/bamboo:sc:rg=artists:mx#5000:\
        :lp=/dev/ttyu5:ms#-parenb cs8 clocal crtscts:rw:mx#5000:\
        :if=/usr/local/libexec/psif:\
        :df=/usr/local/libexec/psdf:
```

The spooling directory is given in the `sd` capability. We will make three megabytes (which is 6144 disk blocks) the amount of free disk space that must exist on the filesystem for **LPD** to accept remote jobs:

```
# echo 6144 > /var/spool/lpd/bamboo/minfree
```

User restrictions

You can control which remote users can print to local printers by specifying the `rs` capability in `/etc/printcap`. When `rs` appears in the entry for a locally-attached printer, **LPD** will accept jobs from remote hosts *if* the user submitting the job also has an account of the same login name on the local host. Otherwise, **LPD** refuses the job.

This capability is particularly useful in an environment where there are (for example) different departments sharing a network, and some users transcend departmental boundaries. By giving them accounts on your systems, they can use your printers from their own departmental systems. If you would rather allow them to use *only* your printers and not your computer resources, you can give them "token" accounts, with no home directory and a useless shell like `/usr/bin/false`.

10.4.5 Accounting for Printer Usage

So, you need to charge for printouts. And why not? Paper and ink cost money. And then there are maintenance costs—printers are loaded with moving parts and tend to break down. You have examined your printers, usage patterns, and maintenance fees and have come up with a per-page (or per-foot, per-meter, or per-whatever) cost. Now, how do you actually start accounting for printouts?

Well, the bad news is the **LPD** spooling system does not provide much help in this department. Accounting is highly dependent on the kind of printer in use, the formats being printed, and *your* equirements in charging for printer usage.

To implement accounting, you have to modify a printer's text filter (to charge for plain text jobs) and the conversion filters (to charge for other file formats), to count pages or query the printer for pages printed. You cannot get away with using the simple output filter, since it cannot do accounting. See section Filters.

Generally, there are two ways to do accounting:

- *Periodic accounting* is the more common way, possibly because it is easier. Whenever someone prints a job, the filter logs the user, host, and number of pages to an accounting file. Every month, semester, year, or whatever time period you prefer, you collect the accounting files for the various printers, tally up the pages printed by users, and charge for usage. Then you truncate all the logging files, starting with a clean slate for the next period.
- *Timely accounting* is less common, probably because it is more difficult. This method has the filters charge users for printouts as soon as they use the printers. Like disk quotas, the accounting is immediate. You can prevent users from printing when their account goes in the red, and might provide a way for users to check and adjust their "print quotas". But this method requires some database code to track users and their quotas.

The **LPD** spooling system supports both methods easily: since you have to provide the filters (well, most of the time), you also have to provide the accounting code. But there is a bright side: you have enormous flexibility in your accounting methods. For example, you choose whether to use periodic or timely accounting. You choose what information to log: user names, host names, job types, pages printed, square footage of paper used, how long the job took to print, and so forth. And you do so by modifying the filters to save this information.

10.4.5.1 Quick and Dirty Printer Accounting

FreeBSD comes with two programs that can get you set up with simple periodic accounting right away. They are the text filter `lpf`, described in section lpf: a Text Filter, and pac(8), a program to gather and total entries from printer accounting files.

As mentioned in the section on filters (Filters), **LPD** starts the text and the conversion filters with the name of the accounting file to use on the filter command line. The filters can use this argument to know where to write an accounting file entry. The name of this file comes from the `af` capability in `/etc/printcap`, and if not specified as an absolute path, is relative to the spooling directory.

LPD starts `lpf` with page width and length arguments (from the `pw` and `pl` capabilities). The `lpf` filter uses these arguments to determine how much paper will be used. After sending the file to the printer, it then writes an accounting entry in the accounting file. The entries look like this:

```
2.00 rose:andy
3.00 rose:kelly
3.00 orchid:mary
5.00 orchid:mary
2.00 orchid:zhang
```

You should use a separate accounting file for each printer, as `lpf` has no file locking logic built into it, and two `lpf`s might corrupt each other's entries if they were to write to the same file at the same time. An easy way to insure a separate accounting file for each printer is to use `af=acct` in `/etc/printcap`. Then, each accounting file will be in the spooling directory for a printer, in a file named `acct`.

When you are ready to charge users for printouts, run the pac(8) program. Just change to the spooling directory for the printer you want to collect on and type `pac`. You will get a dollar-centric summary like the following:

```
  Login               pages/feet   runs    price
orchid:kelly               5.00    1   $  0.10
orchid:mary               31.00    3   $  0.62
orchid:zhang               9.00    1   $  0.18
rose:andy                  2.00    1   $  0.04
rose:kelly               177.00  104   $  3.54
rose:mary                 87.00   32   $  1.74
rose:root                 26.00   12   $  0.52

total                    337.00  154   $  6.74
```

These are the arguments pac(8) expects:

`-P`*`printer`*

> Which *`printer`* to summarize. This option works only if there is an absolute path in the `af` capability in `/etc/printcap`.

`-c`

> Sort the output by cost instead of alphabetically by user name.

`-m`

> Ignore host name in the accounting files. With this option, user `smith` on host `alpha` is the same user `smith` on host `gamma`. Without, they are different users.

`-p`*`price`*

Compute charges with *`price`* dollars per page or per foot instead of the price from the `pc` capability in `/etc/printcap`, or two cents (the default). You can specify *`price`* as a floating point number.

`-r`

Reverse the sort order.

`-s`

Make an accounting summary file and truncate the accounting file.

`name ...`

Print accounting information for the given user *`names`* only.

In the default summary that pac(8) produces, you see the number of pages printed by each user from various hosts. If, at your site, host does not matter (because users can use any host), run `pac -m`, to produce the following summary:

```
  Login               pages/feet   runs    price
andy                        2.00    1   $  0.04
kelly                     182.00  105   $  3.64
mary                      118.00   35   $  2.36
root                       26.00   12   $  0.52
zhang                       9.00    1   $  0.18

total                     337.00  154   $  6.74
```

To compute the dollar amount due, pac(8) uses the `pc` capability in the `/etc/printcap` file (default of 200, or 2 cents per page). Specify, in hundredths of cents, the price per page or per foot you want to charge for printouts in this capability. You can override this value when you run pac(8) with the `-p` option. The units for the `-p` option are in dollars, though, not hundredths of cents. For example,

```
# pac -p1.50
```

makes each page cost one dollar and fifty cents. You can really rake in the profits by using this option.

Finally, running `pac -s` will save the summary information in a summary accounting file, which is named the same as the printer's accounting file, but with `_sum` appended to the name. It then truncates the accounting file. When you run pac(8) again, it rereads the summary file to get starting totals, then adds information from the regular accounting file.

10.4.5.2 How Can You Count Pages Printed?

In order to perform even remotely accurate accounting, you need to be able to determine how much paper a job uses. This is the essential problem of printer accounting.

For plain text jobs, the problem is not that hard to solve: you count how many lines are in a job and compare it to how many lines per page your printer supports. Do not forget to take into account backspaces in the file which overprint lines, or long logical lines that wrap onto one or more additional physical lines.

The text filter `lpf` (introduced in lpf: a Text Filter) takes into account these things when it does accounting. If you are writing a text filter which needs to do accounting, you might want to examine `lpf`'s source code.

How do you handle other file formats, though?

Well, for DVI-to-LaserJet or DVI-to-PostScript conversion, you can have your filter parse the diagnostic output of `dvilj` or `dvips` and look to see how many pages were converted. You might be able to do similar things with other file formats and conversion programs.

But these methods suffer from the fact that the printer may not actually print all those pages. For example, it could jam, run out of toner, or explode—and the user would still get charged.

So, what can you do?

There is only one *sure* way to do *accurate* accounting. Get a printer that can tell you how much paper it uses, and attach it via a serial line or a network connection. Nearly all PostScript printers support this notion. Other makes and models do as well (networked Imagen laser printers, for example). Modify the filters for these printers to get the page usage after they print each job and have them log accounting information based on that value *only*. There is no line counting nor error-prone file examination required.

Of course, you can always be generous and make all printouts free.

10.5 Using Printers

This section tells you how to use printers you have set up with FreeBSD. Here is an overview of the user-level commands:

lpr(1)

: Print jobs

lpq(1)

: Check printer queues

lprm(1)

: Remove jobs from a printer's queue

There is also an administrative command, lpc(8), described in the section Administering Printers, used to control printers and their queues.

All three of the commands lpr(1), lprm(1), and lpq(1) accept an option `-P` *`printer-name`* to specify on which printer/queue to operate, as listed in the `/etc/printcap` file. This enables you to submit, remove, and check on jobs for various printers. If you do not use the `-P` option, then these commands use the printer specified in the `PRINTER` environment variable. Finally, if you do not have a `PRINTER` environment variable, these commands default to the printer named `lp`.

Hereafter, the terminology *default printer* means the printer named in the `PRINTER` environment variable, or the printer named `lp` when there is no `PRINTER` environment variable.

10.5.1 Printing Jobs

To print files, type:

```
% lpr filename ...
```

This prints each of the listed files to the default printer. If you list no files, lpr(1) reads data to print from standard input. For example, this command prints some important system files:

```
% lpr /etc/host.conf /etc/hosts.equiv
```

To select a specific printer, type:

```
% lpr -P printer-name filename ...
```

This example prints a long listing of the current directory to the printer named `rattan`:

```
% ls -l | lpr -P rattan
```

Because no files were listed for the lpr(1) command, `lpr` read the data to print from standard input, which was the output of the `ls -l` command.

The lpr(1) command can also accept a wide variety of options to control formatting, apply file conversions, generate multiple copies, and so forth. For more information, see the section Printing Options.

10.5.2 Checking Jobs

When you print with lpr(1), the data you wish to print is put together in a package called a “print job”, which is sent to the **LPD** spooling system. Each printer has a queue of jobs, and your job waits in that queue along with other jobs from yourself and from other users. The printer prints those jobs in a first-come, first-served order.

To display the queue for the default printer, type lpq(1). For a specific printer, use the `-P` option. For example, the command

```
% lpq -P bamboo
```

shows the queue for the printer named `bamboo`. Here is an example of the output of the `lpq` command:

```
bamboo is ready and printing
Rank   Owner    Job  Files                              Total Size
active kelly    9    /etc/host.conf, /etc/hosts.equiv   88 bytes
2nd    kelly    10   (standard input)                   1635 bytes
3rd    mary     11   ...                                78519 bytes
```

This shows three jobs in the queue for `bamboo`. The first job, submitted by user kelly, got assigned “job number” 9. Every job for a printer gets a unique job number. Most of the time you can ignore the job number, but you will need it if you want to cancel the job; see section Removing Jobs for details.

Job number nine consists of two files; multiple files given on the lpr(1) command line are treated as part of a single job. It is the currently active job (note the word `active` under the “Rank” column), which means the printer should be currently printing that job. The second job consists of data passed as the standard input to the lpr(1) command. The third job came from user `mary`; it is a much larger job. The pathname of the file she is trying to print is too long to fit, so the lpq(1) command just shows three dots.

The very first line of the output from lpq(1) is also useful: it tells what the printer is currently doing (or at least what **LPD** thinks the printer is doing).

The lpq(1) command also support a `-l` option to generate a detailed long listing. Here is an example of `lpq -l`:

```
waiting for bamboo to become ready (offline ?)
kelly: 1st                              [job 009rose]
       /etc/host.conf                    73 bytes
       /etc/hosts.equiv                  15 bytes

kelly: 2nd                              [job 010rose]
       (standard input)                  1635 bytes

mary: 3rd                               [job 011rose]
      /home/orchid/mary/research/venus/alpha-regio/mapping 78519 bytes
```

10.5.3 Removing Jobs

If you change your mind about printing a job, you can remove the job from the queue with the lprm(1) command. Often, you can even use lprm(1) to remove an active job, but some or all of the job might still get printed.

To remove a job from the default printer, first use lpq(1) to find the job number. Then type:

```
% lprm job-number
```

To remove the job from a specific printer, add the `-P` option. The following command removes job number 10 from the queue for the printer `bamboo`:

```
% lprm -P bamboo 10
```

The lprm(1) command has a few shortcuts:

lprm -

> Removes all jobs (for the default printer) belonging to you.

lprm *user*

> Removes all jobs (for the default printer) belonging to *user*. The superuser can remove other users' jobs; you can remove only your own jobs.

lprm

> With no job number, user name, or `-` appearing on the command line, lprm(1) removes the currently active job on the default printer, if it belongs to you. The superuser can remove any active job.

Just use the `-P` option with the above shortcuts to operate on a specific printer instead of the default. For example, the following command removes all jobs for the current user in the queue for the printer named `rattan`:

```
% lprm -P rattan -
```

> **Note:** If you are working in a networked environment, lprm(1) will let you remove jobs only from the host from which the jobs were submitted, even if the same printer is available from other hosts. The following command sequence demonstrates this:
>
> ```
> % lpr -P rattan myfile
> % rlogin orchid
> % lpq -P rattan
> ```

```
Rank   Owner     Job  Files                          Total Size
active seeyan    12    ...                           49123 bytes
2nd    kelly      13   myfile                         12 bytes
% lprm -P rattan 13
rose: Permission denied
% logout
% lprm -P rattan 13
dfA013rose dequeued
cfA013rose dequeued
```

10.5.4 Beyond Plain Text: Printing Options

The lpr(1) command supports a number of options that control formatting text, converting graphic and other file formats, producing multiple copies, handling of the job, and more. This section describes the options.

10.5.4.1 Formatting and Conversion Options

The following lpr(1) options control formatting of the files in the job. Use these options if the job does not contain plain text or if you want plain text formatted through the pr(1) utility.

For example, the following command prints a DVI file (from the TEX typesetting system) named *fish-report.dvi* to the printer named bamboo:

```
% lpr -P bamboo -d fish-report.dvi
```

These options apply to every file in the job, so you cannot mix (say) DVI and ditroff files together in a job. Instead, submit the files as separate jobs, using a different conversion option for each job.

> **Note:** All of these options except `-p` and `-T` require conversion filters installed for the destination printer. For example, the `-d` option requires the DVI conversion filter. Section Conversion Filters gives details.

`-c`

Print cifplot files.

`-d`

Print DVI files.

`-f`

Print FORTRAN text files.

`-g`

Print plot data.

`-i` *number*

Indent the output by *number* columns; if you omit *number*, indent by 8 columns. This option works only with certain conversion filters.

Note: Do not put any space between the `-i` and the number.

`-l`

Print literal text data, including control characters.

`-n`

Print ditroff (device independent troff) data.

`-p`

Format plain text with pr(1) before printing. See pr(1) for more information.

`-T` *title*

Use *title* on the pr(1) header instead of the file name. This option has effect only when used with the `-p` option.

`-t`

Print troff data.

`-v`

Print raster data.

Here is an example: this command prints a nicely formatted version of the ls(1) manual page on the default printer:

```
% zcat /usr/share/man/man1/ls.1.gz | troff -t -man | lpr -t
```

The zcat(1) command uncompresses the source of the ls(1) manual page and passes it to the troff(1) command, which formats that source and makes GNU troff output and passes it to lpr(1), which submits the job to the **LPD** spooler. Because we used the `-t` option to lpr(1), the spooler will convert the GNU troff output into a format the default printer can understand when it prints the job.

10.5.4.2 Job Handling Options

The following options to lpr(1) tell **LPD** to handle the job specially:

`-#` *copies*

Produce a number of *copies* of each file in the job instead of just one copy. An administrator may disable this option to reduce printer wear-and-tear and encourage photocopier usage. See section Restricting Multiple Copies.

This example prints three copies of *parser.c* followed by three copies of *parser.h* to the default printer:

```
% lpr -#3 parser.c parser.h
```

-m

Send mail after completing the print job. With this option, the **LPD** system will send mail to your account when it finishes handling your job. In its message, it will tell you if the job completed successfully or if there was an error, and (often) what the error was.

-s

Do not copy the files to the spooling directory, but make symbolic links to them instead.

If you are printing a large job, you probably want to use this option. It saves space in the spooling directory (your job might overflow the free space on the filesystem where the spooling directory resides). It saves time as well since **LPD** will not have to copy each and every byte of your job to the spooling directory.

There is a drawback, though: since **LPD** will refer to the original files directly, you cannot modify or remove them until they have been printed.

> **Note:** If you are printing to a remote printer, **LPD** will eventually have to copy files from the local host to the remote host, so the `-s` option will save space only on the local spooling directory, not the remote. It is still useful, though.

-r

Remove the files in the job after copying them to the spooling directory, or after printing them with the `-s` option. Be careful with this option!

10.5.4.3 Header Page Options

These options to lpr(1) adjust the text that normally appears on a job's header page. If header pages are suppressed for the destination printer, these options have no effect. See section Header Pages for information about setting up header pages.

-C *text*

Replace the hostname on the header page with *text*. The hostname is normally the name of the host from which the job was submitted.

-J *text*

Replace the job name on the header page with *text*. The job name is normally the name of the first file of the job, or `stdin` if you are printing standard input.

-h

Do not print any header page.

> **Note:** At some sites, this option may have no effect due to the way header pages are generated. See Header Pages for details.

10.5.5 Administering Printers

As an administrator for your printers, you have had to install, set up, and test them. Using the lpc(8) command, you can interact with your printers in yet more ways. With lpc(8), you can

- Start and stop the printers
- Enable and disable their queues
- Rearrange the order of the jobs in each queue.

First, a note about terminology: if a printer is *stopped*, it will not print anything in its queue. Users can still submit jobs, which will wait in the queue until the printer is *started* or the queue is cleared.

If a queue is *disabled*, no user (except `root`) can submit jobs for the printer. An *enabled* queue allows jobs to be submitted. A printer can be *started* for a disabled queue, in which case it will continue to print jobs in the queue until the queue is empty.

In general, you have to have `root` privileges to use the lpc(8) command. Ordinary users can use the lpc(8) command to get printer status and to restart a hung printer only.

Here is a summary of the lpc(8) commands. Most of the commands take a *`printer-name`* argument to tell on which printer to operate. You can use `all` for the *`printer-name`* to mean all printers listed in `/etc/printcap`.

`abort` *`printer-name`*

Cancel the current job and stop the printer. Users can still submit jobs if the queue is enabled.

`clean` *`printer-name`*

Remove old files from the printer's spooling directory. Occasionally, the files that make up a job are not properly removed by **LPD**, particularly if there have been errors during printing or a lot of administrative activity. This command finds files that do not belong in the spooling directory and removes them.

`disable` *`printer-name`*

Disable queuing of new jobs. If the printer is running, it will continue to print any jobs remaining in the queue. The superuser (`root`) can always submit jobs, even to a disabled queue.

This command is useful while you are testing a new printer or filter installation: disable the queue and submit jobs as `root`. Other users will not be able to submit jobs until you complete your testing and re-enable the queue with the `enable` command.

`down` *`printer-name message`*

Take a printer down. Equivalent to `disable` followed by `stop`. The *`message`* appears as the printer's status whenever a user checks the printer's queue with lpq(1) or status with `lpc status`.

`enable` *`printer-name`*

Enable the queue for a printer. Users can submit jobs but the printer will not print anything until it is started.

`help` *`command-name`*

Print help on the command *`command-name`*. With no *`command-name`*, print a summary of the commands available.

`restart` *`printer-name`*

Start the printer. Ordinary users can use this command if some extraordinary circumstance hangs **LPD**, but they cannot start a printer stopped with either the `stop` or `down` commands. The `restart` command is equivalent to `abort` followed by `start`.

`start` *`printer-name`*

Start the printer. The printer will print jobs in its queue.

`stop` *`printer-name`*

Stop the printer. The printer will finish the current job and will not print anything else in its queue. Even though the printer is stopped, users can still submit jobs to an enabled queue.

`topq` *`printer-name job-or-username`*

Rearrange the queue for *`printer-name`* by placing the jobs with the listed *`job`* numbers or the jobs belonging to *`username`* at the top of the queue. For this command, you cannot use `all` as the *`printer-name`*.

`up` *`printer-name`*

Bring a printer up; the opposite of the `down` command. Equivalent to `start` followed by `enable`.

lpc(8) accepts the above commands on the command line. If you do not enter any commands, lpc(8) enters an interactive mode, where you can enter commands until you type `exit`, `quit`, or end-of-file.

10.6 Alternatives to the Standard Spooler

If you have been reading straight through this manual, by now you have learned just about everything there is to know about the **LPD** spooling system that comes with FreeBSD. You can probably appreciate many of its shortcomings, which naturally leads to the question: “What other spooling systems are out there (and work with FreeBSD)?”

LPRng

LPRng, which purportedly means “LPR: the Next Generation” is a complete rewrite of PLP. Patrick Powell and Justin Mason (the principal maintainer of PLP) collaborated to make **LPRng**. The main site for **LPRng** is http://www.lprng.org/.

CUPS

CUPS, the Common UNIX Printing System, provides a portable printing layer for UNIX-based operating systems. It has been developed by Easy Software Products to promote a standard printing solution for all UNIX vendors and users.

CUPS uses the Internet Printing Protocol (IPP) as the basis for managing print jobs and queues. The Line Printer Daemon (LPD), Server Message Block (SMB), and AppSocket (aka JetDirect) protocols are also supported with reduced functionality. CUPS adds network printer browsing and PostScript Printer Description (PPD) based printing options to support real-world printing under UNIX.

The main site for **CUPS** is http://www.cups.org/.

HPLIP

HPLIP, the HP Linux Imaging and Printing system, is an HP-developed suite of programs that supports printing, scanning and fax facilities for HP appliances. This suite of programs utilizes the **CUPS** printing system as a backend for some of its printing features.

The main site for **HPLIP** is http://hplipopensource.com/hplip-web/index.html.

10.7 Troubleshooting

After performing the simple test with lptest(1), you might have gotten one of the following results instead of the correct printout:

It worked, after a while; or, it did not eject a full sheet.

The printer printed the above, but it sat for a while and did nothing. In fact, you might have needed to press a PRINT REMAINING or FORM FEED button on the printer to get any results to appear.

If this is the case, the printer was probably waiting to see if there was any more data for your job before it printed anything. To fix this problem, you can have the text filter send a FORM FEED character (or whatever is necessary) to the printer. This is usually sufficient to have the printer immediately print any text remaining in its internal buffer. It is also useful to make sure each print job ends on a full sheet, so the next job does not start somewhere on the middle of the last page of the previous job.

The following replacement for the shell script `/usr/local/libexec/if-simple` prints a form feed after it sends the job to the printer:

```
#!/bin/sh
#
# if-simple - Simple text input filter for lpd
# Installed in /usr/local/libexec/if-simple
#
# Simply copies stdin to stdout.  Ignores all filter arguments.
# Writes a form feed character (\f) after printing job.

/bin/cat && printf "\f" && exit 0
exit 2
```

It produced the “staircase effect.”

You got the following on paper:

```
!"#$%&'()*+,-./01234
               "#$%&'()*+,-./012345
                             #$%&'()*+,-./0123456
```

You have become another victim of the *staircase effect*, caused by conflicting interpretations of what characters should indicate a new line. UNIX style operating systems use a single character: ASCII code 10, the line feed (LF). MS-DOS, OS/2®, and others uses a pair of characters, ASCII code 10 *and* ASCII code 13 (the carriage return or CR). Many printers use the MS-DOS convention for representing new-lines.

When you print with FreeBSD, your text used just the line feed character. The printer, upon seeing a line feed character, advanced the paper one line, but maintained the same horizontal position on the page for the next character to print. That is what the carriage return is for: to move the location of the next character to print to the left edge of the paper.

Here is what FreeBSD wants your printer to do:

Printer received CR	Printer prints CR
Printer received LF	Printer prints CR + LF

Here are some ways to achieve this:

- Use the printer's configuration switches or control panel to alter its interpretation of these characters. Check your printer's manual to find out how to do this.

  > **Note:** If you boot your system into other operating systems besides FreeBSD, you may have to *reconfigure* the printer to use a an interpretation for CR and LF characters that those other operating systems use. You might prefer one of the other solutions, below.

- Have FreeBSD's serial line driver automatically convert LF to CR+LF. Of course, this works with printers on serial ports *only*. To enable this feature, use the `ms#` capability and set the `onlcr` mode in the `/etc/printcap` file for the printer.
- Send an *escape code* to the printer to have it temporarily treat LF characters differently. Consult your printer's manual for escape codes that your printer might support. When you find the proper escape code, modify the text filter to send the code first, then send the print job.

  Here is an example text filter for printers that understand the Hewlett-Packard PCL escape codes. This filter makes the printer treat LF characters as a LF and CR; then it sends the job; then it sends a form feed to eject the last page of the job. It should work with nearly all Hewlett Packard printers.

  ```
  #!/bin/sh
  #
  # hpif - Simple text input filter for lpd for HP-PCL based printers
  # Installed in /usr/local/libexec/hpif
  #
  # Simply copies stdin to stdout.  Ignores all filter arguments.
  # Tells printer to treat LF as CR+LF.  Ejects the page when done.

  printf "\033&k2G" && cat && printf "\033&l0H" && exit 0
  exit 2
  ```

  Here is an example `/etc/printcap` from a host called `orchid`. It has a single printer attached to its first parallel port, a Hewlett Packard LaserJet 3Si named `teak`. It is using the above script as its text filter:

  ```
  #
  ```

```
#  /etc/printcap for host orchid
#
teak|hp|laserjet|Hewlett Packard LaserJet 3Si:\
        :lp=/dev/lpt0:sh:sd=/var/spool/lpd/teak:mx#0:\
        :if=/usr/local/libexec/hpif:
```

It overprinted each line.

The printer never advanced a line. All of the lines of text were printed on top of each other on one line.

This problem is the "opposite" of the staircase effect, described above, and is much rarer. Somewhere, the LF characters that FreeBSD uses to end a line are being treated as CR characters to return the print location to the left edge of the paper, but not also down a line.

Use the printer's configuration switches or control panel to enforce the following interpretation of LF and CR characters:

Printer receives	Printer prints
CR	CR
LF	CR + LF

The printer lost characters.

While printing, the printer did not print a few characters in each line. The problem might have gotten worse as the printer ran, losing more and more characters.

The problem is that the printer cannot keep up with the speed at which the computer sends data over a serial line (this problem should not occur with printers on parallel ports). There are two ways to overcome the problem:

- If the printer supports XON/XOFF flow control, have FreeBSD use it by specifying the `ixon` mode in the `ms#` capability.
- If the printer supports the Request to Send / Clear to Send hardware handshake (commonly known as `RTS/CTS`), specify the `crtscts` mode in the `ms#` capability. Make sure the cable connecting the printer to the computer is correctly wired for hardware flow control.

It printed garbage.

The printer printed what appeared to be random garbage, but not the desired text.

This is usually another symptom of incorrect communications parameters with a serial printer. Double-check the bps rate in the `br` capability, and the parity setting in the `ms#` capability; make sure the printer is using the same settings as specified in the `/etc/printcap` file.

Nothing happened.

If nothing happened, the problem is probably within FreeBSD and not the hardware. Add the log file (`lf`) capability to the entry for the printer you are debugging in the `/etc/printcap` file. For example, here is the entry for `rattan`, with the `lf` capability:

```
rattan|line|diablo|lp|Diablo 630 Line Printer:\
```

```
:sh:sd=/var/spool/lpd/rattan:\
:lp=/dev/lpt0:\
:if=/usr/local/libexec/if-simple:\
:lf=/var/log/rattan.log
```

Then, try printing again. Check the log file (in our example, `/var/log/rattan.log`) to see any error messages that might appear. Based on the messages you see, try to correct the problem.

If you do not specify a `lf` capability, **LPD** uses `/dev/console` as a default.

Chapter 11 Linux® Binary Compatibility

Restructured and parts updated by Jim Mock. Originally contributed by Brian N. Handy and Rich Murphey.

11.1 Synopsis

FreeBSD provides binary compatibility with Linux, allowing users to install and run Linux binaries on a FreeBSD system. Many companies and developers develop only for Linux, and binary compatibility allows FreeBSD users to run about 90% of all Linux applications without modification. This includes productivity applications, games, and more. It has even been reported that, in some situations, Linux binaries perform better on FreeBSD than they do on Linux.

However, some Linux-specific operating system features are not supported under FreeBSD. For example, Linux binaries will not work on FreeBSD if they overly use i386 specific calls, such as enabling virtual 8086 mode.

After reading this chapter, you will know:

- How to enable Linux binary compatibility on a FreeBSD system.
- How to install additional Linux shared libraries.
- How to install Linux applications on a FreeBSD system.
- The implementation details of Linux compatibility in FreeBSD.

Before reading this chapter, you should:

- Know how to install additional third-party software.

11.2 Installation

Linux libararies are not installed on FreeBSD by default and Linux binary compatibility is not enabled by default. Linux libraries can be installed using the FreeBSD Ports Collection. Alternately, Linux libraries can be installed manually.

Using the Ports Collection is by far the easiest way to install Linux libraries:

```
# cd
/usr/ports/emulators/linux_base-f10 # make install distclean
```

Once the port is installed, enable Linux binary compatibility by loading the `linux` module. Type the following as `root`:

```
# kldload linux
```

In order for Linux compatibility to always be enabled at boot time, add the following line to `/etc/rc.conf`:

```
linux_enable="YES"
```

To verify that the module is loaded, use kldstat(8):

```
% kldstat
```

```
Id Refs Address    Size     Name
 1    2 0xc0100000 16bdb8   kernel
 7    1 0xc24db000 d000     linux.ko
```

Users who prefer to statically link Linux binary compatibility into the kernel should add `options COMPAT_LINUX` to the custom kernel configuration file. Compile and install the new kernel as described in Chapter 9.

11.2.1 Installing Libraries Manually

While using the Ports Collection is recommended, Linux libraries can be installed manually. The Linux shared libraries required by a program and the runtime linker should be copied to `/compat/linux`. Any shared libraries opened by Linux programs run under FreeBSD will look in this directory first. For example, if a Linux program loads `/lib/libc.so`, FreeBSD will first try to open `/compat/linux/lib/libc.so`, and if that does not exist, it will then try `/lib/libc.so`. Shared libraries should be installed to `/compat/linux/lib` rather than to the paths that the Linux `ld.so` reports.

Generally, one will need to look for the shared libraries that Linux binaries depend on only the first few times that a Linux program is installed on FreeBSD. After a while, there will be a sufficient set of Linux shared libraries on the system to be able to run newly imported Linux binaries without any extra work.

11.2.1.1 How to Install Additional Shared Libraries

If the `linux_base` port is installed and an application still complains about missing shared libraries, there are two methods `root` can use to determine which shared libraries the Linux binaries need.

If a Linux system is available, determine which shared libraries the application needs, and copy them to the FreeBSD system.

In this example, FTP was used to download the Linux binary of **Doom** on a Linux system . To check which shared libraries it needs, run `ldd linuxdoom`:

```
% ldd linuxdoom
libXt.so.3 (DLL Jump 3.1) => /usr/X11/lib/libXt.so.3.1.0
libX11.so.3 (DLL Jump 3.1) => /usr/X11/lib/libX11.so.3.1.0
libc.so.4 (DLL Jump 4.5pl26) => /lib/libc.so.4.6.29
```

Copy all the files in the last column into `/compat/linux` on the FreeBSD system, with the names in the first column as symbolic links pointing to them. This example will result in the following files on the FreeBSD system:

```
/compat/linux/usr/X11/lib/libXt.so.3.1.0
/compat/linux/usr/X11/lib/libXt.so.3 -> libXt.so.3.1.0
/compat/linux/usr/X11/lib/libX11.so.3.1.0
/compat/linux/usr/X11/lib/libX11.so.3 -> libX11.so.3.1.0
/compat/linux/lib/libc.so.4.6.29
/compat/linux/lib/libc.so.4 -> libc.so.4.6.29
```

> **Note:** If a Linux shared library already exists with a matching major revision number to the first column of the `ldd` output, it does not need to be copied to the file named in the last column, as the existing library should work. It is advisable to copy the shared library if it is a newer version, though. The old one can be removed, as long as the symbolic link points to the new one. For example, these libraries exist on the system:
>
> ```
> /compat/linux/lib/libc.so.4.6.27
> /compat/linux/lib/libc.so.4 -> libc.so.4.6.27
> ```

and a binary claims to require a later version according to the output of `ldd`:

```
libc.so.4 (DLL Jump 4.5pl26) -> libc.so.4.6.29
```

If it is only one or two versions out of date in the trailing digit, do not worry about copying `/lib/libc.so.4.6.29`, because the program should work fine with the slightly older version. However, it is safe to replace the `libc.so`:

```
/compat/linux/lib/libc.so.4.6.29
/compat/linux/lib/libc.so.4 -> libc.so.4.6.29
```

Note: The symbolic link mechanism is *only* needed for Linux binaries as the FreeBSD runtime linker takes care of looking for matching major revision numbers.

11.2.2 Installing Linux ELF Binaries

ELF binaries sometimes require an extra step of “branding”. If an unbranded ELF binary is executed, it will generate an error message like the following:

```
% ./my-linux-elf-binary
ELF binary type not known
Abort
```

To help the FreeBSD kernel distinguish between a FreeBSD ELF binary and a Linux binary, use brandelf(1):

```
% brandelf -t Linux my-linux-elf-binary
```

Since the GNU toolchain places the appropriate branding information into ELF binaries automatically, this step is usually not necessary.

11.2.3 Installing a Linux RPM Based Application

FreeBSD uses its own package database to track all software installed from the Ports Collection. However, the Linux RPM database is not supported.

In order to install a Linux RPM-based application, first install the `archivers/rpm2cpio` package or port. Once installed, `root` can use this command to install a `.rpm` as follows:

```
# cd /compat/linux
# rpm2cpio -q < /path/to/linux.archive.rpm | cpio -id
```

If necessary, `brandelf` the installed ELF binaries, but *not* the libraries. Note that this will prevent a clean uninstall.

11.2.4 Configuring the Hostname Resolver

If DNS does not work or this error appears:

```
resolv+: "bind" is an invalid keyword resolv+:
"hosts" is an invalid keyword
```

Configure `/compat/linux/etc/host.conf` as follows:

```
order hosts, bind
multi on
```

This order specifies that `/etc/hosts` is searched first and DNS is searched second. When `/compat/linux/etc/host.conf` does not exist, Linux applications use `/etc/host.conf` and complain about the incompatible FreeBSD syntax. Remove `bind` if a name server is not configured using `/etc/resolv.conf`.

11.3 Installing Mathematica®

Updated for Mathematica 5.X by Boris Hollas.

This section describes the process of installing the Linux version of **Mathematica 5.X** onto a FreeBSD system. **Mathematica** is a commercial, computational software program used in scientific, engineering, and mathematical fields. It is available from Wolfram Research (http://www.wolfram.com/mathematica/).

11.3.1 Running the Mathematica Installer

First, tell FreeBSD that **Mathematica**'s Linux binaries use the Linux Application Binary Interface ABI. The easiest way to do this is to set the default ELF brand to Linux for all unbranded binaries with the command:

```
# sysctl kern.fallback_elf_brand=3
```

FreeBSD will now assume that unbranded ELF binaries use the Linux ABI which should allow the installer to execute from the CDROM.

Copy the `MathInstaller` to the hard drive:

```
# mount /cdrom
# cp /cdrom/Unix/Installers/Linux/MathInstaller /localdir/
```

In this file, replace `/bin/sh` in the first line with `/compat/linux/bin/sh`. This ensures that the installer is executed by the Linux version of sh(1). Next, replace all occurrences of `Linux)` with `FreeBSD)` using a text editor or the script below in the next section. This tells the **Mathematica** installer, to treat FreeBSD as a Linux-like operating system. Invoking `MathInstaller` should now install **Mathematica**.

11.3.2 Modifying the Mathematica Executables

The shell scripts that **Mathematica** created during installation have to be modified before use. When using `/usr/local/bin` as the directory for the **Mathematica** executables, symlinks in this directory will point to files

called `math`, `mathematica`, `Mathematica`, and `MathKernel`. In each of these, replace `Linux)` with `FreeBSD)` using a text editor or the following shell script:

```
#!/bin/sh
cd /usr/local/bin
for i in math mathematica Mathematica MathKernel
  do sed 's/Linux)/FreeBSD)/g' $i > $i.tmp
  sed 's/\/bin\/sh/\/compat\/linux\/bin\/sh/g' $i.tmp > $i
  rm $i.tmp
  chmod a+x $i
done
```

11.3.3 Obtaining a Mathematica Password

When **Mathematica** is started for the first time, it will ask for a password. If a password had not yet been obtained from Wolfram Research, run `mathinfo` in the installation directory to obtain the "machine ID". This machine ID is based solely on the MAC address of the first Ethernet card, as the copy of **Mathematica** cannot run on different machines.

When registering with Wolfram Research, provide the "machine ID" and they will respond with a corresponding password consisting of groups of numbers.

11.3.4 Running the Mathematica Frontend over a Network

Mathematica uses some special fonts to display characters not present in any of the standard font sets. **Xorg** requires these fonts to be installed locally. This means that these fonts need to be copied from the CDROM or from a host with **Mathematica** installed to the local machine. These fonts are normally stored in `/cdrom/Unix/Files/SystemFiles/Fonts` on the CDROM, or `/usr/local/mathematica/SystemFiles/Fonts` on the hard drive. The actual fonts are in the subdirectories `Type1` and `X`. There are several ways to use them, as described below.

The first way is to copy the fonts into one of the existing font directories in `/usr/local/lib/X11/fonts` then running mkfontdir(1) within the directory containing the new fonts.

The second way to do this is to copy the directories to `/usr/local/lib/X11/fonts`:

```
# cd /usr/local/lib/X11/fonts
# mkdir X
# mkdir MathType1
# cd /cdrom/Unix/Files/SystemFiles/Fonts
# cp X/* /usr/local/lib/X11/fonts/X
# cp Type1/* /usr/local/lib/X11/fonts/MathType1
# cd /usr/local/lib/X11/fonts/X
# mkfontdir
# cd ../MathType1
# mkfontdir
```

Now add the new font directories to the font path:

```
# xset fp+ /usr/local/lib/X11/fonts/X
# xset fp+ /usr/local/lib/X11/fonts/MathType1
```

```
# xset fp rehash
```

When using the **Xorg** server, these font directories can be loaded automatically by adding them to `/etc/X11/xorg.conf`.

If `/usr/local/lib/X11/fonts/Type1` does not already exist, change the name of the `MathType1` directory in the example above to `Type1`.

11.4 Installing Maple™

Contributed by Aaron Kaplan. Thanks to Robert Getschmann.

Maple™ is a commercial mathematics program similar to **Mathematica**. This software must be purchased and licensed from Maplesoft (http://www.maplesoft.com/products/maple/). To install the Linux version of this software on FreeBSD, follow these steps.

1. Execute the `INSTALL` shell script from the product distribution. Choose the "RedHat" option when prompted by the installation program. A typical installation directory might be `/usr/local/maple`.
2. Copy the license to `/usr/local/maple/license/license.dat`.
3. Install the **FLEXlm** license manager by running the `INSTALL_LIC` install shell script that comes with **Maple**. Specify the primary hostname for the machine for the license server.
4. Patch `/usr/local/maple/bin/maple.system.type` with the following:

```
   ----- snip ------------------
*** maple.system.type.orig      Sun Jul  8 16:35:33 2001
--- maple.system.type   Sun Jul  8 16:35:51 2001
***************
*** 72,77 ****
--- 72,78 ----
          # the IBM RS/6000 AIX case
          MAPLE_BIN="bin.IBM_RISC_UNIX"
          ;;
+     "FreeBSD"|\
      "Linux")
          # the Linux/x86 case
        # We have two Linux implementations, one for Red Hat and
   ----- snip end of patch -----
```

   Note that no whitespace should be present after `"FreeBSD"|\`.

   This patch instructs **Maple** to recognize FreeBSD as a type of Linux system. The `bin/maple` shell script calls the `bin/maple.system.type` shell script which in turn calls `uname -a` to find out the operating system name. Depending on the OS name it will find out which binaries to use.

5. Start the license server.

   The following script, installed as `/usr/local/rtc/rc.d/lmgrd` is a convenient way to start up `lmgrd`:

```
   ----- snip ------------

#! /bin/sh
```

```
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
PATH=${PATH}:/usr/local/maple/bin:/usr/local/maple/FLEXlm/UNIX/LINUX
export PATH

LICENSE_FILE=/usr/local/maple/license/license.dat
LOG=/var/log/lmgrd.log

case "$1" in
start)
       lmgrd -c ${LICENSE_FILE} 2>> ${LOG} 1>&2
       echo -n " lmgrd"
       ;;
stop)
       lmgrd -c ${LICENSE_FILE} -x lmdown 2>> ${LOG} 1>&2
       ;;
*)
       echo "Usage: ‘basename $0‘ {start|stop}" 1>&2
       exit 64
       ;;
esac

exit 0
   ----- snip ------------
```

6. Test that **Maple** starts:

```
% cd /usr/local/maple/bin
% ./xmaple
```

Once everything is working, consider writing Maplesoft to let them know you would like a native FreeBSD version!

11.4.1 Common Pitfalls

- `lmgrd` is known to be picky about the license file and to dump core if there are any problems. A correct license file should look like this:

```
#
============================================================
# License File for UNIX Installations ("Pointer File")
# ============================================================
SERVER chillig ANY
#USE_SERVER
VENDOR maplelmg

FEATURE Maple maplelmg 2000.0831 permanent 1 XXXXXXXXXXX \
         PLATFORMS=i86_r ISSUER="Waterloo Maple Inc." \
         ISSUED=11-may-2000 NOTICE=" Technische Universitat Wien" \
         SN=XXXXXXXXX
```

Note: In this example, the serial number and key were replaced with `X`. `chillig` is the hostname.

Editing the license file works as long as the “FEATURE” line is not edited. That line is protected by the license key.

11.5 Installing MATLAB®

Contributed by Dan Pelleg.

This document describes the process of installing the Linux version of **MATLAB® version 6.5** onto a FreeBSD system. It works quite well, with the exception of the **Java Virtual Machine™** which is described further in Section 11.5.3.

The Linux version of **MATLAB** can be purchased and licensed from MathWorks (http://www.mathworks.com/products/matlab/). Consider letting the company know that you would like a native FreeBSD version of this software.

11.5.1 Installing MATLAB

To install **MATLAB**:

1. Become `root`, as recommended by the installation script. Insert the installation CD and mount it. To start the installation script type:

   ```
   # /compat/linux/bin/sh /cdrom/install
   ```

   Tip: The installer is graphical. If it is not able to open a display, type `setenv HOME ~USER`, where *USER* is the user who ran su(1).

2. When asked for the **MATLAB** root directory, type: **`/compat/linux/usr/local/matlab`**.

   Tip: For easier typing on the rest of the installation process, type this at the shell prompt: `set MATLAB=/compat/linux/usr/local/matlab`.

3. Edit the license file as instructed when obtaining the **MATLAB** license.

   Tip: This file can be prepared in advance using an editor, and copied to `$MATLAB/license.dat` before the installer asks to edit it.

4. Complete the installation process.

At this point the **MATLAB** installation is complete. The following steps apply “glue” to connect it to the FreeBSD system.

11.5.2 License Manager Startup

1. Create symlinks for the license manager scripts:

```
# ln -s $MATLAB/etc/lmboot /usr/local/etc/lmboot_TMW
# ln -s $MATLAB/etc/lmdown /usr/local/etc/lmdown_TMW
```

2. Create a startup file named `/usr/local/etc/rc.d/flexlm`. The example below is a modified version of the distributed `$MATLAB/etc/rc.lm.glnx86`. The changes are file locations and startup of the license manager under Linux emulation.

```
#!/bin/sh
case "$1" in
  start)
        if [ -f /usr/local/etc/lmboot_TMW ]; then
              /compat/linux/bin/sh /usr/local/etc/lmboot_TMW -u username && echo 'MATLAB_lmgrd'
        fi
        ;;
  stop)
       if [ -f /usr/local/etc/lmdown_TMW ]; then
            /compat/linux/bin/sh /usr/local/etc/lmdown_TMW  > /dev/null 2>&1
       fi
        ;;
  *)
       echo "Usage: $0 {start|stop}"
       exit 1
       ;;
esac

exit 0
```

   Important: The file must be made executable:

   ```
   # chmod +x /usr/local/etc/rc.d/flexlm
   ```

   Replace *username* with the name of a valid user on the system which is not `root`.

3. Start the license manager with the command:

```
# service flexlm start
```

11.5.3 Linking the Java Runtime Environment

Change the **Java** Runtime Environment (JRE) link to one working under FreeBSD:

```
# cd $MATLAB/sys/java/jre/glnx86/
# unlink jre; ln -s ./jre1.1.8 ./jre
```

11.5.4 Creating a MATLAB Startup Script

1. Place the following startup script in `/usr/local/bin/matlab`:

   ```
   #!/bin/sh
   /compat/linux/bin/sh /compat/linux/usr/local/matlab/bin/matlab "$@"
   ```

2. Then, type the command `chmod +x /usr/local/bin/matlab`.

Tip: Depending on the version of `emulators/linux_base`, running this script may result in errors. To avoid errors, edit `/compat/linux/usr/local/matlab/bin/matlab`, and change the line that says:

```
if [ ‘expr "$lscmd" : ’.*->.*’‘ -ne 0 ]; then
```

to this line:

```
if test -L $newbase; then
```

11.5.5 Creating a MATLAB Shutdown Script

The following is needed to solve a problem with MATLAB not exiting correctly.

1. Create `$MATLAB/toolbox/local/finish.m` containing the single line:

   ```
   ! $MATLAB/bin/finish.sh
   ```

   Note: The `$MATLAB` is literal.

   Tip: The same directory contains `finishsav.m` and `finishdlg.m`, which allow the workspace to be saved before quitting. If either file is used, insert the line above immediately after the `save` command.

2. Create `$MATLAB/bin/finish.sh` which contains the following:

   ```
   #!/compat/linux/bin/sh
   (sleep 5; killall -1 matlab_helper) &
   exit 0
   ```

3. Make the file executable:

   ```
   # chmod +x $MATLAB/bin/finish.sh
   ```

11.5.6 Using MATLAB

At this point, `matlab` is ready for use.

11.6 Installing Oracle®

Contributed by Marcel Moolenaar.

This document describes the process of installing **Oracle 8.0.5** and **Oracle 8.0.5.1 Enterprise Edition** for Linux onto a FreeBSD machine.

11.6.1 Installing the Linux Environment

Make sure `emulators/linux_base` has been installed from the Ports Collection.

To run the intelligent agent, install the Red Hat Tcl package: `tcl-8.0.3-20.i386.rpm`. The general command for installing RPMs with the `archivers/rpm` port is:

```
# rpm -i --ignoreos --root /compat/linux --dbpath /var/lib/rpm package
```

This command should not generate any errors.

11.6.2 Creating the Oracle Environment

Before installing **Oracle**, set up a proper environment. This section only describes how to install **Oracle** for Linux on FreeBSD, not what has been described in the **Oracle** installation guide.

11.6.2.1 Kernel Tuning

As described in the **Oracle** installation guide, the maximum size of shared memory needs to be set. Do not use `SHMMAX` under FreeBSD as it is calculated from `SHMMAXPGS` and `PGSIZE`. Therefore, define `SHMMAXPGS`. All other options can be used as described in the guide. For example:

```
options SHMMAXPGS=10000
options SHMMNI=100
options SHMSEG=10
options SEMMNS=200
options SEMMNI=70
options SEMMSL=61
```

Set these options to suit the intended use of **Oracle**.

Also, make sure the following options are in the kernel configuration file:

```
options SYSVSHM #SysV shared memory
options SYSVSEM #SysV semaphores
options SYSVMSG #SysV interprocess communication
```

11.6.2.2 Oracle Account

Create a user account to be used as the `oracle` account. Add `/compat/linux/bin/bash` to `/etc/shells` and set the shell for the `oracle` account to `/compat/linux/bin/bash`.

11.6.2.3 Environment

Besides the normal **Oracle** variables, such as `ORACLE_HOME` and `ORACLE_SID` set the following environment variables:

Variable	Value
`LD_LIBRARY_PATH`	`$ORACLE_HOME/lib`
`CLASSPATH`	`$ORACLE_HOME/jdbc/lib/classes111.zip`
`PATH`	`/compat/linux/bin /compat/linux/sbin /compat/linux/usr/bin /compat/linux/usr/sbin /bin /sbin /usr/bin /usr/sbin /usr/local/bin $ORACLE_HOME/bin`

It is advised to set all the environment variables in `~/.profile` as follows:

```
ORACLE_BASE=/oracle; export ORACLE_BASE
ORACLE_HOME=/oracle; export ORACLE_HOME
LD_LIBRARY_PATH=$ORACLE_HOME/lib
export LD_LIBRARY_PATH
ORACLE_SID=ORCL; export ORACLE_SID
ORACLE_TERM=386x; export ORACLE_TERM
CLASSPATH=$ORACLE_HOME/jdbc/lib/classes111.zip
export CLASSPATH
PATH=/compat/linux/bin:/compat/linux/sbin:/compat/linux/usr/bin
PATH=$PATH:/compat/linux/usr/sbin:/bin:/sbin:/usr/bin:/usr/sbin
PATH=$PATH:/usr/local/bin:$ORACLE_HOME/bin
export PATH
```

11.6.3 Installing Oracle

Before starting the installer, create a directory named `/var/tmp/.oracle` which is owned by the `oracle` user. The installation of **Oracle** should work without any problems. If errors are encountered, check the **Oracle** distribution and configuration. Once **Oracle** is installed, apply the patches described in the next two subsections.

A frequent error is that the TCP protocol adapter is not installed correctly. As a consequence, no TCP listeners can be started. The following actions help to solve this problem:

```
# cd $ORACLE_HOME/network/lib
# make -f ins_network.mk ntcontab.o
# cd $ORACLE_HOME/lib
# ar r libnetwork.a ntcontab.o
# cd $ORACLE_HOME/network/lib
# make -f ins_network.mk install
```

Do not forget to run `root.sh` again.

11.6.3.1 Patching `root.sh`

When installing **Oracle**, some actions, which need to be performed as `root`, are recorded in a shell script called `root.sh`. This script is found in `orainst`. Apply the following patch to `root.sh` so that it can find the FreeBSD location of `chown`. Alternatively, run the script under a Linux native shell.

```
*** orainst/root.sh.orig Tue Oct 6 21:57:33 1998
--- orainst/root.sh Mon Dec 28 15:58:53 1998
***************
*** 31,37 ****
# This is the default value for CHOWN
# It will redefined later in this script for those ports
# which have it conditionally defined in ss_install.h
! CHOWN=/bin/chown
#
# Define variables to be used in this script
--- 31,37 ----
# This is the default value for CHOWN
# It will redefined later in this script for those ports
# which have it conditionally defined in ss_install.h
! CHOWN=/usr/sbin/chown
#
# Define variables to be used in this script
```

If **Oracle** is not installed from CD, patch the source for `root.sh`. It is called `rthd.sh` and is located in `orainst` in the source tree.

11.6.3.2 Patching `genclntsh`

The script `genclntsh` is used to create a single shared client library when building the demos. Apply the following patch to comment out the definition of `PATH`:

```
*** bin/genclntsh.orig Wed Sep 30 07:37:19 1998
--- bin/genclntsh Tue Dec 22 15:36:49 1998
***************
*** 32,38 ****
#
# Explicit path to ensure that we're using the correct commands
#PATH=/usr/bin:/usr/ccs/bin export PATH
! PATH=/usr/local/bin:/bin:/usr/bin export PATH
#
# each product MUST provide a $PRODUCT/admin/shrept.lst
--- 32,38 ----
#
# Explicit path to ensure that we're using the correct commands
#PATH=/usr/bin:/usr/ccs/bin export PATH
! #PATH=/usr/local/bin:/bin:/usr/bin: export PATH
#
# each product MUST provide a $PRODUCT/admin/shrept.lst
```

11.6.4 Running Oracle

After following these instructions, **Oracle** should run as if it was running on Linux.

11.7 Advanced Topics

This section describes how Linux binary compatibility works and is based on an email written to FreeBSD chat mailing list (http://lists.FreeBSD.org/mailman/listinfo/freebsd-chat) by Terry Lambert <tlambert@primenet.com> (Message ID: <199906020108.SAA07001@usr09.primenet.com>).

FreeBSD has an abstraction called an “execution class loader”. This is a wedge into the execve(2) system call.

Historically, the UNIX loader examined the magic number (generally the first 4 or 8 bytes of the file) to see if it was a binary known to the system, and if so, invoked the binary loader.

If it was not the binary type for the system, the execve(2) call returned a failure, and the shell attempted to start executing it as shell commands. The assumption was a default of “whatever the current shell is”.

Later, a hack was made for sh(1) to examine the first two characters, and if they were `:\n`, it invoked the csh(1) shell instead.

FreeBSD has a list of loaders, instead of a single loader, with a fallback to the `#!` loader for running shell interpreters or shell scripts.

For the Linux ABI support, FreeBSD sees the magic number as an ELF binary. The ELF loader looks for a specialized *brand*, which is a comment section in the ELF image, and which is not present on SVR4/Solaris ELF binaries.

For Linux binaries to function, they must be *branded* as type `Linux` using brandelf(1):

```
# brandelf -t Linux file
```

When the ELF loader sees the `Linux` brand, the loader replaces a pointer in the `proc` structure. All system calls are indexed through this pointer. In addition, the process is flagged for special handling of the trap vector for the signal trampoline code, and several other (minor) fix-ups that are handled by the Linux kernel module.

The Linux system call vector contains, among other things, a list of `sysent[]` entries whose addresses reside in the kernel module.

When a system call is called by the Linux binary, the trap code dereferences the system call function pointer off the `proc` structure, and gets the Linux, not the FreeBSD, system call entry points.

Linux mode dynamically *reroots* lookups. This is, in effect, equivalent to the `union` option to file system mounts. First, an attempt is made to lookup the file in `/compat/linux/`*`original-path`*. If that fails, the lookup is done in */original-path*. This makes sure that binaries that require other binaries can run. For example, the Linux toolchain can all run under Linux ABI support. It also means that the Linux binaries can load and execute FreeBSD binaries, if there are no corresponding Linux binaries present, and that a uname(1) command can be placed in the `/compat/linux` directory tree to ensure that the Linux binaries can not tell they are not running on Linux.

In effect, there is a Linux kernel in the FreeBSD kernel. The various underlying functions that implement all of the services provided by the kernel are identical to both the FreeBSD system call table entries, and the Linux system call table entries: file system operations, virtual memory operations, signal delivery, and System V IPC. The only difference is that FreeBSD binaries get the FreeBSD *glue* functions, and Linux binaries get the Linux *glue* functions. The FreeBSD *glue* functions are statically linked into the kernel, and the Linux *glue* functions can be statically linked, or they can be accessed via a kernel module.

Technically, this is not really emulation, it is an ABI implementation. It is sometimes called “Linux emulation” because the implementation was done at a time when there was no other word to describe what was going on. Saying that FreeBSD ran Linux binaries was not true, since the code was not compiled in.

III. System Administration

The remaining chapters of the FreeBSD Handbook cover all aspects of FreeBSD system administration. Each chapter starts by describing what you will learn as a result of reading the chapter, and also details what you are expected to know before tackling the material.

These chapters are designed to be read when you need the information. You do not have to read them in any particular order, nor do you need to read all of them before you can begin using FreeBSD.

Chapter 12 Configuration and Tuning

Written by Chern Lee. Based on a tutorial written by Mike Smith. Also based on tuning(7) written by Matt Dillon.

12.1 Synopsis

One of the important aspects of FreeBSD is proper system configuration. This chapter explains much of the FreeBSD configuration process, including some of the parameters which can be set to tune a FreeBSD system.

After reading this chapter, you will know:

- How to efficiently work with file systems and swap partitions.
- The basics of `rc.conf` configuration and `/usr/local/etc/rc.d` startup scripts.
- How to configure and test a network card.
- How to configure virtual hosts on network devices.
- How to use the various configuration files in `/etc`.
- How to tune FreeBSD using sysctl(8) variables.
- How to tune disk performance and modify kernel limitations.

Before reading this chapter, you should:

- Understand UNIX and FreeBSD basics (Chapter 4).
- Be familiar with the basics of kernel configuration and compilation (Chapter 9).

12.2 Initial Configuration

12.2.1 Partition Layout

12.2.1.1 Base Partitions

When laying out file systems with bsdlabel(8) or sysinstall(8), remember that hard drives transfer data faster from the outer tracks to the inner. Thus, smaller and heavier-accessed file systems should be closer to the outside of the drive, while larger partitions like `/usr` should be placed toward the inner parts of the disk. It is a good idea to create partitions in an order similar to: `/`, swap, `/var`, and `/usr`.

The size of the `/var` partition reflects the intended machine's usage. This partition is used to hold mailboxes, log files, and printer spools. Mailboxes and log files can grow to unexpected sizes depending on the number of users and how long log files are kept. On average, most users rarely need more than about a gigabyte of free disk space in `/var`.

> **Note:** Sometimes, a lot of disk space is required in `/var/tmp`. When new software is installed with pkg_add(1), the packaging tools extract a temporary copy of the packages under `/var/tmp`. Large software packages, like **Firefox**, **OpenOffice** or **LibreOffice** may be tricky to install if there is not enough disk space under `/var/tmp`.

The /usr partition holds many of the files which support the system, including the FreeBSD Ports Collection and system source code. At least 2 gigabytes is recommended for this partition.

When selecting partition sizes, keep the space requirements in mind. Running out of space in one partition while barely using another can be a hassle.

> **Note:** The Auto-defaults partition sizer used by sysinstall(8) will sometimes select smaller than adequate /var and / partitions. Partition wisely and generously.

12.2.1.2 Swap Partition

As a rule of thumb, the swap partition should be about double the size of physical memory (RAM) as the kernel's virtual memory (VM) paging algorithms are tuned to perform best when the swap partition is at least two times the size of main memory. Systems with minimal RAM may perform better with more swap. Configuring too little swap can lead to inefficiencies in the VM page scanning code and might create issues later if more memory is added.

On larger systems with multiple SCSI disks or multiple IDE disks operating on different controllers, it is recommended that swap be configured on each drive, up to four drives. The swap partitions should be approximately the same size. The kernel can handle arbitrary sizes but internal data structures scale to 4 times the largest swap partition. Keeping the swap partitions near the same size will allow the kernel to optimally stripe swap space across disks. Large swap sizes are fine, even if swap is not used much. It might be easier to recover from a runaway program before being forced to reboot.

12.2.1.3 Why Partition?

Several users think a single large partition will be fine, but there are several reasons why this is a bad idea. First, each partition has different operational characteristics and separating them allows the file system to tune accordingly. For example, the root and /usr partitions are read-mostly, with few writes, while a lot of reads and writes could occur in /var and /var/tmp.

By properly partitioning a system, fragmentation introduced in the smaller write heavy partitions will not bleed over into the mostly read partitions. Keeping the write loaded partitions closer to the disk's edge will increase I/O performance in the partitions where it occurs the most. While I/O performance in the larger partitions may be needed, shifting them more toward the edge of the disk will not lead to a significant performance improvement over moving /var to the edge. Finally, there are safety concerns. A smaller, neater root partition which is mostly read-only has a greater chance of surviving a bad crash.

12.3 Core Configuration

The principal location for system configuration information is /etc/rc.conf. This file contains a wide range of configuration information and it is read at system startup to configure the system. It provides the configuration information for the rc* files.

The entries in `/etc/rc.conf` override the default settings in `/etc/defaults/rc.conf`. The file containing the default settings should not be edited. Instead, all system-specific changes should be made to `/etc/rc.conf`.

A number of strategies may be applied in clustered applications to separate site-wide configuration from system-specific configuration in order to keep administration overhead down. The recommended approach is to place system-specific configuration into `/etc/rc.conf.local`. For example:

- `/etc/rc.conf`:

  ```
  sshd_enable="YES"
  keyrate="fast"
  defaultrouter="10.1.1.254"
  ```

- `/etc/rc.conf.local`:

  ```
  hostname="node1.example.org"
  ifconfig_fxp0="inet 10.1.1.1/8"
  ```

Distribute `/etc/rc.conf` to every system using `rsync` or a similar program, while `/etc/rc.conf.local` remains unique.

Upgrading the system using sysinstall(8) or `make world` will not overwrite `/etc/rc.conf`, so system configuration information will not be lost.

> **Tip:** The configuration in `/etc/rc.conf` is parsed by sh(1). This allows system operators to create complex configuration scenarios. Refer to rc.conf(5) for further information on this topic.

12.4 Application Configuration

Typically, installed applications have their own configuration files and syntax. It is important that these files be kept separate from the base system, so that they may be easily located and managed by the package management tools.

Typically, these files are installed in `/usr/local/etc`. In the case where an application has a large number of configuration files, a subdirectory will be created to hold them.

Normally, when a port or package is installed, sample configuration files are also installed. These are usually identified with a suffix such as `.sample`. If there are no existing configuration files for the application, they can be created by copying the sample files.

For example, consider the contents of the directory `/usr/local/etc/apache`:

```
-rw-r--r--  1 root  wheel   2184 May 20  1998 access.conf
-rw-r--r--  1 root  wheel   2184 May 20  1998 access.conf.default
-rw-r--r--  1 root  wheel   9555 May 20  1998 httpd.conf
-rw-r--r--  1 root  wheel   9555 May 20  1998 httpd.conf.default
-rw-r--r--  1 root  wheel  12205 May 20  1998 magic
-rw-r--r--  1 root  wheel  12205 May 20  1998 magic.default
-rw-r--r--  1 root  wheel   2700 May 20  1998 mime.types
-rw-r--r--  1 root  wheel   2700 May 20  1998 mime.types.default
-rw-r--r--  1 root  wheel   7980 May 20  1998 srm.conf
-rw-r--r--  1 root  wheel   7933 May 20  1998 srm.conf.default
```

The file sizes show that only `srm.conf` has been changed. A later update of the **Apache** port would not overwrite this changed file.

12.5 Starting Services

Contributed by Tom Rhodes.

Many users install third party software on FreeBSD from the Ports Collection and require the installed services to be started upon system initialization. Services, such as `mail/postfix` or `www/apache22` are just two of the many software packages which may be started during system initialization. This section explains the procedures available for starting third party software.

In FreeBSD, most included services, such as cron(8), are started through the system start up scripts.

12.5.1 Extended Application Configuration

Now that FreeBSD includes `rc.d`, configuration of application startup is easier and provides more features. Using the key words discussed in Section 12.7, applications can be set to start after certain other services and extra flags can be passed through `/etc/rc.conf` in place of hard coded flags in the start up script. A basic script may look similar to the following:

```
#!/bin/sh
#
# PROVIDE: utility
# REQUIRE: DAEMON
# KEYWORD: shutdown

. /etc/rc.subr

name=utility
rcvar=utility_enable

command="/usr/local/sbin/utility"

load_rc_config $name

#
# DO NOT CHANGE THESE DEFAULT VALUES HERE
# SET THEM IN THE /etc/rc.conf FILE
#
utility_enable=${utility_enable-"NO"}
pidfile=${utility_pidfile-"/var/run/utility.pid"}

run_rc_command "$1"
```

This script will ensure that the provided `utility` will be started after the `DAEMON` pseudo-service. It also provides a method for setting and tracking the process ID (PID).

This application could then have the following line placed in `/etc/rc.conf`:

```
utility_enable="YES"
```

This method allows for easier manipulation of command line arguments, inclusion of the default functions provided in `/etc/rc.subr`, compatibility with rcorder(8), and provides for easier configuration via `rc.conf`.

12.5.2 Using Services to Start Services

Other services can be started using inetd(8). Working with inetd(8) and its configuration is described in depth in Section 30.2.

In some cases, it may make more sense to use cron(8) to start system services. This approach has a number of advantages as cron(8) runs these processes as the owner of the crontab(5). This allows regular users to start and maintain their own applications.

The `@reboot` feature of cron(8), may be used in place of the time specification. This causes the job to run when cron(8) is started, normally during system initialization.

12.6 Configuring cron(8)

Contributed by Tom Rhodes.

One of the most useful utilities in FreeBSD is cron(8). This utility runs in the background and regularly checks `/etc/crontab` for tasks to execute and searches `/var/cron/tabs` for custom crontab(5) files. These files store information about specific functions which cron(8) is supposed to perform at certain times.

Two different types of configuration files are used by cron(8): the system `crontab` and user `crontab`s. These formats only differ in the sixth field and later. In the system `crontab`, cron(8) runs the command as the user specified in the sixth field. In a user `crontab`, all commands run as the user who created the `crontab`, so the sixth field is the last field; this is an important security feature. The final field is always the command to run.

> **Note:** User crontabs allow individual users to schedule tasks without the need for `root` privileges. Commands in a user's crontab run with the permissions of the user who owns the crontab.
>
> The `root` user can have a user `crontab` just like any other user. The `root` user `crontab` is separate from the system `crontab`, `/etc/crontab`. Because the system `crontab` invokes the specified commands as `root`, there is usually no need to create a user `crontab` for `root`.

Here is a sample entry from `/etc/crontab`:

```
# /etc/crontab - root’s crontab for FreeBSD
#
# $FreeBSD: head/en_US.ISO8859-1/books/handbook/config/chapter.xml 42014 2013-06-23 22:37:08Z gjb
# ❶
#
SHELL=/bin/sh
PATH=/etc:/bin:/sbin:/usr/bin:/usr/sbin ❷
#
#minute        hour    mday    month   wday    who     command ❸
#
*/5    *       *       *       *       root    /usr/libexec/atrun ❹
```

❶ Like most FreeBSD configuration files, lines that begin with the # character are comments. A comment can be placed in the file as a reminder of what and why a desired action is performed. Comments cannot be on the same line as a command or else they will be interpreted as part of the command; they must be on a new line. Blank lines are ignored.

❷ The equals (=) character is used to define any environment settings. In this example, it is used to define the `SHELL` and `PATH`. If the `SHELL` is omitted, cron(8) will use the default of sh(1). If the `PATH` is omitted, no default will be used and file locations will need to be absolute.

❸ This line defines a total of seven fields: `minute`, `hour`, `mday`, `month`, `wday`, `who`, and `command`. These are almost all self explanatory. `minute` is the time in minutes when the specified command will be run. `hour` is the hour when the specified command will be run. `mday` stands for day of the month and `month` designates the month. The `wday` option stands for day of the week. These fields must be numeric values, representing the twenty-four hour clock, or a `*`, representing all values for that field. The `who` field only exists in the system crontab. This field specifies which user the command should be run as. The last field is the command to be executed.

❹ This last line defines the values discussed above. This example has a `*/5` listing,followed by several more `*` characters. These `*` characters mean "first-last", and can be interpreted as *every* time. In this example, atrun(8) is invoked by `root` every five minutes, regardless of the day or month.

Commands can have any number of flags passed to them; however, commands which extend to multiple lines need to be broken with the backslash "\" continuation character.

This is the basic setup for every crontab(5). However, field number six, which specifies the username, only exists in the system crontab(5). This field should be omitted for individual user crontab(5) files.

12.6.1 Installing a Crontab

> **Important:** Do not use the procedure described here to edit and install the system `crontab`, `/etc/crontab`. Instead, use an editor and cron(8) will notice that the file has changed and immediately begin using the updated version. See this FAQ entry (http://www.FreeBSD.org/doc/en_US.ISO8859-1/books/faq/admin.html#root-not-found-cron-errors) for more information.

To install a freshly written user crontab(5), use an editor to create and save a file in the proper format. Then, specify the file name with crontab(1):

```
% crontab crontab-file
```

In this example, `crontab-file` is the filename of a crontab(5) that was previously created.

To list installed crontab(5) files, pass `-l` to crontab(1).

Users who wish to begin their own `crontab` file from scratch, without the use of a template, can use `crontab -e`. This will invoke the default editor with an empty file. When this file is saved, it will be automatically installed by crontab(1).

In order to remove a user crontab(5) completely, use `crontab -r`.

12.7 Using rc(8) Under FreeBSD

Contributed by Tom Rhodes.

In 2002, FreeBSD integrated the NetBSD rc(8) system for system initialization. The files listed in `/etc/rc.d` provide basic services which can be controlled with the `start`, `stop`, and `restart` options to service(8). For instance, sshd(8) can be restarted with the following command:

```
# service sshd restart
```

This procedure can be used to start services on a running system. Services will be started automatically at boot time as specified in rc.conf(5). For example, to enable natd(8) at system startup, add the following line to `/etc/rc.conf`:

```
natd_enable="YES"
```

If a `natd_enable="NO"` line is already present, change the `NO` to `YES`. The rc(8) scripts will automatically load any dependent services during the next boot, as described below.

Since the rc(8) system is primarily intended to start and stop services at system startup and shutdown time, the `start`, `stop` and `restart` options will only perform their action if the appropriate `/etc/rc.conf` variable is set. For instance, `sshd restart` will only work if `sshd_enable` is set to `YES` in `/etc/rc.conf`. To `start`, `stop` or `restart` a service regardless of the settings in `/etc/rc.conf`, these commands should be prefixed with “one”. For instance, to restart sshd(8) regardless of the current `/etc/rc.conf` setting, execute the following command:

```
# service sshd onerestart
```

To check if a service is enabled in `/etc/rc.conf`, run the appropriate rc(8) script with `rcvar`. This example checks to see if sshd(8) is enabled in `/etc/rc.conf`:

```
# service sshd rcvar
# sshd
$sshd_enable=YES
```

> **Note:** The `# sshd` line is output from the above command, not a `root` console.

To determine whether or not a service is running, use `status`. For instance, to verify that sshd(8) is running:

```
# service sshd status
sshd is running as pid 433.
```

In some cases, it is also possible to `reload` a service. This attempts to send a signal to an individual service, forcing the service to reload its configuration files. In most cases, this means sending the service a `SIGHUP` signal. Support for this feature is not included for every service.

The rc(8) system is used for network services and it also contributes to most of the system initialization. For instance, when the `/etc/rc.d/bgfsck` script is executed, it prints out the following message:

```
Starting background file system checks in 60 seconds.
```

This script is used for background file system checks, which occur only during system initialization.

Many system services depend on other services to function properly. For example, yp(8) and other RPC-based services may fail to start until after the rpcbind(8) service has started. To resolve this issue, information about dependencies and other meta-data is included in the comments at the top of each startup script. The rcorder(8) program is used to parse these comments during system initialization to determine the order in which system services should be invoked to satisfy the dependencies.

The following key word must be included in all startup scripts as it is required by rc.subr(8) to "enable" the startup script:

- `PROVIDE`: Specifies the services this file provides.

The following key words may be included at the top of each startup script. They are not strictly necessary, but are useful as hints to rcorder(8):

- `REQUIRE`: Lists services which are required for this service. The script containing this key word will run *after* the specified services.
- `BEFORE`: Lists services which depend on this service. The script containing this key word will run *before* the specified services.

By carefully setting these keywords for each startup script, an administrator has a fine-grained level of control of the startup order of the scripts, without the need for "runlevels" used by some UNIX operating systems.

Additional information can be found in rc(8) and rc.subr(8). Refer to this article (http://www.FreeBSD.org/doc/en_US.ISO8859-1/articles/rc-scripting) for instructions on how to create custom rc(8) scripts.

12.8 Setting Up Network Interface Cards

Contributed by Marc Fonvieille.

Adding and configuring a network interface card (NIC) is a common task for any FreeBSD administrator.

12.8.1 Locating the Correct Driver

First, determine the model of the NIC and the chip it uses. FreeBSD supports a wide variety of NICs. Check the Hardware Compatibility List for the FreeBSD release to see if the NIC is supported.

If the NIC is supported, determine the name of the FreeBSD driver for the NIC. Refer to `/usr/src/sys/conf/NOTES` and `/usr/src/sys/`*`arch`*`/conf/NOTES` for the list of NIC drivers with some information about the supported chipsets. When in doubt, read the manual page of the driver as it will provide more information about the supported hardware and any known limitations of the driver.

The drivers for common NICs are already present in the `GENERIC` kernel, meaning the NIC should show up during boot. In this example, two NICs using the dc(4) driver are present on the system:

```
dc0: <82c169 PNIC 10/100BaseTX> port 0xa000-0xa0ff mem 0xd3800000-0xd38
000ff irq 15 at device 11.0 on pci0
miibus0: <MII bus> on dc0
bmtphy0: <BCM5201 10/100baseTX PHY> PHY 1 on miibus0
bmtphy0:  10baseT, 10baseT-FDX, 100baseTX, 100baseTX-FDX, auto
dc0: Ethernet address: 00:a0:cc:da:da:da
```

```
dc0: [ITHREAD]
dc1: <82c169 PNIC 10/100BaseTX> port 0x9800-0x98ff mem 0xd3000000-0xd30
000ff irq 11 at device 12.0 on pci0
miibus1: <MII bus> on dc1
bmtphy1: <BCM5201 10/100baseTX PHY> PHY 1 on miibus1
bmtphy1:  10baseT, 10baseT-FDX, 100baseTX, 100baseTX-FDX, auto
dc1: Ethernet address: 00:a0:cc:da:da:db
dc1: [ITHREAD]
```

If the driver for the NIC is not present in `GENERIC`, but a driver is available, the driver will need to be loaded before the NIC can be configured and used. This may be accomplished in one of two ways:

- The easiest way is to load a kernel module for the NIC using kldload(8). To also automatically load the driver at boot time, add the appropriate line to `/boot/loader.conf`. Not all NIC drivers are available as modules.
- Alternatively, statically compile support for the NIC into a custom kernel. Refer to `/usr/src/sys/conf/NOTES`, `/usr/src/sys/`*`arch`*`/conf/NOTES` and the manual page of the driver to determine which line to add to the custom kernel configuration file. For more information about recompiling the kernel, refer to Chapter 9. If the NIC was detected at boot, the kernel does not need to be recompiled.

12.8.1.1 Using Windows NDIS Drivers

Unfortunately, there are still many vendors that do not provide schematics for their drivers to the open source community because they regard such information as trade secrets. Consequently, the developers of FreeBSD and other operating systems are left with two choices: develop the drivers by a long and pain-staking process of reverse engineering or using the existing driver binaries available for Microsoft Windows platforms.

FreeBSD provides “native” support for the Network Driver Interface Specification (NDIS). It includes ndisgen(8) which can be used to convert a Windows XP driver into a format that can be used on FreeBSD. Because the ndis(4) driver uses a Windows XP binary, it only runs on i386 and amd64 systems. PCI, CardBus, PCMCIA, and USB devices are supported.

To use ndisgen(8), three things are needed:

1. FreeBSD kernel sources.
2. A Windows XP driver binary with a `.SYS` extension.
3. A Windows XP driver configuration file with a `.INF` extension.

Download the `.SYS` and `.INF` files for the specific NIC. Generally, these can be found on the driver CD or at the vendor’s website. The following examples use `W32DRIVER.SYS` and `W32DRIVER.INF`.

The driver bit width must match the version of FreeBSD. For FreeBSD/i386, use a Windows 32-bit driver. For FreeBSD/amd64, a Windows 64-bit driver is needed.

The next step is to compile the driver binary into a loadable kernel module. As `root`, use ndisgen(8):

```
# ndisgen /path/to/W32DRIVER.INF /path/to/W32DRIVER.SYS
```

This command is interactive and prompts for any extra information it requires. A new kernel module will be generated in the current directory. Use kldload(8) to load the new module:

```
# kldload ./W32DRIVER_SYS.ko
```

In addition to the generated kernel module, the `ndis.ko` and `if_ndis.ko` modules must be loaded. This should happen automatically when any module that depends on ndis(4) is loaded. If not, load them manually, using the following commands:

```
# kldload ndis
# kldload if_ndis
```

The first command loads the ndis(4) miniport driver wrapper and the second loads the generated NIC driver.

Check dmesg(8) to see if there were any load errors. If all went well, the output should be similar to the following:

```
ndis0: <Wireless-G PCI Adapter> mem 0xf4100000-0xf4101fff irq 3 at device 8.0 on pci1
ndis0: NDIS API version: 5.0
ndis0: Ethernet address: 0a:b1:2c:d3:4e:f5
ndis0: 11b rates: 1Mbps 2Mbps 5.5Mbps 11Mbps
ndis0: 11g rates: 6Mbps 9Mbps 12Mbps 18Mbps 36Mbps 48Mbps 54Mbps
```

From here, `ndis0` can be configured like any other NIC.

To configure the system to load the ndis(4) modules at boot time, copy the generated module, `W32DRIVER_SYS.ko`, to `/boot/modules`. Then, add the following line to `/boot/loader.conf`:

```
W32DRIVER_SYS_load="YES"
```

12.8.2 Configuring the Network Card

Once the right driver is loaded for the NIC, the card needs to be configured. It may have been configured at installation time by sysinstall(8).

To display the NIC configuration, enter the following command:

```
% ifconfig
dc0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
        options=80008<VLAN_MTU,LINKSTATE>
        ether 00:a0:cc:da:da:da
        inet 192.168.1.3 netmask 0xffffff00 broadcast 192.168.1.255
        media: Ethernet autoselect (100baseTX <full-duplex>)
        status: active
dc1: flags=8802<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
        options=80008<VLAN_MTU,LINKSTATE>
        ether 00:a0:cc:da:da:db
        inet 10.0.0.1 netmask 0xffffff00 broadcast 10.0.0.255
        media: Ethernet 10baseT/UTP
        status: no carrier
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> metric 0 mtu 16384
        options=3<RXCSUM,TXCSUM>
        inet6 fe80::1%lo0 prefixlen 64 scopeid 0x4
        inet6 ::1 prefixlen 128
        inet 127.0.0.1 netmask 0xff000000
        nd6 options=3<PERFORMNUD,ACCEPT_RTADV>
```

In this example, the following devices were displayed:

- `dc0`: The first Ethernet interface.
- `dc1`: The second Ethernet interface.
- `lo0`: The loopback device.

FreeBSD uses the driver name followed by the order in which the card is detected at boot to name the NIC. For example, `sis2` is the third NIC on the system using the sis(4) driver.

In this example, `dc0` is up and running. The key indicators are:

1. `UP` means that the card is configured and ready.
2. The card has an Internet (`inet`) address, `192.168.1.3`.
3. It has a valid subnet mask (`netmask`), where `0xffffff00` is the same as `255.255.255.0`.
4. It has a valid broadcast address, `192.168.1.255`.
5. The MAC address of the card (`ether`) is `00:a0:cc:da:da:da`.
6. The physical media selection is on autoselection mode (`media: Ethernet autoselect (100baseTX <full-duplex>)`). In this example, `dc1` is configured to run with `10baseT/UTP` media. For more information on available media types for a driver, refer to its manual page.
7. The status of the link (`status`) is `active`, indicating that the carrier signal is detected. For `dc1`, the `status: no carrier` status is normal when an Ethernet cable is not plugged into the card.

If the ifconfig(8) output had shown something similar to:

```
dc0: flags=8843<BROADCAST,SIMPLEX,MULTICAST> metric 0 mtu 1500
        options=80008<VLAN_MTU,LINKSTATE>
        ether 00:a0:cc:da:da:da
        media: Ethernet autoselect (100baseTX <full-duplex>)
        status: active
```

it would indicate the card has not been configured.

The card must be configured as `root`. The NIC configuration can be performed from the command line with ifconfig(8) but will not persist after a reboot unless the configuration is also added to `/etc/rc.conf`. Add a line for each NIC present on the system, as seen in this example:

```
ifconfig_dc0="inet 192.168.1.3 netmask 255.255.255.0"
ifconfig_dc1="inet 10.0.0.1 netmask 255.255.255.0 media 10baseT/UTP"
```

Replace `dc0` and `dc1` and the IP address information with the correct values for the system. Refer to the man page for the driver, ifconfig(8), and rc.conf(5) for more details about the allowed options and the syntax of `/etc/rc.conf`.

If the network was configured during installation, some entries for the NIC(s) may be already present. Double check `/etc/rc.conf` before adding any lines.

If the network is not using DNS, edit `/etc/hosts` to add the names and IP addresses of of the hosts on the LAN, if they are not already there. For more information, refer to hosts(5) and to `/usr/share/examples/etc/hosts`.

> **Note:** If there is no DHCP server and access to the Internet is needed, manually configure the default gateway and the nameserver:
>
> ```
> # echo 'defaultrouter="your_default_router"' >> /etc/rc.conf
> ```

```
# echo 'nameserver your_DNS_server' >> /etc/resolv.conf
```

12.8.3 Testing and Troubleshooting

Once the necessary changes to `/etc/rc.conf` are saved, a reboot can be used to test the network configuration and to verify that the system restarts without any configuration errors. Alternatively, apply the settings to the networking system with this command:

```
# service netif restart
```

Note: If a default gateway has been set in `/etc/rc.conf`, also issue this command:

```
# service routing restart
```

Once the networking system has been relaunched, test the NICs.

12.8.3.1 Testing the Ethernet Card

To verify that an Ethernet card is configured correctly, ping(8) the interface itself, and then ping(8) another machine on the LAN:

```
% ping -c5 192.168.1.3
PING 192.168.1.3 (192.168.1.3): 56 data bytes
64 bytes from 192.168.1.3: icmp_seq=0 ttl=64 time=0.082 ms
64 bytes from 192.168.1.3: icmp_seq=1 ttl=64 time=0.074 ms
64 bytes from 192.168.1.3: icmp_seq=2 ttl=64 time=0.076 ms
64 bytes from 192.168.1.3: icmp_seq=3 ttl=64 time=0.108 ms
64 bytes from 192.168.1.3: icmp_seq=4 ttl=64 time=0.076 ms

--- 192.168.1.3 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.074/0.083/0.108/0.013 ms

% ping -c5 192.168.1.2
PING 192.168.1.2 (192.168.1.2): 56 data bytes
64 bytes from 192.168.1.2: icmp_seq=0 ttl=64 time=0.726 ms
64 bytes from 192.168.1.2: icmp_seq=1 ttl=64 time=0.766 ms
64 bytes from 192.168.1.2: icmp_seq=2 ttl=64 time=0.700 ms
64 bytes from 192.168.1.2: icmp_seq=3 ttl=64 time=0.747 ms
64 bytes from 192.168.1.2: icmp_seq=4 ttl=64 time=0.704 ms

--- 192.168.1.2 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.700/0.729/0.766/0.025 ms
```

To test network resolution, use the host name instead of the IP address. If there is no DNS server on the network, `/etc/hosts` must first be configured.

12.8.3.2 Troubleshooting

When troubleshooting hardware and software configurations, check the simple things first. Is the network cable plugged in? Are the network services properly configured? Is the firewall configured correctly? Is the NIC supported by FreeBSD? Before sending a bug report, always check the Hardware Notes, update the version of FreeBSD to the latest STABLE version, check the mailing list archives, and search the Internet.

If the card works, yet performance is poor, read through tuning(7). Also, check the network configuration as incorrect network settings can cause slow connections.

Some users experience one or two `device timeout` messages, which is normal for some cards. If they continue, or are bothersome, determine if the device is conflicting with another device. Double check the cable connections. Consider trying another card.

To resolve `watchdog timeout` errors, first check the network cable. Many cards require a PCI slot which supports bus mastering. On some old motherboards, only one PCI slot allows it, usually slot 0. Check the NIC and the motherboard documentation to determine if that may be the problem.

`No route to host` messages occur if the system is unable to route a packet to the destination host. This can happen if no default route is specified or if a cable is unplugged. Check the output of `netstat -rn` and make sure there is a valid route to the host. If there is not, read Chapter 32.

`ping: sendto: Permission denied` error messages are often caused by a misconfigured firewall. If a firewall is enabled on FreeBSD but no rules have been defined, the default policy is to deny all traffic, even ping(8). Refer to Chapter 31 for more information.

Sometimes performance of the card is poor or below average. In these cases, try setting the media selection mode from `autoselect` to the correct media selection. While this works for most hardware, it may or may not resolve the issue. Again, check all the network settings, and refer to tuning(7).

12.9 Virtual Hosts

A common use of FreeBSD is virtual site hosting, where one server appears to the network as many servers. This is achieved by assigning multiple network addresses to a single interface.

A given network interface has one “real” address, and may have any number of “alias” addresses. These aliases are normally added by placing alias entries in `/etc/rc.conf`, as seen in this example:

```
ifconfig_fxp0_alias0="inet xxx.xxx.xxx.xxx netmask xxx.xxx.xxx.xxx"
```

Alias entries must start with `alias`*`0`* using a sequential number such as `alias0`, `alias1`, and so on. The configuration process will stop at the first missing number.

The calculation of alias netmasks is important. For a given interface, there must be one address which correctly represents the network’s netmask. Any other addresses which fall within this network must have a netmask of all `1`s, expressed as either `255.255.255.255` or `0xffffffff`.

For example, consider the case where the `fxp0` interface is connected to two networks: `10.1.1.0` with a netmask of `255.255.255.0` and `202.0.75.16` with a netmask of `255.255.255.240`. The system is to be configured to appear in the ranges `10.1.1.1` through `10.1.1.5` and `202.0.75.17` through `202.0.75.20`. Only the first address in a given network range should have a real netmask. All the rest (`10.1.1.2` through `10.1.1.5` and `202.0.75.18` through `202.0.75.20`) must be configured with a netmask of `255.255.255.255`.

The following /etc/rc.conf entries configure the adapter correctly for this scenario:

```
ifconfig_fxp0="inet 10.1.1.1 netmask 255.255.255.0"
ifconfig_fxp0_alias0="inet 10.1.1.2 netmask 255.255.255.255"
ifconfig_fxp0_alias1="inet 10.1.1.3 netmask 255.255.255.255"
ifconfig_fxp0_alias2="inet 10.1.1.4 netmask 255.255.255.255"
ifconfig_fxp0_alias3="inet 10.1.1.5 netmask 255.255.255.255"
ifconfig_fxp0_alias4="inet 202.0.75.17 netmask 255.255.255.240"
ifconfig_fxp0_alias5="inet 202.0.75.18 netmask 255.255.255.255"
ifconfig_fxp0_alias6="inet 202.0.75.19 netmask 255.255.255.255"
ifconfig_fxp0_alias7="inet 202.0.75.20 netmask 255.255.255.255"
```

12.10 Configuring the System Logger, syslogd

Contributed by Niclas Zeising.

System logging is an important aspect of system administration. It is used to detect hardware and software issues and errors in the system. It plays an important role in security auditing and incident response. System daemons without a controlling terminal usually log information to a system logging facility or other log file.

This section describes how to configure and use the FreeBSD system logger, syslogd(8), and how to perform log rotation and log management using newsyslog(8). Focus will be on setting up and using syslogd(8) on a local machine. For more advanced setups using a separate loghost, see Section 30.12.

12.10.1 Using syslogd

In the default FreeBSD configuration, syslogd(8) is started at boot. This is controlled by the variable syslogd_enable in /etc/rc.conf. There are numerous application arguments that affect the behavior of syslogd(8). To change them, use syslogd_flags in /etc/rc.conf. Refer to syslogd(8) for more information on the arguments, and rc.conf(5), Section 12.3 and Section 12.7 for more information about /etc/rc.conf and the rc(8) subsystem.

12.10.2 Configuring syslogd

The configuration file, by default /etc/syslog.conf, controls what syslogd(8) does with the log entries once they are received. There are several parameters to control the handling of incoming events, of which the most basic are *facility* and *level*. The facility describes which subsystem generated the message, such as the kernel or a daemon, and the level describes the severity of the event that occurred. This makes it possible to log the message to different log files, or discard it, depending on the facility and level. It is also possible to take action depending on the application that sent the message, and in the case of remote logging, the hostname of the machine generating the logging event.

The configuration file for syslogd(8) contains one line per action, and the syntax for each line is a selector field followed by an action field. The syntax of the selector field is *facility.level* which will match log messages from *facility* at level *level* or higher. It is also possible to add an optional comparison flag before the level to specify more precisely what is logged. Multiple selector fields can be used for the same action, and are separated with a semicolon (;). Using * will match everything. The action field denotes where to send the log message, such as to a file or remote log host. As an example, here is the default syslog.conf from FreeBSD:

```
# $FreeBSD$
```

```
#
#       Spaces ARE valid field separators in this file. However,
#       other *nix-like systems still insist on using tabs as field
#       separators. If you are sharing this file between systems, you
#       may want to use only tabs as field separators here.
#       Consult the syslog.conf(5) manpage.
*.err;kern.warning;auth.notice;mail.crit                /dev/console ❶
*.notice;authpriv.none;kern.debug;lpr.info;mail.crit;news.err   /var/log/messages
security.*                                              /var/log/security
auth.info;authpriv.info                                 /var/log/auth.log
mail.info                                               /var/log/maillog ❷
lpr.info                                                /var/log/lpd-errs
ftp.info                                                /var/log/xferlog
cron.*                                                  /var/log/cron
*.=debug                                                /var/log/debug.log ❸
*.emerg                                                 *
# uncomment this to log all writes to /dev/console to /var/log/console.log
#console.info                                           /var/log/console.log
# uncomment this to enable logging of all log messages to /var/log/all.log
# touch /var/log/all.log and chmod it to mode 600 before it will work
#*.*                                                    /var/log/all.log
# uncomment this to enable logging to a remote loghost named loghost
#*.*                                                    @loghost
# uncomment these if you're running inn
# news.crit                                             /var/log/news/news.crit
# news.err                                              /var/log/news/news.err
# news.notice                                           /var/log/news/news.notice
!ppp ❹
*.*                                                     /var/log/ppp.log
!*
```

❶ Match all messages with a level of `err` or higher, as well as `kern.warning`, `auth.notice` and `mail.crit`, and send these log messages to the console (`/dev/console`).

❷ Match all messages from the `mail` facility at level `info` or above, and log the messages to `/var/log/maillog`.

❸ This line uses a comparison flag, `=` to only match messages at level `debug`, and log them in `/var/log/debug.log`.

❹ Here is an example usage of a *program specification*. This makes the rules following it only valid for the program in the program specification. In this case, this and the following lines log all messages from ppp(8), but no other programs, to `/var/log/ppp.log`.

This example shows that there are plenty of levels and subsystems. The levels are, in order from most to least critical: `emerg`, `alert`, `crit`, `err`, `warning`, `notice`, `info`, and `debug`.

The facilities are, in no particular order: `auth`, `authpriv`, `console`, `cron`, `daemon`, `ftp`, `kern`, `lpr`, `mail`, `mark`, `news`, `security`, `syslog`, `user`, `uucp`, and `local0` through `local7`. Be aware that other operating systems might have different facilities.

With this knowledge, it is easy to add a new line to `/etc/syslog.conf` to log everything from the different daemons on level `notice` and higher to `/var/log/daemon.log`. Just add the following:

```
daemon.notice                                           /var/log/daemon.log
```

For more information about the different levels and facilities, refer to syslog(3) and syslogd(8). For more information about `/etc/syslog.conf`, its syntax, and more advanced usage examples, see syslog.conf(5) and Section 30.12.

12.10.3 Log Management and Rotation with `newsyslog`

Log files tend to grow quickly and accumulate steadily. This leads to the files being full of less immediately useful information while filling up the hard drive. Log management attempts to mitigate this. In FreeBSD, newsyslog(8) is used to manage log files. This program periodically rotates and compresses log files, and optionally creates missing log files and signals programs when log files are moved. The log files are not necessarily generated by syslogd(8) as newsyslog(8) works with any logs written from any program. While newsyslog(8) is normally run from cron(8), it is not a system daemon. In the default configuration, it is run every hour.

12.10.3.1 Configuring `newsyslog`

To know which actions to take, newsyslog(8) reads its configuration file, by default `/etc/newsyslog.conf`. This configuration file contains one line for each file that newsyslog(8) manages. Each line states the file owner, permissions, when to rotate that file, optional flags that affect log rotation, such as compression, and programs to signal when the log is rotated. Here is the default configuration in FreeBSD:

```
# configuration file for newsyslog
# $FreeBSD: head/en_US.ISO8859-1/books/handbook/config/chapter.xml 42014 2013-06-23 22:37:08Z gjb
#
# Entries which do not specify the '/pid_file' field will cause the
# syslogd process to be signalled when that log file is rotated.  This
# action is only appropriate for log files which are written to by the
# syslogd process (ie, files listed in /etc/syslog.conf).  If there
# is no process which needs to be signalled when a given log file is
# rotated, then the entry for that file should include the 'N' flag.
#
# The 'flags' field is one or more of the letters: BCDGJNUXZ or a '-'.
#
# Note: some sites will want to select more restrictive protections than the
# defaults.  In particular, it may be desirable to switch many of the 644
# entries to 640 or 600.  For example, some sites will consider the
# contents of maillog, messages, and lpd-errs to be confidential.  In the
# future, these defaults may change to more conservative ones.
#
# logfilename          [owner:group]    mode count size when  flags [/pid_file] [sig_num]
/var/log/all.log                        600  7     *    @T00  J
/var/log/amd.log                        644  7     100  *     J
/var/log/auth.log                       600  7     100  @0101T JC
/var/log/console.log                    600  5     100  *     J
/var/log/cron                           600  3     100  *     JC
/var/log/daily.log                      640  7     *    @T00  JN
/var/log/debug.log                      600  7     100  *     JC
/var/log/kerberos.log                   600  7     100  *     J
/var/log/lpd-errs                       644  7     100  *     JC
/var/log/maillog                        640  7     *    @T00  JC
/var/log/messages                       644  5     100  @0101T JC
/var/log/monthly.log                    640  12    *    $M1D0 JN
/var/log/pflog                          600  3     100  *     JB    /var/run/pflogd.pid
```

```
/var/log/ppp.log        root:network    640  3     100  *     JC
/var/log/security                       600  10    100  *     JC
/var/log/sendmail.st                    640  10    *    168   B
/var/log/utx.log                        644  3     *    @01T05 B
/var/log/weekly.log                     640  5     1    $W6D0 JN
/var/log/xferlog                        600  7     100  *     JC
```

Each line starts with the name of the file to be rotated, optionally followed by an owner and group for both rotated and newly created files. The `mode` field sets the permissions on the log file and `count` denotes how many rotated log files should be kept. The `size` and `when` fields tell newsyslog(8) when to rotate the file. A log file is rotated when either its size is larger than the `size` field, or when the time in the `when` filed has passed. `*` means that this field is ignored. The *`flags`* field gives newsyslog(8) further instructions, such as how to compress the rotated file or to create the log file if it is missing. The last two fields are optional, and specify the PID file of a process and a signal number to send to that process when the file is rotated. For more information on all fields, valid flags, and how to specify the rotation time, refer to newsyslog.conf(5). Since newsyslog(8) is run from cron(8), it can not rotate files more often than it is run from cron(8).

12.11 Configuration Files

12.11.1 `/etc` Layout

There are a number of directories in which configuration information is kept. These include:

`/etc`	Generic system-specific configuration information.
`/etc/defaults`	Default versions of system configuration files.
`/etc/mail`	Extra sendmail(8) configuration and other MTA configuration files.
`/etc/ppp`	Configuration for both user- and kernel-ppp programs.
`/etc/namedb`	Default location for named(8) data. Normally `named.conf` and zone files are stored here.
`/usr/local/etc`	Configuration files for installed applications. May contain per-application subdirectories.
`/usr/local/etc/rc.d`	rc(8) scripts for installed applications.
`/var/db`	Automatically generated system-specific database files, such as the package database and the locate(1) database.

12.11.2 Hostnames

12.11.2.1 `/etc/resolv.conf`

How a FreeBSD system accesses the Internet Domain Name System (DNS) is controlled by resolv.conf(5).

The most common entries to `/etc/resolv.conf` are:

`nameserver`	The IP address of a name server the resolver should query. The servers are queried in the order listed with a maximum of three.
`search`	Search list for hostname lookup. This is normally determined by the domain of the local hostname.
`domain`	The local domain name.

A typical `/etc/resolv.conf` looks like this:

```
search example.com
nameserver 147.11.1.11
nameserver 147.11.100.30
```

Note: Only one of the `search` and `domain` options should be used.

When using DHCP, dhclient(8) usually rewrites `/etc/resolv.conf` with information received from the DHCP server.

12.11.2.2 `/etc/hosts`

`/etc/hosts` is a simple text database which works in conjunction with DNS and NIS to provide host name to IP address mappings. Entries for local computers connected via a LAN can be added to this file for simplistic naming purposes instead of setting up a named(8) server. Additionally, `/etc/hosts` can be used to provide a local record of Internet names, reducing the need to query external DNS servers for commonly accessed names.

```
# $FreeBSD$
#
#
# Host Database
#
# This file should contain the addresses and aliases for local hosts that
# share this file.  Replace 'my.domain' below with the domainname of your
# machine.
#
# In the presence of the domain name service or NIS, this file may
# not be consulted at all; see /etc/nsswitch.conf for the resolution order.
#
#
::1                     localhost localhost.my.domain
127.0.0.1               localhost localhost.my.domain
#
# Imaginary network.
#10.0.0.2               myname.my.domain myname
#10.0.0.3               myfriend.my.domain myfriend
#
# According to RFC 1918, you can use the following IP networks for
# private nets which will never be connected to the Internet:
#
#      10.0.0.0        -   10.255.255.255
```

```
#      172.16.0.0      -   172.31.255.255
#      192.168.0.0     -   192.168.255.255
#
# In case you want to be able to connect to the Internet, you need
# real official assigned numbers.  Do not try to invent your own network
# numbers but instead get one from your network provider (if any) or
# from your regional registry (ARIN, APNIC, LACNIC, RIPE NCC, or AfriNIC.)
#
```

The format of `/etc/hosts` is as follows:

```
[Internet address] [official hostname] [alias1] [alias2] ...
```

For example:

```
10.0.0.1 myRealHostname.example.com myRealHostname foobar1 foobar2
```

Consult hosts(5) for more information.

12.12 Tuning with sysctl(8)

sysctl(8) is used to make changes to a running FreeBSD system. This includes many advanced options of the TCP/IP stack and virtual memory system that can dramatically improve performance for an experienced system administrator. Over five hundred system variables can be read and set using sysctl(8).

At its core, sysctl(8) serves two functions: to read and to modify system settings.

To view all readable variables:

```
% sysctl -a
```

To read a particular variable, specify its name:

```
% sysctl kern.maxproc
kern.maxproc: 1044
```

To set a particular variable, use the *`variable=value`* syntax:

```
# sysctl kern.maxfiles=5000
kern.maxfiles: 2088 -> 5000
```

Settings of sysctl variables are usually either strings, numbers, or booleans, where a a boolean is `1` for yes or `0` for no.

To automatically set some variables each time the machine boots, add them to `/etc/sysctl.conf`. For more information, refer to sysctl.conf(5) and Section 12.12.1.

12.12.1 `sysctl.conf`

The configuration file for sysctl(8), `/etc/sysctl.conf`, looks much like `/etc/rc.conf`. Values are set in a `variable=value` form. The specified values are set after the system goes into multi-user mode. Not all variables are settable in this mode.

For example, to turn off logging of fatal signal exits and prevent users from seeing processes started by other users, the following tunables can be set in `/etc/sysctl.conf`:

```
# Do not log fatal signal exits (e.g., sig 11)
kern.logsigexit=0

# Prevent users from seeing information about processes that
# are being run under another UID.
security.bsd.see_other_uids=0
```

12.12.2 sysctl(8) Read-only

Contributed by Tom Rhodes.

In some cases it may be desirable to modify read-only sysctl(8) values, which will require a reboot of the system.

For instance, on some laptop models the cardbus(4) device will not probe memory ranges and will fail with errors similar to:

```
cbb0: Could not map register memory
device_probe_and_attach: cbb0 attach returned 12
```

The fix requires the modification of a read-only sysctl(8) setting. Add `hw.pci.allow_unsupported_io_range=1` to `/boot/loader.conf` and reboot. Now cardbus(4) should work properly.

12.13 Tuning Disks

The following section will discuss various tuning mechanisms and options which may be applied to disk devices. In many cases, disks with mechanical parts, such as SCSI drives, will be the bottleneck driving down the overall system performance. While a solution is to install a drive without mechanical parts, such as a solid state drive, mechanical drives are not going away anytime in the near future. When tuning disks, it is advisable to utilize the features of the iostat(8) command to test various changes to the system. This command will allow the user to obtain valuable information on system IO.

12.13.1 Sysctl Variables

12.13.1.1 `vfs.vmiodirenable`

The `vfs.vmiodirenable` sysctl(8) variable may be set to either `0` (off) or `1` (on). It is set to `1` by default. This variable controls how directories are cached by the system. Most directories are small, using just a single fragment (typically 1 K) in the file system and typically 512 bytes in the buffer cache. With this variable turned off, the buffer cache will only cache a fixed number of directories, even if the system has a huge amount of memory. When turned on, this sysctl(8) allows the buffer cache to use the VM page cache to cache the directories, making all the memory available for caching directories. However, the minimum in-core memory used to cache a directory is the physical page size (typically 4 K) rather than 512 bytes. Keeping this option enabled is recommended if the system is running any services which manipulate large numbers of files. Such services can include web caches, large mail systems, and

news systems. Keeping this option on will generally not reduce performance, even with the wasted memory, but one should experiment to find out.

12.13.1.2 `vfs.write_behind`

The `vfs.write_behind` sysctl(8) variable defaults to `1` (on). This tells the file system to issue media writes as full clusters are collected, which typically occurs when writing large sequential files. This avoids saturating the buffer cache with dirty buffers when it would not benefit I/O performance. However, this may stall processes and under certain circumstances should be turned off.

12.13.1.3 `vfs.hirunningspace`

The `vfs.hirunningspace` sysctl(8) variable determines how much outstanding write I/O may be queued to disk controllers system-wide at any given instance. The default is usually sufficient, but on machines with many disks, try bumping it up to four or five *megabytes*. Setting too high a value which exceeds the buffer cache's write threshold can lead to bad clustering performance. Do not set this value arbitrarily high as higher write values may add latency to reads occurring at the same time.

There are various other buffer cache and VM page cache related sysctl(8) values. Modifying these values is not recommended as the VM system does a good job of automatically tuning itself.

12.13.1.4 `vm.swap_idle_enabled`

The `vm.swap_idle_enabled` sysctl(8) variable is useful in large multi-user systems with many active login users and lots of idle processes. Such systems tend to generate continuous pressure on free memory reserves. Turning this feature on and tweaking the swapout hysteresis (in idle seconds) via `vm.swap_idle_threshold1` and `vm.swap_idle_threshold2` depresses the priority of memory pages associated with idle processes more quickly then the normal pageout algorithm. This gives a helping hand to the pageout daemon. Only turn this option on if needed, because the tradeoff is essentially pre-page memory sooner rather than later which eats more swap and disk bandwidth. In a small system this option will have a determinable effect, but in a large system that is already doing moderate paging, this option allows the VM system to stage whole processes into and out of memory easily.

12.13.1.5 `hw.ata.wc`

Turning off IDE write caching reduces write bandwidth to IDE disks, but may sometimes be necessary due to data consistency issues introduced by hard drive vendors. The problem is that some IDE drives lie about when a write completes. With IDE write caching turned on, IDE hard drives write data to disk out of order and will sometimes delay writing some blocks indefinitely when under heavy disk load. A crash or power failure may cause serious file system corruption. Check the default on the system by observing the `hw.ata.wc` sysctl(8) variable. If IDE write caching is turned off, one can set this read-only variable to `1` in `/boot/loader.conf` in order to enable it at boot time.

For more information, refer to ata(4).

12.13.1.6 SCSI_DELAY (kern.cam.scsi_delay)

The SCSI_DELAY kernel configuration option may be used to reduce system boot times. The defaults are fairly high and can be responsible for 15 seconds of delay in the boot process. Reducing it to 5 seconds usually works with modern drives. The kern.cam.scsi_delay boot time tunable should be used. The tunable and kernel configuration option accept values in terms of *milliseconds* and *not seconds*.

12.13.2 Soft Updates

To fine-tune a file system, use tunefs(8). This program has many different options. To toggle Soft Updates on and off, use:

```
# tunefs -n enable /filesystem
# tunefs -n disable /filesystem
```

A file system cannot be modified with tunefs(8) while it is mounted. A good time to enable Soft Updates is before any partitions have been mounted, in single-user mode.

Soft Updates is recommended for UFS file systems as it drastically improves meta-data performance, mainly file creation and deletion, through the use of a memory cache. There are two downsides to Soft Updates to be aware of. First, Soft Updates guarantee file system consistency in the case of a crash, but could easily be several seconds or even a minute behind updating the physical disk. If the system crashes, unwritten data may be lost. Secondly, Soft Updates delay the freeing of file system blocks. If the root file system is almost full, performing a major update, such as make installworld, can cause the file system to run out of space and the update to fail.

12.13.2.1 More Details About Soft Updates

Meta-data updates are updates to non-content data like inodes or directories. There are two traditional approaches to writing a file system's meta-data back to disk.

Historically, the default behavior was to write out meta-data updates synchronously. If a directory changed, the system waited until the change was actually written to disk. The file data buffers (file contents) were passed through the buffer cache and backed up to disk later on asynchronously. The advantage of this implementation is that it operates safely. If there is a failure during an update, meta-data is always in a consistent state. A file is either created completely or not at all. If the data blocks of a file did not find their way out of the buffer cache onto the disk by the time of the crash, fsck(8) recognizes this and repairs the file system by setting the file length to 0. Additionally, the implementation is clear and simple. The disadvantage is that meta-data changes are slow. For example, rm -r touches all the files in a directory sequentially, but each directory change will be written synchronously to the disk. This includes updates to the directory itself, to the inode table, and possibly to indirect blocks allocated by the file. Similar considerations apply for unrolling large hierarchies using tar -x.

The second approach is to use asynchronous meta-data updates. This is the default for a UFS file system mounted with mount -o async. Since all meta-data updates are also passed through the buffer cache, they will be intermixed with the updates of the file content data. The advantage of this implementation is there is no need to wait until each meta-data update has been written to disk, so all operations which cause huge amounts of meta-data updates work much faster than in the synchronous case. This implementation is still clear and simple, so there is a low risk for bugs creeping into the code. The disadvantage is that there is no guarantee for a consistent state of the file system. If there is a failure during an operation that updated large amounts of meta-data, like a power failure or someone pressing the reset button, the file system will be left in an unpredictable state. There is no opportunity to examine the state of the file system when the system comes up again as the data blocks of a file could already have been written to the disk

while the updates of the inode table or the associated directory were not. It is impossible to implement a fsck(8) which is able to clean up the resulting chaos because the necessary information is not available on the disk. If the file system has been damaged beyond repair, the only choice is to reformat it and restore from backup.

The usual solution for this problem is to implement *dirty region logging*, which is also referred to as *journaling*. Meta-data updates are still written synchronously, but only into a small region of the disk. Later on, they are moved to their proper location. Because the logging area is a small, contiguous region on the disk, there are no long distances for the disk heads to move, even during heavy operations, so these operations are quicker than synchronous updates. Additionally, the complexity of the implementation is limited, so the risk of bugs being present is low. A disadvantage is that all meta-data is written twice, once into the logging region and once to the proper location, so performance "pessimization" might result. On the other hand, in case of a crash, all pending meta-data operations can be either quickly rolled back or completed from the logging area after the system comes up again, resulting in a fast file system startup.

Kirk McKusick, the developer of Berkeley FFS, solved this problem with Soft Updates. All pending meta-data updates are kept in memory and written out to disk in a sorted sequence ("ordered meta-data updates"). This has the effect that, in case of heavy meta-data operations, later updates to an item "catch" the earlier ones which are still in memory and have not already been written to disk. All operations are generally performed in memory before the update is written to disk and the data blocks are sorted according to their position so that they will not be on the disk ahead of their meta-data. If the system crashes, an implicit "log rewind" causes all operations which were not written to the disk appear as if they never happened. A consistent file system state is maintained that appears to be the one of 30 to 60 seconds earlier. The algorithm used guarantees that all resources in use are marked as such in their blocks and inodes. After a crash, the only resource allocation error that occurs is that resources are marked as "used" which are actually "free". fsck(8) recognizes this situation, and frees the resources that are no longer used. It is safe to ignore the dirty state of the file system after a crash by forcibly mounting it with `mount -f`. In order to free resources that may be unused, fsck(8) needs to be run at a later time. This is the idea behind the *background fsck(8)*: at system startup time, only a *snapshot* of the file system is recorded and fsck(8) is run afterwards. All file systems can then be mounted "dirty", so the system startup proceeds in multi-user mode. Then, background fsck(8) is scheduled for all file systems where this is required, to free resources that may be unused. File systems that do not use Soft Updates still need the usual foreground fsck(8).

The advantage is that meta-data operations are nearly as fast as asynchronous updates and are faster than *logging*, which has to write the meta-data twice. The disadvantages are the complexity of the code, a higher memory consumption, and some idiosyncrasies. After a crash, the state of the file system appears to be somewhat "older". In situations where the standard synchronous approach would have caused some zero-length files to remain after the fsck(8), these files do not exist at all with Soft Updates because neither the meta-data nor the file contents have been written to disk. Disk space is not released until the updates have been written to disk, which may take place some time after running rm(1). This may cause problems when installing large amounts of data on a file system that does not have enough free space to hold all the files twice.

12.14 Tuning Kernel Limits

12.14.1 File/Process Limits

12.14.1.1 `kern.maxfiles`

The `kern.maxfiles` sysctl(8) variable can be raised or lowered based upon system requirements. This variable indicates the maximum number of file descriptors on the system. When the file descriptor table is full, `file: table is full` will show up repeatedly in the system message buffer, which can be viewed using dmesg(8).

Each open file, socket, or fifo uses one file descriptor. A large-scale production server may easily require many thousands of file descriptors, depending on the kind and number of services running concurrently.

In older FreeBSD releases, the default value of `kern.maxfiles` is derived from `maxusers` in the kernel configuration file. `kern.maxfiles` grows proportionally to the value of `maxusers`. When compiling a custom kernel, consider setting this kernel configuration option according to the use of the system. From this number, the kernel is given most of its pre-defined limits. Even though a production machine may not have 256 concurrent users, the resources needed may be similar to a high-scale web server.

The read-only sysctl(8) variable `kern.maxusers` is automatically sized at boot based on the amount of memory available in the system, and may be determined at run-time by inspecting the value of `kern.maxusers`. Some systems require larger or smaller values of `kern.maxusers` and values of `64`, `128`, and `256` are not uncommon. Going above `256` is not recommended unless a huge number of file descriptors is needed. Many of the tunable values set to their defaults by `kern.maxusers` may be individually overridden at boot-time or run-time in `/boot/loader.conf`. Refer to loader.conf(5) and `/boot/defaults/loader.conf` for more details and some hints.

In older releases, the system will auto-tune `maxusers` if it is set to `0`. [1]. When setting this option, set `maxusers` to at least `4`, especially if the system runs **Xorg** or is used to compile software. The most important table set by `maxusers` is the maximum number of processes, which is set to `20 + 16 * maxusers`. If `maxusers` is set to `1`, there can only be `36` simultaneous processes, including the `18` or so that the system starts up at boot time and the `15` or so used by **Xorg**. Even a simple task like reading a manual page will start up nine processes to filter, decompress, and view it. Setting `maxusers` to `64` allows up to `1044` simultaneous processes, which should be enough for nearly all uses. If, however, the proc table full error is displayed when trying to start another program, or a server is running with a large number of simultaneous users, increase the number and rebuild.

> **Note:** `maxusers` does *not* limit the number of users which can log into the machine. It instead sets various table sizes to reasonable values considering the maximum number of users on the system and how many processes each user will be running.

12.14.1.2 `kern.ipc.somaxconn`

The `kern.ipc.somaxconn` sysctl(8) variable limits the size of the listen queue for accepting new `TCP` connections. The default value of `128` is typically too low for robust handling of new connections on a heavily loaded web server. For such environments, it is recommended to increase this value to `1024` or higher. A service such as sendmail(8), or **Apache** may itself limit the listen queue size, but will often have a directive in its configuration file to adjust the queue size. Large listen queues do a better job of avoiding Denial of Service (DoS) attacks.

12.14.2 Network Limits

The NMBCLUSTERS kernel configuration option dictates the amount of network Mbufs available to the system. A heavily-trafficked server with a low number of Mbufs will hinder performance. Each cluster represents approximately 2 K of memory, so a value of 1024 represents 2 megabytes of kernel memory reserved for network buffers. A simple calculation can be done to figure out how many are needed. A web server which maxes out at 1000 simultaneous connections where each connection uses a 6 K receive and 16 K send buffer, requires approximately 32 MB worth of network buffers to cover the web server. A good rule of thumb is to multiply by 2, so 2x32 MB / 2 KB = 64 MB / 2 kB = 32768. Values between 4096 and 32768 are recommended for machines with greater amounts of memory. Never specify an arbitrarily high value for this parameter as it could lead to a boot time crash. To observe network cluster usage, use -m with netstat(1).

The kern.ipc.nmbclusters loader tunable should be used to tune this at boot time. Only older versions of FreeBSD will require the use of the NMBCLUSTERS kernel config(8) option.

For busy servers that make extensive use of the sendfile(2) system call, it may be necessary to increase the number of sendfile(2) buffers via the NSFBUFS kernel configuration option or by setting its value in /boot/loader.conf (see loader(8) for details). A common indicator that this parameter needs to be adjusted is when processes are seen in the sfbufa state. The sysctl(8) variable kern.ipc.nsfbufs is read-only. This parameter nominally scales with kern.maxusers, however it may be necessary to tune accordingly.

> **Important:** Even though a socket has been marked as non-blocking, calling sendfile(2) on the non-blocking socket may result in the sendfile(2) call blocking until enough struct sf_buf's are made available.

12.14.2.1 net.inet.ip.portrange.*

The net.inet.ip.portrange.* sysctl(8) variables control the port number ranges automatically bound to TCP and UDP sockets. There are three ranges: a low range, a default range, and a high range. Most network programs use the default range which is controlled by net.inet.ip.portrange.first and net.inet.ip.portrange.last, which default to 1024 and 5000, respectively. Bound port ranges are used for outgoing connections and it is possible to run the system out of ports under certain circumstances. This most commonly occurs when running a heavily loaded web proxy. The port range is not an issue when running a server which handles mainly incoming connections, such as a web server, or has a limited number of outgoing connections, such as a mail relay. For situations where there is a shortage of ports, it is recommended to increase net.inet.ip.portrange.last modestly. A value of 10000, 20000 or 30000 may be reasonable. Consider firewall effects when changing the port range. Some firewalls may block large ranges of ports, usually low-numbered ports, and expect systems to use higher ranges of ports for outgoing connections. For this reason, it is not recommended that the value of net.inet.ip.portrange.first be lowered.

12.14.2.2 TCP Bandwidth Delay Product

TCP bandwidth delay product limiting can be enabled by setting the net.inet.tcp.inflight.enable sysctl(8) variable to 1. This instructs the system to attempt to calculate the bandwidth delay product for each connection and limit the amount of data queued to the network to just the amount required to maintain optimum throughput.

This feature is useful when serving data over modems, Gigabit Ethernet, high speed WAN links, or any other link with a high bandwidth delay product, especially when also using window scaling or when a large send window has been configured. When enabling this option, also set net.inet.tcp.inflight.debug to 0 to disable debugging. For production use, setting net.inet.tcp.inflight.min to at least 6144 may be beneficial. Setting high minimums

may effectively disable bandwidth limiting, depending on the link. The limiting feature reduces the amount of data built up in intermediate route and switch packet queues and reduces the amount of data built up in the local host's interface queue. With fewer queued packets, interactive connections, especially over slow modems, will operate with lower *Round Trip Times*. This feature only effects server side data transmission such as uploading. It has no effect on data reception or downloading.

Adjusting `net.inet.tcp.inflight.stab` is *not* recommended. This parameter defaults to `20`, representing 2 maximal packets added to the bandwidth delay product window calculation. The additional window is required to stabilize the algorithm and improve responsiveness to changing conditions, but it can also result in higher ping(8) times over slow links, though still much lower than without the inflight algorithm. In such cases, try reducing this parameter to `15`, `10`, or `5` and reducing `net.inet.tcp.inflight.min` to a value such as `3500` to get the desired effect. Reducing these parameters should be done as a last resort only.

12.14.3 Virtual Memory

12.14.3.1 `kern.maxvnodes`

A vnode is the internal representation of a file or directory. Increasing the number of vnodes available to the operating system reduces disk I/O. Normally, this is handled by the operating system and does not need to be changed. In some cases where disk I/O is a bottleneck and the system is running out of vnodes, this setting needs to be increased. The amount of inactive and free RAM will need to be taken into account.

To see the current number of vnodes in use:

```
# sysctl vfs.numvnodes
vfs.numvnodes: 91349
```

To see the maximum vnodes:

```
# sysctl kern.maxvnodes
kern.maxvnodes: 100000
```

If the current vnode usage is near the maximum, try increasing `kern.maxvnodes` by a value of `1000`. Keep an eye on the number of `vfs.numvnodes`. If it climbs up to the maximum again, `kern.maxvnodes` will need to be increased further. Otherwise, a shift in memory usage as reported by top(1) should be visible and more memory should be active.

12.15 Adding Swap Space

Sometimes a system requires more swap space. There are three ways to increase swap space: add a new hard drive, enable swap over `NFS`, or create a swap file on an existing partition.

For information on how to encrypt swap space, which options exist, and why it should be done, refer to Section 19.15.

12.15.1 Swap on a New or Existing Hard Drive

Adding a new hard drive for swap gives better performance than adding a partition on an existing drive. Setting up partitions and hard drives is explained in Section 19.3 while Section 12.2 discusses partition layouts and swap partition size considerations.

Use swapon(8) to add a swap partition to the system. For example:

```
# swapon /dev/ada1s1b
```

> **Warning:** It is possible to use any partition not currently mounted, even if it already contains data. Using swapon(8) on a partition that contains data will overwrite and destroy that data. Make sure that the partition to be added as swap is really the intended partition before running swapon(8).

To automatically add this swap partition on boot, add an entry to `/etc/fstab`:

```
/dev/ada1s1b    none    swap    sw      0       0
```

See fstab(5) for an explanation of the entries in `/etc/fstab`.

12.15.2 Swapping over `NFS`

Swapping over `NFS` is only recommended when there is no local hard disk to swap to. `NFS` swapping will be limited by the available network bandwidth and puts an additional burden on nfsd(8).

12.15.3 Swapfiles

To create a swap file, specify its size. The following example creates a 64MB file named `/usr/swap0`.

Example 12-1. Creating a Swapfile on FreeBSD

1. The `GENERIC` kernel already includes the memory disk driver (md(4)) required for this operation. When building a custom kernel, make sure to include the following line in the custom configuration file:
   ```
   device   md
   ```
   For information on building a custom kernel, refer to Chapter 9.
2. First, create the swapfile `/usr/swap0`:
   ```
   # dd if=/dev/zero of=/usr/swap0 bs=1024k count=64
   ```
3. Then, set proper permissions on `/usr/swap0`:
   ```
   # chmod 0600 /usr/swap0
   ```
4. Enable the swap file in `/etc/rc.conf`:
   ```
   swapfile="/usr/swap0"   # Set to name of swapfile if aux swapfile desired.
   ```
5. Reboot the machine or, to enable the swap file immediately, type:
   ```
   # mdconfig -a -t vnode -f /usr/swap0 -u 0 && swapon /dev/md0
   ```

12.16 Power and Resource Management

Written by Hiten Pandya and Tom Rhodes.

It is important to utilize hardware resources in an efficient manner. Before the Advanced Configuration and Power Interface (ACPI) was introduced, it was difficult and inflexible for operating systems to manage the power usage and thermal properties of a system. The hardware was managed by the BIOS and the user had less control and visibility into the power management settings. Some limited configurability was available via *Advanced Power Management (APM)*. Power and resource management allows the operating system to monitor system limits and to possibly provide an alert if the system temperature increases unexpectedly.

This section provides comprehensive information about ACPI. References will be provided for further reading.

12.16.1 What Is ACPI?

ACPI is a standard written by an alliance of vendors to provide a standard interface for hardware resources and power management. It is a key element in *Operating System-directed configuration and Power Management* as it provides more control and flexibility to the operating system. Modern systems "stretched" the limits of the current Plug and Play interfaces prior to the introduction of ACPI. ACPI is the direct successor to APM.

12.16.2 Shortcomings of Advanced Power Management

The APM facility controls the power usage of a system based on its activity. The APM BIOS is supplied by the vendor and is specific to the hardware platform. An APM driver in the operating system mediates access to the *APM Software Interface*, which allows management of power levels. APM should still be used for systems manufactured at or before the year 2000.

There are four major problems in APM. First, power management is done by the vendor-specific BIOS, separate from the operating system. For example, the user can set idle-time values for a hard drive in the APM BIOS so that, when exceeded, the BIOS spins down the hard drive without the consent of the operating system. Second, the APM logic is embedded in the BIOS, and it operates outside the scope of the operating system. This means that users can only fix problems in the APM BIOS by flashing a new one into the ROM, which is a dangerous procedure with the potential to leave the system in an unrecoverable state if it fails. Third, APM is a vendor-specific technology, meaning that there is a lot of duplication of efforts and bugs found in one vendor's BIOS may not be solved in others. Lastly, the APM BIOS did not have enough room to implement a sophisticated power policy or one that can adapt well to the purpose of the machine.

The *Plug and Play BIOS (PNPBIOS)* was unreliable in many situations. PNPBIOS is 16-bit technology, so the operating system has to use 16-bit emulation in order to interface with PNPBIOS methods.

The FreeBSD APM driver is documented in apm(4).

12.16.3 Configuring ACPI

The acpi(4) driver is loaded by default at start up by loader(8) and should *not* be compiled into the kernel. The reasoning is that modules are easier to work with and do not require a kernel rebuild. This has the advantage of making testing easier. Another reason is that starting ACPI after a system has been brought up often does not work well. If experiencing problems, ACPI can be disabled altogether. This driver should not and can not be unloaded because the system bus uses it for various hardware interactions. ACPI can be disabled by rebooting after setting `hint.acpi.0.disabled="1"` in `/boot/loader.conf` or by setting this variable at the loader(8) prompt.

Note: ACPI and APM cannot coexist and should be used separately. The last one to load will terminate if the driver notices the other is running.

ACPI can be used to put the system into a sleep mode with acpiconf(8), the `-s` flag, and a `1-5` option. Most users only need `1` (quick suspend to RAM) or `3` (suspend to RAM). Option `5` performs a soft-off which is the same action as:

```
# halt -p
```

Other options are available via sysctl(8). Refer to acpi(4) and acpiconf(8) for more information.

12.17 Using and Debugging FreeBSD ACPI

Written by Nate Lawson. With contributions from Peter Schultz and Tom Rhodes.

ACPI is a fundamentally new way of discovering devices, managing power usage, and providing standardized access to various hardware previously managed by the BIOS. Progress is being made toward ACPI working on all systems, but bugs in some motherboards' *ACPI Machine Language* (AML) bytecode, incompleteness in FreeBSD's kernel subsystems, and bugs in the Intel ACPI-CA interpreter continue to appear.

This section is intended to help users assist the FreeBSD ACPI maintainers in identifying the root cause of problems and in debugging and developing a solution.

12.17.1 Submitting Debugging Information

Note: Before submitting a problem, ensure the latest BIOS version is installed and, if available, the embedded controller firmware version.

When submitting a problem, send the following information to freebsd-acpi@FreeBSD.org (mailto:freebsd-acpi@FreeBSD.org):

- Description of the buggy behavior, including system type and model and anything that causes the bug to appear. Note as accurately as possible when the bug began occurring if it is new.
- The output of dmesg(8) after running `boot -v`, including any error messages generated by the bug.
- The dmesg(8) output from `boot -v` with ACPI disabled, if disabling it helps to fix the problem.
- Output from `sysctl hw.acpi`. This lists which features the system offers.
- The URL to a pasted version of the *ACPI Source Language* (ASL). Do *not* send the ASL directly to the list as it can be very large. Generate a copy of the ASL by running this command:

  ```
  # acpidump -dt > name-system.asl
  ```

  Substitute the login name for *`name`* and manufacturer/model for *`system`*. For example, use `njl-FooCo6000.asl`.

Most FreeBSD developers watch FreeBSD-CURRENT mailing list (http://lists.FreeBSD.org/mailman/listinfo/freebsd-current), but one should submit problems to freebsd-acpi

(http://lists.FreeBSD.org/mailman/listinfo/freebsd-acpi) to be sure it is seen. Be patient when waiting for a response. If the bug is not immediately apparent, submit a PR using send-pr(1). When entering a PR, include the same information as requested above. This helps developers to track the problem and resolve it. Do not send a PR without emailing freebsd-acpi (http://lists.FreeBSD.org/mailman/listinfo/freebsd-acpi) first as it is likely that the problem has been reported before.

12.17.2 Background

ACPI is present in all modern computers that conform to the ia32 (x86), ia64 (Itanium), and amd64 (AMD) architectures. The full standard has many features including CPU performance management, power planes control, thermal zones, various battery systems, embedded controllers, and bus enumeration. Most systems implement less than the full standard. For instance, a desktop system usually only implements bus enumeration while a laptop might have cooling and battery management support as well. Laptops also have suspend and resume, with their own associated complexity.

An ACPI-compliant system has various components. The BIOS and chipset vendors provide various fixed tables, such as FADT, in memory that specify things like the APIC map (used for SMP), config registers, and simple configuration values. Additionally, a bytecode table, the *Differentiated System Description Table* DSDT, specifies a tree-like name space of devices and methods.

The ACPI driver must parse the fixed tables, implement an interpreter for the bytecode, and modify device drivers and the kernel to accept information from the ACPI subsystem. For FreeBSD, Intel has provided an interpreter (ACPI-CA) that is shared with Linux and NetBSD. The path to the ACPI-CA source code is `src/sys/contrib/dev/acpica`. The glue code that allows ACPI-CA to work on FreeBSD is in `src/sys/dev/acpica/Osd`. Finally, drivers that implement various ACPI devices are found in `src/sys/dev/acpica`.

12.17.3 Common Problems

For ACPI to work correctly, all the parts have to work correctly. Here are some common problems, in order of frequency of appearance, and some possible workarounds or fixes.

12.17.3.1 Mouse Issues

In some cases, resuming from a suspend operation will cause the mouse to fail. A known work around is to add `hint.psm.0.flags="0x3000"` to `/boot/loader.conf`. If this does not work, consider sending a bug report using send-pr(1).

12.17.3.2 Suspend/Resume

ACPI has three suspend to RAM (STR) states, `S1`-`S3`, and one suspend to disk state (`STD`), called `S4`. `S5` is “soft off” and is the normal state the system is in when plugged in but not powered up. `S4` can be implemented in two separate ways. `S4`BIOS is a BIOS-assisted suspend to disk. `S4`OS is implemented entirely by the operating system.

Start by checking `sysctl hw.acpi` for the suspend-related items. Here are the results for a Thinkpad:

```
hw.acpi.supported_sleep_state: S3 S4 S5
hw.acpi.s4bios: 0
```

Use `acpiconf -s` to test `S3`, `S4`OS, and `S5`. An `s4bios` of one (`1`), indicates `S4`BIOS support instead of `S4` OS.

When testing suspend/resume, start with `S1`, if supported. This state is most likely to work since it does not require much driver support. No one has implemented `S2`, which is similar to `S1`. Next, try `S3`. This is the deepest STR state and requires a lot of driver support to properly reinitialize the hardware. If there are problems resuming, email freebsd-acpi (http://lists.FreeBSD.org/mailman/listinfo/freebsd-acpi). However, the problem may not be resolved quickly since due to the amount of drivers and hardware that need more testing and work.

A common problem with suspend/resume is that many device drivers do not save, restore, or reinitialize their firmware, registers, or device memory properly. As a first attempt at debugging the problem, try:

```
# sysctl debug.bootverbose=1
# sysctl debug.acpi.suspend_bounce=1
# acpiconf -s 3
```

This test emulates the suspend/resume cycle of all device drivers without actually going into `S3` state. In some cases, problems such as losing firmware state, device watchdog time out, and retrying forever, can be captured with this method. Note that the system will not really enter `S3` state, which means devices may not lose power, and many will work fine even if suspend/resume methods are totally missing, unlike real `S3` state.

Harder cases require additional hardware, such as a serial port and cable for debugging through a serial console, a Firewire port and cable for using dcons(4), and kernel debugging skills.

To help isolate the problem, remove as many drivers from the kernel as possible. If it works, narrow down which driver is the problem by loading drivers until it fails again. Typically, binary drivers like `nvidia.ko`, display drivers, and USB will have the most problems while Ethernet interfaces usually work fine. If drivers can be properly loaded and unloaded, automate this by putting the appropriate commands in `/etc/rc.suspend` and `/etc/rc.resume`. Try setting `hw.acpi.reset_video` to `0` if the display is messed up after resume. Try setting longer or shorter values for `hw.acpi.sleep_delay` to see if that helps.

Try loading a recent Linux distribution to see if suspend/resume works on the same hardware. If it works on Linux, it is likely a FreeBSD driver problem. Narrowing down which driver causes the problem will assist developers in fixing the problem. Since the ACPI maintainers rarely maintain other drivers, such as sound or ATA, any driver problems should also be posted to the freebsd-current (http://lists.FreeBSD.org/mailman/listinfo/freebsd-current) list and mailed to the driver maintainer. Advanced users can include debugging printf(3)s in a problematic driver to track down where in its resume function it hangs.

Finally, try disabling ACPI and enabling APM instead. If suspend/resume works with APM, stick with APM, especially on older hardware (pre-2000). It took vendors a while to get ACPI support correct and older hardware is more likely to have BIOS problems with ACPI.

12.17.3.3 System Hangs

Most system hangs are a result of lost interrupts or an interrupt storm. Chipsets may have problems based on boot, how the BIOS configures interrupts before correctness of the APIC (MADT) table, and routing of the *System Control Interrupt* (SCI).

Interrupt storms can be distinguished from lost interrupts by checking the output of `vmstat -i` and looking at the line that has `acpi0`. If the counter is increasing at more than a couple per second, there is an interrupt storm. If the system appears hung, try breaking to DDB (**CTRL**+**ALT**+**ESC** on console) and type `show interrupts`.

When dealing with interrupt problems, try disabling APIC support with `hint.apic.0.disabled="1"` in `/boot/loader.conf`.

12.17.3.4 Panics

Panics are relatively rare for ACPI and are the top priority to be fixed. The first step is to isolate the steps to reproduce the panic, if possible, and get a backtrace. Follow the advice for enabling `options DDB` and setting up a serial console in Section 27.6.5.3 or setting up a dump(8) partition. To get a backtrace in DDB, use `tr`. When handwriting the backtrace, get at least the last five and the top five lines in the trace.

Then, try to isolate the problem by booting with ACPI disabled. If that works, isolate the ACPI subsystem by using various values of `debug.acpi.disable`. See acpi(4) for some examples.

12.17.3.5 System Powers Up After Suspend or Shutdown

First, try setting `hw.acpi.disable_on_poweroff="0"` in loader.conf(5). This keeps ACPI from disabling various events during the shutdown process. Some systems need this value set to `1` (the default) for the same reason. This usually fixes the problem of a system powering up spontaneously after a suspend or poweroff.

12.17.3.6 Other Problems

For other problems with ACPI, such as it not working with a docking station or devices not being detected, email a description to freebsd-acpi (http://lists.FreeBSD.org/mailman/listinfo/freebsd-acpi). Some issues may be related to unfinished parts of the ACPI subsystem which might take a while to be implemented. Be patient and prepared to test patches.

12.17.4 ASL, acpidump(8), and IASL

Some BIOS vendors provide incorrect or buggy bytecode. This is usually manifested by kernel console messages like this:

```
ACPI-1287: *** Error: Method execution failed [\\_SB_.PCI0.LPC0.FIGD._STA] \\
(Node 0xc3f6d160), AE_NOT_FOUND
```

Often, these problems may be resolved by updating the BIOS to the latest revision. Most console messages are harmless, but if there are other problems like the battery status is not working, these messages are a good place to start looking for problems. The bytecode, known as AML, is compiled from a source language called ASL. The AML is found in the table known as the DSDT. To get a copy of the system's ASL, use acpidump(8). Include both `-t`, to show the contents of the fixed tables, and `-d`, to disassemble the AML. Refer to Section 12.17.1 for an example syntax.

The simplest first check is to recompile the ASL to check for errors. Warnings can usually be ignored, but errors are bugs that will usually prevent ACPI from working correctly. To recompile the ASL, issue the following command:

```
# iasl your.asl
```

12.17.5 Fixing the ASL

The goal of FreeBSD is for everyone to have working ACPI without any user intervention. At this point, workarounds are still being developed for common mistakes made by BIOS vendors. The Microsoft interpreter (`acpi.sys` and `acpiec.sys`) does not strictly check for adherence to the standard, and thus many BIOS vendors

who only test ACPI under Windows never fix their ASL. FreeBSD developers continue to identify and document which non-standard behavior is allowed by Microsoft's interpreter and replicate it so that FreeBSD can work without forcing users to fix the ASL. As a workaround, and to help identify behavior, fix the ASL manually. If this works, send a diff(1) of the old and new ASL so developers can possibly work around the buggy behavior in ACPI-CA.

Here is a list of common error messages, their cause, and how to fix them:

12.17.5.1 Operating System Dependencies

Some AML versions assume the user is running Windows. To override this, set `hw.acpi.osname="Windows 2001"` in `/boot/loader.conf`, using the strings in the ASL.

12.17.5.2 Missing Return Statements

Some methods do not explicitly return a value as the standard requires. While ACPI-CA does not handle this, FreeBSD has a workaround that allows it to return the value implicitly. Explicit return statements can be added where required if the value which should be returned is known. To force iasl(8) to compile the ASL, use the `-f` flag.

12.17.5.3 Overriding the Default AML

After customizing `your.asl`, compile it with this command:

```
# iasl your.asl
```

Adding the `-f` flag forces creation of the AML, even if there are errors during compilation. Some errors, such as missing return statements, are automatically worked around by the interpreter.

The default output filename for iasl(8) is `DSDT.aml`. Load this file instead of the BIOS's buggy copy, which is still present in flash memory, by editing `/boot/loader.conf` as follows:

```
acpi_dsdt_load="YES"
acpi_dsdt_name="/boot/DSDT.aml"
```

Be sure to copy `DSDT.aml` to `/boot`.

12.17.6 Getting Debugging Output from ACPI

The ACPI driver has a flexible debugging facility. A set of subsystems and the level of verbosity can be specified. The subsystems to debug are specified as "layers" and are broken down into ACPI-CA components (ACPI_ALL_COMPONENTS) and ACPI hardware support (ACPI_ALL_DRIVERS). The verbosity of debugging output is specified as the "level" and ranges from ACPI_LV_ERROR (just report errors) to ACPI_LV_VERBOSE (everything). The "level" is a bitmask so multiple options can be set at once, separated by spaces. In practice, a serial console should be used to log the output so it is not lost as the console message buffer flushes. A full list of the individual layers and levels is found in acpi(4).

Debugging output is not enabled by default. To enable it, add `options ACPI_DEBUG` to the kernel configuration file if ACPI is compiled into the kernel. Add `ACPI_DEBUG=1` to `/etc/make.conf` to enable it globally. If it is a module, recompile just the `acpi.ko` module as follows:

```
# cd /sys/modules/acpi/acpi
&& make clean &&
make ACPI_DEBUG=1
```

Install `acpi.ko` in `/boot/kernel` and add the desired level and layer to `/boot/loader.conf`. This example enables debug messages for all ACPI-CA components and all ACPI hardware drivers such as (CPU and LID. It only outputs error messages at the least verbose level.

```
debug.acpi.layer="ACPI_ALL_COMPONENTS ACPI_ALL_DRIVERS"
debug.acpi.level="ACPI_LV_ERROR"
```

If the required information is triggered by a specific event, such as a suspend and then resume, leave out changes to `/boot/loader.conf` and instead use sysctl(8) to specify the layer and level after booting and preparing the system for the specific event. The variables which can be set using sysctl(8) are named the same as the tunables in `/boot/loader.conf`.

12.17.7 References

More information about ACPI may be found in the following locations:

- The FreeBSD ACPI mailing list (http://lists.FreeBSD.org/mailman/listinfo/freebsd-acpi)
- The ACPI Mailing List Archives http://lists.freebsd.org/pipermail/freebsd-acpi/
- The old ACPI Mailing List Archives http://home.jp.FreeBSD.org/mail-list/acpi-jp/
- The ACPI 2.0 Specification http://acpi.info/spec.htm
- acpi(4), acpi_thermal(4), acpidump(8), iasl(8), and acpidb(8)
- DSDT debugging resource (http://www.cpqlinux.com/acpi-howto.html#fix_broken_dsdt).

Notes

1. The auto-tuning algorithm sets `maxusers` equal to the amount of memory in the system, with a minimum of `32`, and a maximum of `384`.

Chapter 13 The FreeBSD Booting Process

13.1 Synopsis

The process of starting a computer and loading the operating system is referred to as “the bootstrap process”, or simply “booting”. FreeBSD’s boot process provides a great deal of flexibility in customizing what happens when the system starts, including the ability to select from different operating systems installed on the same computer, different versions of the same operating system, or a different installed kernel.

This chapter details the configuration options that can be set. It demonstrates how to customize the FreeBSD boot process, including everything that happens until the FreeBSD kernel has started, probed for devices, and started init(8). This occurs when the text color of the boot messages changes from bright white to grey.

After reading this chapter, you will recognize:

- The components of the FreeBSD bootstrap system and how they interact.
- The options that can be passed to the components in the FreeBSD bootstrap in order to control the boot process.
- The basics of device.hints(5).

Note: This chapter only describes the boot process for FreeBSD running on Intel x86 systems.

13.2 The Booting Problem

Turning on a computer and starting the operating system poses an interesting dilemma. By definition, the computer does not know how to do anything until the operating system is started. This includes running programs from the disk. If the computer can not run a program from the disk without the operating system, and the operating system programs are on the disk, how is the operating system started?

This problem parallels one in the book *The Adventures of Baron Munchausen*. A character had fallen part way down a manhole, and pulled himself out by grabbing his bootstraps, and lifting. In the early days of computing the term *bootstrap* was applied to the mechanism used to load the operating system, which has become shortened to “booting”.

On x86 hardware the Basic Input/Output System (BIOS) is responsible for loading the operating system. To do this, the BIOS looks on the hard disk for the Master Boot Record (MBR), which must be located in a specific place on the disk. The BIOS has enough knowledge to load and run the MBR, and assumes that the MBR can then carry out the rest of the tasks involved in loading the operating system, possibly with the help of the BIOS.

The code within the MBR is usually referred to as a *boot manager*, especially when it interacts with the user. In this case, the boot manager usually has more code in the first *track* of the disk or within the file system of some operating systems. A boot manager is sometimes also called a *boot loader*, but FreeBSD uses that term for a later stage of booting. Popular boot managers include **boot0**, also called **Boot Easy**, the standard FreeBSD boot manager, **Grub**, **GAG**, and **LILO**. Only **boot0** fits within the MBR.

If only one operating system is installed, a standard PC MBR will suffice. This MBR searches for the first bootable (active) slice on the disk, and then runs the code on that slice to load the remainder of the operating system. By default, the MBR installed by fdisk(8) is such an MBR and is based on `/boot/mbr`.

If multiple operating systems are present, a different boot manager can be installed which displays the list of operating systems so that the user can choose which one to boot from. Two boot managers are discussed in the next subsection.

The remainder of the FreeBSD bootstrap system is divided into three stages. The first stage is run by the MBR, which knows just enough to get the computer into a specific state and run the second stage. The second stage can do a little bit more, before running the third stage. The third stage finishes the task of loading the operating system. The work is split into three stages because PC standards put limits on the size of the programs that can be run at stages one and two. Chaining the tasks together allows FreeBSD to provide a more flexible loader.

The kernel is then started and it begins to probe for devices and initialize them for use. Once the kernel boot process is finished, the kernel passes control to the user process init(8), which then makes sure the disks are in a usable state. init(8) then starts the user-level resource configuration which mounts file systems, sets up network cards to communicate on the network, and starts the processes which have been configured to run on a FreeBSD system at startup.

13.3 The Boot Manager and Boot Stages

13.3.1 The Boot Manager

The code in the MBR or boot manager is sometimes referred to as *stage zero* of the boot process. This section discusses two boot managers: **boot0** and **LILO**.

The boot0 Boot Manager: The MBR installed by FreeBSD's installer or boot0cfg(8) is based on `/boot/boot0`. The size and capability of **boot0** is restricted to 446 bytes due to the slice table and `0x55AA` identifier at the end of the MBR. If **boot0** and multiple operating systems are installed, a message similar to this example will be displayed at boot time:

Example 13-1. `boot0` Screenshot

```
F1 Win
F2 FreeBSD

Default: F2
```

Other operating systems, in particular Windows, will overwrite an existing MBR if they are installed after FreeBSD. If this happens, or to replace the existing MBR with the FreeBSD MBR, use the following command:

```
# fdisk -B -b /boot/boot0 device
```

where *`device`* is the boot disk, such as `ad0` for the first IDE disk, `ad2` for the first IDE disk on a second IDE controller, or `da0` for the first SCSI disk. To create a custom configuration of the MBR, refer to boot0cfg(8).

The LILO Boot Manager: To install this boot manager so it will also boot FreeBSD, boot into Linux and add the following to the existing `/etc/lilo.conf` configuration:

```
other=/dev/hdXY
```

```
table=/dev/hdX
loader=/boot/chain.b
label=FreeBSD
```

Specify FreeBSD's primary partition and drive using Linux specifiers, replacing *X* with the Linux drive letter and *Y* with the Linux primary partition number. For a SCSI drive, change */dev/hd* to */dev/sd*. The `loader=/boot/chain.b` line can be omitted if both operating systems are installed on the same drive. Next, run `/sbin/lilo -v` to commit the new changes. Verify these are correct by checking the screen messages.

13.3.2 Stage One, `/boot/boot1`, and Stage Two, `/boot/boot2`

Conceptually, the first and second stages are part of the same program, on the same area of the disk. Because of space constraints, they have been split into two, but are always installed together. They are copied from the combined `/boot/boot` by the installer or bsdlabel(8).

They are located outside file systems, in the first track of the boot slice, starting with the first sector. This is where boot0 (Section 13.3.1), or any other boot manager, expects to find a program to run which will continue the boot process. The number of sectors used is easily determined from the size of `/boot/boot`.

`boot1` is very simple, since it can only be 512 bytes in size, and knows just enough about the FreeBSD *bsdlabel*, which stores information about the slice, to find and execute `boot2`.

`boot2` is slightly more sophisticated, and understands the FreeBSD file system enough to find files, and can provide a simple interface to choose the kernel or loader to run.

However, loader(8) is much more sophisticated and provides a boot configuration which is run by `boot2`.

Example 13-2. `boot2` Screenshot

```
>> FreeBSD/i386 BOOT
Default: 0:ad(0,a)/boot/loader
boot:
```

bsdlabel(8) can be used to replace the installed `boot1` and `boot2`:

```
# bsdlabel -B diskslice
```

where *`diskslice`* is the disk and slice to boot from, such as `ad0s1` for the first slice on the first IDE disk.

> **Dangerously Dedicated Mode:** If just the disk name is used, such as `ad0`, bsdlabel(8) will create a "dangerously dedicated" disk, without slices. This is probably not the desired action, so double check the *`diskslice`* passed to bsdlabel(8) before pressing **Return**.

13.3.3 Stage Three, `/boot/loader`

The loader is the final stage of the three-stage bootstrap, and is located on the file system, usually as `/boot/loader`.

The loader is intended as an interactive method for configuration, using a built-in command set, backed up by a more powerful interpreter which has a more complex command set.

13.3.3.1 Loader Program Flow

During initialization, the loader will probe for a console and for disks, and figure out which disk it is booting from. It will set variables accordingly, and an interpreter is started where user commands can be passed from a script or interactively.

The loader will then read `/boot/loader.rc`, which by default reads in `/boot/defaults/loader.conf` which sets reasonable defaults for variables and reads `/boot/loader.conf` for local changes to those variables. `loader.rc` then acts on these variables, loading whichever modules and kernel are selected.

Finally, by default, the loader issues a 10 second wait for key presses, and boots the kernel if it is not interrupted. If interrupted, the user is presented with a prompt which understands the command set, where the user may adjust variables, unload all modules, load modules, and then finally boot or reboot.

13.3.3.2 Loader Built-In Commands

These are the most commonly used loader commands. For a complete discussion of all available commands, refer to loader(8).

autoboot *`seconds`*

> Proceeds to boot the kernel if not interrupted within the time span given, in seconds. It displays a countdown, and the default time span is 10 seconds.

boot [`-`*`options`*] [*`kernelname`*]

> Immediately proceeds to boot the kernel, with any specified options or kernel name. Providing a kernel name on the command-line is only applicable after an *unload* command has been issued; otherwise the previously-loaded kernel will be used.

boot-conf

> Goes through the same automatic configuration of modules based on specified variables, most commonly `kernel`. This only makes sense if `unload` is used first, before changing some variables.

help [*`topic`*]

> Shows help messages read from `/boot/loader.help`. If the topic given is `index`, the list of available topics is displayed.

include *`filename`* `...`

> Processes the file with the given filename. The file is read in and interpreted line by line. An error immediately stops the include command.

load [`-t` *`type`*] *`filename`*

> Loads the kernel, kernel module, or file of the type given, with the specified filename. Any arguments after *`filename`* are passed to the file.

ls [`-l`] [*`path`*]

> Displays a listing of files in the given path, or the root directory, if the path is not specified. If `-l` is specified, file sizes will also be shown.

lsdev [-v]

Lists all of the devices from which it may be possible to load modules. If -v is specified, more details are printed.

lsmod [-v]

Displays loaded modules. If -v is specified, more details are shown.

more *filename*

Displays the files specified, with a pause at each LINES displayed.

reboot

Immediately reboots the system.

set *variable*
set *variable*=*value*

Sets the loader's environment variables.

unload

Removes all loaded modules.

13.3.3.3 Loader Examples

Here are some practical examples of loader usage:

- To boot the usual kernel in single-user mode:

  ```
  boot -s
  ```

- To unload the usual kernel and modules, and then load the previous or another kernel:

  ```
  unload
  load kernel.old
  ```

  Use kernel.GENERIC to refer to the default kernel that comes with an installation, or kernel.old to refer to the previously installed kernel before a system upgrade or before configuring a custom kernel.

  > **Note:** Use the following to load the usual modules with another kernel:
  >
  > ```
  > unload
  > set kernel="kernel.old"
  > boot-conf
  > ```

- To load an automated kernel configuration script:

  ```
  load -t userconfig_script /boot/kernel.conf
  ```

13.3.3.4 Boot Time Splash Screens

Contributed by Joseph J. Barbish.

The splash screen creates an alternate boot screen. The splash screen hides all the boot probe messages and service startup messages before displaying either a command line or graphical login prompt.

There are two basic environments available in FreeBSD. The first is the default legacy virtual console command line environment. After the system finishes booting, a console login prompt is presented. The second environment is the graphical environment as described in Chapter 6. Refer to that chapter for more information on how to install and configure a graphical display manager and a graphical login manager.

13.3.3.4.1 Splash Screen Function

The splash screen function supports 256-colors in the bitmap (`.bmp`), ZSoft PCX (`.pcx`), or TheDraw (`.bin`) formats. The splash image files must have a resolution of 320 by 200 pixels or less in order to work on standard VGA adapters.

To use larger images, up to the maximum resolution of 1024 by 768 pixels, load the VESA module during system boot. For a custom kernel, as described in Chapter 9, include the `VESA` kernel configuration option. Loading VESA support provides the ability to display a splash screen image that fills the whole display screen.

While the splash screen is being displayed during the booting process, it can be turned off any time by hitting any key on the keyboard.

The splash screen also defaults to being a screen saver outside. After a time period of non-use, the splash screen will be displayed and will cycle through steps of changing intensity of the image, from bright to very dark and over again. The configuration of the splash screen saver can be overridden by adding a `saver=` line to `/etc/rc.conf`. Several built-in screen savers are available and described in splash(4). The `saver=` option only applies to virtual consoles and has no effect on graphical display managers.

A few boot loader messages, including the boot options menu and a timed wait count down prompt, are displayed at boot time, even when the splash screen is enabled.

Sample splash screen files can be downloaded from the gallery at http://artwork.freebsdgr.org (http://artwork.freebsdgr.org/node/3/). By installing the `sysutils/bsd-splash-changer` port, splash images can be chosen from a collection randomly at each boot.

13.3.3.4.2 Enabling the Splash Screen Function

The splash screen `.bmp`, `.pcx`, or `.bin` image has to be placed on the root partition, for example in `/boot`.

For the default boot display resolution of 256-colors and 320 by 200 pixels or less, edit `/boot/loader.conf` so it contains the following:

```
splash_bmp_load="YES"
bitmap_load="YES"
bitmap_name="/boot/splash.bmp"
```

For larger video resolutions up to the maximum of 1024 by 768 pixels, edit `/boot/loader.conf`, so it contains the following:

```
vesa_load="YES"
splash_bmp_load="YES"
bitmap_load="YES"
```

```
bitmap_name="/boot/splash.bmp"
```

This example assumes that */boot/splash.bmp* is used for the splash screen. To use a PCX file, use the following statements, plus the `vesa_load="YES"` line, depending on the resolution:

```
splash_pcx_load="YES"
bitmap_load="YES"
bitmap_name="/boot/splash.pcx"
```

Beginning with FreeBSD 8.3, another option is to use ASCII art in TheDraw (https://en.wikipedia.org/wiki/TheDraw) format.

```
splash_txt="YES"
bitmap_load="YES"
bitmap_name="/boot/splash.bin"
```

The file name is not restricted to “splash” as shown in the above example. It can be anything as long as it is one of the supported types such as, *splash_640x400*`.bmp` or *bluewave*`.pcx`.

Other interesting `loader.conf` options include:

`beastie_disable="YES"`

: This will stop the boot options menu from being displayed, but the timed wait count down prompt will still be present. Even with the display of the boot options menu disabled, entering an option selection at the timed wait count down prompt will enact the corresponding boot option.

`loader_logo="beastie"`

: This will replace the default words “FreeBSD”, which are displayed to the right of the boot options menu with the colored beastie logo.

For more information, refer to splash(4), loader.conf(5), and vga(4).

13.4 Kernel Interaction During Boot

Once the kernel is loaded by either the default loader (Section 13.3.3) or by boot2 (Section 13.3.2), which bypasses the loader, it examines any boot flags and adjusts its behavior as necessary.

13.4.1 Kernel Boot Flags

Here are the more common boot flags:

`-a`

: During kernel initialization, ask for the device to mount as the root file system.

`-C`

: Boot from CDROM.

`-c`

Run UserConfig, the boot-time kernel configurator.

`-s`

Boot into single-user mode.

`-v`

Be more verbose during kernel startup.

Note: Refer to boot(8) for more information on the other boot flags.

13.5 Device Hints

Contributed by Tom Rhodes.

During initial system startup, the boot loader(8) reads device.hints(5). This file stores kernel boot information known as variables, sometimes referred to as “device hints”. These “device hints” are used by device drivers for device configuration.

Device hints may also be specified at the Stage 3 boot loader prompt, as demonstrated in Section 13.3.3. Variables can be added using `set`, removed with `unset`, and viewed `show`. Variables set in `/boot/device.hints` can also be overridden. Device hints entered at the boot loader are not permanent and will not be applied on the next reboot.

Once the system is booted, kenv(1) can be used to dump all of the variables.

The syntax for `/boot/device.hints` is one variable per line, using the hash “#” as comment markers. Lines are constructed as follows:

```
hint.driver.unit.keyword="value"
```

The syntax for the Stage 3 boot loader is:

```
set hint.driver.unit.keyword=value
```

where `driver` is the device driver name, `unit` is the device driver unit number, and `keyword` is the hint keyword. The keyword may consist of the following options:

- `at`: specifies the bus which the device is attached to.
- `port`: specifies the start address of the I/O to be used.
- `irq`: specifies the interrupt request number to be used.
- `drq`: specifies the DMA channel number.
- `maddr`: specifies the physical memory address occupied by the device.
- `flags`: sets various flag bits for the device.
- `disabled`: if set to `1` the device is disabled.

Since device drivers may accept or require more hints not listed here, viewing a driver's manual page is recommended. For more information, refer to device.hints(5), kenv(1), loader.conf(5), and loader(8).

13.6 Init: Process Control Initialization

Once the kernel has finished booting, it passes control to the user process init(8), which is located at `/sbin/init`, or the program path specified in the `init_path` variable in `loader`.

13.6.1 Automatic Reboot Sequence

The automatic reboot sequence makes sure that the file systems available on the system are consistent. If they are not, and fsck(8) cannot fix the inconsistencies of a UFS file system, init(8) drops the system into single-user mode (Section 13.6.2) so that the system administrator can resolve the problem directly.

13.6.2 Single-User Mode

This mode can be reached through the automatic reboot sequence (Section 13.6.1), the user booting with `-s`, or by setting the `boot_ single` variable in loader(8).

It can also be reached by calling shutdown(8) from multi-user mode (Section 13.6.3) without including `-r` or `-h`.

If the system `console` is set to `insecure` in `/etc/ttys`, the system will prompt for the `root` password before initiating single-user mode.

Example 13-3. An Insecure Console in `/etc/ttys`

```
# name  getty                           type    status          comments
#
# If console is marked "insecure", then init will ask for the root password
# when going to single-user mode.
console none                            unknown off insecure
```

> **Note:** An `insecure` console means that physical security to the console is considered to be insecure, so only someone who knows the `root` password may use single-user mode. Thus, to add this measure of security, choose `insecure`, instead of the default of `secure`.

13.6.3 Multi-User Mode

If init(8) finds the file systems to be in order, or once the user has finished their commands in single-user mode (Section 13.6.2), the system enters multi-user mode, in which it starts the resource configuration of the system.

13.6.3.1 Resource Configuration

The resource configuration system reads in configuration defaults from `/etc/defaults/rc.conf`, and system-specific details from `/etc/rc.conf`, and then proceeds to mount the system file systems listed in

`/etc/fstab`. It starts up networking services, miscellaneous system daemons, then the startup scripts of locally installed packages.

To learn more about the resource configuration system, refer to rc(8) and examine the scripts themselves.

13.7 Shutdown Sequence

Upon controlled shutdown using shutdown(8), init(8) will attempt to run the script `/etc/rc.shutdown`, and then proceed to send all processes the `TERM` signal, and subsequently the `KILL` signal to any that do not terminate in a timely manner.

To power down a FreeBSD machine on architectures and systems that support power management, use `shutdown -p now` to turn the power off immediately. To reboot a FreeBSD system, use `shutdown -r now`. One must be `root` or a member of `operator` in order to run shutdown(8). One can also use halt(8) and reboot(8). Refer to their manual pages and to shutdown(8) for more information.

> **Note:** Power management requires acpi(4) to be loaded as a module or staticly compiled into a custom kernel.

Chapter 14 Users and Basic Account Management

Contributed by Neil Blakey-Milner.

14.1 Synopsis

FreeBSD allows multiple users to use the computer at the same time. While only one user can sit in front of the screen and use the keyboard at any one time, any number of users can log in to the system through the network. To use the system, every user must have a user account.

After reading this chapter, you will know:

- The differences between the various user accounts on a FreeBSD system.
- How to add and remove user accounts.
- How to change account details, such as the user's full name or preferred shell.
- How to set limits on a per-account basis to control the resources, such as memory and CPU time, that accounts and groups of accounts are allowed to access.
- How to use groups to make account management easier.

Before reading this chapter, you should:

- Understand the basics of UNIX and FreeBSD.

14.2 Introduction

Since all access to the FreeBSD system is achieved via accounts and all processes are run by users, user and account management is important.

Every account on a FreeBSD system has certain information associated with it to identify the account.

User name

The user name is typed at the `login:` prompt. User names must be unique on the system as no two users can have the same user name. There are a number of rules for creating valid user names, documented in passwd(5). Typically user names consist of eight or fewer all lower case characters in order to maintain backwards compatibility with applications.

Password

Each account has an associated password. While the password can be blank, this is highly discouraged and every account should have a password.

User ID (UID)

The User ID (UID) is a number, traditionally from 0 to 65535[1], used to uniquely identify the user to the system. Internally, FreeBSD uses the UID to identify users. Commands that allow a user name to be specified will first

convert it to the UID. Though unlikely, it is possible for several accounts with different user names to share the same UID. As far as FreeBSD is concerned, these accounts are one user.

Group ID (GID)

The Group ID (GID) is a number, traditionally from 0 to 65535[1], used to uniquely identify the primary group that the user belongs to. Groups are a mechanism for controlling access to resources based on a user's GID rather than their UID. This can significantly reduce the size of some configuration files. A user may also be a member of more than one group.

Login class

Login classes are an extension to the group mechanism that provide additional flexibility when tailoring the system to different users.

Password change time

By default FreeBSD does not force users to change their passwords periodically. Password expiration can be enforced on a per-user basis, forcing some or all users to change their passwords after a certain amount of time has elapsed.

Account expiry time

By default FreeBSD does not expire accounts. When creating accounts that need a limited lifespan, such as student accounts in a school, specify the account expiry date. After the expiry time has elapsed, the account cannot be used to log in to the system, although the account's directories and files will remain.

User's full name

The user name uniquely identifies the account to FreeBSD, but does not necessarily reflect the user's real name. This information can be associated with the account.

Home directory

The home directory is the full path to a directory on the system. This is the user's starting directory when the user logs in. A common convention is to put all user home directories under `/home/`*`username`* or `/usr/home/`*`username`*. Each user stores their personal files and subdirectories in their own home directory.

User shell

The shell provides the default environment users use to interact with the system. There are many different kinds of shells, and experienced users will have their own preferences, which can be reflected in their account settings.

There are three main types of accounts: the superuser, system accounts, and user accounts. The superuser account, usually called `root`, is used to manage the system with no limitations on privileges. System accounts are used to run services. User accounts are assigned to real people and are used to log in and use the system.

14.2.1 The Superuser Account

The superuser account, usually called `root`, is used to perform system administration tasks and should not be used for day-to-day tasks like sending and receiving mail, general exploration of the system, or programming.

This is because the superuser, unlike normal user accounts, can operate without limits, and misuse of the superuser account may result in spectacular disasters. User accounts are unable to destroy the system by mistake, so it is generally best to use normal user accounts whenever possible, unless extra privilege is required.

Always double and triple-check any commands issued as the superuser, since an extra space or missing character can mean irreparable data loss.

Always create a user account for the system administrator and use this account to log in to the system for general usage. This applies equally to multi-user or single-user systems. Later sections will discuss how to create additional accounts and how to change between the normal user and superuser.

14.2.2 System Accounts

System accounts are used to run services such as DNS, mail, and web servers. The reason for this is security; if all services ran as the superuser, they could act without restriction.

Examples of system accounts are `daemon`, `operator`, `bind`, `news`, and `www`.

`nobody` is the generic unprivileged system account. However, the more services that use `nobody`, the more files and processes that user will become associated with, and hence the more privileged that user becomes.

14.2.3 User Accounts

User accounts are the primary means of access for real people to the system. User accounts insulate the user and the environment, preventing users from damaging the system or other users, and allowing users to customize their environment without affecting others.

Every person accessing the system should have a unique user account. This allows the administrator to find out who is doing what, prevents users from clobbering each others' settings or reading each others' mail, and so forth.

Each user can set up their own environment to accommodate their use of the system, by using alternate shells, editors, key bindings, and language.

14.3 Modifying Accounts

FreeBSD provides a variety of different commands to manage user accounts. The most common commands are summarized below, followed by more detailed examples of their usage.

Command	Summary
adduser(8)	The recommended command-line application for adding new users.
rmuser(8)	The recommended command-line application for removing users.
chpass(1)	A flexible tool for changing user database information.
passwd(1)	The simple command-line tool to change user passwords.
pw(8)	A powerful and flexible tool for modifying all aspects of user accounts.

14.3.1 `adduser`

adduser(8) is a simple program for adding new users When a new user is added, this program automatically updates `/etc/passwd` and `/etc/group`. It also creates a home directory for the new user, copies in the default configuration files from `/usr/share/skel`, and can optionally mail the new user a welcome message.

Example 14-1. Adding a User on FreeBSD

```
# adduser
Username: jru
Full name: J. Random User
Uid (Leave empty for default):
Login group [jru]:
Login group is jru. Invite jru into other groups? []: wheel
Login class [default]:
Shell (sh csh tcsh zsh nologin) [sh]: zsh
Home directory [/home/jru]:
Home directory permissions (Leave empty for default):
Use password-based authentication? [yes]:
Use an empty password? (yes/no) [no]:
Use a random password? (yes/no) [no]:
Enter password:
Enter password again:
Lock out the account after creation? [no]:
Username   : jru
Password   : ****
Full Name  : J. Random User
Uid        : 1001
Class      :
Groups     : jru wheel
Home       : /home/jru
Shell      : /usr/local/bin/zsh
Locked     : no
OK? (yes/no): yes
adduser: INFO: Successfully added (jru) to the user database.
Add another user? (yes/no): no
Goodbye!
#
```

> **Note:** Since the password is not echoed when typed, be careful to not mistype the password when creating the user account.

14.3.2 `rmuser`

To completely remove a user from the system use rmuser(8). This command performs the following steps:

1. Removes the user's crontab(1) entry if one exists.
2. Removes any at(1) jobs belonging to the user.
3. Kills all processes owned by the user.
4. Removes the user from the system's local password file.
5. Removes the user's home directory, if it is owned by the user.
6. Removes the incoming mail files belonging to the user from `/var/mail`.

7. Removes all files owned by the user from temporary file storage areas such as `/tmp`.
8. Finally, removes the username from all groups to which it belongs in `/etc/group`.

> **Note:** If a group becomes empty and the group name is the same as the username, the group is removed. This complements the per-user unique groups created by adduser(8).

rmuser(8) cannot be used to remove superuser accounts since that is almost always an indication of massive destruction.

By default, an interactive mode is used, as shown in the following example.

Example 14-2. `rmuser` Interactive Account Removal

```
# rmuser jru
Matching password entry:
jru:*:1001:1001::0:0:J. Random User:/home/jru:/usr/local/bin/zsh
Is this the entry you wish to remove? y
Remove user's home directory (/home/jru)? y
Updating password file, updating databases, done.
Updating group file: trusted (removing group jru -- personal group is empty) done.
Removing user's incoming mail file /var/mail/jru: done.
Removing files belonging to jru from /tmp: done.
Removing files belonging to jru from /var/tmp: done.
Removing files belonging to jru from /var/tmp/vi.recover: done.
#
```

14.3.3 `chpass`

chpass(1) can be used to change user database information such as passwords, shells, and personal information.

Only the superuser can change other users' information and passwords with chpass(1).

When passed no options, aside from an optional username, chpass(1) displays an editor containing user information. When the user exists from the editor, the user database is updated with the new information.

> **Note:** You will be asked for your password after exiting the editor if you are not the superuser.

Example 14-3. Interactive `chpass` by Superuser

```
#Changing user database information for jru.
Login: jru
Password: *
Uid [#]: 1001
Gid [# or name]: 1001
Change [month day year]:
Expire [month day year]:
Class:
```

```
Home directory: /home/jru
Shell: /usr/local/bin/zsh
Full Name: J. Random User
Office Location:
Office Phone:
Home Phone:
Other information:
```

A user can change only a small subset of this information, and only for their own user account.

Example 14-4. Interactive `chpass` by Normal User

```
#Changing user database information for jru.
Shell: /usr/local/bin/zsh
Full Name: J. Random User
Office Location:
Office Phone:
Home Phone:
Other information:
```

> **Note:** chfn(1) and chsh(1) are links to chpass(1), as are ypchpass(1), ypchfn(1), and ypchsh(1). NIS support is automatic, so specifying the `yp` before the command is not necessary. How to configure NIS is covered in Chapter 30.

14.3.4 `passwd`

passwd(1) is the usual way to change your own password as a user, or another user's password as the superuser.

> **Note:** To prevent accidental or unauthorized changes, the user must enter their original password before a new password can be set. This is not the case when the superuser changes a user's password.

Example 14-5. Changing Your Password

```
% passwd
Changing local password for jru.
Old password:
New password:
Retype new password:
passwd: updating the database...
passwd: done
```

Example 14-6. Changing Another User's Password as the Superuser

```
# passwd jru
Changing local password for jru.
New password:
Retype new password:
passwd: updating the database...
passwd: done
```

> **Note:** As with chpass(1), yppasswd(1) is a link to passwd(1), so NIS works with either command.

14.3.5 pw

pw(8) is a command line utility to create, remove, modify, and display users and groups. It functions as a front end to the system user and group files. pw(8) has a very powerful set of command line options that make it suitable for use in shell scripts, but new users may find it more complicated than the other commands presented in this section.

14.4 Limiting Users

FreeBSD provides several methods for an administrator to limit the amount of system resources an individual may use. These limits are discussed in two sections: disk quotas and other resource limits.

Disk quotas limit the amount of disk space available to users and provide a way to quickly check that usage without calculating it every time. Quotas are discussed in Section 19.13.

The other resource limits include ways to limit the amount of CPU, memory, and other resources a user may consume. These are defined using login classes and are discussed here.

Login classes are defined in `/etc/login.conf` and are described in detail in login.conf(5). Each user account is assigned to a login class, `default` by default, and each login class has a set of login capabilities associated with it. A login capability is a *`name=value`* pair, where *`name`* is a well-known identifier and *`value`* is an arbitrary string which is processed accordingly depending on the *`name`*. Setting up login classes and capabilities is rather straightforward and is also described in login.conf(5).

> **Note:** FreeBSD does not normally read the configuration in `/etc/login.conf` directly, but instead reads the `/etc/login.conf.db` database which provides faster lookups. Whenever `/etc/login.conf` is edited, the `/etc/login.conf.db` must be updated by executing the following command:
>
> ```
> # cap_mkdb /etc/login.conf
> ```

Resource limits differ from the default login capabilities in two ways. First, for every limit, there is a soft (current) and hard limit. A soft limit may be adjusted by the user or application, but may not be set higher than the hard limit. The hard limit may be lowered by the user, but can only be raised by the superuser. Second, most resource limits apply per process to a specific user, not to the user as a whole. These differences are mandated by the specific handling of the limits, not by the implementation of the login capability framework.

Below are the most commonly used resource limits. The rest of the limits, along with all the other login capabilities, can be found in login.conf(5).

`coredumpsize`

The limit on the size of a core file generated by a program is subordinate to other limits on disk usage, such as `filesize`, or disk quotas. This limit is often used as a less-severe method of controlling disk space consumption. Since users do not generate core files themselves, and often do not delete them, setting this may save them from running out of disk space should a large program crash.

`cputime`

The maximum amount of CPU time a user's process may consume. Offending processes will be killed by the kernel.

> **Note:** This is a limit on CPU *time* consumed, not percentage of the CPU as displayed in some fields by top(1) and ps(1).

`filesize`

The maximum size of a file the user may own. Unlike disk quotas, this limit is enforced on individual files, not the set of all files a user owns.

`maxproc`

The maximum number of processes a user can run. This includes foreground and background processes. This limit may not be larger than the system limit specified by the `kern.maxproc` sysctl(8). Setting this limit too small may hinder a user's productivity as it is often useful to be logged in multiple times or to execute pipelines. Some tasks, such as compiling a large program, spawn multiple processes and other intermediate preprocessors.

`memorylocked`

The maximum amount of memory a process may request to be locked into main memory using mlock(2). Some system-critical programs, such as amd(8), lock into main memory so that if the system begins to swap, they do not contribute to disk thrashing.

`memoryuse`

The maximum amount of memory a process may consume at any given time. It includes both core memory and swap usage. This is not a catch-all limit for restricting memory consumption, but is a good start.

`openfiles`

The maximum number of files a process may have open. In FreeBSD, files are used to represent sockets and IPC channels, so be careful not to set this too low. The system-wide limit for this is defined by the `kern.maxfiles` sysctl(8).

`sbsize`

The limit on the amount of network memory, and thus mbufs, a user may consume in order to limit network communications.

`stacksize`

The maximum size of a process stack. This alone is not sufficient to limit the amount of memory a program may use so it should be used in conjunction with other limits.

There are a few other things to remember when setting resource limits. Following are some general tips, suggestions, and miscellaneous comments.

- Processes started at system startup by `/etc/rc` are assigned to the `daemon` login class.
- Although the `/etc/login.conf` that comes with the system is a good source of reasonable values for most limits, they may not be appropriate for every system. Setting a limit too high may open the system up to abuse, while setting it too low may put a strain on productivity.
- Users of **Xorg** should probably be granted more resources than other users. **Xorg** by itself takes a lot of resources, but it also encourages users to run more programs simultaneously.
- Many limits apply to individual processes, not the user as a whole. For example, setting `openfiles` to 50 means that each process the user runs may open up to 50 files. The total amount of files a user may open is the value of `openfiles` multiplied by the value of `maxproc`. This also applies to memory consumption.

For further information on resource limits and login classes and capabilities in general, refer to cap_mkdb(1), getrlimit(2), and login.conf(5).

14.5 Groups

A group is a list of users. A group is identified by its group name and GID. In FreeBSD, the kernel uses the UID of a process, and the list of groups it belongs to, to determine what the process is allowed to do. Most of the time, the GID of a user or process usually means the first group in the list.

The group name to GID mapping is listed in `/etc/group`. This is a plain text file with four colon-delimited fields. The first field is the group name, the second is the encrypted password, the third the GID, and the fourth the comma-delimited list of members. For a more complete description of the syntax, refer to group(5).

The superuser can modify `/etc/group` using a text editor. Alternatively, pw(8) can be used to add and edit groups. For example, to add a group called `teamtwo` and then confirm that it exists:

Example 14-7. Adding a Group Using pw(8)

```
# pw groupadd teamtwo
# pw groupshow teamtwo
teamtwo:*:1100:
```

In this example, `1100` is the GID of `teamtwo`. Right now, `teamtwo` has no members. This command will add `jru` as a member of `teamtwo`.

Example 14-8. Adding User Accounts to a New Group Using pw(8)

```
# pw groupmod teamtwo -M jru
# pw groupshow teamtwo
teamtwo:*:1100:jru
```

The argument to `-M` is a comma-delimited list of users to be added to a new (empty) group or to replace the members of an existing group. To the user, this group membership is different from (and in addition to) the user's primary group listed in the password file. This means that the user will not show up as a member when using `groupshow` with pw(8), but will show up when the information is queried via id(1) or a similar tool. When pw(8) is used to add a user to a group, it only manipulates `/etc/group` and does not attempt to read additional data from `/etc/passwd`.

Example 14-9. Adding a New Member to a Group Using pw(8)

```
# pw groupmod teamtwo -m db
# pw groupshow teamtwo
teamtwo:*:1100:jru,db
```

In this example, the argument to `-m` is a comma-delimited list of users who are to be added to the group. Unlike the previous example, these users are appended to the group list and do not replace the list of existing users in the group.

Example 14-10. Using id(1) to Determine Group Membership

```
% id jru
uid=1001(jru) gid=1001(jru) groups=1001(jru), 1100(teamtwo)
```

In this example, `jru` is a member of the groups `jru` and `teamtwo`.

For more information about this command and the format of `/etc/group`, refer to pw(8) and group(5).

14.6 Becoming Superuser

There are several ways to do things as the superuser. The worst way is to log in as `root` directly. Usually very little activity requires `root` so logging off and logging in as `root`, performing tasks, then logging off and on again as a normal user is a waste of time.

A better way is to use su(1) without providing a login but using `-` to inherit the root environment. Not providing a login will imply super user. For this to work the login that must be in the `wheel` group. An example of a typical software installation would involve the administrator unpacking the software as a normal user and then elevating their privileges for the build and installation of the software.

Example 14-11. Install a Program As The Superuser

```
% configure
% make
% su -
Password:
# make install
# exit
%
```

Note in this example the transition to `root` is less painful than logging off and back on twice.

Using su(1) works well for single systems or small networks with just one system administrator. For more complex environments (or even for these simple environments) `sudo` should be used. It is provided as a port, `security/sudo`. It allows for things like activity logging, granting users the ability to only run certain commands as the superuser, and several other options.

Notes

1. It is possible to use UIDs/GIDs as large as 4294967295, but such IDs can cause serious problems with software that makes assumptions about the values of IDs.

Chapter 15 Security

Much of this chapter has been taken from the security(7) manual page by Matthew Dillon.

15.1 Synopsis

This chapter provides a basic introduction to system security concepts, some general good rules of thumb, and some advanced topics under FreeBSD. Many of the topics covered here can be applied to system and Internet security in general. Securing a system is imperative to protect data, intellectual property, time, and much more from the hands of hackers and the like.

FreeBSD provides an array of utilities and mechanisms to protect the integrity and security of the system and network.

After reading this chapter, you will know:

- Basic FreeBSD system security concepts.
- The various crypt mechanisms available in FreeBSD.
- How to set up one-time password authentication.
- How to configure TCP Wrappers for use with inetd(8).
- How to set up **Kerberos** on FreeBSD.
- How to configure IPsec and create a VPN.
- How to configure and use **OpenSSH** on FreeBSD.
- How to use filesystem ACLs.
- How to use **portaudit** to audit third party software packages installed from the Ports Collection.
- How to utilize FreeBSD security advisories.
- What Process Accounting is and how to enable it on FreeBSD.
- Understand the resource limits database and how to utilize it to control user resources.

Before reading this chapter, you should:

- Understand basic FreeBSD and Internet concepts.

Additional security topics are covered elsewhere in this Handbook. For example, Mandatory Access Control is discussed in Chapter 17 and Internet firewalls are discussed in Chapter 31.

15.2 Introduction

Security is a function that begins and ends with the system administrator. While FreeBSD provides some inherent security, the job of configuring and maintaining additional security mechanisms is probably one of the single largest undertakings of the sysadmin.

System security also pertains to dealing with various forms of attack, including attacks that attempt to crash, or otherwise make a system unusable, but do not attempt to compromise the `root` account. Security concerns can be split up into several categories:

1. Denial of service attacks.
2. User account compromises.
3. Root compromise through accessible services.
4. Root compromise via user accounts.
5. Backdoor creation.

A Denial of Service DoS attack is an action that deprives the machine of needed resources. Typically, DoS attacks are brute-force mechanisms that attempt to crash or otherwise make a machine unusable by overwhelming its services or network stack. Attacks on servers can often be fixed by properly specifying options to limit the load the servers incur on the system under adverse conditions. Brute-force network attacks are harder to deal with. This type of attack may not be able to take the machine down, but it can saturate the Internet connection.

A user account compromise is more common than a DoS attack. Many sysadmins still run unencrypted services, meaning that users logging into the system from a remote location are vulnerable to having their password sniffed. The attentive sysadmin analyzes the remote access logs looking for suspicious source addresses and suspicious logins.

In a well secured and maintained system, access to a user account does not necessarily give the attacker access to `root`. Without `root` access, the attacker cannot generally hide his tracks and may, at best, be able to do nothing more than mess with the user's files or crash the machine. User account compromises are common because users tend not to take the precautions that sysadmins take.

There are potentially many ways to break `root`: the attacker may know the `root` password, the attacker may exploit a bug in a service which runs as `root`, or the attacker may know of a bug in a SUID-root program. An attacker may utilize a program known as a backdoor to search for vulnerable systems, take advantage of unpatched exploits to access a system, and hide traces of illegal activity.

Security remedies should always be implemented with a multi-layered "onion peel" approach and can be categorized as follows:

1. Secure `root` and staff accounts.
2. Secure `root`–run servers and SUID/SGID binaries.
3. Secure user accounts.
4. Secure the password file.
5. Secure the kernel core, raw devices, and filesystems.
6. Quick detection of inappropriate changes made to the system.
7. Paranoia.

The next section covers these items in greater depth.

15.3 Securing FreeBSD

This section describes methods for securing a FreeBSD system against the attacks that were mentioned in the previous section.

15.3.1 Securing the `root` Account

Most systems have a password assigned to the `root` account. Assume that this password is *always* at risk of being compromised. This does not mean that the password should be disabled as the password is almost always necessary for console access to the machine. However, it should not be possible to use this password outside of the console or possibly even with su(1). For example, setting the entries in `/etc/ttys` to `insecure` prevents `root` logins to the specified terminals. In FreeBSD, `root` logins using ssh(1) are disabled by default as `PermitRootLogin` is set to `no` in `/etc/ssh/sshd_config`. Consider every access method as services such as FTP often fall through the cracks. Direct `root` logins should only be allowed via the system console.

Since a sysadmin needs access to `root`, additional password verification should be configured. One method is to add appropriate user accounts to `wheel` in `/etc/group`. Members of `wheel` are allowed to su(1) to `root`. Only those users who actually need to have `root` access should be placed in `wheel`. When using Kerberos for authentication, create a `.k5login` in the home directory of `root` to allow ksu(1) to be used without having to place anyone in `wheel`.

To lock an account completely, use pw(8):

```
# pw lock staff
```

This will prevent the specified user from logging in using any mechanism, including ssh(1).

Another method of blocking access to accounts would be to replace the encrypted password with a single "`*`" character. This character would never match the encrypted password and thus blocks user access. For example, the entry for the following account:

```
foobar:R9DT/Fa1/LV9U:1000:1000::0:0:Foo Bar:/home/foobar:/usr/local/bin/tcsh
```

could be changed to this using vipw(8):

```
foobar:*:1000:1000::0:0:Foo Bar:/home/foobar:/usr/local/bin/tcsh
```

This prevents `foobar` from logging in using conventional methods. This method for access restriction is flawed on sites using **Kerberos** or in situations where the user has set up keys with ssh(1).

These security mechanisms assume that users are logging in from a more restrictive server to a less restrictive server. For example, if the server is running network services, the workstation should not be running any. In order for a workstation to be reasonably secure, run zero or as few services as possible and run a password-protected screensaver. Of course, given physical access to any system, an attacker can break any sort of security. Fortunately, many break-ins occur remotely, over a network, from people who do not have physical access to the system.

Using Kerberos provides the ability to disable or change the password for a user in one place, and have it immediately affect all the machines on which the user has an account. If an account is compromised, the ability to instantly change the associated password on all machines should not be underrated. Additional restrictions can be imposed with Kerberos: a Kerberos ticket can be configured to timeout and the Kerberos system can require that the user choose a new password after a configurable period of time.

15.3.2 Securing Root-run Servers and SUID/SGID Binaries

The prudent sysadmin only enables required services and is aware that third party servers are often the most bug-prone. Never run a server that has not been checked out carefully. Think twice before running any service as `root` as many daemons can be run as a separate service account or can be started in a *sandbox*. Do not activate insecure services such as telnetd(8) or rlogind(8).

Another potential security hole is SUID-root and SGID binaries. Most of these binaries, such as rlogin(1), reside in `/bin`, `/sbin`, `/usr/bin`, or `/usr/sbin`. While nothing is 100% safe, the system-default SUID and SGID binaries can be considered reasonably safe. It is recommended to restrict SUID binaries to a special group that only staff can access, and to delete any unused SUID binaries. SGID binaries can be almost as dangerous. If an intruder can break an SGID-kmem binary, the intruder might be able to read `/dev/kmem` and thus read the encrypted password file, potentially compromising user accounts. Alternatively, an intruder who breaks group `kmem` can monitor keystrokes sent through ptys, including ptys used by users who login through secure methods. An intruder that breaks the `tty` group can write to almost any user's tty. If a user is running a terminal program or emulator with a keyboard-simulation feature, the intruder can potentially generate a data stream that causes the user's terminal to echo a command, which is then run as that user.

15.3.3 Securing User Accounts

User accounts are usually the most difficult to secure. Be vigilant in the monitoring of user accounts. Use of ssh(1) and Kerberos for user accounts requires extra administration and technical support, but provides a good solution compared to an encrypted password file.

15.3.4 Securing the Password File

The only sure fire way is to star out as many passwords as possible and use ssh(1) or Kerberos for access to those accounts. Even though the encrypted password file (`/etc/spwd.db`) can only be read by `root`, it may be possible for an intruder to obtain read access to that file even if the attacker cannot obtain root-write access.

Security scripts should be used to check for and report changes to the password file as described in the Checking file integrity section.

15.3.5 Securing the Kernel Core, Raw Devices, and Filesystems

Most modern kernels have a packet sniffing device driver built in. Under FreeBSD it is called `bpf`. This device is needed for DHCP, but can be removed in the custom kernel configuration file of systems that do not provide or use DHCP.

Even if `bpf` is disabled, `/dev/mem` and `/dev/kmem` are still problematic. An intruder can still write to raw disk devices. An enterprising intruder can use kldload(8) to install his own `bpf`, or another sniffing device, on a running kernel. To avoid these problems, run the kernel at a higher security level, at least security level 1.

The security level of the kernel can be set in a variety of ways. The simplest way of raising the security level of a running kernel is to set `kern.securelevel`:

```
# sysctl kern.securelevel=1
```

By default, the FreeBSD kernel boots with a security level of -1. This is called "insecure mode" because immutable file flags may be turned off and all devices may be read from or written to. The security level will remain at -1 unless

it is altered, either by the administrator or by init(8), because of a setting in the startup scripts. The security level may be raised during system startup by setting `kern_securelevel_enable` to `YES` in `/etc/rc.conf`, and the value of `kern_securelevel` to the desired security level.

Once the security level is set to 1 or a higher value, the append-only and immutable files are honored, they cannot be turned off, and access to raw devices is denied. Higher levels restrict even more operations. For a full description of the effect of various security levels, refer to security(7) and init(8).

> **Note:** Bumping the security level to 1 or higher may cause a few problems to **Xorg**, as access to `/dev/io` will be blocked, or to the installation of FreeBSD built from source as `installworld` needs to temporarily reset the append-only and immutable flags of some files. In the case of **Xorg**, it may be possible to work around this by starting xdm(1) early in the boot process, when the security level is still low enough. Workarounds may not be possible for all secure levels or for all the potential restrictions they enforce. A bit of forward planning is a good idea. Understanding the restrictions imposed by each security level is important as they severely diminish the ease of system use. It will also make choosing a default setting much simpler and prevent any surprises.

If the kernel's security level is raised to 1 or a higher value, it may be useful to set the `schg` flag on critical startup binaries, directories, script files, and everything that gets run up to the point where the security level is set. A less strict compromise is to run the system at a higher security level but skip setting the `schg` flag. Another possibility is to mount `/` and `/usr` read-only. It should be noted that being too draconian about what is permitted may prevent detection of an intrusion.

15.3.6 Checking File Integrity

One can only protect the core system configuration and control files so much before the convenience factor rears its ugly head. For example, using chflags(1) to set the `schg` bit on most of the files in `/` and `/usr` is probably counterproductive, because while it may protect the files, it also closes an intrusion detection window. Security measures are useless or, worse, present a false sense of security, if potential intrusions cannot be detected. Half the job of security is to slow down, not stop, an attacker, in order to catch him in the act.

The best way to detect an intrusion is to look for modified, missing, or unexpected files. The best way to look for modified files is from another, often centralized, limited-access system. Writing security scripts on the extra-security limited-access system makes them mostly invisible to potential attackers. In order to take maximum advantage, the limited-access box needs significant access to the other machines, usually either through a read-only NFS export or by setting up ssh(1) key-pairs. Except for its network traffic, NFS is the least visible method, allowing the administrator to monitor the filesystems on each client box virtually undetected. If a limited-access server is connected to the client boxes through a switch, the NFS method is often the better choice. If a limited-access server is connected to the client boxes through several layers of routing, the NFS method may be too insecure and ssh(1) may be the better choice.

Once a limited-access box has been given at least read access to the client systems it is supposed to monitor, create the monitoring scripts. Given an NFS mount, write scripts out of simple system utilities such as find(1) and md5(1). It is best to physically md5(1) the client system's files at least once a day, and to test control files such as those found in `/etc` and `/usr/local/etc` even more often. When mismatches are found, relative to the base md5 information the limited-access machine knows is valid, it should alert the sysadmin. A good security script will also check for inappropriate SUID binaries and for new or deleted files on system partitions such as `/` and `/usr`.

When using ssh(1) rather than NFS, writing the security script is more difficult. For example, scp(1) is needed to send the scripts to the client box in order to run them. The ssh(1) client on the client box may already be

compromised. Using ssh(1) may be necessary when running over insecure links, but it is harder to deal with.

A good security script will also check for changes to hidden configuration files, such as `.rhosts` and `.ssh/authorized_keys`, as these files might fall outside the purview of the `MD5` check.

For a large amount of user disk space, it may take too long to run through every file on those partitions. In this case, consider setting mount flags to disallow SUID binaries by using `nosuid` with mount(8). Scan these partitions at least once a week, since the objective is to detect a break-in attempt, whether or not the attempt succeeds.

Process accounting (see accton(8)) is a relatively low-overhead feature of FreeBSD which might help as a post-break-in evaluation mechanism. It is especially useful in tracking down how an intruder broke into a system, assuming the file is still intact after the break-in has occurred.

Finally, security scripts should process the log files, and the logs themselves should be generated in as secure a manner as possible and sent to a remote syslog server. An intruder will try to cover his tracks, and log files are critical to the sysadmin trying to track down the time and method of the initial break-in. One way to keep a permanent record of the log files is to run the system console to a serial port and collect the information to a secure machine monitoring the consoles.

15.3.7 Paranoia

A little paranoia never hurts. As a rule, a sysadmin can add any number of security features which do not affect convenience and can add security features that *do* affect convenience with some added thought. More importantly, a security administrator should mix it up a bit. If recommendations, such as those mentioned in this section, are applied verbatim, those methodologies are given to the prospective attacker who also has access to this document.

15.3.8 Denial of Service Attacks

A DoS attack is typically a packet attack. While there is not much one can do about spoofed packet attacks that saturate a network, one can generally limit the damage by ensuring that the attack cannot take down servers by:

1. Limiting server forks.
2. Limiting springboard attacks such as ICMP response attacks and ping broadcasts.
3. Overloading the kernel route cache.

A common DoS attack scenario is to force a forking server to spawn so many child processes that the host system eventually runs out of memory and file descriptors, and then grinds to a halt. There are several options to inetd(8) to limit this sort of attack. It should be noted that while it is possible to prevent a machine from going down, it is not generally possible to prevent a service from being disrupted by the attack. Read inetd(8) carefully and pay specific attention to `-c`, `-C`, and `-R`. Spoofed IP attacks will circumvent `-C` to inetd(8), so typically a combination of options must be used. Some standalone servers have self-fork-limitation parameters.

Sendmail provides `-OMaxDaemonChildren`, which tends to work better than trying to use **Sendmail**'s load limiting options due to the load lag. Specify a `MaxDaemonChildren` when starting **Sendmail** which is high enough to handle the expected load, but not so high that the computer cannot handle that number of **Sendmail** instances. It is prudent to run **Sendmail** in queued mode using `-ODeliveryMode=queued` and to run the daemon (`sendmail -bd`) separate from the queue-runs (`sendmail -q15m`). For real-time delivery, run the queue at a much lower interval, such as `-q1m`, but be sure to specify a reasonable `MaxDaemonChildren` to prevent cascade failures.

syslogd(8) can be attacked directly and it is strongly recommended to use `-s` whenever possible, and `-a` otherwise.

Be careful with connect-back services such as reverse-identd, which can be attacked directly. The reverse-ident feature of **TCP Wrappers** is not recommended for this reason.

It is recommended to protect internal services from external access by firewalling them at the border routers. This is to prevent saturation attacks from outside the LAN, not so much to protect internal services from network-based `root` compromise. Always configure an exclusive firewall which denies everything by default except for traffic which is explicitly allowed. The range of port numbers used for dynamic binding in FreeBSD is controlled by several `net.inet.ip.portrange` sysctl(8) variables.

Another common DoS attack, called a springboard attack, causes the server to generate responses which overloads the server, the local network, or some other machine. The most common attack of this nature is the *ICMP ping broadcast attack*. The attacker spoofs ping packets sent to the LAN's broadcast address with the source IP address set to the machine to attack. If the border routers are not configured to drop ping packets sent to broadcast addresses, the LAN generates sufficient responses to the spoofed source address to saturate the victim, especially when the attack is against several dozen broadcast addresses over several dozen different networks at once. A second common springboard attack constructs packets that generate ICMP error responses which can saturate a server's incoming network and cause the server to saturate its outgoing network with ICMP responses. This type of attack can crash the server by running it out of memory, especially if the server cannot drain the ICMP responses it generates fast enough. Use the sysctl(8) variable `net.inet.icmp.icmplim` to limit these attacks. The last major class of springboard attacks is related to certain internal inetd(8) services such as the UDP echo service. An attacker spoofs a UDP packet with a source address of server A's echo port and a destination address of server B's echo port, where server A and B on the same LAN. The two servers bounce this one packet back and forth between each other. The attacker can overload both servers and the LAN by injecting a few packets in this manner. Similar problems exist with the **chargen** port. These inetd-internal test services should remain disabled.

Spoofed packet attacks may be used to overload the kernel route cache. Refer to the `net.inet.ip.rtexpire`, `rtminexpire`, and `rtmaxcache` sysctl(8) parameters. A spoofed packet attack that uses a random source IP will cause the kernel to generate a temporary cached route in the route table, viewable with `netstat -rna | fgrep W3`. These routes typically timeout in 1600 seconds or so. If the kernel detects that the cached route table has gotten too big, it will dynamically reduce the `rtexpire` but will never decrease it to less than `rtminexpire`. This creates two problems:

1. The kernel does not react quickly enough when a lightly loaded server is suddenly attacked.
2. The `rtminexpire` is not low enough for the kernel to survive a sustained attack.

If the servers are connected to the Internet via a T3 or better, it may be prudent to manually override both `rtexpire` and `rtminexpire` via sysctl(8). Never set either parameter to zero as this could crash the machine. Setting both parameters to 2 seconds should be sufficient to protect the route table from attack.

15.3.9 Access Issues with Kerberos and ssh(1)

There are a few issues with both Kerberos and ssh(1) that need to be addressed if they are used. Kerberos is an excellent authentication protocol, but there are bugs in the kerberized versions of telnet(1) and rlogin(1) that make them unsuitable for dealing with binary streams. By default, Kerberos does not encrypt a session unless `-x` is used whereas ssh(1) encrypts everything.

While ssh(1) works well, it forwards encryption keys by default. This introduces a security risk to a user who uses ssh(1) to access an insecure machine from a secure workstation. The keys themselves are not exposed, but ssh(1) installs a forwarding port for the duration of the login. If an attacker has broken `root` on the insecure machine, he can utilize that port to gain access to any other machine that those keys unlock.

It is recommended that ssh(1) is used in combination with Kerberos whenever possible for staff logins and ssh(1) can be compiled with Kerberos support. This reduces reliance on potentially exposed SSH keys while protecting passwords via Kerberos. Keys should only be used for automated tasks from secure machines as this is something that Kerberos is unsuited to. It is recommended to either turn off key-forwarding in the SSH configuration, or to make use of `from=IP/DOMAIN` in `authorized_keys` to make the key only usable to entities logging in from specific machines.

15.4 DES, Blowfish, MD5, SHA256, SHA512, and Crypt

Parts rewritten and updated by Bill Swingle.

Every user on a UNIX system has a password associated with their account. In order to keep these passwords secret, they are encrypted with a "one-way hash", as they can be easily encrypted but not decrypted. The operating system itself does not know the password. It only knows the *encrypted* form of the password. The only way to get the "plain-text" password is by a brute force search of the space of possible passwords.

Originally, the only secure way to encrypt passwords in UNIX was based on the Data Encryption Standard (DES). Since the source code for DES could not be exported outside the US, FreeBSD had to find a way to both comply with US law and retain compatibility with other UNIX variants that used DES. The solution was MD5 which is believed to be more secure than DES.

15.4.1 Recognizing the Crypt Mechanism

Currently the library supports DES, MD5, Blowfish, SHA256, and SHA512 hash functions. To identify which encryption method FreeBSD is set up to use, examine the encrypted passwords in `/etc/master.passwd`. Passwords encrypted with the MD5 hash are longer than those encrypted with the DES hash and begin with the characters `$1$`. Passwords starting with `$2a$` are encrypted with the Blowfish hash function. DES password strings do not have any particular identifying characteristics, but they are shorter than MD5 passwords, and are coded in a 64-character alphabet which does not include the `$` character, so a relatively short string which does not begin with a dollar sign is very likely a DES password. Both SHA256 and SHA512 begin with the characters `$6$`.

The password format used for new passwords is controlled by the `passwd_format` login capability in `/etc/login.conf`, which takes values of `des`, `md5`, `blf`, `sha256` or `sha512`. Refer to login.conf(5) for more information about login capabilities.

15.5 One-time Passwords

By default, FreeBSD includes support for One-time Passwords In Everything (OPIE), which uses the MD5 hash by default.

There are three different types of passwords. The first is the usual UNIX style or Kerberos password. The second is the one-time password which is generated by opiekey(1) and accepted by opiepasswd(1) and the login prompt. The final type of password is the "secret password" used by opiekey(1), and sometimes opiepasswd(1), to generate one-time passwords.

The secret password has nothing to do with the UNIX password. They can be the same, but this is not recommended. OPIE secret passwords are not limited to 8 characters like old UNIX passwords[1]. Passwords of six or seven word

long phrases are fairly common. For the most part, the OPIE system operates completely independently of the UNIX password system.

Besides the password, there are two other pieces of data that are important to OPIE. One is the "seed" or "key", consisting of two letters and five digits. The other is the "iteration count", a number between 1 and 100. OPIE creates the one-time password by concatenating the seed and the secret password, applying the MD5 hash as many times as specified by the iteration count, and turning the result into six short English words. These six English words are the one-time password. The authentication system (primarily PAM) keeps track of the last one-time password used, and the user is authenticated if the hash of the user-provided password is equal to the previous password. Because a one-way hash is used, it is impossible to generate future one-time passwords if a successfully used password is captured. The iteration count is decremented after each successful login to keep the user and the login program in sync. When the iteration count gets down to 1, OPIE must be reinitialized.

There are a few programs involved in this process. opiekey(1) accepts an iteration count, a seed, and a secret password, and generates a one-time password or a consecutive list of one-time passwords. In addition to initializing OPIE, opiepasswd(1) is used to change passwords, iteration counts, or seeds. It takes either a secret passphrase, or an iteration count, seed, and a one-time password. The relevant credential files in `/etc/opiekeys` are examined by opieinfo(1) which prints out the invoking user's current iteration count and seed.

There are four different sorts of operations. The first is to use opiepasswd(1) over a secure connection to set up one-time-passwords for the first time, or to change the password or seed. The second operation is to use opiepasswd(1) over an insecure connection, in conjunction with opiekey(1) over a secure connection, to do the same. The third is to use opiekey(1) to log in over an insecure connection. The fourth is to use opiekey(1) to generate a number of keys which can be written down or printed out to carry to insecure locations in order to make a connection to anywhere.

15.5.1 Secure Connection Initialization

To initialize OPIE for the first time, execute opiepasswd(1):

```
% opiepasswd -c
[grimreaper] ~ $ opiepasswd -f -c
Adding unfurl:
Only use this method from the console; NEVER from remote. If you are using
telnet, xterm, or a dial-in, type ^C now or exit with no password.
Then run opiepasswd without the -c parameter.
Using MD5 to compute responses.
Enter new secret pass phrase:
Again new secret pass phrase:

ID unfurl OTP key is 499 to4268
MOS MALL GOAT ARM AVID COED
```

At the `Enter new secret pass phrase:` or `Enter secret password:` prompt, enter a password or phrase. This is not the login password as this password is used to generate the one-time login keys. The "ID" line gives the parameters of the instance: the login name, iteration count, and seed. When logging in, the system will remember these parameters and display them, meaning that they do not have to be memorized. The last line gives the particular one-time password which corresponds to those parameters and the secret password. At the next login, this one-time password is the one to use.

15.5.2 Insecure Connection Initialization

To initialize or change the secret password over an insecure connection, a secure connection is needed to some place where opiekey(1) can be run. This might be a shell prompt on a trusted machine. An iteration count is needed, where 100 is probably a good value, and the seed can either be specified or the randomly-generated one used. On the insecure connection, the machine being initialized, use opiepasswd(1):

```
% opiepasswd

Updating unfurl:
You need the response from an OTP generator.
Old secret pass phrase:
       otp-md5 498 to4268 ext
       Response: GAME GAG WELT OUT DOWN CHAT
New secret pass phrase:
       otp-md5 499 to4269
       Response: LINE PAP MILK NELL BUOY TROY

ID mark OTP key is 499 gr4269
LINE PAP MILK NELL BUOY TROY
```

To accept the default seed, press **Return**. Before entering an access password, move over to the secure connection and give it the same parameters:

```
% opiekey 498 to4268
Using the MD5 algorithm to compute response.
Reminder: Do not use opiekey from telnet or dial-in sessions.
Enter secret pass phrase:
GAME GAG WELT OUT DOWN CHAT
```

Switch back over to the insecure connection, and copy the generated one-time password over to the relevant program.

15.5.3 Generating a Single One-time Password

After initializing OPIE and logging in, a prompt like this will be displayed:

```
% telnet example.com
Trying 10.0.0.1...
Connected to example.com
Escape character is '^]'.

FreeBSD/i386 (example.com) (ttypa)

login: <username>
otp-md5 498 gr4269 ext
Password:
```

The OPIE prompts provides a useful feature. If **Return** is pressed at the password prompt, the prompt will turn echo on and display what is typed. This can be useful when attempting to type in a password by hand from a printout.

At this point, generate the one-time password to answer this login prompt. This must be done on a trusted system where it is safe to run opiekey(1). There are versions of this command for Windows, Mac OS and FreeBSD. This

command needs the iteration count and the seed as command line options. Use cut-and-paste from the login prompt on the machine being logged in to.

On the trusted system:

```
% opiekey 498 to4268
Using the MD5 algorithm to compute response.
Reminder: Do not use opiekey from telnet or dial-in sessions.
Enter secret pass phrase:
GAME GAG WELT OUT DOWN CHAT
```

Once the one-time password is generated, continue to log in.

15.5.4 Generating Multiple One-time Passwords

Sometimes there is no access to a trusted machine or secure connection. In this case, it is possible to use opiekey(1) to generate a number of one-time passwords beforehand. For example:

```
% opiekey -n 5 30 zz99999
Using the MD5 algorithm to compute response.
Reminder: Do not use opiekey from telnet or dial-in sessions.
Enter secret pass phrase: <secret password>
26: JOAN BORE FOSS DES NAY QUIT
27: LATE BIAS SLAY FOLK MUCH TRIG
28: SALT TIN ANTI LOON NEAL USE
29: RIO ODIN GO BYE FURY TIC
30: GREW JIVE SAN GIRD BOIL PHI
```

The `-n 5` requests five keys in sequence, and `30` specifies what the last iteration number should be. Note that these are printed out in *reverse* order of use. The really paranoid might want to write the results down by hand; otherwise, print the list. Each line shows both the iteration count and the one-time password. Scratch off the passwords as they are used.

15.5.5 Restricting Use of UNIX® Passwords

OPIE can restrict the use of UNIX passwords based on the IP address of a login session. The relevant file is `/etc/opieaccess`, which is present by default. Refer to opieaccess(5) for more information on this file and which security considerations to be aware of when using it.

Here is a sample `opieaccess`:

```
permit 192.168.0.0 255.255.0.0
```

This line allows users whose IP source address (which is vulnerable to spoofing) matches the specified value and mask, to use UNIX passwords at any time.

If no rules in `opieaccess` are matched, the default is to deny non-OPIE logins.

15.6 TCP Wrappers

Written by Tom Rhodes.

TCP Wrappers extends the abilities of Section 30.2 to provide support for every server daemon under its control. It can be configured to provide logging support, return messages to connections, and permit a daemon to only accept internal connections. While some of these features can be provided by implementing a firewall, TCP Wrappers adds an extra layer of protection and goes beyond the amount of control a firewall can provide.

TCP Wrappers should not be considered a replacement for a properly configured firewall. TCP Wrappers should be used in conjunction with a firewall and other security enhancements.

15.6.1 Initial Configuration

To enable TCP Wrappers in FreeBSD, ensure the inetd(8) server is started from `/etc/rc.conf` with `-Ww`. Then, properly configure `/etc/hosts.allow`.

> **Note:** Unlike other implementations of TCP Wrappers, the use of `hosts.deny` has been deprecated. All configuration options should be placed in `/etc/hosts.allow`.

In the simplest configuration, daemon connection policies are set to either be permitted or blocked depending on the options in `/etc/hosts.allow`. The default configuration in FreeBSD is to allow a connection to every daemon started with inetd(8).

Basic configuration usually takes the form of `daemon : address : action`, where `daemon` is the daemon which inetd(8) started, `address` is a valid hostname, IP address, or an IPv6 address enclosed in brackets ([ ]), and `action` is either `allow` or `deny`. TCP Wrappers uses a first rule match semantic, meaning that the configuration file is scanned in ascending order for a matching rule. When a match is found, the rule is applied and the search process stops.

For example, to allow POP3 connections via the `mail/qpopper` daemon, the following lines should be appended to `hosts.allow`:

```
# This line is required for POP3 connections:
qpopper : ALL : allow
```

After adding this line, inetd(8) needs to be restarted:

```
# service inetd restart
```

15.6.2 Advanced Configuration

TCP Wrappers provides advanced options to allow more control over the way connections are handled. In some cases, it may be appropriate to return a comment to certain hosts or daemon connections. In other cases, a log entry should be recorded or an email sent to the administrator. Other situations may require the use of a service for local connections only. This is all possible through the use of configuration options known as `wildcards`, expansion characters and external command execution.

15.6.2.1 External Commands

Suppose that a situation occurs where a connection should be denied yet a reason should be sent to the individual who attempted to establish that connection. That action is possible with `twist`. When a connection attempt is made, `twist` executes a shell command or script. An example exists in `hosts.allow`:

```
# The rest of the daemons are protected.
ALL : ALL \
       : severity auth.info \
       : twist /bin/echo "You are not welcome to use %d from %h."
```

In this example, the message "You are not allowed to use `daemon` from `hostname`." will be returned for any daemon not previously configured in the access file. This is useful for sending a reply back to the connection initiator right after the established connection is dropped. Any message returned *must* be wrapped in quote (") characters.

> **Warning:** It may be possible to launch a denial of service attack on the server if an attacker, or group of attackers, could flood these daemons with connection requests.

Another possibility is to use `spawn`. Like `twist`, `spawn` implicitly denies the connection and may be used to run external shell commands or scripts. Unlike `twist`, `spawn` will not send a reply back to the individual who established the connection. For example, consider the following configuration line:

```
# We do not allow connections from example.com:
ALL : .example.com \
       : spawn (/bin/echo %a from %h attempted to access %d >> \
         /var/log/connections.log) \
       : deny
```

This will deny all connection attempts from `*.example.com` and log the hostname, IP address, and the daemon to which access was attempted to `/var/log/connections.log`.

This example uses the substitution characters `%a` and `%h`. Refer to hosts_access(5) for the complete list.

15.6.2.2 Wildcard Options

The `ALL` option may be used to match every instance of a daemon, domain, or an IP address. Another wildcard is `PARANOID` which may be used to match any host which provides an IP address that may be forged. For example, `PARANOID` may be used to define an action to be taken whenever a connection is made from an IP address that differs from its hostname. In this example, all connection requests to sendmail(8) which have an IP address that varies from its hostname will be denied:

```
# Block possibly spoofed requests to sendmail:
sendmail : PARANOID : deny
```

> **Caution:** Using the `PARANOID` wildcard may severely cripple servers if the client or server has a broken DNS setup. Administrator discretion is advised.

To learn more about wildcards and their associated functionality, refer to hosts_access(5).

Before any of the specific configuration lines above will work, the first configuration line should be commented out in `hosts.allow`.

15.7 Kerberos5

Contributed by Tillman Hodgson. Based on a contribution by Mark Murray.

Kerberos is a network add-on system/protocol that allows users to authenticate themselves through the services of a secure server. **Kerberos** can be described as an identity-verifying proxy system. It can also be described as a trusted third-party authentication system. After a user authenticates with **Kerberos**, their communications can be encrypted to assure privacy and data integrity.

The only function of **Kerberos** is to provide the secure authentication of users on the network. It does not provide authorization functions (what users are allowed to do) or auditing functions (what those users did). It is recommended that **Kerberos** be used with other security methods which provide authorization and audit services.

This section provides a guide on how to set up **Kerberos** as distributed for FreeBSD. Refer to the relevant manual pages for more complete descriptions.

For purposes of demonstrating a **Kerberos** installation, the various name spaces will be as follows:

- The DNS domain (“zone”) will be `example.org`.
- The **Kerberos** realm will be `EXAMPLE.ORG`.

> **Note:** Use real domain names when setting up **Kerberos** even if it will run internally. This avoids DNS problems and assures inter-operation with other **Kerberos** realms.

15.7.1 History

Kerberos was created by MIT as a solution to network security problems. The **Kerberos** protocol uses strong cryptography so that a client can prove its identity to a server (and vice versa) across an insecure network connection.

Kerberos is both the name of a network authentication protocol and an adjective to describe programs that implement it, such as **Kerberos** telnet. The current version of the protocol is version 5, described in RFC 1510.

Several free implementations of this protocol are available, covering a wide range of operating systems. The Massachusetts Institute of Technology (MIT), where **Kerberos** was originally developed, continues to develop their **Kerberos** package. It is commonly used in the US as a cryptography product, and has historically been affected by US export regulations. The MIT **Kerberos** is available as the `security/krb5` package or port. Heimdal **Kerberos** is another version 5 implementation, and was explicitly developed outside of the US to avoid export regulations. The Heimdal **Kerberos** distribution is available as a the `security/heimdal` package or port, and a minimal installation is included in the base FreeBSD install.

These instructions assume the use of the Heimdal distribution included in FreeBSD.

15.7.2 Setting up a Heimdal KDC

The Key Distribution Center (KDC) is the centralized authentication service that **Kerberos** provides. It is the computer that issues **Kerberos** tickets. The KDC is considered "trusted" by all other computers in the **Kerberos** realm, and thus has heightened security concerns.

While running the **Kerberos** server requires very few computing resources, a dedicated machine acting only as a KDC is recommended for security reasons.

To begin setting up a KDC, ensure that `/etc/rc.conf` contains the correct settings to act as a KDC. As required, adjust paths to reflect the system:

```
kerberos5_server_enable="YES"
kadmind5_server_enable="YES"
```

Next, edit `/etc/krb5.conf` as follows:

```
[libdefaults]
    default_realm = EXAMPLE.ORG
[realms]
    EXAMPLE.ORG = {
        kdc = kerberos.example.org
        admin_server = kerberos.example.org
    }
[domain_realm]
    .example.org = EXAMPLE.ORG
```

This `/etc/krb5.conf` implies that the KDC will use the fully-qualified hostname `kerberos.example.org`. Add a CNAME (alias) entry to the zone file to accomplish this if the KDC has a different hostname.

> **Note:** For large networks with a properly configured DNS server, the above example could be trimmed to:
>
> ```
> [libdefaults]
>       default_realm = EXAMPLE.ORG
> ```
>
> With the following lines being appended to the `example.org` zone file:
>
> ```
> _kerberos._udp      IN  SRV     01 00 88 kerberos.example.org.
> _kerberos._tcp      IN  SRV     01 00 88 kerberos.example.org.
> _kpasswd._udp       IN  SRV     01 00 464 kerberos.example.org.
> _kerberos-adm._tcp  IN  SRV     01 00 749 kerberos.example.org.
> _kerberos           IN  TXT     EXAMPLE.ORG
> ```

> **Note:** For clients to be able to find the **Kerberos** services, it *must* have either a fully configured `/etc/krb5.conf` or a minimally configured `/etc/krb5.conf` *and* a properly configured DNS server.

Next, create the **Kerberos** database which contains the keys of all principals encrypted with a master password. It is not required to remember this password as it will be stored in `/var/heimdal/m-key`. To create the master key, run kstash(8) and enter a password.

Once the master key has been created, initialize the database using `kadmin -l`. This option instructs kadmin(8) to modify the local database files directly rather than going through the kadmind(8) network service. This handles the chicken-and-egg problem of trying to connect to the database before it is created. At the kadmin(8) prompt, use `init` to create the realm's initial database.

Lastly, while still in kadmin(8), create the first principal using `add`. Stick to the default options for the principal for now, as these can be changed later with `modify`. Type `?` at the kadmin(8) prompt to see the available options.

A sample database creation session is shown below:

```
# kstash
Master key: xxxxxxxx
Verifying password - Master key: xxxxxxxx

# kadmin -l
kadmin> init EXAMPLE.ORG
Realm max ticket life [unlimited]:
kadmin> add tillman
Max ticket life [unlimited]:
Max renewable life [unlimited]:
Attributes []:
Password: xxxxxxxx
Verifying password - Password: xxxxxxxx
```

Next, start the KDC services. Run `service kerberos start` and `service kadmind start` to bring up the services. While there will not be any kerberized daemons running at this point, it is possible to confirm that the KDC is functioning by obtaining and listing a ticket for the principal (user) that was just created from the command-line of the KDC itself:

```
% kinit tillman
tillman@EXAMPLE.ORG's Password:

% klist
Credentials cache: FILE:/tmp/krb5cc_500
       Principal: tillman@EXAMPLE.ORG

  Issued           Expires          Principal
Aug 27 15:37:58  Aug 28 01:37:58  krbtgt/EXAMPLE.ORG@EXAMPLE.ORG
```

The ticket can then be revoked when finished:

```
% kdestroy
```

15.7.3 Kerberos Enabling a Server with Heimdal Services

First, copy `/etc/krb5.conf` from the KDC to the client computer in a secure fashion, such as scp(1), or physically via a removable media.

Next, create `/etc/krb5.keytab`. This is the major difference between a server providing **Kerberos** enabled daemons and a workstation: the server must have a `keytab`. This file contains the server's host key, which allows it and the KDC to verify each others identity. It must be transmitted to the server in a secure fashion, as the security of the server can be broken if the key is made public.

Typically, the `keytab` is transferred to the server using kadmin(8). This is handy because the host principal, the KDC end of the `krb5.keytab`, is also created using kadmin(8).

A ticket must already be obtained and this ticket must be allowed to use the kadmin(8) interface in the `kadmind.acl`. See the section titled “Remote administration” in`info heimdal` for details on designing access control lists. Instead of enabling remote kadmin(8) access, the administrator can securely connect to the KDC via the local console or ssh(1), and perform administration locally using `kadmin -l`.

After installing `/etc/krb5.conf`, use `add --random-key` from the **Kerberos** server. This adds the server’s host principal. Then, use `ext` to extract the server’s host principal to its own keytab. For example:

```
# kadmin
kadmin> add --random-key host/myserver.example.org
Max ticket life [unlimited]:
Max renewable life [unlimited]:
Attributes []:
kadmin> ext host/myserver.example.org
kadmin> exit
```

Note that `ext` stores the extracted key in `/etc/krb5.keytab` by default.

If kadmind(8) is not running on the KDC and there is no access to kadmin(8) remotely, add the host principal (`host/myserver.EXAMPLE.ORG`) directly on the KDC and then extract it to a temporary file to avoid overwriting the `/etc/krb5.keytab` on the KDC, using something like this:

```
# kadmin
kadmin> ext --keytab=/tmp/example.keytab host/myserver.example.org
kadmin> exit
```

The keytab can then be securely copied to the server using scp(1) or a removable media. Be sure to specify a non-default keytab name to avoid overwriting the keytab on the KDC.

At this point, the server can communicate with the KDC using `krb5.conf` and it can prove its own identity with `krb5.keytab`. It is now ready for the **Kerberos** services to be enabled. For this example, the telnetd(8) service is enabled in `/etc/inetd.conf` and inetd(8) has been restarted with `service inetd restart`:

```
telnet    stream  tcp     nowait  root    /usr/libexec/telnetd  telnetd -a user
```

The critical change is that the `-a` authentication type is set to user. Refer to telnetd(8) for more details.

15.7.4 Kerberos Enabling a Client with Heimdal

Setting up a client computer is easy as only `/etc/krb5.conf` is needed. Securely copy this file over to the client computer from the KDC.

Test the client by attempting to use kinit(1), klist(1), and kdestroy(1) from the client to obtain, show, and then delete a ticket for the principal created above. **Kerberos** applications should also be able to connect to **Kerberos** enabled servers. If that does not work but obtaining a ticket does, the problem is likely with the server and not with the client or the KDC.

When testing a Kerberized application, try using a packet sniffer such as tcpdump(1) to confirm that the password is not sent in the clear.

Various non-core **Kerberos** client applications are available. The “minimal” installation in FreeBSD installs telnetd(8) as the only **Kerberos** enabled service.

The Heimdal port installs **Kerberos** enabled versions of ftpd(8), rshd(8), rcp(1), rlogind(8), and a few other less common programs. The MIT port also contains a full suite of **Kerberos** client applications.

15.7.5 User Configuration Files: `.k5login` and `.k5users`

Users within a realm typically have their **Kerberos** principal mapped to a local user account. Occasionally, one needs to grant access to a local user account to someone who does not have a matching **Kerberos** principal. For example, `tillman@EXAMPLE.ORG` may need access to the local user account `webdevelopers`. Other principals may also need access to that local account.

The `.k5login` and `.k5users` files, placed in a user’s home directory, can be used to solve this problem. For example, if `.k5login` with the following contents is placed in the home directory of `webdevelopers`, both principals listed will have access to that account without requiring a shared password.:

```
tillman@example.org
jdoe@example.org
```

Refer to ksu(1) for more information about `.k5users`.

15.7.6 Kerberos Tips, Tricks, and Troubleshooting

- When using either the Heimdal or MIT **Kerberos** ports, ensure that the `PATH` lists the **Kerberos** versions of the client applications before the system versions.
- If all the computers in the realm do not have synchronized time settings, authentication may fail. Section 30.11 describes how to synchronize clocks using NTP.
- MIT and Heimdal interoperate except for kadmin(8), which is not standardized.
- If the hostname is changed, the `host/` principal must be changed and the keytab updated. This also applies to special keytab entries like the `www/` principal used for Apache’s `www/mod_auth_kerb`.
- All hosts in the realm must be both forward and reverse resolvable in DNS or, at a minimum, in `/etc/hosts`. CNAMEs will work, but the A and PTR records must be correct and in place. The error message for unresolvable hosts is not intuitive: `Kerberos5 refuses authentication because Read req failed: Key table entry not found`.
- Some operating systems that act as clients to the KDC do not set the permissions for ksu(1) to be setuid `root`. This means that ksu(1) does not work. This is not a KDC error.
- With MIT **Kerberos**, in order to allow a principal to have a ticket life longer than the default ten hours, use `modify_principal` at the kadmin(8) prompt to change the maxlife of both the principal in question and the `krbtgt` principal. Then the principal can use `kinit -l` to request a ticket with a longer lifetime.
-

  Note: When running a packet sniffer on the KDC to aid in troubleshooting while running kinit(1) from a workstation, the Ticket Granting Ticket (TGT) is sent immediately upon running kinit(1), even before the password is typed. This is because the **Kerberos** server freely transmits a TGT to any unauthorized request.

However, every TGT is encrypted in a key derived from the user's password. When a user types their password, it is not sent to the KDC, it is instead used to decrypt the TGT that kinit(1) already obtained. If the decryption process results in a valid ticket with a valid time stamp, the user has valid **Kerberos** credentials. These credentials include a session key for establishing secure communications with the **Kerberos** server in the future, as well as the actual TGT, which is encrypted with the **Kerberos** server's own key. This second layer of encryption allows the **Kerberos** server to verify the authenticity of each TGT.

- To use long ticket lifetimes, such as a week, when using **OpenSSH** to connect to the machine where the ticket is stored, make sure that **Kerberos** `TicketCleanup` is set to `no` in `sshd_config` or else tickets will be deleted at log out.
- Host principals can have a longer ticket lifetime. If the user principal has a lifetime of a week but the host being connected to has a lifetime of nine hours, the user cache will have an expired host principal and the ticket cache will not work as expected.
- When setting up `krb5.dict` to prevent specific bad passwords from being used as described in kadmind(8), remember that it only applies to principals that have a password policy assigned to them. The format used in `krb5.dict` is one string per line. Creating a symbolic link to `/usr/share/dict/words` might be useful.

15.7.7 Differences with the MIT Port

The major difference between MIT and Heimdal relates to kadmin(8) which has a different, but equivalent, set of commands and uses a different protocol. If the KDC is MIT, the Heimdal version of kadmin(8) cannot be used to administer the KDC remotely, and vice versa.

The client applications may also use slightly different command line options to accomplish the same tasks. Following the instructions on the MIT **Kerberos** web site (http://web.mit.edu/Kerberos/www/) is recommended. Be careful of path issues: the MIT port installs into `/usr/local/` by default, and the "normal" system applications run instead of MIT versions if `PATH` lists the system directories first.

Note: With the FreeBSD MIT `security/krb5` port, be sure to read `/usr/local/share/doc/krb5/README.FreeBSD` installed by the port to understand why logins via telnetd(8) and `klogind` behave somewhat oddly. Correcting the "incorrect permissions on cache file" behavior requires that the `login.krb5` binary be used for authentication so that it can properly change ownership for the forwarded credentials.

The following edits should also be made to `rc.conf`:

```
kerberos5_server="/usr/local/sbin/krb5kdc"
kadmind5_server="/usr/local/sbin/kadmind"
kerberos5_server_enable="YES"
kadmind5_server_enable="YES"
```

This is done because the applications for MIT Kerberos installs binaries in the `/usr/local` hierarchy.

15.7.8 Mitigating Limitations Found in Kerberos

15.7.8.1 Kerberos is an All or Nothing Approach

Every service enabled on the network must be modified to work with **Kerberos**, or be otherwise secured against network attacks, or else the user's credentials could be stolen and re-used. An example of this would be **Kerberos** enabling all remote shells but not converting the POP3 mail server which sends passwords in plain text.

15.7.8.2 Kerberos is Intended for Single-User Workstations

In a multi-user environment, **Kerberos** is less secure. This is because it stores the tickets in `/tmp`, which is readable by all users. If a user is sharing a computer with other users, it is possible that the user's tickets can be stolen or copied by another user.

This can be overcome with the `-c` command-line option or, preferably, the `KRB5CCNAME` environment variable. Storing the ticket in the user's home directory and using file permissions are commonly used to mitigate this problem.

15.7.8.3 The KDC is a Single Point of Failure

By design, the KDC must be as secure as its master password database. The KDC should have absolutely no other services running on it and should be physically secure. The danger is high because **Kerberos** stores all passwords encrypted with the same "master" key which is stored as a file on the KDC.

A compromised master key is not quite as bad as one might fear. The master key is only used to encrypt the **Kerberos** database and as a seed for the random number generator. As long as access to the KDC is secure, an attacker cannot do much with the master key.

Additionally, if the KDC is unavailable, network services are unusable as authentication cannot be performed. This can be alleviated with a single master KDC and one or more slaves, and with careful implementation of secondary or fall-back authentication using PAM.

15.7.8.4 Kerberos Shortcomings

Kerberos allows users, hosts and services to authenticate between themselves. It does not have a mechanism to authenticate the KDC to the users, hosts or services. This means that a trojanned kinit(1) could record all user names and passwords. Filesystem integrity checking tools like `security/tripwire` can alleviate this.

15.7.9 Resources and Further Information

- The **Kerberos** FAQ (http://www.faqs.org/faqs/Kerberos-faq/general/preamble.html)
- Designing an Authentication System: a Dialog in Four Scenes (http://web.mit.edu/Kerberos/www/dialogue.html)
- RFC 1510, The **Kerberos** Network Authentication Service (V5) (http://www.ietf.org/rfc/rfc1510.txt?number=1510)
- MIT **Kerberos** home page (http://web.mit.edu/Kerberos/www/)
- Heimdal **Kerberos** home page (http://www.pdc.kth.se/heimdal/)

15.8 OpenSSL

Written by Tom Rhodes.

The **OpenSSL** toolkit is included in FreeBSD. It provides an encryption transport layer on top of the normal communications layer, allowing it to be intertwined with many network applications and services.

Some uses of **OpenSSL** may include encrypted authentication of mail clients and web based transactions such as credit card payments. Many ports such as `www/apache22`, and `mail/claws-mail` offer compilation support for building with **OpenSSL**.

> **Note:** In most cases, the Ports Collection will attempt to build the `security/openssl` port unless `WITH_OPENSSL_BASE` is explicitly set to “yes”.

The version of **OpenSSL** included in FreeBSD supports Secure Sockets Layer v2/v3 (SSLv2/SSLv3) and Transport Layer Security v1 (TLSv1) network security protocols and can be used as a general cryptographic library.

> **Note:** While **OpenSSL** supports the IDEA algorithm, it is disabled by default due to United States patents. To use it, the license should be reviewed and, if the restrictions are acceptable, the `MAKE_IDEA` variable must be set in `/etc/make.conf`.

One of the most common uses of **OpenSSL** is to provide certificates for use with software applications. These certificates ensure that the credentials of the company or individual are valid and not fraudulent. If the certificate in question has not been verified by a “Certificate Authority” (CA), a warning is produced. A CA is a company, such as VeriSign (http://www.verisign.com), signs certificates in order to validate the credentials of individuals or companies. This process has a cost associated with it and is not a requirement for using certificates; however, it can put users at ease.

15.8.1 Generating Certificates

To generate a certificate, the following command is available:

```
# openssl req -new -nodes -out req.pem -keyout cert.pem
Generating a 1024 bit RSA private key
..............++++++
.....................................++++++
writing new private key to 'cert.pem'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:PA
Locality Name (eg, city) []:Pittsburgh
Organization Name (eg, company) [Internet Widgits Pty Ltd]:My Company
```

```
Organizational Unit Name (eg, section) []:Systems Administrator
Common Name (eg, YOUR name) []:localhost.example.org
Email Address []:trhodes@FreeBSD.org

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:SOME PASSWORD
An optional company name []:Another Name
```

Notice the response directly after the "Common Name" prompt shows a domain name. This prompt requires a server name to be entered for verification purposes and placing anything but a domain name yields a useless certificate. Other options, such as the expire time and alternate encryption algorithms, are available. A complete list of options is described in openssl(1).

Two files should now exist in the directory in which this command was issued. The certificate request, `req.pem`, may be sent to a CA who will validate the entered credentials, sign the request, and return the signed certificate. The second file is named `cert.pem` and is the private key for the certificate and should be protected at all costs. If this falls in the hands of others it can be used to impersonate the user or the server.

In cases where a signature from a CA is not required, a self signed certificate can be created. First, generate the RSA key:

```
# openssl dsaparam -rand -genkey -out myRSA.key 1024
```

Next, generate the CA key:

```
# openssl gendsa -des3 -out myca.key myRSA.key
```

Use this key to create the certificate:

```
# openssl req -new -x509 -days 365 -key myca.key -out new.crt
```

Two new files should appear in the directory: a certificate authority signature file, `myca.key` and the certificate itself, `new.crt`. These should be placed in a directory, preferably under `/etc`, which is readable only by `root`. Permissions of 0700 are appropriate and can be set using chmod(1).

15.8.2 Using Certificates

One use for a certificate is to encrypt connections to the **Sendmail** MTA. This prevents the use of clear text authentication for users who send mail via the local MTA.

> **Note:** Some MUAs will display error if the user has not installed the certificate locally. Refer to the documentation included with the software for more information on certificate installation.

To configure **Sendmail**, the following lines should be placed in the local `.mc` file:

```
dnl SSL Options
define('confCACERT_PATH','/etc/certs')dnl
define('confCACERT','/etc/certs/new.crt')dnl
define('confSERVER_CERT','/etc/certs/new.crt')dnl
define('confSERVER_KEY','/etc/certs/myca.key')dnl
```

```
define(‘confTLS_SRV_OPTIONS’, ‘V’)dnl
```

In this example, `/etc/certs/` stores the certificate and key files locally. After saving the edits, rebuild the local `.cf` file by typing `make install` within `/etc/mail`. Follow that up with `make restart` which should start the **Sendmail** daemon.

If all went well, there will be no error messages in `/var/log/maillog` and **Sendmail** will show up in the process list.

For a simple test, connect to the mail server using telnet(1):

```
# telnet example.com 25
Trying 192.0.34.166...
Connected to example.com.
Escape character is ’^]’.
220 example.com ESMTP Sendmail 8.12.10/8.12.10; Tue, 31 Aug 2004 03:41:22 -0400 (EDT)
ehlo example.com
250-example.com Hello example.com [192.0.34.166], pleased to meet you
250-ENHANCEDSTATUSCODES
250-PIPELINING
250-8BITMIME
250-SIZE
250-DSN
250-ETRN
250-AUTH LOGIN PLAIN
250-STARTTLS
250-DELIVERBY
250 HELP
quit
221 2.0.0 example.com closing connection
Connection closed by foreign host.
```

If the “STARTTLS” line appears in the output, everything is working correctly.

15.9 VPN over IPsec

Written by Nik Clayton.

15.9.1 Understanding IPsec

Written by Hiten M. Pandya.

This section demonstrates the process of setting up IPsec. It assumes familiarity with the concepts of building a custom kernel (see Chapter 9).

IPsec is a protocol which sits on top of the Internet Protocol (IP) layer. It allows two or more hosts to communicate in a secure manner. The FreeBSD IPsec “network stack” is based on the KAME (http://www.kame.net/) implementation, which has support for both IPv4 and IPv6.

IPsec consists of two sub-protocols:

- *Encapsulated Security Payload ESP)*: this protocol protects the IP packet data from third party interference by encrypting the contents using symmetric cryptography algorithms such as Blowfish and 3DES.
- *Authentication Header (AH)*: this protocol protects the IP packet header from third party interference and spoofing by computing a cryptographic checksum and hashing the IP packet header fields with a secure hashing function. This is then followed by an additional header that contains the hash, to allow the information in the packet to be authenticated.

ESP and AH can either be used together or separately, depending on the environment.

IPsec can either be used to directly encrypt the traffic between two hosts using *Transport Mode* or to build "virtual tunnels" using *Tunnel Mode*. The latter mode is more commonly known as a *Virtual Private Network (VPN)*. Consult ipsec(4) for detailed information on the IPsec subsystem in FreeBSD.

To add IPsec support to the kernel, add the following options to the custom kernel configuration file:

```
options   IPSEC        #IP security
device    crypto
```

If IPsec debugging support is desired, the following kernel option should also be added:

```
options   IPSEC_DEBUG  #debug for IP security
```

15.9.2 VPN Between a Home and Corporate Network

There is no standard for what constitutes a VPN. VPNs can be implemented using a number of different technologies, each of which has their own strengths and weaknesses. This section presents the strategies used for implementing a VPN for the following scenario:

- There are at least two sites where each site is using IP internally.
- Both sites are connected to the Internet through a gateway that is running FreeBSD.
- The gateway on each network has at least one public IP address.
- The internal addresses of the two networks can be either public or private IP addresses. However, the address space must not collide. For example, both networks cannot use `192.168.1.x`.

15.9.2.1 Configuring IPsec on FreeBSD

Written by Tom Rhodes.

To begin, `security/ipsec-tools` must be installed from the Ports Collection. This software provides a number of applications which support the configuration.

The next requirement is to create two gif(4) pseudo-devices which will be used to tunnel packets and allow both networks to communicate properly. As `root`, run the following commands, replacing *`internal`* and *`external`* with the real IP addresses of the internal and external interfaces of the two gateways:

```
# ifconfig gif0 create

# ifconfig gif0 internal1 internal2

# ifconfig gif0 tunnel external1 external2
```

In this example, the corporate LAN's external IP address is `172.16.5.4` and its internal IP address is `10.246.38.1`. The home LAN's external IP address is `192.168.1.12` and its internal private IP address is `10.0.0.5`.

If this is confusing, review the following example output from ifconfig(8):

```
Gateway 1:

gif0: flags=8051 mtu 1280
tunnel inet 172.16.5.4 --> 192.168.1.12
inet6 fe80::2e0:81ff:fe02:5881%gif0 prefixlen 64 scopeid 0x6
inet 10.246.38.1 --> 10.0.0.5 netmask 0xffffff00

Gateway 2:

gif0: flags=8051 mtu 1280
tunnel inet 192.168.1.12 --> 172.16.5.4
inet 10.0.0.5 --> 10.246.38.1 netmask 0xffffff00
inet6 fe80::250:bfff:fe3a:c1f%gif0 prefixlen 64 scopeid 0x4
```

Once complete, both internal IP addresses should be reachable using ping(8):

```
priv-net# ping 10.0.0.5
PING 10.0.0.5 (10.0.0.5): 56 data bytes
64 bytes from 10.0.0.5: icmp_seq=0 ttl=64 time=42.786 ms
64 bytes from 10.0.0.5: icmp_seq=1 ttl=64 time=19.255 ms
64 bytes from 10.0.0.5: icmp_seq=2 ttl=64 time=20.440 ms
64 bytes from 10.0.0.5: icmp_seq=3 ttl=64 time=21.036 ms
--- 10.0.0.5 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 19.255/25.879/42.786/9.782 ms

corp-net# ping 10.246.38.1
PING 10.246.38.1 (10.246.38.1): 56 data bytes
64 bytes from 10.246.38.1: icmp_seq=0 ttl=64 time=28.106 ms
64 bytes from 10.246.38.1: icmp_seq=1 ttl=64 time=42.917 ms
64 bytes from 10.246.38.1: icmp_seq=2 ttl=64 time=127.525 ms
64 bytes from 10.246.38.1: icmp_seq=3 ttl=64 time=119.896 ms
64 bytes from 10.246.38.1: icmp_seq=4 ttl=64 time=154.524 ms
--- 10.246.38.1 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 28.106/94.594/154.524/49.814 ms
```

As expected, both sides have the ability to send and receive ICMP packets from the privately configured addresses. Next, both gateways must be told how to route packets in order to correctly send traffic from either network. The following command will achieve this goal:

```
# corp-net# route add 10.0.0.0 10.0.0.5 255.255.255.0

# corp-net# route add net 10.0.0.0: gateway 10.0.0.5

# priv-net# route add 10.246.38.0 10.246.38.1 255.255.255.0

# priv-net# route add host 10.246.38.0: gateway 10.246.38.1
```

At this point, internal machines should be reachable from each gateway as well as from machines behind the gateways. Again, use ping(8) to confirm:

```
corp-net# ping 10.0.0.8
PING 10.0.0.8 (10.0.0.8): 56 data bytes
64 bytes from 10.0.0.8: icmp_seq=0 ttl=63 time=92.391 ms
64 bytes from 10.0.0.8: icmp_seq=1 ttl=63 time=21.870 ms
64 bytes from 10.0.0.8: icmp_seq=2 ttl=63 time=198.022 ms
64 bytes from 10.0.0.8: icmp_seq=3 ttl=63 time=22.241 ms
64 bytes from 10.0.0.8: icmp_seq=4 ttl=63 time=174.705 ms
--- 10.0.0.8 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 21.870/101.846/198.022/74.001 ms

priv-net# ping 10.246.38.107
PING 10.246.38.1 (10.246.38.107): 56 data bytes
64 bytes from 10.246.38.107: icmp_seq=0 ttl=64 time=53.491 ms
64 bytes from 10.246.38.107: icmp_seq=1 ttl=64 time=23.395 ms
64 bytes from 10.246.38.107: icmp_seq=2 ttl=64 time=23.865 ms
64 bytes from 10.246.38.107: icmp_seq=3 ttl=64 time=21.145 ms
64 bytes from 10.246.38.107: icmp_seq=4 ttl=64 time=36.708 ms
--- 10.246.38.107 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 21.145/31.721/53.491/12.179 ms
```

Setting up the tunnels is the easy part. Configuring a secure link is a more in depth process. The following configuration uses pre-shared (PSK) RSA keys. Other than the IP addresses, the `/usr/local/etc/racoon/racoon.conf` on both gateways will be identical and look similar to:

```
path    pre_shared_key  "/usr/local/etc/racoon/psk.txt"; #location of pre-shared key file
log     debug; #log verbosity setting: set to 'notify' when testing and debugging is complete

padding        # options are not to be changed
{
        maximum_length  20;
        randomize       off;
        strict_check    off;
        exclusive_tail  off;
}

timer  # timing options. change as needed
{
        counter         5;
        interval        20 sec;
        persend         1;
#       natt_keepalive  15 sec;
        phase1          30 sec;
        phase2          15 sec;
}

listen # address [port] that racoon will listening on
{
        isakmp          172.16.5.4 [500];
```

```
        isakmp_natt     172.16.5.4 [4500];
}

remote  192.168.1.12 [500]
{
        exchange_mode   main,aggressive;
        doi             ipsec_doi;
        situation       identity_only;
        my_identifier   address 172.16.5.4;
        peers_identifier        address 192.168.1.12;
        lifetime        time 8 hour;
        passive         off;
        proposal_check  obey;
#       nat_traversal   off;
        generate_policy off;

                        proposal {
                                encryption_algorithm    blowfish;
                                hash_algorithm          md5;
                                authentication_method   pre_shared_key;
                                lifetime time           30 sec;
                                dh_group                1;
                        }
}

sainfo  (address 10.246.38.0/24 any address 10.0.0.0/24 any)   # address $network/$netmask $type a
{                                                               # $network must be the two interna
        pfs_group       1;
        lifetime        time    36000 sec;
        encryption_algorithm    blowfish,3des,des;
        authentication_algorithm        hmac_md5,hmac_sha1;
        compression_algorithm   deflate;
}
```

For descriptions of each available option, refer to the manual page for `racoon.conf`.

The Security Policy Database (SPD) needs to be configured so that FreeBSD and **racoon** are able to encrypt and decrypt network traffic between the hosts.

This can be achieved with a shell script, similar to the following, on the corporate gateway. This file will be used during system initialization and should be saved as `/usr/local/etc/racoon/setkey.conf`.

```
flush;
spdflush;
# To the home network
spdadd 10.246.38.0/24 10.0.0.0/24 any -P out ipsec esp/tunnel/172.16.5.4-192.168.1.12/use;
spdadd 10.0.0.0/24 10.246.38.0/24 any -P in ipsec esp/tunnel/192.168.1.12-172.16.5.4/use;
```

Once in place, **racoon** may be started on both gateways using the following command:

```
# /usr/local/sbin/racoon -F -f /usr/local/etc/racoon/racoon.conf -l /var/log/racoon.log
```

The output should be similar to the following:

```
corp-net# /usr/local/sbin/racoon -F -f /usr/local/etc/racoon/racoon.conf
Foreground mode.
2006-01-30 01:35:47: INFO: begin Identity Protection mode.
2006-01-30 01:35:48: INFO: received Vendor ID: KAME/racoon
2006-01-30 01:35:55: INFO: received Vendor ID: KAME/racoon
2006-01-30 01:36:04: INFO: ISAKMP-SA established 172.16.5.4[500]-192.168.1.12[500] spi:623b9b3bd2
2006-01-30 01:36:05: INFO: initiate new phase 2 negotiation: 172.16.5.4[0]192.168.1.12[0]
2006-01-30 01:36:09: INFO: IPsec-SA established: ESP/Tunnel 192.168.1.12[0]->172.16.5.4[0] spi=28
2006-01-30 01:36:09: INFO: IPsec-SA established: ESP/Tunnel 172.16.5.4[0]->192.168.1.12[0] spi=47
2006-01-30 01:36:13: INFO: respond new phase 2 negotiation: 172.16.5.4[0]192.168.1.12[0]
2006-01-30 01:36:18: INFO: IPsec-SA established: ESP/Tunnel 192.168.1.12[0]->172.16.5.4[0] spi=12
2006-01-30 01:36:18: INFO: IPsec-SA established: ESP/Tunnel 172.16.5.4[0]->192.168.1.12[0] spi=17
```

To ensure the tunnel is working properly, switch to another console and use tcpdump(1) to view network traffic using the following command. Replace `em0` with the network interface card as required:

```
# tcpdump -i em0 host 172.16.5.4 and dst 192.168.1.12
```

Data similar to the following should appear on the console. If not, there is an issue and debugging the returned data will be required.

```
01:47:32.021683 IP corporatenetwork.com > 192.168.1.12.privatenetwork.com: ESP(spi=0x02acbf9f,seq
01:47:33.022442 IP corporatenetwork.com > 192.168.1.12.privatenetwork.com: ESP(spi=0x02acbf9f,seq
01:47:34.024218 IP corporatenetwork.com > 192.168.1.12.privatenetwork.com: ESP(spi=0x02acbf9f,seq
```

At this point, both networks should be available and seem to be part of the same network. Most likely both networks are protected by a firewall. To allow traffic to flow between them, rules need to be added to pass packets. For the ipfw(8) firewall, add the following lines to the firewall configuration file:

```
ipfw add 00201 allow log esp from any to any
ipfw add 00202 allow log ah from any to any
ipfw add 00203 allow log ipencap from any to any
ipfw add 00204 allow log udp from any 500 to any
```

Note: The rule numbers may need to be altered depending on the current host configuration.

For users of pf(4) or ipf(8), the following rules should do the trick:

```
pass in quick proto esp from any to any
pass in quick proto ah from any to any
pass in quick proto ipencap from any to any
pass in quick proto udp from any port = 500 to any port = 500
pass in quick on gif0 from any to any
pass out quick proto esp from any to any
pass out quick proto ah from any to any
pass out quick proto ipencap from any to any
pass out quick proto udp from any port = 500 to any port = 500
pass out quick on gif0 from any to any
```

Finally, to allow the machine to start support for the VPN during system initialization, add the following lines to `/etc/rc.conf`:

```
ipsec_enable="YES"
ipsec_program="/usr/local/sbin/setkey"
ipsec_file="/usr/local/etc/racoon/setkey.conf" # allows setting up spd policies on boot
racoon_enable="yes"
```

15.10 OpenSSH

Contributed by Chern Lee.

OpenSSH is a set of network connectivity tools used to access remote machines securely. Additionally, TCP/IP connections can be tunneled/forwarded securely through SSH connections. **OpenSSH** encrypts all traffic to effectively eliminate eavesdropping, connection hijacking, and other network-level attacks.

OpenSSH is maintained by the OpenBSD project and is installed by default in FreeBSD. It is compatible with both SSH version 1 and 2 protocols.

15.10.1 Advantages of Using OpenSSH

When data is sent over the network in an unencrypted form, network sniffers anywhere in between the client and server can steal user/password information or data transferred during the session. **OpenSSH** offers a variety of authentication and encryption methods to prevent this from happening.

15.10.2 Enabling The SSH Server

To see if sshd(8) is enabled, check `/etc/rc.conf` for this line:

```
sshd_enable="YES"
```

This will start sshd(8), the daemon program for **OpenSSH**, the next time the system initializes. Alternatively, it is possible to use service(8) to start **OpenSSH** now:

```
# service sshd start
```

15.10.3 The SSH Client

To use ssh(1) to connect to a system running sshd(8), specify the username and host to log into:

```
# ssh user@example.com
Host key not found from the list of known hosts.
Are you sure you want to continue connecting (yes/no)? yes
Host 'example.com' added to the list of known hosts.
user@example.com's password: *******
```

SSH utilizes a key fingerprint system to verify the authenticity of the server when the client connects. The user is prompted to type `yes` when connecting for the first time. Future attempts to login are verified against the saved

fingerprint key and the ssh(1) client will display an alert if the saved fingerprint differs from the received fingerprint on future login attempts. The fingerprints are saved in `~/.ssh/known_hosts`.

By default, recent versions of sshd(8) only accept SSH v2 connections. The client will use version 2 if possible and will fall back to version 1. The client can also be forced to use one or the other by passing it the `-1` or `-2` for version 1 or version 2, respectively. The version 1 compatibility is maintained in the client for backwards compatibility with older versions.

15.10.4 Secure Copy

Use scp(1) to copy a file to or from a remote machine in a secure fashion.

```
#  scp user@example.com:/COPYRIGHT COPYRIGHT
user@example.com's password: *******
COPYRIGHT            100% |*****************************|  4735
00:00
#
```

Since the fingerprint was already saved for this host in the previous example, it is verified when using scp(1) here.

The arguments passed to scp(1) are similar to cp(1), with the file or files to copy in the first argument, and the destination in the second. Since the file is fetched over the network, through an SSH, connection, one or more of the file arguments takes the form `user@host:<path_to_remote_file>`.

15.10.5 Configuration

The system-wide configuration files for both the **OpenSSH** daemon and client reside in `/etc/ssh`.

`ssh_config` configures the client settings, while `sshd_config` configures the daemon. Each file has its own manual page which describes the available configuration options.

15.10.6 ssh-keygen(1)

Instead of using passwords, ssh-keygen(1) can be used to generate DSA or RSA keys to authenticate a user:

```
% ssh-keygen -t dsa
Generating public/private dsa key pair.
Enter file in which to save the key (/home/user/.ssh/id_dsa):
Created directory '/home/user/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/user/.ssh/id_dsa.
Your public key has been saved in /home/user/.ssh/id_dsa.pub.
The key fingerprint is:
bb:48:db:f2:93:57:80:b6:aa:bc:f5:d5:ba:8f:79:17 user@host.example.com
```

ssh-keygen(1) will create a public and private key pair for use in authentication. The private key is stored in `~/.ssh/id_dsa` or `~/.ssh/id_rsa`, whereas the public key is stored in `~/.ssh/id_dsa.pub` or `~/.ssh/id_rsa.pub`, respectively for the DSA and RSA key types. The public key must be placed in the `~/.ssh/authorized_keys` file of the remote machine for both RSA or DSA keys in order for the setup to work.

This setup allows connections to the remote machine based upon SSH keys instead of passwords.

> **Warning:** Many users believe that keys are secure by design and will use a key without a passphrase. This is *dangerous* behavior and the method an administrator may use to verify keys have a passphrase is to view the key manually. If the private key file contains the word `ENCRYPTED` the key owner is using a passphrase. While it may still be a weak passphrase, at least if the system is compromised, access to other sites will still require some level of password guessing. In addition, to better secure end users, the `from` may be placed in the public key file. For example, adding `from="192.168.10.5` in the front of `ssh-rsa` or `rsa-dsa` prefix will only allow that specific user to login from that host IP.

If a passphrase is used in ssh-keygen(1), the user will be prompted for the passphrase each time in order to use the private key. ssh-agent(1) can alleviate the strain of repeatedly entering long passphrases, and is explored in Section 15.10.7.

> **Warning:** The various options and files can be different according to the **OpenSSH** version. To avoid problems, consult ssh-keygen(1).

15.10.7 Using SSH Agent To Cache Keys

To load SSH keys into memory for use, without needing to type the passphrase each time, use ssh-agent(1) and ssh-add(1).

Authentication is handled by ssh-agent(1), using the private key(s) that are loaded into it. Then, ssh-agent(1) should be used to launch another application. At the most basic level, it could spawn a shell or a window manager.

To use ssh-agent(1) in a shell, start it with a shell as an argument. Next, add the identity by running ssh-add(1) and providing it the passphrase for the private key. Once these steps have been completed, the user will be able to ssh(1) to any host that has the corresponding public key installed. For example:

```
% ssh-agent csh
% ssh-add
Enter passphrase for /home/user/.ssh/id_dsa:
Identity added: /home/user/.ssh/id_dsa (/home/user/.ssh/id_dsa)
%
```

To use ssh-agent(1) in **Xorg**, a call to ssh-agent(1) needs to be placed in `~/.xinitrc`. This provides the ssh-agent(1) services to all programs launched in **Xorg**. An example `~/.xinitrc` file might look like this:

```
exec ssh-agent startxfce4
```

This launches ssh-agent(1), which in turn launches **XFCE**, every time **Xorg** starts. Once **Xorg** has been restarted so that the changes can take effect, run ssh-add(1) to load all of the SSH keys.

15.10.8 SSH Tunneling

OpenSSH has the ability to create a tunnel to encapsulate another protocol in an encrypted session.

The following command tells ssh(1) to create a tunnel for telnet(1):

```
% ssh -2 -N -f -L 5023:localhost:23 user@foo.example.com
%
```

This example uses the following options:

`-2`

Forces ssh(1) to use version 2 to connect to the server.

`-N`

Indicates no command, or tunnel only. If omitted, ssh(1) initiates a normal session.

`-f`

Forces ssh(1) to run in the background.

`-L`

Indicates a local tunnel in *`localport:remotehost:remoteport`* format.

`user@foo.example.com`

The login name to use on the specified remote SSH server.

An SSH tunnel works by creating a listen socket on `localhost` on the specified port. It then forwards any connections received on the local host/port via the SSH connection to the specified remote host and port.

In the example, port *`5023`* on `localhost` is forwarded to port *`23`* on `localhost` of the remote machine. Since *`23`* is used by telnet(1), this creates an encrypted telnet(1) session through an SSH tunnel.

This can be used to wrap any number of insecure TCP protocols such as SMTP, POP3, and FTP.

Example 15-1. Using ssh(1) to Create a Secure Tunnel for SMTP

```
% ssh -2 -N -f -L 5025:localhost:25 user@mailserver.example.com
user@mailserver.example.com's password: *****
% telnet localhost 5025
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 mailserver.example.com ESMTP
```

This can be used in conjunction with ssh-keygen(1) and additional user accounts to create a more seamless SSH tunneling environment. Keys can be used in place of typing a password, and the tunnels can be run as a separate user.

15.10.8.1 Practical SSH Tunneling Examples

15.10.8.1.1 Secure Access of a POP3 Server

In this example, there is an SSH server that accepts connections from the outside. On the same network resides a mail server running a POP3 server. To check email in a secure manner, create an SSH connection to the SSH server, and tunnel through to the mail server.

```
% ssh -2 -N -f -L 2110:mail.example.com:110 user@ssh-server.example.com
user@ssh-server.example.com's password: ******
```

Once the tunnel is up and running, point the email client to send POP3 requests to `localhost` on port 2110. This connection will be forwarded securely across the tunnel to `mail.example.com`.

15.10.8.1.2 Bypassing a Draconian Firewall

Some network administrators impose firewall rules which filter both incoming and outgoing connections. For example, it might limit access from remote machines to ports 22 and 80 to only allow ssh(1) and web surfing. This prevents access to any other service which uses a port other than 22 or 80.

The solution is to create an SSH connection to a machine outside of the network's firewall and use it to tunnel to the desired service.

```
% ssh -2 -N -f -L 8888:music.example.com:8000 user@unfirewalled-system.example.org
user@unfirewalled-system.example.org's password: *******
```

In this example, a streaming Ogg Vorbis client can now be pointed to `localhost` port 8888, which will be forwarded over to `music.example.com` on port 8000, successfully bypassing the firewall.

15.10.9 The `AllowUsers` Option

It is often a good idea to limit which users can log in and from where using `AllowUsers`. For example, to only allow `root` to log in from `192.168.1.32`, add this line to `/etc/ssh/sshd_config`:

```
AllowUsers root@192.168.1.32
```

To allow `admin` to log in from anywhere, list that username by itself:

```
AllowUsers admin
```

Multiple users should be listed on the same line, like so:

```
AllowUsers root@192.168.1.32 admin
```

> **Note:** It is important to list each user that needs to log into this machine; otherwise, they will be locked out.

After making changes to `/etc/ssh/sshd_config`, tell sshd(8) to reload its configuration file by running:

```
# service sshd reload
```

15.10.10 Further Reading

The OpenSSH (http://www.openssh.com/) website.

ssh(1), scp(1), ssh-keygen(1), ssh-agent(1), ssh-add(1), and ssh_config(5) for client options.

sshd(8), sftp-server(8), and sshd_config(5) for server options.

15.11 Filesystem Access Control Lists (ACLs)

Contributed by Tom Rhodes.

Filesystem Access Control Lists (ACLs) extend the standard UNIX permission model in a POSIX.1e compatible way. This permits an administrator to make use of and take advantage of a more sophisticated security model.

The FreeBSD `GENERIC` kernel provides ACL support for UFS file systems. Users who prefer to compile a custom kernel must include the following option in their custom kernel configuration file:

```
options UFS_ACL
```

If this option is not compiled in, a warning message will be displayed when attempting to mount a filesystem supporting ACLs. ACLs rely on extended attributes being enabled on the filesystem. Extended attributes are natively supported in UFS2.

> **Note:** A higher level of administrative overhead is required to configure extended attributes on UFS1 than on UFS2. The performance of extended attributes on UFS2 is also substantially higher. As a result, UFS2 is recommended for use with ACLs.

ACLs are enabled by the mount-time administrative flag, `acls`, which may be added to `/etc/fstab`. The mount-time flag can also be automatically set in a persistent manner using tunefs(8) to modify a superblock ACLs flag in the filesystem header. In general, it is preferred to use the superblock flag for several reasons:

- The mount-time ACLs flag cannot be changed by a remount using `mount -u`. It requires a complete umount(8) and fresh mount(8). This means that ACLs cannot be enabled on the root filesystem after boot. It also means that the disposition of a filesystem cannot be changed once it is in use.
- Setting the superblock flag will cause the filesystem to always be mounted with ACLs enabled, even if there is not an `fstab` entry or if the devices re-order. This prevents accidental mounting of the filesystem without ACLs enabled, which can result in the security problem of ACLs being improperly enforced.

> **Note:** It is desirable to discourage accidental mounting without ACLs enabled, because nasty things can happen if ACLs are enabled, then disabled, then re-enabled without flushing the extended attributes. In general, once ACLs are enabled on a filesystem, they should not be disabled, as the resulting file protections may not be compatible with those intended by the users of the system, and re-enabling ACLs may re-attach the previous ACLs to files that have since had their permissions changed, resulting in unpredictable behavior.

Filesystems with ACLs enabled will show a + (plus) sign in their permission settings when viewed. For example:

```
drwx------  2 robert  robert  512 Dec 27 11:54 private
drwxrwx---+ 2 robert  robert  512 Dec 23 10:57 directory1
drwxrwx---+ 2 robert  robert  512 Dec 22 10:20 directory2
drwxrwx---+ 2 robert  robert  512 Dec 27 11:57 directory3
drwxr-xr-x  2 robert  robert  512 Nov 10 11:54 public_html
```

In this example, `directory1`, `directory2`, and `directory3` are all taking advantage of ACLs, whereas `public_html` is not.

15.11.1 Making Use of ACLs

Filesystem ACLs can be viewed using getfacl(1). For instance, to view the ACL settings on `test`:

```
% getfacl test
        #file:test
        #owner:1001
        #group:1001
        user::rw-
        group::r--
        other::r--
```

To change the ACL settings on this file, use setfacl(1):

```
% setfacl -k test
```

To remove all of the currently defined ACLs from a file or filesystem, one can use `-k`. However, the preferred method is to use `-b` as it leaves the basic fields required for ACLs to work.

```
% setfacl -m u:trhodes:rwx,group:web:r--,o::--- test
```

In this example, `-m` is used to modify the default ACL entries. Since there were no pre-defined entries, as they were removed by the previous command, it restores the default options and assign the options listed. If a user or group is added which does not exist on the system, an `Invalid argument` error will be displayed.

15.12 Monitoring Third Party Security Issues

Contributed by Tom Rhodes.

In recent years, the security world has made many improvements to how vulnerability assessment is handled. The threat of system intrusion increases as third party utilities are installed and configured for virtually any operating system available today.

Vulnerability assessment is a key factor in security. While FreeBSD releases advisories for the base system, doing so for every third party utility is beyond the FreeBSD Project's capability. There is a way to mitigate third party vulnerabilities and warn administrators of known security issues. A FreeBSD add on utility known as **portaudit** exists solely for this purpose.

The `ports-mgmt/portaudit` port polls a database, which is updated and maintained by the FreeBSD Security Team and ports developers, for known security issues.

To install **portaudit** from the Ports Collection:

```
# cd /usr/ports/ports-mgmt/portaudit && make install clean
```

During the installation, the configuration files for periodic(8) will be updated, permitting **portaudit** output in the daily security runs. Ensure that the daily security run emails, which are sent to `root`'s email account, are being read. No other configuration is required.

After installation, an administrator can update the database and view known vulnerabilities in installed packages by invoking the following command:

```
# portaudit -Fda
```

> **Note:** The database is automatically updated during the periodic(8) run. The above command is optional and can be used to manually update the database now.

To audit the third party utilities installed as part of the Ports Collection at anytime, an administrator can run the following command:

```
# portaudit -a
```

portaudit will display messages for any installed vulnerable packages:

```
Affected package: cups-base-1.1.22.0_1
Type of problem: cups-base -- HPGL buffer overflow vulnerability.
Reference: <http://www.FreeBSD.org/ports/portaudit/40a3bca2-6809-11d9-a9e7-0001020eed82.html>

1 problem(s) in your installed packages found.

You are advised to update or deinstall the affected package(s) immediately.
```

By pointing a web browser to the displayed URL, an administrator may obtain more information about the vulnerability. This will include the versions affected, by FreeBSD port version, along with other web sites which may contain security advisories.

portaudit is a powerful utility and is extremely useful when coupled with the **portmaster** port.

15.13 FreeBSD Security Advisories

Contributed by Tom Rhodes.

Like many production quality operating systems, FreeBSD publishes “Security Advisories”. These advisories are usually mailed to the security lists and noted in the Errata only after the appropriate releases have been patched. This section explains what an advisory is, how to understand it, and what measures to take in order to patch a system.

15.13.1 What Does an Advisory Look Like?

FreeBSD security advisories use the format seen in this example:

```
=============================================================================
FreeBSD-SA-XX:XX.UTIL                                          Security Advisory
                                                            The FreeBSD Project

Topic:          denial of service due to some problem ❶

Category:       core ❷
Module:         sys ❸
```

```
Announced:      2003-09-23 ❹
Credits:        Person ❺
Affects:        All releases of FreeBSD ❻
                FreeBSD 4-STABLE prior to the correction date
Corrected:      2003-09-23 16:42:59 UTC (RELENG_4, 4.9-PRERELEASE)
                2003-09-23 20:08:42 UTC (RELENG_5_1, 5.1-RELEASE-p6)
                2003-09-23 20:07:06 UTC (RELENG_5_0, 5.0-RELEASE-p15)
                2003-09-23 16:44:58 UTC (RELENG_4_8, 4.8-RELEASE-p8)
                2003-09-23 16:47:34 UTC (RELENG_4_7, 4.7-RELEASE-p18)
                2003-09-23 16:49:46 UTC (RELENG_4_6, 4.6-RELEASE-p21)
                2003-09-23 16:51:24 UTC (RELENG_4_5, 4.5-RELEASE-p33)
                2003-09-23 16:52:45 UTC (RELENG_4_4, 4.4-RELEASE-p43)
                2003-09-23 16:54:39 UTC (RELENG_4_3, 4.3-RELEASE-p39) ❼
CVE Name:       CVE-XXXX-XXXX ❽

For general information regarding FreeBSD Security Advisories,
including descriptions of the fields above, security branches, and the
following sections, please visit
http://www.FreeBSD.org/security/.

I.   Background ❾


II.  Problem Description (10)


III. Impact (11)


IV.  Workaround (12)


V.   Solution (13)


VI.  Correction details (14)


VII. References (15)
```

❶ The `Topic` field specifies the problem. It provides an introduction to the security advisory and notes the utility affected by the vulnerability.

❷ The `Category` refers to the affected part of the system which may be one of `core`, `contrib`, or `ports`. The `core` category means that the vulnerability affects a core component of the FreeBSD operating system. The `contrib` category means that the vulnerability affects software contributed to the FreeBSD Project, such as **Sendmail**. The `ports` category indicates that the vulnerability affects add on software available through the Ports Collection.

❸ The `Module` field refers to the component location. In this example, the `sys` module is affected; therefore, this vulnerability affects a component used within the kernel.

❹ The `Announced` field reflects the date the security advisory was published, or announced to the world. This means that the security team has verified that the problem exists and that a patch has been committed to the FreeBSD source code repository.

❺ The `Credits` field gives credit to the individual or organization who noticed the vulnerability and reported it.

❻ The `Affects` field explains which releases of FreeBSD are affected by this vulnerability. For the kernel, a quick look over the output from ident(1) on the affected files will help in determining the revision. For ports, the version number is listed after the port name in `/var/db/pkg`. If the system does not sync with the FreeBSD Subversion repository and is not rebuilt daily, chances are that it is affected.

❼ The `Corrected` field indicates the date, time, time offset, and release that was corrected.

❽ Reserved for the identification information used to look up vulnerabilities in the Common Vulnerabilities and Exposures (http://cve.mitre.org) database.

❾ The `Background` field gives information about the affected utility. Most of the time this is why the utility exists in FreeBSD, what it is used for, and a bit of information on how the utility came to be.

(10) The `Problem Description` field explains the security hole in depth. This can include information on flawed code, or even how the utility could be maliciously used to open a security hole.

(11) The `Impact` field describes what type of impact the problem could have on a system. For example, this could be anything from a denial of service attack, to extra privileges available to users, or even giving the attacker superuser access.

(12) The `Workaround` field offers a workaround to system administrators who cannot upgrade the system due to time constraints, network availability, or other reasons. Security should not be taken lightly, and an affected system should either be patched or the workaround implemented.

(13) The `Solution` field offers instructions for patching the affected system. This is a step by step tested and verified method for getting a system patched and working securely.

(14) The `Correction Details` field displays the Subversion branch or release name with the periods changed to underscore characters. It also shows the revision number of the affected files within each branch.

(15) The `References` field usually offers sources of other information. This can include web URLs, books, mailing lists, and newsgroups.

15.14 Process Accounting

Contributed by Tom Rhodes.

Process accounting is a security method in which an administrator may keep track of system resources used and their allocation among users, provide for system monitoring, and minimally track a user's commands.

This indeed has both positive and negative points. One of the positives is that an intrusion may be narrowed down to the point of entry. A negative is the amount of logs generated by process accounting, and the disk space they may require. This section walks an administrator through the basics of process accounting.

15.14.1 Enabling and Utilizing Process Accounting

Before using process accounting, it must be enabled using the following commands:

```
# touch /var/account/acct

# accton /var/account/acct

# echo 'accounting_enable="YES"' >> /etc/rc.conf
```

Once enabled, accounting will begin to track information such as CPU statistics and executed commands. All accounting logs are in a non-human readable format which can be viewed using sa(8). If issued without any options, sa(8) prints information relating to the number of per-user calls, the total elapsed time in minutes, total CPU and user time in minutes, and the average number of I/O operations.

To view information about commands being issued, use lastcomm(1). This command displays the commands issued by users on specific ttys(5). For example, this command prints out all known usage of ls(1) by `trhodes` on the `ttyp1` terminal:

```
# lastcomm ls
 trhodes ttyp1
```

Many other useful options exist and are explained in the lastcomm(1), acct(5), and sa(8).

15.15 Resource Limits

Contributed by Tom Rhodes.

For years, FreeBSD has used a resource limits database controlled through a flat file, `/etc/login.conf`. While it has been discussed previously and is still supported, it is not the most optimal method of controlling resources. The flat file requires users to be divided into various group labels known as classes, which require changes not only to this flat file but also the password database. Potentially a single, more constrained user would require an additional label added, the resource database needs to be built using `cap_mkdb`, edits made to the `/etc/master.passwd` file. In addition, the password database must be rebuilt using `pwd_mkdb`. This multi-step process could be very time consuming depending on how many users must be singled out.

A new command in FreeBSD, rctl(8), allows for a more fine grained method of controlling resources limits for users. This command will support much more than users, it will also set resource constraints on processes, jails, and the original login class. These advanced features provide administrators and users with methods to control resources through the command line and set rules on system initialization using a configuration file.

To enable this feature, add these lines to `GENERIC`, or the custom kernel configuration file, and rebuild.:

```
options         RACCT
options         RCTL
```

The entire system will need rebuilt. See Chapter 9, which will provide instructions for the process. Once this is complete, the `rctl` may be used to set rules for the system.

Rule syntax is simple, controlled through the use of a *subject*, a *subject-id*, *resource*, and *action*. Take the following example rule:

```
user:trhodes:maxproc:deny=10/user
```

This rule shows a basic premise of a rule, here the subject is `user` and the subject-id is `trhodes`. The maxproc is, of course, max number of processes, which is considered the action. The action here is set to `deny`, which blocks any new processes from being created. In the previous example, the user, `trhodes` will be constrained to `10` (ten) processes and no greater. Other actions are available and could be log to the console, pass a notification to devd(8), or send a sigterm to the process.

Some care must be taken while adding rules. The one above will unfortunately block my user from doing the most simple tasks after I have logged in and executed a `screen` session. When a resource limit has been hit, an error will be printed, as in this example:

```
% man test
    /usr/bin/man: Cannot fork: Resource temporarily unavailable
eval: Cannot fork: Resource temporarily unavailable
```

For another example, rctl(8) can be used to prevent a jail from exceeding a memory limit. This rule could be written as:

```
# rctl -a jail:httpd:memoryuse:deny=2G/jail
```

Rules may also persist across reboots if they have been added to `/etc/rctl.conf` file. The format is a rule, without the preceding command. For example, the previous rule could be added like the following:

```
# Block jail from using more than 2G memory:
jail:httpd:memoryuse:deny=2G/jail
```

To remove a rule, just ask `rctl` to remove it from the list:

```
# rctl -r user:trhodes:maxproc:deny=10/user
```

The manual page shows a method for removing all rules; however, if removing all rules for a single user is required, this command may be issued:

```
# rctl -r user:trhodes
```

Many other resources exist which can be used to excert additional control over various `subjects`. See rctl(8) to learn about them.

Notes

1. Under FreeBSD the standard login password may be up to 128 characters in length.

Chapter 16 Jails

Contributed by Matteo Riondato.

16.1 Synopsis

This chapter will provide an explanation of what FreeBSD jails are and how to use them. Jails, sometimes referred to as an enhanced replacement of *chroot environments*, are a very powerful tool for system administrators, but their basic usage can also be useful for advanced users.

> **Important:** Jails are a powerful tool, but they are not a security panacea. It is particularly important to note that while it is not possible for a jailed process to break out on its own, there are several ways in which an unprivileged user outside the jail can cooperate with a privileged user inside the jail and thereby obtain elevated privileges in the host environment.
>
> Most of these attacks can be mitigated by ensuring that the jail root is not accessible to unprivileged users in the host environment. Regardless, as a general rule, untrusted users with privileged access to a jail should not be given access to the host environment.

After reading this chapter, you will know:

- What a jail is, and what purpose it may serve in FreeBSD installations.
- How to build, start, and stop a jail.
- The basics of jail administration, both from inside and outside the jail.

Other sources of useful information about jails are:

- The jail(8) manual page. This is the full reference of the `jail` utility — the administrative tool which can be used in FreeBSD to start, stop, and control FreeBSD jails.
- The mailing lists and their archives. The archives of the FreeBSD general questions mailing list (http://lists.FreeBSD.org/mailman/listinfo/freebsd-questions) and other mailing lists hosted by the FreeBSD list server (http://lists.FreeBSD.org/mailman/listinfo) already contain a wealth of material for jails. It should always be engaging to search the archives, or post a new question to the freebsd-questions (http://lists.FreeBSD.org/mailman/listinfo/freebsd-questions) mailing list.

16.2 Terms Related to Jails

To facilitate better understanding of parts of the FreeBSD system related to jails, their internals and the way they interact with the rest of FreeBSD, the following terms are used further in this chapter:

chroot(8) (command)

> Utility, which uses chroot(2) FreeBSD system call to change the root directory of a process and all its descendants.

chroot(2) (environment)

The environment of processes running in a “chroot”. This includes resources such as the part of the file system which is visible, user and group IDs which are available, network interfaces and other IPC mechanisms, etc.

jail(8) (command)

The system administration utility which allows launching of processes within a jail environment.

host (system, process, user, etc.)

The controlling system of a jail environment. The host system has access to all the hardware resources available, and can control processes both outside of and inside a jail environment. One of the important differences of the host system from a jail is that the limitations which apply to superuser processes inside a jail are not enforced for processes of the host system.

hosted (system, process, user, etc.)

A process, user or other entity, whose access to resources is restricted by a FreeBSD jail.

16.3 Introduction

Since system administration is a difficult and perplexing task, many powerful tools were developed to make life easier for the administrator. These tools mostly provide enhancements of some sort to the way systems are installed, configured and maintained. Part of the tasks which an administrator is expected to do is to properly configure the security of a system, so that it can continue serving its real purpose, without allowing security violations.

One of the tools which can be used to enhance the security of a FreeBSD system are *jails*. Jails were introduced in FreeBSD 4.X by Poul-Henning Kamp <phk@FreeBSD.org>, but were greatly improved in FreeBSD 5.X to make them a powerful and flexible subsystem. Their development still goes on, enhancing their usefulness, performance, reliability, and security.

16.3.1 What is a Jail

BSD-like operating systems have had chroot(2) since the time of 4.2BSD. The chroot(8) utility can be used to change the root directory of a set of processes, creating a safe environment, separate from the rest of the system. Processes created in the chrooted environment can not access files or resources outside of it. For that reason, compromising a service running in a chrooted environment should not allow the attacker to compromise the entire system. The chroot(8) utility is good for easy tasks which do not require much flexibility or complex, advanced features. Since the inception of the chroot concept, however, many ways have been found to escape from a chrooted environment and, although they have been fixed in modern versions of the FreeBSD kernel, it was clear that chroot(2) was not the ideal solution for securing services. A new subsystem had to be implemented.

This is one of the main reasons why *jails* were developed.

Jails improve on the concept of the traditional chroot(2) environment in several ways. In a traditional chroot(2) environment, processes are only limited in the part of the file system they can access. The rest of the system resources (like the set of system users, the running processes, or the networking subsystem) are shared by the chrooted processes and the processes of the host system. Jails expand this model by virtualizing not only access to the file system, but also the set of users, the networking subsystem of the FreeBSD kernel and a few other things. A more complete set of fine-grained controls available for tuning the access of a jailed environment is described in Section 16.5.

A jail is characterized by four elements:

- A directory subtree — the starting point from which a jail is entered. Once inside the jail, a process is not permitted to escape outside of this subtree. Traditional security issues which plagued the original chroot(2) design will not affect FreeBSD jails.
- A hostname — the hostname which will be used within the jail. Jails are mainly used for hosting network services, therefore having a descriptive hostname for each jail can really help the system administrator.
- An IP address — this will be assigned to the jail and cannot be changed in any way during the jail's life span. The IP address of a jail is usually an alias address for an existing network interface, but this is not strictly necessary.
- A command — the path name of an executable to run inside the jail. The path is relative to the root directory of the jail environment.

Apart from these, jails can have their own set of users and their own `root` user. Naturally, the powers of the `root` user are limited within the jail environment and, from the point of view of the host system, the jail `root` user is not an omnipotent user. In addition, the `root` user of a jail is not allowed to perform critical operations to the system outside of the associated jail(8) environment. More information about capabilities and restrictions of the `root` user will be discussed in Section 16.5 below.

16.4 Creating and Controlling Jails

Some administrators divide jails into the following two types: "complete" jails, which resemble a real FreeBSD system, and "service" jails, dedicated to one application or service, possibly running with privileges. This is only a conceptual division and the process of building a jail is not affected by it. The jail(8) manual page is quite clear about the procedure for building a jail:

```
# setenv D /here/is/the/jail
# mkdir -p $D       ❶
# cd /usr/src
# make buildworld   ❷
# make installworld DESTDIR=$D   ❸
# make distribution DESTDIR=$D   ❹
# mount -t devfs devfs $D/dev    ❺
```

❶ Selecting a location for a jail is the best starting point. This is where the jail will physically reside within the file system of the jail's host. A good choice can be `/usr/jail/`*`jailname`*, where *`jailname`* is the hostname identifying the jail. The `/usr/` file system usually has enough space for the jail file system, which for "complete" jails is, essentially, a replication of every file present in a default installation of the FreeBSD base system.

❷ If you have already rebuilt your userland using `make world` or `make buildworld`, you can skip this step and install your existing userland into the new jail.

❸ This command will populate the directory subtree chosen as jail's physical location on the file system with the necessary binaries, libraries, manual pages and so on.

❹ The `distribution` target for **make** installs every needed configuration file. In simple words, it installs every installable file of `/usr/src/etc/` to the `/etc` directory of the jail environment: `$D/etc/`.

❺ Mounting the devfs(8) file system inside a jail is not required. On the other hand, any, or almost any application requires access to at least one device, depending on the purpose of the given application. It is very important to control access to devices from inside a jail, as improper settings could permit an attacker to do nasty things in the jail. Control over devfs(8) is managed through rulesets which are described in the devfs(8) and devfs.conf(5) manual pages.

Once a jail is installed, it can be started by using the jail(8) utility. The jail(8) utility takes four mandatory arguments which are described in the Section 16.3.1. Other arguments may be specified too, e.g., to run the jailed process with the credentials of a specific user. The *command* argument depends on the type of the jail; for a *virtual system*, `/etc/rc` is a good choice, since it will replicate the startup sequence of a real FreeBSD system. For a *service* jail, it depends on the service or application that will run within the jail.

Jails are often started at boot time and the FreeBSD `rc` mechanism provides an easy way to do this.

1. A list of the jails which are enabled to start at boot time should be added to the rc.conf(5) file:

   ```
   jail_enable="YES"   # Set to NO to disable starting of any jails
   jail_list="www"     # Space separated list of names of jails
   ```

   Note: Jail names in `jail_list` should contain alphanumeric characters only.

2. For each jail listed in `jail_list`, a group of rc.conf(5) settings, which describe the particular jail, should be added:

   ```
   jail_www_rootdir="/usr/jail/www"     # jail's root directory
   jail_www_hostname="www.example.org"  # jail's hostname
   jail_www_ip="192.168.0.10"           # jail's IP address
   jail_www_devfs_enable="YES"          # mount devfs in the jail
   jail_www_devfs_ruleset="www_ruleset" # devfs ruleset to apply to jail
   ```

   The default startup of jails configured in rc.conf(5), will run the `/etc/rc` script of the jail, which assumes the jail is a complete virtual system. For service jails, the default startup command of the jail should be changed, by setting the `jail_`*`jailname`*`_exec_start` option appropriately.

   Note: For a full list of available options, please see the rc.conf(5) manual page.

service(8) can be used to start or stop a jail by hand, if an entry for it exists in `rc.conf`:

```
# service jail start www
# service jail stop www
```

A clean way to shut down a jail(8) is not available at the moment. This is because commands normally used to accomplish a clean system shutdown cannot be used inside a jail. The best way to shut down a jail is to run the following command from within the jail itself or using the jexec(8) utility from outside the jail:

```
# sh /etc/rc.shutdown
```

More information about this can be found in the jail(8) manual page.

16.5 Fine Tuning and Administration

There are several options which can be set for any jail, and various ways of combining a host FreeBSD system with jails, to produce higher level applications. This section presents:

- Some of the options available for tuning the behavior and security restrictions implemented by a jail installation.
- Some of the high-level applications for jail management, which are available through the FreeBSD Ports Collection, and can be used to implement overall jail-based solutions.

16.5.1 System Tools for Jail Tuning in FreeBSD

Fine tuning of a jail's configuration is mostly done by setting sysctl(8) variables. A special subtree of sysctl exists as a basis for organizing all the relevant options: the `security.jail.*` hierarchy of FreeBSD kernel options. Here is a list of the main jail-related sysctls, complete with their default value. Names should be self-explanatory, but for more information about them, please refer to the jail(8) and sysctl(8) manual pages.

- `security.jail.set_hostname_allowed: 1`
- `security.jail.socket_unixiproute_only: 1`
- `security.jail.sysvipc_allowed: 0`
- `security.jail.enforce_statfs: 2`
- `security.jail.allow_raw_sockets: 0`
- `security.jail.chflags_allowed: 0`
- `security.jail.jailed: 0`

These variables can be used by the system administrator of the *host system* to add or remove some of the limitations imposed by default on the `root` user. Note that there are some limitations which cannot be removed. The `root` user is not allowed to mount or unmount file systems from within a jail(8). The `root` inside a jail may not load or unload devfs(8) rulesets, set firewall rules, or do many other administrative tasks which require modifications of in-kernel data, such as setting the `securelevel` of the kernel.

The base system of FreeBSD contains a basic set of tools for viewing information about the active jails, and attaching to a jail to run administrative commands. The jls(8) and jexec(8) commands are part of the base FreeBSD system, and can be used to perform the following simple tasks:

- Print a list of active jails and their corresponding jail identifier (JID), IP address, hostname and path.
- Attach to a running jail, from its host system, and run a command inside the jail or perform administrative tasks inside the jail itself. This is especially useful when the `root` user wants to cleanly shut down a jail. The jexec(8) utility can also be used to start a shell in a jail to do administration in it; for example:

  ```
  # jexec 1 tcsh
  ```

16.5.2 High-Level Administrative Tools in the FreeBSD Ports Collection

Among the many third-party utilities for jail administration, one of the most complete and useful is `sysutils/jailutils`. It is a set of small applications that contribute to jail(8) management. Please refer to its web page for more information.

16.6 Application of Jails

16.6.1 Service Jails

Contributed by Daniel Gerzo.

This section is based upon an idea originally presented by Simon L. Nielsen <simon@FreeBSD.org> at http://simon.nitro.dk/service-jails.html, and an updated article written by Ken Tom <locals@gmail.com>. This section illustrates how to set up a FreeBSD system that adds an additional layer of security, using the jail(8) feature. It is also assumed that the given system is at least running RELENG_6_0 and the information provided earlier in this chapter has been well understood.

16.6.1.1 Design

One of the major problems with jails is the management of their upgrade process. This tends to be a problem because every jail has to be rebuilt from scratch whenever it is updated. This is usually not a problem for a single jail, since the update process is fairly simple, but can be quite time consuming and tedious if a lot of jails are created.

> **Warning:** This setup requires advanced experience with FreeBSD and usage of its features. If the presented steps below look too complicated, it is advised to take a look at a simpler system such as sysutils/ezjail, which provides an easier method of administering FreeBSD jails and is not as sophisticated as this setup.

This idea has been presented to resolve such issues by sharing as much as is possible between jails, in a safe way — using read-only mount_nullfs(8) mounts, so that updating will be simpler, and putting single services into individual jails will become more attractive. Additionally, it provides a simple way to add or remove jails as well as a way to upgrade them.

> **Note:** Examples of services in this context are: an HTTP server, a DNS server, a SMTP server, and so forth.

The goals of the setup described in this section are:

- Create a simple and easy to understand jail structure. This implies *not* having to run a full installworld on each and every jail.
- Make it easy to add new jails or remove existing ones.
- Make it easy to update or upgrade existing jails.
- Make it possible to run a customized FreeBSD branch.
- Be paranoid about security, reducing as much as possible the possibility of compromise.
- Save space and inodes, as much as possible.

As it has been already mentioned, this design relies heavily on having a single master template which is read-only (known as **nullfs**) mounted into each jail and one read-write device per jail. A device can be a separate physical disc, a partition, or a vnode backed md(4) device. In this example, we will use read-write **nullfs** mounts.

The file system layout is described in the following list:

- Each jail will be mounted under the `/home/j` directory.
- `/home/j/mroot` is the template for each jail and the read-only partition for all of the jails.
- A blank directory will be created for each jail under the `/home/j` directory.
- Each jail will have a `/s` directory, that will be linked to the read-write portion of the system.
- Each jail shall have its own read-write system that is based upon `/home/j/skel`.
- Each jailspace (read-write portion of each jail) shall be created in `/home/js`.

> **Note:** This assumes that the jails are based under the `/home` partition. This can, of course, be changed to anything else, but this change will have to be reflected in each of the examples below.

16.6.1.2 Creating the Template

This section will describe the steps needed to create the master template that will be the read-only portion for the jails to use.

It is always a good idea to update the FreeBSD system to the latest -RELEASE branch. Check the corresponding Handbook Chapter (http://www.FreeBSD.org/doc/en_US.ISO8859-1/books/handbook/makeworld.html) to accomplish this task. In the case the update is not feasible, the buildworld will be required in order to be able to proceed. Additionally, the `sysutils/cpdup` package will be required. We will use the portsnap(8) utility to download the FreeBSD Ports Collection. The Handbook Portsnap Chapter (http://www.FreeBSD.org/doc/en_US.ISO8859-1/books/handbook/portsnap.html) is always good reading for newcomers.

1. First, create a directory structure for the read-only file system which will contain the FreeBSD binaries for our jails, then change directory to the FreeBSD source tree and install the read-only file system to the jail template:

```
# mkdir /home/j /home/j/mroot
# cd /usr/src
# make installworld DESTDIR=/home/j/mroot
```

2. Next, prepare a FreeBSD Ports Collection for the jails as well as a FreeBSD source tree, which is required for **mergemaster**:

```
# cd /home/j/mroot
# mkdir usr/ports
# portsnap -p /home/j/mroot/usr/ports fetch extract
# cpdup /usr/src /home/j/mroot/usr/src
```

3. Create a skeleton for the read-write portion of the system:

```
# mkdir /home/j/skel /home/j/skel/home /home/j/skel/usr-X11R6 /home/j/skel/distfiles
# mv etc /home/j/skel
# mv usr/local /home/j/skel/usr-local
# mv tmp /home/j/skel
# mv var /home/j/skel
# mv root /home/j/skel
```

4. Use **mergemaster** to install missing configuration files. Then get rid of the extra directories that **mergemaster** creates:

```
# mergemaster -t /home/j/skel/var/tmp/temproot -D /home/j/skel -i
# cd /home/j/skel
# rm -R bin boot lib libexec mnt proc rescue sbin sys usr dev
```

5. Now, symlink the read-write file system to the read-only file system. Please make sure that the symlinks are created in the correct `s/` locations. Real directories or the creation of directories in the wrong locations will cause the installation to fail.

```
# cd /home/j/mroot
# mkdir s
# ln -s s/etc etc
# ln -s s/home home
# ln -s s/root root
# ln -s ../s/usr-local usr/local
# ln -s ../s/usr-X11R6 usr/X11R6
# ln -s ../../s/distfiles usr/ports/distfiles
# ln -s s/tmp tmp
# ln -s s/var var
```

6. As a last step, create a generic `/home/j/skel/etc/make.conf` with its contents as shown below:

```
WRKDIRPREFIX?=  /s/portbuild
```

Having `WRKDIRPREFIX` set up this way will make it possible to compile FreeBSD ports inside each jail. Remember that the ports directory is part of the read-only system. The custom path for `WRKDIRPREFIX` allows builds to be done in the read-write portion of every jail.

16.6.1.3 Creating Jails

Now that we have a complete FreeBSD jail template, we can setup and configure the jails in `/etc/rc.conf`. This example demonstrates the creation of 3 jails: “NS”, “MAIL” and “WWW”.

1. Put the following lines into the `/etc/fstab` file, so that the read-only template for the jails and the read-write space will be available in the respective jails:

```
/home/j/mroot   /home/j/ns     nullfs  ro  0   0
/home/j/mroot   /home/j/mail   nullfs  ro  0   0
/home/j/mroot   /home/j/www    nullfs  ro  0   0
/home/js/ns     /home/j/ns/s   nullfs  rw  0   0
/home/js/mail   /home/j/mail/s nullfs  rw  0   0
/home/js/www    /home/j/www/s  nullfs  rw  0   0
```

> **Note:** Partitions marked with a 0 pass number are not checked by fsck(8) during boot, and partitions marked with a 0 dump number are not backed up by dump(8). We do not want **fsck** to check **nullfs** mounts or **dump** to back up the read-only nullfs mounts of the jails. This is why they are marked with “0 0” in the last two columns of each `fstab` entry above.

2. Configure the jails in `/etc/rc.conf`:

```
jail_enable="YES"
jail_set_hostname_allow="NO"
jail_list="ns mail www"
jail_ns_hostname="ns.example.org"
```

```
jail_ns_ip="192.168.3.17"
jail_ns_rootdir="/usr/home/j/ns"
jail_ns_devfs_enable="YES"
jail_mail_hostname="mail.example.org"
jail_mail_ip="192.168.3.18"
jail_mail_rootdir="/usr/home/j/mail"
jail_mail_devfs_enable="YES"
jail_www_hostname="www.example.org"
jail_www_ip="62.123.43.14"
jail_www_rootdir="/usr/home/j/www"
jail_www_devfs_enable="YES"
```

> **Warning:** The reason why the `jail_name_rootdir` variable is set to `/usr/home` instead of `/home` is that the physical path of the `/home` directory on a default FreeBSD installation is `/usr/home`. The `jail_name_rootdir` variable must *not* be set to a path which includes a symbolic link, otherwise the jails will refuse to start. Use the realpath(1) utility to determine a value which should be set to this variable. Please see the FreeBSD-SA-07:01.jail Security Advisory for more information.

3. Create the required mount points for the read-only file system of each jail:

   ```
   # mkdir /home/j/ns /home/j/mail /home/j/www
   ```

4. Install the read-write template into each jail. Note the use of `sysutils/cpdup`, which helps to ensure that a correct copy is done of each directory:

   ```
   # mkdir /home/js
   # cpdup /home/j/skel /home/js/ns
   # cpdup /home/j/skel /home/js/mail
   # cpdup /home/j/skel /home/js/www
   ```

5. In this phase, the jails are built and prepared to run. First, mount the required file systems for each jail, and then start them using the jail rc script.

   ```
   # mount -a
   # service jail start
   ```

The jails should be running now. To check if they have started correctly, use the jls(8) command. Its output should be similar to the following:

```
# jls
   JID  IP Address      Hostname                      Path
     3  192.168.3.17    ns.example.org                /home/j/ns
     2  192.168.3.18    mail.example.org              /home/j/mail
     1  62.123.43.14    www.example.org               /home/j/www
```

At this point, it should be possible to log onto each jail, add new users or configure daemons. The `JID` column indicates the jail identification number of each running jail. Use the following command in order to perform administrative tasks in the jail whose `JID` is 3:

```
# jexec 3 tcsh
```

16.6.1.4 Upgrading

In time, there will be a need to upgrade the system to a newer version of FreeBSD, either because of a security issue, or because new features have been implemented which are useful for the existing jails. The design of this setup provides an easy way to upgrade existing jails. Additionally, it minimizes their downtime, as the jails will be brought down only in the very last minute. Also, it provides a way to roll back to the older versions should any problems occur.

1. The first step is to upgrade the host system in the usual manner. Then create a new temporary read-only template in `/home/j/mroot2`.

   ```
   # mkdir /home/j/mroot2
   # cd /usr/src
   # make installworld DESTDIR=/home/j/mroot2
   # cd /home/j/mroot2
   # cpdup /usr/src usr/src
   # mkdir s
   ```

   The `installworld` run creates a few unnecessary directories, which should be removed:

   ```
   # chflags -R 0 var
   # rm -R etc var root usr/local tmp
   ```

2. Recreate the read-write symlinks for the master file system:

   ```
   # ln -s s/etc etc
   # ln -s s/root root
   # ln -s s/home home
   # ln -s ../s/usr-local usr/local
   # ln -s ../s/usr-X11R6 usr/X11R6
   # ln -s s/tmp tmp
   # ln -s s/var var
   ```

3. The right time to stop the jails is now:

   ```
   # service jail stop
   ```

4. Unmount the original file systems:

   ```
   # umount /home/j/ns/s
   # umount /home/j/ns
   # umount /home/j/mail/s
   # umount /home/j/mail
   # umount /home/j/www/s
   # umount /home/j/www
   ```

   Note: The read-write systems are attached to the read-only system (`/s`) and must be unmounted first.

5. Move the old read-only file system and replace it with the new one. This will serve as a backup and archive of the old read-only file system should something go wrong. The naming convention used here corresponds to when a new read-only file system has been created. Move the original FreeBSD Ports Collection over to the new file system to save some space and inodes:

   ```
   # cd /home/j
   # mv mroot mroot.20060601
   # mv mroot2 mroot
   ```

```
# mv mroot.20060601/usr/ports mroot/usr
```

6. At this point the new read-only template is ready, so the only remaining task is to remount the file systems and start the jails:

```
# mount -a
# service jail start
```

Use jls(8) to check if the jails started correctly. Do not forget to run mergemaster in each jail. The configuration files will need to be updated as well as the rc.d scripts.

Chapter 17 Mandatory Access Control

Written by Tom Rhodes.

17.1 Synopsis

FreeBSD 5.X introduced new security extensions from the TrustedBSD Project (http://www.trustedbsd.org) based on the POSIX.1e draft. Two of the most significant new security mechanisms are file system Access Control Lists (ACLs) and Mandatory Access Control (MAC) facilities. MAC allows new access control modules to be loaded, implementing new security policies. Some modules provide protections for a narrow subset of the system, hardening a particular service. Others provide comprehensive labeled security across all subjects and objects. The mandatory part of the definition indicates that enforcement of controls is performed by administrators and the operating system. This is in contrast to the default security mechanism of Discretionary Access Control (DAC where enforcement is left to the discretion of users.

This chapter focuses on the MAC framework and the set of pluggable security policy modules FreeBSD provides for enabling various security mechanisms.

After reading this chapter, you will know:

- Which MAC security policy modules are included in FreeBSD and their associated mechanisms.
- The capabilities of MAC security policy modules as well as the difference between a labeled and non-labeled policy.
- How to efficiently configure a system to use the MAC framework.
- How to configure the different security policy modules included with the MAC framework.
- How to implement a more secure environment using the MAC framework.
- How to test the MAC configuration to ensure the framework has been properly implemented.

Before reading this chapter, you should:

- Understand UNIX and FreeBSD basics (Chapter 4).
- Have some familiarity with security and how it pertains to FreeBSD (Chapter 15).

> **Warning:** Improper MAC configuration may cause loss of system access, aggravation of users, or inability to access the features provided by **Xorg**. More importantly, MAC should not be relied upon to completely secure a system. The MAC framework only augments an existing security policy. Without sound security practices and regular security checks, the system will never be completely secure.
>
> The examples contained within this chapter are for demonstration purposes and the example settings should *not* be implemented on a production system. Implementing any security policy takes a good deal of understanding, proper design, and thorough testing.

17.1.1 What Will Not Be Covered

This chapter covers a broad range of security issues relating to the MAC framework. The development of new MAC security policy modules will not be covered. A number of security policy modules included with the MAC framework have specific characteristics which are provided for both testing and new module development. These include mac_test(4), mac_stub(4) and mac_none(4). For more information on these security policy modules and the various mechanisms they provide, refer to their manual pages.

17.2 Key Terms in This Chapter

Before reading this chapter, a few key terms must be explained:

- *compartment*: a set of programs and data to be partitioned or separated, where users are given explicit access to specific component of a system. A compartment represents a grouping, such as a work group, department, project, or topic. Compartments make it possible to implement a need-to-know-basis security policy.
- *high-watermark*: this type of policy permits the raising of security levels for the purpose of accessing higher level information. In most cases, the original level is restored after the process is complete. Currently, the FreeBSD MAC framework does not include this type of policy.
- *integrity*: the level of trust which can be placed on data. As the integrity of the data is elevated, so does the ability to trust that data.
- *label*: a security attribute which can be applied to files, directories, or other items in the system. It could be considered a confidentiality stamp. When a label is placed on a file, it describes the security properties of that file and will only permit access by files, users, and resources with a similar security setting. The meaning and interpretation of label values depends on the policy configuration. Some policies treat a label as representing the integrity or secrecy of an object while other policies might use labels to hold rules for access.
- *level*: the increased or decreased setting of a security attribute. As the level increases, its security is considered to elevate as well.
- *low-watermark*: this type of policy permits lowering security levels for the purpose of accessing information which is less secure. In most cases, the original security level of the user is restored after the process is complete. The only security policy module in FreeBSD to use this is mac_lomac(4).
- *multilabel*: this property is a file system option which can be set in single user mode using tunefs(8), during boot using fstab(5), or during the creation of a new file system. This option permits an administrator to apply different MAC labels on different objects. This option only applies to security policy modules which support labeling.
- *object*: an entity through which information flows under the direction of a *subject*. This includes directories, files, fields, screens, keyboards, memory, magnetic storage, printers or any other data storage or moving device. An object is a data container or a system resource. Access to an *object* effectively means access to its data.
- *policy*: a collection of rules which defines how objectives are to be achieved. A *policy* usually documents how certain items are to be handled. This chapter considers the term *policy* to be a *security policy*, or a collection of rules which controls the flow of data and information and defines who has access to that data and information.
- *sensitivity*: usually used when discussing Multilevel Security MLS. A sensitivity level describes how important or secret the data should be. As the sensitivity level increases, so does the importance of the secrecy, or confidentiality, of the data.

- *single label*: a policy where the entire file system uses one label to enforce access control over the flow of data. Whenever `multilabel` is not set, all files will conform to the same label setting.
- *subject*: any active entity that causes information to flow between *objects* such as a user, user process, or system process. On FreeBSD, this is almost always a thread acting in a process on behalf of a user.

17.3 Explanation of MAC

With all of these new terms in mind, consider how the MAC framework augments the security of the system as a whole. The various security policy modules provided by the MAC framework could be used to protect the network and file systems or to block users from accessing certain ports and sockets. Perhaps the best use of the policy modules is to load several security policy modules at a time in order to provide a MLS environment. This approach differs from a hardening policy, which typically hardens elements of a system which are used only for specific purposes. The downside to MLS is increased administrative overhead.

The overhead is minimal when compared to the lasting effect of a framework which provides the ability to pick and choose which policies are required for a specific configuration and which keeps performance overhead down. The reduction of support for unneeded policies can increase the overall performance of the system as well as offer flexibility of choice. A good implementation would consider the overall security requirements and effectively implement the various security policy modules offered by the framework.

A system utilizing MAC guarantees that a user will not be permitted to change security attributes at will. All user utilities, programs, and scripts must work within the constraints of the access rules provided by the selected security policy modules and total control of the MAC access rules are in the hands of the system administrator.

It is the duty of the system administrator to carefully select the correct security policy modules. For an environment that needs to limit access control over the network, the mac_portacl(4), mac_ifoff(4), and mac_biba(4) policy modules make good starting points. For an environment where strict confidentiality of file system objects is required, consider the mac_bsdextended(4) and mac_mls(4) policy modules.

Policy decisions could be made based on network configuration. If only certain users should be permitted access to ssh(1), the mac_portacl(4) policy module is a good choice. In the case of file systems, access to objects might be considered confidential to some users, but not to others. As an example, a large development team might be broken off into smaller projects where developers in project A might not be permitted to access objects written by developers in project B. Yet both projects might need to access objects created by developers in project C. Using the different security policy modules provided by the MAC framework, users could be divided into these groups and then given access to the appropriate objects.

Each security policy module has a unique way of dealing with the overall security of a system. Module selection should be based on a well thought out security policy which may require revision and reimplementation. Understanding the different security policy modules offered by the MAC framework will help administrators choose the best policies for their situations.

> **Caution:** Implementing MAC is much like implementing a firewall, care must be taken to prevent being completely locked out of the system. The ability to revert back to a previous configuration should be considered and the implementation of MAC remotely should be done with extreme caution.

17.4 Understanding MAC Labels

A MAC label is a security attribute which may be applied to subjects and objects throughout the system.

When setting a label, the administrator must be able to comprehend what exactly is being done and understand any implications in order to prevent unexpected or undesired behavior of the system. The attributes available on an object depend on the loaded policy module as policy modules interpret their attributes in different ways.

The security label on an object is used as a part of a security access control decision by a policy. With some policies, the label contains all of the information necessary to make a decision. In other policies, the labels may be processed as part of a larger rule set. For instance, setting the label of `biba/low` on a file will represent a label maintained by the Biba security policy module, with a value of “low”.

A few policy modules which support the labeling feature in FreeBSD offer three specific predefined labels: low, high, and equal. Such policy modules enforce access control in a different manner with each policy module, where the low label is the lowest setting, the equal label sets the subject or object to be disabled or unaffected, and the high label enforces the highest setting available in the Biba and MLS policy modules.

Within single label file system environments, only one label may be used on objects. This label enforces one set of access permissions across the entire system and in many environments may be all that is required. There are a few cases where multiple labels may be set on objects or subjects in the file system by passing `multilabel` to tunefs(8).

In the case of Biba and MLS, a numeric label may be set to indicate the precise level of hierarchical control. This numeric level is used to partition or sort information into different groups of classification only permitting access to that group or a higher group level.

In most cases, the administrator will set up a single label to use throughout the file system. This is similar to DAC to some extent as `root` is the one in control and who configures the policies so that users are placed in the appropriate categories/access levels. Alas, many policy modules can restrict the `root` user as well. Basic control over objects will then be released to the group, but `root` may revoke or modify the settings at any time. This is the hierarchical/clearance model covered by policies such as Biba and MLS.

17.4.1 Label Configuration

Virtually all aspects of label policy module configuration will be performed using the base system utilities. These commands provide a simple interface for object or subject configuration or the manipulation and verification of the configuration.

All configuration may be done using setfmac(8) and setpmac(8). `setfmac` is used to set MAC labels on system objects while `setpmac` is used to set the labels on system subjects. Observe:

```
# setfmac biba/high test
```

If the configuration is successful, the prompt will be returned without error. A common error is `Permission denied` which usually occurs when the label is being set or modified on an object which is restricted.[1] The system administrator may use the following commands to overcome this:

```
# setfmac biba/high test
Permission denied
# setpmac biba/low setfmac biba/high test
# getfmac test
test: biba/high
```

`setpmac` can be used to override the policy module's settings by assigning a different label to the invoked process. `getpmac` is usually used with currently running processes, such as **sendmail**. It takes a process ID in place of a command. If users attempt to manipulate a file not in their access, subject to the rules of the loaded policy modules, the `Operation not permitted` error will be displayed by the `mac_set_link` function.

17.4.1.1 Common Label Types

For the mac_biba(4), mac_mls(4) and mac_lomac(4) policy modules, the ability to assign simple labels is provided. These take the form of high, equal, and low, where:

- The `low` label is considered the lowest label setting an object or subject may have. Setting this on objects or subjects blocks their access to objects or subjects marked high.
- The `equal` label should only be placed on objects considered to be exempt from the policy.
- The `high` label grants an object or subject the highest possible setting.

With respect to each policy module, each of those settings will establish a different information flow directive. Refer to the manual pages of the module to determine the traits of these generic label configurations.

17.4.1.1.1 Advanced Label Configuration

Numeric grade labels are used for `comparison:compartment+compartment`. For example:

```
biba/10:2+3+6(5:2+3-20:2+3+4+5+6)
```

may be interpreted as "Biba Policy Label"/"Grade 10":"Compartments 2, 3 and 6": ("grade 5 ...")

In this example, the first grade would be considered the "effective grade" with "effective compartments", the second grade is the low grade, and the last one is the high grade. In most configurations, these settings will not be used as they are advanced configurations.

System objects only have a current grade/compartment. System subjects reflect the range of available rights in the system, and network interfaces, where they are used for access control.

The grade and compartments in a subject and object pair are used to construct a relationship known as "dominance", in which a subject dominates an object, the object dominates the subject, neither dominates the other, or both dominate each other. The "both dominate" case occurs when the two labels are equal. Due to the information flow nature of Biba, a user has rights to a set of compartments that might correspond to projects, but objects also have a set of compartments. Users may have to subset their rights using `su` or `setpmac` in order to access objects in a compartment from which they are not restricted.

17.4.1.2 Users and Label Settings

Users are required to have labels so that their files and processes properly interact with the security policy defined on the system. This is configured in `login.conf` using login classes. Every policy module that uses labels will implement the user class setting.

An example entry containing every policy module setting is displayed below:

```
default:\
       :copyright=/etc/COPYRIGHT:\
```

```
:welcome=/etc/motd:\
:setenv=MAIL=/var/mail/$,BLOCKSIZE=K:\
:path=~/bin:/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/sbin:/usr/local/bin:\
:manpath=/usr/share/man /usr/local/man:\
:nologin=/usr/sbin/nologin:\
:cputime=1h30m:\
:datasize=8M:\
:vmemoryuse=100M:\
:stacksize=2M:\
:memorylocked=4M:\
:memoryuse=8M:\
:filesize=8M:\
:coredumpsize=8M:\
:openfiles=24:\
:maxproc=32:\
:priority=0:\
:requirehome:\
:passwordtime=91d:\
:umask=022:\
:ignoretime@:\
:label=partition/13,mls/5,biba/10(5-15),lomac/10[2]:
```

To set the user class default label which will be enforced by MAC, use `label`. Users are never permitted to modify this value. In a real configuration, however, the administrator would never enable every policy module. It is recommended that the rest of this chapter be reviewed before any configuration is implemented.

> **Note:** Users may change their label after they login, subject to the constraints of the policy. The example above tells the Biba policy that a process's minimum integrity is 5, its maximum is 15, and the default effective label is 10. The process will run at 10 until it chooses to change label, perhaps due to the user using setpmac(8), which will be constrained by Biba to the configured range.

After any change to `login.conf`, the login class capability database must be rebuilt using `cap_mkdb`.

Many sites have a large number of users requiring several different user classes. In depth planning is required as this may get extremely difficult to manage.

17.4.1.3 Network Interfaces and Label Settings

Labels may be set on network interfaces to help control the flow of data across the network. Policies using network interface labels function in the same way that policies function with respect to objects. Users at high settings in `biba`, for example, will not be permitted to access network interfaces with a label of low.

`maclabel` may be passed to `ifconfig` when setting the MAC label on network interfaces. For example:

```
# ifconfig bge0 maclabel biba/equal
```

will set the MAC label of `biba/equal` on the bge(4) interface. When using a setting similar to `biba/high(low-high)`, the entire label should be quoted to prevent an error from being returned.

Each policy module which supports labeling has a tunable which may be used to disable the MAC label on network interfaces. Setting the label to `equal` will have a similar effect. Review the output of `sysctl`, the policy manual pages, and the information in this chapter for more information on those tunables.

17.4.2 Singlelabel or Multilabel?

By default, the system will use `singlelabel`. For the administrator, there are several differences which offer pros and cons to the flexibility in the system's security model.

A security policy which uses `singlelabel` only permits one label, such as `biba/high`, to be used for each subject or object. This provides lower administration overhead, but decreases the flexibility of policies which support labeling.

`multilabel` permits each subject or object to have its own independent MAC label. The decision to use `multilabel` or `singlelabel` is only required for the policies which implement the labeling feature, including the Biba, Lomac, and MLS policies.

In many cases, `multilabel` may not be needed. Consider the following situation and security model:

- FreeBSD web-server using the MAC framework and a mix of the various policies.
- This machine only requires one label, `biba/high`, for everything in the system. This file system would not require `multilabel` as a single label will always be in effect.
- But, this machine will be a web server and should have the web server run at `biba/low` to prevent write up capabilities. The server could use a separate partition set at `biba/low` for most if not all of its runtime state.

If any of the non-labeling policies are to be used, `multilabel` would not be required. These include the `seeotheruids`, `portacl` and `partition` policies.

Using `multilabel` with a partition and establishing a security model based on `multilabel` functionality could increase administrative overhead as everything in the file system has a label. This includes directories, files, and even device nodes.

The following command will set `multilabel` on the file systems to have multiple labels. This may only be done in single user mode and is not a requirement for the swap file system:

```
# tunefs -l enable /
```

> **Note:** Some users have experienced problems with setting the `multilabel` flag on the root partition. If this is the case, please review the Section 17.17 of this chapter.

17.5 Planning the Security Configuration

Whenever a new technology is implemented, a planning phase is recommended. During the planning stages, an administrator should consider the implementation requirements and the implementation goals.

For MAC installations, these include:

- How to classify information and resources available on the target systems.
- Which information or resources to restrict access to along with the type of restrictions that should be applied.
- Which MAC module or modules will be required to achieve this goal.

Good planning helps to ensure a trouble-free and efficient trusted system implementation. A trial run of the trusted system and its configuration should occur *before* a MAC implementation is used on production systems. The idea of just letting loose on a system with MAC is like setting up for failure.

Different environments have different needs and requirements. Establishing an in depth and complete security profile will decrease the need of changes once the system goes live. The rest of this chapter covers the available modules, describes their use and configuration, and in some cases, provides insight on applicable situations. For instance, a web server might use the mac_biba(4) and mac_bsdextended(4) policies. In the case of a machine with few local users, mac_partition(4) might be a good choice.

17.6 Module Configuration

Beginning with FreeBSD 8.0, the default FreeBSD kernel includes `options MAC`. This means that every module included with the MAC framework may be loaded as a run-time kernel module. The recommended method is to add the module name to `/boot/loader.conf` so that it will load during boot. Each module also provides a kernel option for those administrators who choose to compile their own custom kernel.

Some modules support the use of labeling, which is controlling access by enforcing a label such as “this is allowed and this is not”. A label configuration file may control how files may be accessed, network communication can be exchanged, and more. The previous section showed how the `multilabel` flag could be set on file systems to enable per-file or per-partition access control.

A single label configuration enforces only one label across the system, that is why the `tunefs` option is called `multilabel`.

17.7 The MAC See Other UIDs Policy

Module name: `mac_seeotheruids.ko`

Kernel configuration line: `options MAC_SEEOTHERUIDS`

Boot option: `mac_seeotheruids_load="YES"`

The mac_seeotheruids(4) module mimics and extends the `security.bsd.see_other_uids` and `security.bsd.see_other_gids` `sysctl` tunables. This option does not require any labels to be set before configuration and can operate transparently with the other modules.

After loading the module, the following `sysctl` tunables may be used to control the features:

- `security.mac.seeotheruids.enabled` enables the module and uses the default settings which deny users the ability to view processes and sockets owned by other users.
- `security.mac.seeotheruids.specificgid_enabled` allows certain groups to be exempt from this policy. To exempt specific groups from this policy, use the `security.mac.seeotheruids.specificgid=`*`XXX`* `sysctl` tunable. Replace *`XXX`* with the numeric group ID to be exempted.

- `security.mac.seeotheruids.primarygroup_enabled` is used to exempt specific primary groups from this policy. When using this tunable, `security.mac.seeotheruids.specificgid_enabled` may not be set.

17.8 The MAC BSD Extended Policy

Module name: `mac_bsdextended.ko`

Kernel configuration line: `options MAC_BSDEXTENDED`

Boot option: `mac_bsdextended_load="YES"`

The mac_bsdextended(4) module enforces the file system firewall. This module's policy provides an extension to the standard file system permissions model, permitting an administrator to create a firewall-like ruleset to protect files, utilities, and directories in the file system hierarchy. When access to a file system object is attempted, the list of rules is iterated until either a matching rule is located or the end is reached. This behavior may be changed by the use of a sysctl(8) parameter, `security.mac.bsdextended.firstmatch_enabled`. Similar to other firewall modules in FreeBSD, a file containing the access control rules can be created and read by the system at boot time using an rc.conf(5) variable.

The rule list may be entered using ugidfw(8) which has a syntax similar to ipfw(8). More tools can be written by using the functions in the libugidfw(3) library.

Extreme caution should be taken when working with this module as incorrect use could block access to certain parts of the file system.

17.8.1 Examples

After the mac_bsdextended(4) module has been loaded, the following command may be used to list the current rule configuration:

```
# ugidfw list
0 slots, 0 rules
```

By default, no rules are defined and everything is completely accessible. To create a rule which will block all access by users but leave `root` unaffected, run the following command:

```
# ugidfw add subject not uid root new object not uid root mode n
```

This is a very bad idea as it will block all users from issuing even the most simple commands, such as `ls`. The next example will block `user1` any and all access, including directory listings, to *`user2`*'s home directory:

```
# ugidfw set 2 subject uid user1 object uid user2 mode n
# ugidfw set 3 subject uid user1 object gid user2 mode n
```

Instead of `user1`, `not uid` *`user2`* could be used. This enforces the same access restrictions for all users instead of just one user.

> **Note:** The `root` user is unaffected by these changes.

For more information, refer to mac_bsdextended(4) and ugidfw(8)

17.9 The MAC Interface Silencing Policy

Module name: `mac_ifoff.ko`

Kernel configuration line: `options MAC_IFOFF`

Boot option: `mac_ifoff_load="YES"`

The mac_ifoff(4) module exists solely to disable network interfaces on the fly and keep network interfaces from being brought up during system boot. It does not require any labels to be set up on the system, nor does it depend on other MAC modules.

Most of this module's control is performed through the `sysctl` tunables listed below.

- `security.mac.ifoff.lo_enabled` enables or disables all traffic on the loopback (lo(4)) interface.
- `security.mac.ifoff.bpfrecv_enabled` enables or disables all traffic on the Berkeley Packet Filter interface (bpf(4))
- `security.mac.ifoff.other_enabled` enables or disables traffic on all other interfaces.

One of the most common uses of mac_ifoff(4) is network monitoring in an environment where network traffic should not be permitted during the boot sequence. Another suggested use would be to write a script which uses `security/aide` to automatically block network traffic if it finds new or altered files in protected directories.

17.10 The MAC Port Access Control List Policy

Module name: `mac_portacl.ko`

Kernel configuration line: `MAC_PORTACL`

Boot option: `mac_portacl_load="YES"`

The mac_portacl(4) module is used to limit binding to local TCP and UDP ports using a variety of `sysctl` variables. mac_portacl(4) makes it possible to allow non-`root` users to bind to specified privileged ports below 1024.

Once loaded, this module enables the MAC policy on all sockets. The following tunables are available:

- `security.mac.portacl.enabled` enables or disables the policy completely.
- `security.mac.portacl.port_high` sets the highest port number that mac_portacl(4) protects.
- `security.mac.portacl.suser_exempt`, when set to a non-zero value, exempts the `root` user from this policy.
- `security.mac.portacl.rules` specifies the mac_portacl policy, which is a text string of the form: `rule[,rule,...]` with as many rules as needed. Each rule is of the form: `idtype:id:protocol:port`. The *`idtype`* parameter can be `uid` or `gid` and is used to interpret the *`id`* parameter as either a user id or group id, respectively. The *`protocol`* parameter is used to determine if the rule should apply to TCP or UDP by setting the parameter to `tcp` or `udp`. The final *`port`* parameter is the port number to allow the specified user or group to bind to.

  Note: Since the ruleset is interpreted directly by the kernel, only numeric values can be used for the user ID, group ID, and port parameters. Names cannot be used for users, groups, or services.

By default, ports below 1024 can only be used by or bound to privileged processes, which run as `root`. For mac_portacl(4) to allow non-privileged processes to bind to ports below 1024, this restriction has to be disabled by setting the sysctl(8) variables `net.inet.ip.portrange.reservedlow` and `net.inet.ip.portrange.reservedhigh` to zero:

```
# sysctl security.mac.portacl.port_high=1023
# sysctl net.inet.ip.portrange.reservedlow=0
net.inet.ip.portrange.reservedhigh=0
```

See the examples below or refer to mac_portacl(4) for further information.

17.10.1 Examples

Since the `root` user should not be crippled by this policy, this example starts by setting the `security.mac.portacl.suser_exempt` to a non-zero value.

```
# sysctl security.mac.portacl.suser_exempt=1
```

Next, allow the user with UID 80 to bind to port 80. This allows the `www` user to run a web server without ever having `root` privilege.

```
# sysctl security.mac.portacl.rules=uid:80:tcp:80
```

The next example permits the user with the UID of 1001 to bind to the TCP ports 110 (“pop3”) and 995 (“pop3s”). This permits this user to start a server that accepts connections on ports 110 and 995.

```
# sysctl security.mac.portacl.rules=uid:1001:tcp:110,uid:1001:tcp:995
```

17.11 The MAC Partition Policy

Module name: `mac_partition.ko`

Kernel configuration line: `options MAC_PARTITION`

Boot option: `mac_partition_load="YES"`

The mac_partition(4) policy will drop processes into specific “partitions” based on their MAC label. This module should be added to loader.conf(5) so that it loads and enables the policy at system boot.

Most configuration for this policy is done using setpmac(8). One `sysctl` tunable is available for this policy:

- `security.mac.partition.enabled` enables the enforcement of MAC process partitions.

When this policy is enabled, users will only be permitted to see their processes, and any others within their partition, but will not be permitted to work with utilities outside the scope of this partition. For instance, a user in the `insecure` class will not be permitted to access `top` as well as many other commands that must spawn a process.

To set or drop utilities into a partition label, use the `setpmac` utility:

```
# setpmac partition/13 top
```

This example adds `top` to the label set on users in the `insecure` class. All processes spawned by users in the `insecure` class will stay in the `partition/13` label.

17.11.1 Examples

The following command will display the partition label and the process list:

```
# ps Zax
```

This command will display another user's process partition label and that user's currently running processes:

```
# ps -ZU trhodes
```

> **Note:** Users can see processes in `root`'s label unless the mac_seeotheruids(4) policy is loaded.

A really crafty implementation could have all of the services disabled in `/etc/rc.conf` and started by a script that starts them with the proper labeling set.

> **Note:** The following policies support integer settings in place of the three default labels offered. These options, including their limitations, are further explained in the module manual pages.

17.12 The MAC Multi-Level Security Module

Module name: `mac_mls.ko`

Kernel configuration line: `options MAC_MLS`

Boot option: `mac_mls_load="YES"`

The mac_mls(4) policy controls access between subjects and objects in the system by enforcing a strict information flow policy.

In MLS environments, a "clearance" level is set in the label of each subject or object, along with compartments. Since these clearance or sensibility levels can reach numbers greater than several thousand; it would be a daunting task for any system administrator to thoroughly configure each subject or object. Thankfully, three "instant" labels are included in this policy.

These labels are `mls/low`, `mls/equal` and `mls/high`. Since these labels are described in depth in the manual page, they will only get a brief description here:

- The `mls/low` label contains a low configuration which permits it to be dominated by all other objects. Anything labeled with `mls/low` will have a low clearance level and not be permitted to access information of a higher level. This label also prevents objects of a higher clearance level from writing or passing information on to them.
- The `mls/equal` label should be placed on objects considered to be exempt from the policy.

- The `mls/high` label is the highest level of clearance possible. Objects assigned this label will hold dominance over all other objects in the system; however, they will not permit the leaking of information to objects of a lower class.

MLS provides:

- A hierarchical security level with a set of non hierarchical categories.
- Fixed rules of `no read up, no write down`. This means that a subject can have read access to objects on its own level or below, but not above. Similarly, a subject can have write access to objects on its own level or above but not beneath.
- Secrecy, or the prevention of inappropriate disclosure of data.
- A basis for the design of systems that concurrently handle data at multiple sensitivity levels without leaking information between secret and confidential.

The following `sysctl` tunables are available for the configuration of special services and interfaces:

- `security.mac.mls.enabled` is used to enable or disable the MLS policy.
- `security.mac.mls.ptys_equal` labels all pty(4) devices as `mls/equal` during creation.
- `security.mac.mls.revocation_enabled` revokes access to objects after their label changes to a label of a lower grade.
- `security.mac.mls.max_compartments` sets the maximum number of compartment levels allowed on a system.

To manipulate the MLS labels, use setfmac(8). To assign a label to an object, issue the following command:

```
# setfmac mls/5 test
```

To get the MLS label for the file `test`, issue the following command:

```
# getfmac test
```

Another approach is to create a master policy file in `/etc/` which specifies the MLS policy information and to feed that file to `setfmac`. This method will be explained after all policies are covered.

17.12.1 Planning Mandatory Sensitivity

When using the MLS policy module, an administrator plans to control the flow of sensitive information. The default `block read up block write down` sets everything to a low state. Everything is accessible and an administrator slowly augments the confidentiality of the information during the configuration stage;.

Beyond the three basic label options, an administrator may group users and groups as required to block the information flow between them. It might be easier to look at the information in clearance levels using descriptive words, such as classifications of `Confidential`, `Secret`, and `Top Secret`. Some administrators instead create different groups based on project levels. Regardless of the classification method, a well thought out plan must exist before implementing such a restrictive policy.

Some example situations for the MLS policy module include an e-commerce web server, a file server holding critical company information, and financial institution environments.

17.13 The MAC Biba Module

Module name: `mac_biba.ko`

Kernel configuration line: `options MAC_BIBA`

Boot option: `mac_biba_load="YES"`

The mac_biba(4) module loads the MAC Biba policy. This policy is similar to the MLS policy with the exception that the rules for information flow are slightly reversed. This is to prevent the downward flow of sensitive information whereas the MLS policy prevents the upward flow of sensitive information. Much of this section can apply to both policies.

In Biba environments, an "integrity" label is set on each subject or object. These labels are made up of hierarchical grades and non-hierarchical components. As an grade ascends, so does its integrity.

Supported labels are `biba/low`, `biba/equal`, and `biba/high`; as explained below:

- The `biba/low` label is considered the lowest integrity an object or subject may have. Setting this on objects or subjects will block their write access to objects or subjects marked high. They still have read access though.
- The `biba/equal` label should only be placed on objects considered to be exempt from the policy.
- The `biba/high` label will permit writing to objects set at a lower label, but not permit reading that object. It is recommended that this label be placed on objects that affect the integrity of the entire system.

Biba provides:

- Hierarchical integrity level with a set of non hierarchical integrity categories.
- Fixed rules are `no write up, no read down`, the opposite of MLS. A subject can have write access to objects on its own level or below, but not above. Similarly, a subject can have read access to objects on its own level or above, but not below.
- Integrity by preventing inappropriate modification of data.
- Integrity levels instead of MLS sensitivity levels.

The following `sysctl` tunables can be used to manipulate the Biba policy:

- `security.mac.biba.enabled` is used to enable or disable enforcement of the Biba policy on the target machine.
- `security.mac.biba.ptys_equal` is used to disable the Biba policy on pty(4) devices.
- `security.mac.biba.revocation_enabled` forces the revocation of access to objects if the label is changed to dominate the subject.

To access the Biba policy setting on system objects, use `setfmac` and `getfmac`:

```
# setfmac biba/low test
# getfmac test
test: biba/low
```

17.13.1 Planning Mandatory Integrity

Integrity, which is different from sensitivity, guarantees that the information will never be manipulated by untrusted parties. This includes information passed between subjects, objects, and both. It ensures that users will only be able to modify or access information they explicitly need to.

The mac_biba(4) security policy module permits an administrator to address which files and programs a user may see and invoke while assuring that the programs and files are free from threats and trusted by the system for that user.

During the initial planning phase, an administrator must be prepared to partition users into grades, levels, and areas. Users will be blocked access not only to data but to programs and utilities both before and after they start. The system will default to a high label once this policy module is enabled, and it is up to the administrator to configure the different grades and levels for users. Instead of using clearance levels, a good planning method could include topics. For instance, only allow developers modification access to the source code repository, source code compiler, and other development utilities. Other users would be grouped into other categories such as testers, designers, or end users and would only be permitted read access.

A lower integrity subject is unable to write to a higher integrity subject and a higher integrity subject cannot observe or read a lower integrity object. Setting a label at the lowest possible grade could make it inaccessible to subjects. Some prospective environments for this security policy module would include a constrained web server, a development and test machine, and a source code repository. A less useful implementation would be a personal workstation, a machine used as a router, or a network firewall.

17.14 The MAC LOMAC Module

Module name: `mac_lomac.ko`

Kernel configuration line: `options MAC_LOMAC`

Boot option: `mac_lomac_load="YES"`

Unlike the MAC Biba policy, the mac_lomac(4) policy permits access to lower integrity objects only after decreasing the integrity level to not disrupt any integrity rules.

The MAC version of the Low-watermark integrity policy works almost identically to Biba, but with the exception of using floating labels to support subject demotion via an auxiliary grade compartment. This secondary compartment takes the form `[auxgrade]`. When assigning a LOMAC policy with an auxiliary grade, use the syntax `lomac/10[2]` where the number two (2) is the auxiliary grade.

The MAC LOMAC policy relies on the ubiquitous labeling of all system objects with integrity labels, permitting subjects to read from low integrity objects and then downgrading the label on the subject to prevent future writes to high integrity objects using `[auxgrade]`. The policy may provide for greater compatibility and require less initial configuration than Biba.

17.14.1 Examples

Like the Biba and MLS policies, `setfmac` and `setpmac` are used to place labels on system objects:

```
# setfmac /usr/home/trhodes lomac/high[low]
# getfmac /usr/home/trhodes lomac/high[low]
```

The auxiliary grade `low` is a feature provided only by the MAC LOMAC policy.

17.15 Nagios in a MAC Jail

The following demonstration implements a secure environment using various MAC modules with properly configured policies. This is only a test as implementing a policy and ignoring it could be disastrous in a production environment.

Before beginning this process, `multilabel` must be set on each file system as not doing so will result in errors. This example assumes that `net-mngt/nagios-plugins`, `net-mngt/nagios`, and `www/apache22` are all installed, configured, and working correctly.

17.15.1 Create an Insecure User Class

Begin the procedure by adding the following user class to `/etc/login.conf`:

```
insecure:\
:copyright=/etc/COPYRIGHT:\
:welcome=/etc/motd:\
:setenv=MAIL=/var/mail/$,BLOCKSIZE=K:\
:path=~/bin:/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/sbin:/usr/local/bin
:manpath=/usr/share/man /usr/local/man:\
:nologin=/usr/sbin/nologin:\
:cputime=1h30m:\
:datasize=8M:\
:vmemoryuse=100M:\
:stacksize=2M:\
:memorylocked=4M:\
:memoryuse=8M:\
:filesize=8M:\
:coredumpsize=8M:\
:openfiles=24:\
:maxproc=32:\
:priority=0:\
:requirehome:\
:passwordtime=91d:\
:umask=022:\
:ignoretime@:\
:label=biba/10(10-10):
```

Add the following line to the default user class:

```
:label=biba/high:
```

Next, issue the following command to rebuild the database:

```
# cap_mkdb /etc/login.conf
```

17.15.2 Boot Configuration

Add the following lines to `/boot/loader.conf`:

```
mac_biba_load="YES"
mac_seeotheruids_load="YES"
```

17.15.3 Configure Users

Set the `root` user to the default class using:

```
# pw usermod root -L default
```

All user accounts that are not `root` or system users will now require a login class. The login class is required otherwise users will be refused access to common commands such as vi(1). The following `sh` script should do the trick:

```
# for x in 'awk -F: '($3 >= 1001) && ($3 != 65534) { print $1 }' \
        /etc/passwd'; do pw usermod $x -L default; done;
```

Drop the `nagios` and `www` users into the insecure class:

```
# pw usermod nagios -L insecure

# pw usermod www -L insecure
```

17.15.4 Create the Contexts File

A contexts file should now be created as `/etc/policy.contexts`.

```
# This is the default BIBA policy for this system.

# System:
/var/run                        biba/equal
/var/run/*                      biba/equal

/dev                            biba/equal
/dev/*                          biba/equal

/var                           biba/equal
/var/spool                      biba/equal
/var/spool/*                    biba/equal

/var/log                        biba/equal
/var/log/*                      biba/equal

/tmp                           biba/equal
/tmp/*                         biba/equal
/var/tmp                       biba/equal
/var/tmp/*                     biba/equal

/var/spool/mqueue              biba/equal
/var/spool/clientmqueue                biba/equal

# For Nagios:
/usr/local/etc/nagios
/usr/local/etc/nagios/*         biba/10

/var/spool/nagios               biba/10
```

```
/var/spool/nagios/*             biba/10

# For apache
/usr/local/etc/apache           biba/10
/usr/local/etc/apache/*         biba/10
```

This policy enforces security by setting restrictions on the flow of information. In this specific configuration, users, including `root`, should never be allowed to access **Nagios**. Configuration files and processes that are a part of **Nagios** will be completely self contained or jailed.

This file will be read by the system by issuing the following command:

```
# setfsmac -ef /etc/policy.contexts /
# setfsmac -ef /etc/policy.contexts /
```

> **Note:** The above file system layout will differ depending upon the environment and must be run on every file system.

`/etc/mac.conf` requires the following modifications in the main section:

```
default_labels file ?biba
default_labels ifnet ?biba
default_labels process ?biba
default_labels socket ?biba
```

17.15.5 Enable Networking

Add the following line to `/boot/loader.conf`:

```
security.mac.biba.trust_all_interfaces=1
```

And the following to the network card configuration stored in `rc.conf`. If the primary Internet configuration is done via DHCP, this may need to be configured manually after every system boot:

```
maclabel biba/equal
```

17.15.6 Testing the Configuration

Ensure that the web server and **Nagios** will not be started on system initialization and reboot. Ensure the `root` user cannot access any of the files in the **Nagios** configuration directory. If `root` can issue an ls(1) command on `/var/spool/nagios`, something is wrong. Otherwise a “permission denied” error should be returned.

If all seems well, **Nagios**, **Apache**, and **Sendmail** can now be started:

```
# cd /etc/mail && make stop && \
setpmac biba/equal make start && setpmac biba/10\(10-10\) apachectl start && \
setpmac biba/10\(10-10\) /usr/local/etc/rc.d/nagios.sh forcestart
```

Double check to ensure that everything is working properly. If not, check the log files for error messages. Use sysctl(8) to disable the mac_biba(4) security policy module enforcement and try starting everything again as usual.

> **Note:** The `root` user can still change the security enforcement and edit its configuration files. The following command will permit the degradation of the security policy to a lower grade for a newly spawned shell:
>
> ```
> # setpmac biba/10 csh
> ```
>
> To block this from happening, force the user into a range using login.conf(5). If setpmac(8) attempts to run a command outside of the compartment's range, an error will be returned and the command will not be executed. In this case, set root to `biba/high(high-high)`.

17.16 User Lock Down

This example considers a relatively small storage system with fewer than fifty users. Users will have login capabilities, and be permitted to store data and access resources.

For this scenario, the mac_bsdextended(4) and mac_seeotheruids(4) policy modules could co-exist and block access to system objects while hiding user processes.

Begin by adding the following line to `/boot/loader.conf`:

```
mac_seeotheruids_load="YES"
```

The mac_bsdextended(4) security policy module may be activated by adding this line to `/etc/rc.conf`:

```
ugidfw_enable="YES"
```

Default rules stored in `/etc/rc.bsdextended` will be loaded at system initialization. However, the default entries may need modification. Since this machine is expected only to service users, everything may be left commented out except the last two lines in order to force the loading of user owned system objects by default.

Add the required users to this machine and reboot. For testing purposes, try logging in as a different user across two consoles. Run `ps aux` to see if processes of other users are visible. Verify that running ls(1) on another user's home directory fails.

Do not try to test with the `root` user unless the specific `sysctl`s have been modified to block super user access.

> **Note:** When a new user is added, their mac_bsdextended(4) rule will not be in the ruleset list. To update the ruleset quickly, unload the security policy module and reload it again using kldunload(8) and kldload(8).

17.17 Troubleshooting the MAC Framework

This section discusses common configuration issues.

- The `multilabel` flag does not stay enabled on my root (`/`) partition!

The following steps may resolve this transient error:

1. Edit `/etc/fstab` and set the root partition to `ro` for read-only.
2. Reboot into single user mode.
3. Run `tunefs -l enable` on `/`.
4. Reboot the system.
5. Run `mount -urw /` and change the `ro` back to `rw` in `/etc/fstab` and reboot the system again.
6. Double-check the output from `mount` to ensure that `multilabel` has been properly set on the root file system.

- After establishing a secure environment with MAC, I am no longer able to start Xorg!

  This could be caused by the MAC `partition` policy or by a mislabeling in one of the MAC labeling policies. To debug, try the following:

  1. Check the error message; if the user is in the `insecure` class, the `partition` policy may be the culprit. Try setting the user's class back to the `default` class and rebuild the database with `cap_mkdb`. If this does not alleviate the problem, go to step two.
  2. Double-check the label policies. Ensure that the policies are set correctly for the user, the Xorg application, and the `/dev` entries.
  3. If neither of these resolve the problem, send the error message and a description of the environment to the FreeBSD general questions mailing list (http://lists.FreeBSD.org/mailman/listinfo/freebsd-questions) mailing list.

- The error: `_secure_path: unable to stat .login_conf` shows up.

  When a user attempts to switch from the `root` user to another user in the system, the error message `_secure_path: unable to stat .login_conf` appears.

  This message is usually shown when the user has a higher label setting than that of the user they are attempting to become. For instance, `joe` has a default label of `biba/low`. The `root` user, who has a label of `biba/high`, cannot view `joe`'s home directory. This will happen whether or not `root` has used `su` to become `joe` as the Biba integrity model will not permit `root` to view objects set at a lower integrity level.

- The system no longer recognizes the `root` user.

  In normal or even single user mode, the `root` is not recognized, `whoami` returns 0 (zero), and `su` returns `who are you?`.

  This can happen if a labeling policy has been disabled, either by a sysctl(8) or the policy module was unloaded. If the policy is disabled, the login capabilities database needs to be reconfigured with `label` removed. Double check `login.conf` to ensure that all `label` options have been removed and rebuild the database with `cap_mkdb`.

  This may also happen if a policy restricts access to `master.passwd`. This is usually caused by an administrator altering the file under a label which conflicts with the general policy being used by the system. In these cases, the user information would be read by the system and access would be blocked as the file has inherited the new label. Disable the policy using sysctl(8) and everything should return to normal.

Notes

1. Other conditions may produce different failures. For instance, the file may not be owned by the user attempting to relabel the object, the object may not exist, or the object may be read only. A mandatory policy will not allow the process to relabel the file, maybe because of a property of the file, a property of the process, or a property of the proposed new label value. For example, a user running at low integrity tries to change the label of a high integrity file. Or perhaps a user running at low integrity tries to change the label of a low integrity file to a high integrity label.

Chapter 18 Security Event Auditing

Written by Tom Rhodes and Robert Watson.

18.1 Synopsis

The FreeBSD operating system includes support for fine-grained security event auditing. Event auditing allows the reliable, fine-grained, and configurable logging of a variety of security-relevant system events, including logins, configuration changes, and file and network access. These log records can be invaluable for live system monitoring, intrusion detection, and postmortem analysis. FreeBSD implements Sun's published BSM API and file format, and is interoperable with both Sun's Solaris and Apple's Mac OS X audit implementations.

This chapter focuses on the installation and configuration of Event Auditing. It explains audit policies, and provides an example audit configuration.

After reading this chapter, you will know:

- What Event Auditing is and how it works.
- How to configure Event Auditing on FreeBSD for users and processes.
- How to review the audit trail using the audit reduction and review tools.

Before reading this chapter, you should:

- Understand UNIX and FreeBSD basics (Chapter 4).
- Be familiar with the basics of kernel configuration/compilation (Chapter 9).
- Have some familiarity with security and how it pertains to FreeBSD (Chapter 15).

> **Warning:** The audit facility has some known limitations which include that not all security-relevant system events are currently auditable, and that some login mechanisms, such as X11-based display managers and third party daemons, do not properly configure auditing for user login sessions.
>
> The security event auditing facility is able to generate very detailed logs of system activity: on a busy system, trail file data can be very large when configured for high detail, exceeding gigabytes a week in some configurations. Administrators should take into account disk space requirements associated with high volume audit configurations. For example, it may be desirable to dedicate a file system to the `/var/audit` tree so that other file systems are not affected if the audit file system becomes full.

18.2 Key Terms in This Chapter

Before reading this chapter, a few key audit-related terms must be explained:

- *event*: An auditable event is any event that can be logged using the audit subsystem. Examples of security-relevant events include the creation of a file, the building of a network connection, or a user logging in. Events are either "attributable", meaning that they can be traced to an authenticated user, or "non-attributable" if they cannot be.

Examples of non-attributable events are any events that occur before authentication in the login process, such as bad password attempts.

- *class*: Event classes are named sets of related events, and are used in selection expressions. Commonly used classes of events include “file creation” (fc), “exec” (ex) and “login_logout” (lo).
- *record*: A record is an audit log entry describing a security event. Records contain a record event type, information on the subject (user) performing the action, date and time information, information on any objects or arguments, and a success or failure condition.
- *trail*: An audit trail, or log file, consists of a series of audit records describing security events. Typically, trails are in roughly chronological order with respect to the time events completed. Only authorized processes are allowed to commit records to the audit trail.
- *selection expression*: A selection expression is a string containing a list of prefixes and audit event class names used to match events.
- *preselection*: The process by which the system identifies which events are of interest to the administrator in order to avoid generating audit records describing events that are not of interest. The preselection configuration uses a series of selection expressions to identify which classes of events to audit for which users, as well as global settings that apply to both authenticated and unauthenticated processes.
- *reduction*: The process by which records from existing audit trails are selected for preservation, printing, or analysis. Likewise, the process by which undesired audit records are removed from the audit trail. Using reduction, administrators can implement policies for the preservation of audit data. For example, detailed audit trails might be kept for one month, but after that, trails might be reduced in order to preserve only login information for archival purposes.

18.3 Installing Audit Support

User space support for Event Auditing is installed as part of the base FreeBSD operating system. Kernel support for Event Auditing is compiled in by default, but support for this feature must be explicitly compiled into the custom kernel by adding the following line to the kernel configuration file:

```
options AUDIT
```

Rebuild and reinstall the kernel via the normal process explained in Chapter 9.

Once an audit-enabled kernel is built, installed, and the system has been rebooted, enable the audit daemon by adding the following line to rc.conf(5):

```
auditd_enable="YES"
```

Audit support must then be started by a reboot, or by manually starting the audit daemon:

```
service auditd start
```

18.4 Audit Configuration

All configuration files for security audit are found in `/etc/security`. The following files must be present before the audit daemon is started:

- `audit_class` - Contains the definitions of the audit classes.
- `audit_control` - Controls aspects of the audit subsystem, such as default audit classes, minimum disk space to leave on the audit log volume, maximum audit trail size, etc.
- `audit_event` - Textual names and descriptions of system audit events, as well as a list of which classes each event is in.
- `audit_user` - User-specific audit requirements, which are combined with the global defaults at login.
- `audit_warn` - A customizable shell script used by auditd(8) to generate warning messages in exceptional situations, such as when space for audit records is running low or when the audit trail file has been rotated.

> **Warning:** Audit configuration files should be edited and maintained carefully, as errors in configuration may result in improper logging of events.

18.4.1 Event Selection Expressions

Selection expressions are used in a number of places in the audit configuration to determine which events should be audited. Expressions contain a list of event classes to match, each with a prefix indicating whether matching records should be accepted or ignored, and optionally to indicate if the entry is intended to match successful or failed operations. Selection expressions are evaluated from left to right, and two expressions are combined by appending one onto the other.

The following list contains the default audit event classes present in `audit_class`:

- `all` - *all* - Match all event classes.
- `ad` - *administrative* - Administrative actions performed on the system as a whole.
- `ap` - *application* - Application defined action.
- `cl` - *file close* - Audit calls to the `close` system call.
- `ex` - *exec* - Audit program execution. Auditing of command line arguments and environmental variables is controlled via audit_control(5) using the `argv` and `envv` parameters to the `policy` setting.
- `fa` - *file attribute access* - Audit the access of object attributes such as stat(1), pathconf(2) and similar events.
- `fc` - *file create* - Audit events where a file is created as a result.
- `fd` - *file delete* - Audit events where file deletion occurs.
- `fm` - *file attribute modify* - Audit events where file attribute modification occurs, such as chown(8), chflags(1), flock(2), etc.
- `fr` - *file read* - Audit events in which data is read, files are opened for reading, etc.
- `fw` - *file write* - Audit events in which data is written, files are written or modified, etc.
- `io` - *ioctl* - Audit use of the ioctl(2) system call.
- `ip` - *ipc* - Audit various forms of Inter-Process Communication, including POSIX pipes and System V IPC operations.
- `lo` - *login_logout* - Audit login(1) and logout(1) events occurring on the system.
- `na` - *non attributable* - Audit non-attributable events.

- `no` - *invalid class* - Match no audit events.
- `nt` - *network* - Audit events related to network actions, such as connect(2) and accept(2).
- `ot` - *other* - Audit miscellaneous events.
- `pc` - *process* - Audit process operations, such as exec(3) and exit(3).

These audit event classes may be customized by modifying the `audit_class` and `audit_ event` configuration files.

Each audit class in the list is combined with a prefix indicating whether successful/failed operations are matched, and whether the entry is adding or removing matching for the class and type.

- (none) Audit both successful and failed instances of the event.
- `+` Audit successful events in this class.
- `-` Audit failed events in this class.
- `^` Audit neither successful nor failed events in this class.
- `^+` Do not audit successful events in this class.
- `^-` Do not audit failed events in this class.

The following example selection string selects both successful and failed login/logout events, but only successful execution events:

```
lo,+ex
```

18.4.2 Configuration Files

In most cases, administrators will need to modify only two files when configuring the audit system: `audit_ control` and `audit_user`. The first controls system-wide audit properties and policies; the second may be used to fine-tune auditing by user.

18.4.2.1 The `audit_control` File

A number of defaults for the audit subsystem are specified in `audit_control`:

```
dir:/var/audit
flags:lo
minfree:20
naflags:lo
policy:cnt
filesz:0
```

The `dir` entry is used to set one or more directories where audit logs will be stored. If more than one directory entry appears, they will be used in order as they fill. It is common to configure audit so that audit logs are stored on a dedicated file system, in order to prevent interference between the audit subsystem and other subsystems if the file system fills.

The `flags` field sets the system-wide default preselection mask for attributable events. In the example above, successful and failed login and logout events are audited for all users.

The `minfree` entry defines the minimum percentage of free space for the file system where the audit trail is stored. When this threshold is exceeded, a warning will be generated. The above example sets the minimum free space to twenty percent.

The `naflags` entry specifies audit classes to be audited for non-attributed events, such as the login process and system daemons.

The `policy` entry specifies a comma-separated list of policy flags controlling various aspects of audit behavior. The default `cnt` flag indicates that the system should continue running despite an auditing failure (this flag is highly recommended). Another commonly used flag is `argv`, which causes command line arguments to the execve(2) system call to be audited as part of command execution.

The `filesz` entry specifies the maximum size in bytes to allow an audit trail file to grow to before automatically terminating and rotating the trail file. The default, 0, disables automatic log rotation. If the requested file size is non-zero and below the minimum 512k, it will be ignored and a log message will be generated.

18.4.2.2 The `audit_user` File

The administrator can specify further audit requirements for specific users in `audit_user`. Each line configures auditing for a user via two fields: the first is the `alwaysaudit` field, which specifies a set of events that should always be audited for the user, and the second is the `neveraudit` field, which specifies a set of events that should never be audited for the user.

The following example `audit_user` audits login/logout events and successful command execution for `root`, and audits file creation and successful command execution for `www`. If used with the above example `audit_control`, the `lo` entry for `root` is redundant, and login/logout events will also be audited for `www`.

```
root:lo,+ex:no
www:fc,+ex:no
```

18.5 Administering the Audit Subsystem

18.5.1 Viewing Audit Trails

Audit trails are stored in the BSM binary format, so tools must be used to modify or convert to text. The praudit(1) command converts trail files to a simple text format; the auditreduce(1) command may be used to reduce the audit trail file for analysis, archiving, or printing purposes. A variety of selection parameters are supported by auditreduce(1), including event type, event class, user, date or time of the event, and the file path or object acted on.

For example, praudit(1) will dump the entire contents of a specified audit log in plain text:

```
# praudit /var/audit/AUDITFILE
```

Where *`AUDITFILE`* is the audit log to dump.

Audit trails consist of a series of audit records made up of tokens, which praudit(1) prints sequentially one per line. Each token is of a specific type, such as `header` holding an audit record header, or `path` holding a file path from a name lookup. The following is an example of an `execve` event:

```
header,133,10,execve(2),0,Mon Sep 25 15:58:03 2006, + 384 msec
exec arg,finger,doug
path,/usr/bin/finger
attribute,555,root,wheel,90,24918,104944
subject,robert,root,wheel,root,wheel,38439,38032,42086,128.232.9.100
return,success,0
trailer,133
```

This audit represents a successful `execve` call, in which the command `finger doug` has been run. The arguments token contains both the processed command line presented by the shell to the kernel. The `path` token holds the path to the executable as looked up by the kernel. The `attribute` token describes the binary, and in particular, includes the file mode which can be used to determine if the application was setuid. The `subject` token describes the subject process, and stores in sequence the audit user ID, effective user ID and group ID, real user ID and group ID, process ID, session ID, port ID, and login address. Notice that the audit user ID and real user ID differ: the user `robert` has switched to the `root` account before running this command, but it is audited using the original authenticated user. Finally, the `return` token indicates the successful execution, and the `trailer` concludes the record.

XML output format is also supported by praudit(1), and can be selected using `-x`.

18.5.2 Reducing Audit Trails

Since audit logs may be very large, an administrator will likely want to select a subset of records for using, such as records associated with a specific user:

```
# auditreduce -u trhodes /var/audit/AUDITFILE | praudit
```

This will select all audit records produced for `trhodes` stored in *`AUDITFILE`*.

18.5.3 Delegating Audit Review Rights

Members of the `audit` group are given permission to read audit trails in `/var/audit`; by default, this group is empty, so only the `root` user may read audit trails. Users may be added to the `audit` group in order to delegate audit review rights to the user. As the ability to track audit log contents provides significant insight into the behavior of users and processes, it is recommended that the delegation of audit review rights be performed with caution.

18.5.4 Live Monitoring Using Audit Pipes

Audit pipes are cloning pseudo-devices in the device file system which allow applications to tap the live audit record stream. This is primarily of interest to authors of intrusion detection and system monitoring applications. However, for the administrator the audit pipe device is a convenient way to allow live monitoring without running into problems with audit trail file ownership or log rotation interrupting the event stream. To track the live audit event stream, use the following command line:

```
# praudit /dev/auditpipe
```

By default, audit pipe device nodes are accessible only to the `root` user. To make them accessible to the members of the `audit` group, add a `devfs` rule to `devfs.rules`:

```
add path 'auditpipe*' mode 0440 group audit
```

See devfs.rules(5) for more information on configuring the devfs file system.

> **Warning:** It is easy to produce audit event feedback cycles, in which the viewing of each audit event results in the generation of more audit events. For example, if all network I/O is audited, and praudit(1) is run from an SSH session, then a continuous stream of audit events will be generated at a high rate, as each event being printed will generate another event. It is advisable to run praudit(1) on an audit pipe device from sessions without fine-grained I/O auditing in order to avoid this happening.

18.5.5 Rotating Audit Trail Files

Audit trails are written to only by the kernel, and managed only by the audit daemon, auditd(8). Administrators should not attempt to use newsyslog.conf(5) or other tools to directly rotate audit logs. Instead, the audit(8) management tool may be used to shut down auditing, reconfigure the audit system, and perform log rotation. The following command causes the audit daemon to create a new audit log and signal the kernel to switch to using the new log. The old log will be terminated and renamed, at which point it may then be manipulated by the administrator.

```
# audit -n
```

> **Warning:** If auditd(8) is not currently running, this command will fail and an error message will be produced.

Adding the following line to `/etc/crontab` will force the rotation every twelve hours from cron(8):

```
0     */12       *       *       *       root    /usr/sbin/audit -n
```

The change will take effect once you have saved the new `/etc/crontab`.

Automatic rotation of the audit trail file based on file size is possible using `filesz` in audit_control(5), and is described in the configuration files section of this chapter.

18.5.6 Compressing Audit Trails

As audit trail files can become very large, it is often desirable to compress or otherwise archive trails once they have been closed by the audit daemon. The `audit_warn` script can be used to perform customized operations for a variety of audit-related events, including the clean termination of audit trails when they are rotated. For example, the following may be added to the `audit_warn` script to compress audit trails on close:

```
#
# Compress audit trail files on close.
#
if [ "$1" = closefile ]; then
        gzip -9 $2
fi
```

Other archiving activities might include copying trail files to a centralized server, deleting old trail files, or reducing the audit trail to remove unneeded records. The script will be run only when audit trail files are cleanly terminated, so will not be run on trails left unterminated following an improper shutdown.

Chapter 19 Storage

19.1 Synopsis

This chapter covers the use of disks in FreeBSD. This includes memory-backed disks, network-attached disks, standard SCSI/IDE storage devices, and devices using the USB interface.

After reading this chapter, you will know:

- The terminology FreeBSD uses to describe the organization of data on a physical disk.
- How to add additional hard disks to a FreeBSD system.
- How to configure FreeBSD to use USB storage devices.
- How to set up virtual file systems, such as memory disks.
- How to use quotas to limit disk space usage.
- How to encrypt disks to secure them against attackers.
- How to create and burn CDs and DVDs on FreeBSD.
- How to use the backup programs available under FreeBSD.
- What file system snapshots are and how to use them efficiently.

Before reading this chapter, you should:

- Know how to configure and install a new FreeBSD kernel.

19.2 Device Names

The following is a list of physical storage devices supported in FreeBSD and their associated device names.

Table 19-1. Physical Disk Naming Conventions

Drive type	Drive device name
IDE hard drives	`ad` or `ada`
IDE CD-ROM drives	`acd` or `cd`
SATA hard drives	`ad` or `ada`
SATA CD-ROM drives	`acd` or `cd`
SCSI hard drives and USB Mass storage devices	`da`
SCSI CD-ROM drives	`cd`
Assorted non-standard CD-ROM drives	`mcd` for Mitsumi CD-ROM and `scd` for Sony CD-ROM devices
Floppy drives	`fd`
SCSI tape drives	`sa`
IDE tape drives	`ast`

Drive type	Drive device name
Flash drives	`fla` for DiskOnChip® Flash device
RAID drives	`aacd` for Adaptec® AdvancedRAID, `mlxd` and `mlyd` for Mylex®, `amrd` for AMI MegaRAID®, `idad` for Compaq Smart RAID, `twed` for 3ware® RAID.

19.3 Adding Disks

Originally contributed by David O'Brien.

This section describes how to add a new SATA disk to a machine that currently only has a single drive. First, turn off the computer and install the drive in the computer following the instructions of the computer, controller, and drive manufacturers. Reboot the system and become `root`.

Inspect `/var/run/dmesg.boot` to ensure the new disk was found. In this example, the newly added SATA drive will appear as `ada1`.

For this example, a single large partition will be created on the new disk. The GPT (http://en.wikipedia.org/wiki/GUID_Partition_Table) partitioning scheme will be used in preference to the older and less versatile MBR scheme.

> **Note:** If the disk to be added is not blank, old partition information can be removed with `gpart delete`. See gpart(8) for details.

The partition scheme is created, and then a single partition is added:

```
# gpart create -s GPT ada1
# gpart add -t freebsd-ufs ada1
```

Depending on use, several smaller partitions may be desired. See gpart(8) for options to create partitions smaller than a whole disk.

A file system is created on the new blank disk:

```
# newfs -U /dev/ada1p1
```

An empty directory is created as a *mountpoint*, a location for mounting the new disk in the original disk's file system:

```
# mkdir /newdisk
```

Finally, an entry is added to `/etc/fstab` so the new disk will be mounted automatically at startup:

```
/dev/ada1p1     /newdisk        ufs     rw      2       2
```

The new disk can be mounted manually, without restarting the system:

```
# mount /newdisk
```

19.4 USB Storage Devices

Contributed by Marc Fonvieille.

Many external storage solutions, such as hard drives, USB thumbdrives, and CD/DVD burners, use the Universal Serial Bus (USB). FreeBSD provides support for these devices.

19.4.1 Configuration

The USB mass storage devices driver, umass(4), is built into the `GENERIC` kernel and provides support for USB storage devices. For a custom kernel, be sure that the following lines are present in the kernel configuration file:

```
device scbus
device da
device pass
device uhci
device ohci
device ehci
device usb
device umass
```

Since the umass(4) driver uses the SCSI subsystem to access the USB storage devices, any USB device will be seen as a SCSI device by the system. Depending on the USB chipset on the motherboard, `device uhci` or `device ohci` is used to provide USB 1.X support. Support for USB 2.0 controllers is provided by `device ehci`.

> **Note:** If the USB device is a CD or DVD burner, cd(4), must be added to the kernel via the line:
>
> ```
> device cd
> ```
>
> Since the burner is seen as a SCSI drive, the driver atapicam(4) should not be used in the kernel configuration.

19.4.2 Testing the Configuration

To test the USB configuration, plug in the USB device. In the system message buffer, dmesg(8), the drive should appear as something like:

```
umass0: USB Solid state disk, rev 1.10/1.00, addr 2
GEOM: create disk da0 dp=0xc2d74850
da0 at umass-sim0 bus 0 target 0 lun 0
da0: <Generic Traveling Disk 1.11> Removable Direct Access SCSI-2 device
da0: 1.000MB/s transfers
da0: 126MB (258048 512 byte sectors: 64H 32S/T 126C)
```

The brand, device node (`da0`), and other details will differ according to the device.

Since the USB device is seen as a SCSI one, `camcontrol` can be used to list the USB storage devices attached to the system:

```
# camcontrol devlist
<Generic Traveling Disk 1.11>      at scbus0 target 0 lun 0 (da0,pass0)
```

If the drive comes with a file system, it can be mounted. Refer to Section 19.3 for instructions on how to format and create partitions on the USB drive.

> **Warning:** Allowing untrusted users to mount arbitrary media, by enabling `vfs.usermount` as described below, should not be considered safe from a security point of view. Most file systems in FreeBSD were not built to safeguard against malicious devices.

To make the device mountable as a normal user, one solution is to make all users of the device a member of the `operator` group using pw(8). Next, ensure that the `operator` group is able to read and write the device by adding these lines to `/etc/devfs.rules`:

```
[localrules=5]
add path 'da*' mode 0660 group operator
```

> **Note:** If SCSI disks are installed in the system, change the second line as follows:
>
> ```
> add path 'da[3-9]*' mode 0660 group operator
> ```
>
> This will exclude the first three SCSI disks (`da0` to `da2`)from belonging to the `operator` group.

Next, enable the devfs.rules(5) ruleset in `/etc/rc.conf`:

```
devfs_system_ruleset="localrules"
```

Next, instruct the running kernel to allow regular users to mount file systems. The easiest way is to add the following line to `/etc/sysctl.conf`:

```
vfs.usermount=1
```

Since this only takes effect after the next reboot use sysctl(8) to set this variable now.

The final step is to create a directory where the file system is to be mounted. This directory needs to be owned by the user that is to mount the file system. One way to do that is for `root` to create a subdirectory owned by that user as `/mnt/`*`username`*. In the following example, replace *`username`* with the login name of the user and *`usergroup`* with the user's primary group:

```
# mkdir /mnt/username
# chown username:usergroup /mnt/username
```

Suppose a USB thumbdrive is plugged in, and a device `/dev/da0s1` appears. If the device is preformatted with a FAT file system, it can be mounted using:

```
% mount -t msdosfs -o -m=644,-M=755 /dev/da0s1 /mnt/username
```

Before the device can be unplugged, it *must* be unmounted first. After device removal, the system message buffer will show messages similar to the following:

```
umass0: at uhub0 port 1 (addr 2) disconnected
(da0:umass-sim0:0:0:0): lost device
(da0:umass-sim0:0:0:0): removing device entry
```

```
GEOM: destroy disk da0 dp=0xc2d74850
umass0: detached
```

19.4.3 Further Reading

Beside the Adding Disks and Mounting and Unmounting File Systems sections, reading various manual pages may be also useful: umass(4), camcontrol(8), and usbconfig(8) under FreeBSD 8.X or usbdevs(8) under earlier versions of FreeBSD.

19.5 Creating and Using CD Media

Contributed by Mike Meyer.

19.5.1 Introduction

CD media provide a number of features that differentiate them from conventional disks. Initially, they were not writable by the user. They are designed so that they can be read continuously without delays to move the head between tracks. They are also much easier to transport between systems.

CD media do have tracks, but this refers to a section of data to be read continuously and not a physical property of the disk. For example, to produce a CD on FreeBSD, prepare the data files that are going to make up the tracks on the CD, then write the tracks to the CD.

The ISO 9660 file system was designed to deal with these differences. To overcome the original file system limits, it provides an extension mechanism that allows properly written CDs to exceed those limits while still working with systems that do not support those extensions.

The `sysutils/cdrtools` port includes mkisofs(8), a program that can be used to produce a data file containing an ISO 9660 file system. It has options that support various extensions, and is described below.

Which tool to use to burn the CD depends on whether the CD burner is ATAPI or something else. ATAPI CD burners use `burncd` which is part of the base system. SCSI and USB CD burners should use `cdrecord` from the `sysutils/cdrtools` port. It is also possible to use `cdrecord` and other tools for SCSI drives on ATAPI hardware with the ATAPI/CAM module.

For CD burning software with a graphical user interface, consider **X-CD-Roast** or **K3b**. These tools are available as packages or from the `sysutils/xcdroast` and `sysutils/k3b` ports. **X-CD-Roast** and **K3b** require the ATAPI/CAM module with ATAPI hardware.

19.5.2 mkisofs

The `sysutils/cdrtools` port also installs mkisofs(8), which produces an ISO 9660 file system that is an image of a directory tree in the UNIX file system name space. The simplest usage is:

```
# mkisofs -o imagefile.iso /path/to/tree
```

This command creates an *imagefile.iso* containing an ISO 9660 file system that is a copy of the tree at */path/to/tree*. In the process, it maps the file names to names that fit the limitations of the standard ISO 9660 file system, and will exclude files that have names uncharacteristic of ISO file systems.

A number of options are available to overcome these restrictions. In particular, `-R` enables the Rock Ridge extensions common to UNIX systems, `-J` enables Joliet extensions used by Microsoft systems, and `-hfs` can be used to create HFS file systems used by Mac OS.

For CDs that are going to be used only on FreeBSD systems, `-U` can be used to disable all filename restrictions. When used with `-R`, it produces a file system image that is identical to the specified FreeBSD tree, though it may violate the ISO 9660 standard in a number of ways.

The last option of general use is `-b`. This is used to specify the location of the boot image for use in producing an "El Torito" bootable CD. This option takes an argument which is the path to a boot image from the top of the tree being written to the CD. By default, mkisofs(8) creates an ISO image in "floppy disk emulation" mode, and thus expects the boot image to be exactly 1200, 1440 or 2880 KB in size. Some boot loaders, like the one used by the FreeBSD distribution disks, do not use emulation mode. In this case, `-no-emul-boot` should be used. So, if `/tmp/myboot` holds a bootable FreeBSD system with the boot image in `/tmp/myboot/boot/cdboot`, this command would produce the image of an ISO 9660 file system as `/tmp/bootable.iso`:

```
# mkisofs -R -no-emul-boot -b boot/cdboot -o /tmp/bootable.iso /tmp/myboot
```

If `md` is configured in the kernel, the file system can be mounted as a memory disk with:

```
# mdconfig -a -t vnode -f /tmp/bootable.iso -u 0
# mount -t cd9660 /dev/md0 /mnt
```

One can then verify that `/mnt` and `/tmp/myboot` are identical.

There are many other options available for mkisofs(8) to fine-tune its behavior. Refer to mkisofs(8) for details.

19.5.3 burncd

For an ATAPI CD burner, `burncd` can be used to burn an ISO image onto a CD. `burncd` is part of the base system, installed as `/usr/sbin/burncd`. Usage is very simple, as it has few options:

```
# burncd -f cddevice data imagefile.iso fixate
```

This command will burn a copy of *imagefile.iso* on *cddevice*. The default device is `/dev/acd0`. See burncd(8) for options to set the write speed, eject the CD after burning, and write audio data.

19.5.4 cdrecord

For systems without an ATAPI CD burner, `cdrecord` can be used to burn CDs. `cdrecord` is not part of the base system and must be installed from either the `sysutils/cdrtools` package or port. Changes to the base system can cause binary versions of this program to fail, possibly resulting in a "coaster". It is recommended to either upgrade the port when the system is upgraded, or for users tracking -STABLE, to upgrade the port when a new version becomes available.

While `cdrecord` has many options, basic usage is simple. Burning an ISO 9660 image is done with:

```
# cdrecord dev=device imagefile.iso
```

The tricky part of using `cdrecord` is finding the `dev` to use. To find the proper setting, use `-scanbus` which might produce results like this:

```
# cdrecord -scanbus
Cdrecord-Clone 2.01 (i386-unknown-freebsd7.0) Copyright (C) 1995-2004 Jörg Schilling
Using libscg version 'schily-0.1'
scsibus0:
        0,0,0     0) 'SEAGATE ' 'ST39236LW       ' '0004' Disk
        0,1,0     1) 'SEAGATE ' 'ST39173W        ' '5958' Disk
        0,2,0     2) *
        0,3,0     3) 'iomega  ' 'jaz 1GB         ' 'J.86' Removable Disk
        0,4,0     4) 'NEC     ' 'CD-ROM DRIVE:466' '1.26' Removable CD-ROM
        0,5,0     5) *
        0,6,0     6) *
        0,7,0     7) *
scsibus1:
        1,0,0   100) *
        1,1,0   101) *
        1,2,0   102) *
        1,3,0   103) *
        1,4,0   104) *
        1,5,0   105) 'YAMAHA  ' 'CRW4260         ' '1.0q' Removable CD-ROM
        1,6,0   106) 'ARTEC   ' 'AM12S           ' '1.06' Scanner
        1,7,0   107) *
```

This lists the appropriate `dev` value for the devices on the list. Locate the CD burner, and use the three numbers separated by commas as the value for `dev`. In this case, the CRW device is 1,5,0, so the appropriate input is `dev=1,5,0`. Refer to cdrecord(1) for easier ways to specify this value and for information on writing audio tracks and controlling the write speed.

19.5.5 Duplicating Audio CDs

To duplicate an audio CD, extract the audio data from the CD to a series of files, then write these files to a blank CD. The process is slightly different for ATAPI and SCSI drives.

SCSI Drives

1. Use `cdda2wav` to extract the audio:

   ```
   % cdda2wav -vall -D2,0 -B -Owav
   ```

2. Use `cdrecord` to write the `.wav` files:

   ```
   % cdrecord -v dev=2,0 -dao -useinfo  *.wav
   ```

   Make sure that *`2,0`* is set appropriately, as described in Section 19.5.4.

ATAPI Drives

> **Note:** With the help of the ATAPI/CAM module, `cdda2wav` can also be used on ATAPI drives. This tool is usually a better choice for most of users, as it supports jitter correction and endianness, than the method proposed below.

1. The ATAPI CD driver makes each track available as `/dev/acd`*`d`*`t`*`nn`*, where *d* is the drive number, and *nn* is the track number written with two decimal digits, prefixed with zero as needed. So the first track on the first disk is `/dev/acd0t01`, the second is `/dev/acd0t02`, the third is `/dev/acd0t03`, and so on.

   Make sure the appropriate files exist in `/dev`. If the entries are missing, force the system to retaste the media:

   ```
   # dd if=/dev/acd0 of=/dev/null count=1
   ```

2. Extract each track using dd(1), making sure to specify a block size when extracting the files:

   ```
   # dd if=/dev/acd0t01 of=track1.cdr bs=2352
   # dd if=/dev/acd0t02 of=track2.cdr bs=2352
   ...
   ```

3. Burn the extracted files to disk using `burncd`. Specify that these are audio files, and that `burncd` should fixate the disk when finished:

   ```
   # burncd -f /dev/acd0 audio track1.cdr track2.cdr ... fixate
   ```

19.5.6 Duplicating Data CDs

It is possible to copy a data CD to an image file that is functionally equivalent to the image file created with mkisofs(8), and then use it to duplicate any data CD. The example given here assumes that the CD-ROM device is `acd0`. Substitute the correct CD-ROM device.

```
# dd if=/dev/acd0 of=file.iso bs=2048
```

Now that there is an image, it can be burned to CD as described above.

19.5.7 Using Data CDs

It is possible to mount and read the data on a standard data CD. By default, mount(8) assumes that a file system is of type `ufs`. Running this command:

```
# mount /dev/cd0 /mnt
```

will generate an error about `Incorrect super block`, and will fail to mount the CD. The CD does not use the `UFS` file system, so attempts to mount it as such will fail. Instead, tell mount(8) that the file system is of type `ISO9660` by specifying `-t cd9660` to mount(8). For example, to mount the CD-ROM device, `/dev/cd0`, under `/mnt`, use:

```
# mount -t cd9660 /dev/cd0 /mnt
```

Replace `/dev/cd0` with the device name for the CD device. Also, `-t cd9660` executes mount_cd9660(8), meaning the above command is equivalent to:

```
# mount_cd9660 /dev/cd0 /mnt
```

While data CD-ROMs from any vendor can be mounted this way, disks with certain ISO 9660 extensions might behave oddly. For example, Joliet disks store all filenames in two-byte Unicode characters. The FreeBSD kernel does not speak Unicode, but the FreeBSD CD9660 driver is able to convert Unicode characters on the fly. If some

non-English characters show up as question marks, specify the local charset with `-C`. For more information, refer to mount_cd9660(8).

> **Note:** In order to do this character conversion with the help of `-C`, the kernel requires the `cd9660_iconv.ko` module to be loaded. This can be done either by adding this line to `loader.conf`:
>
> ```
> cd9660_iconv_load="YES"
> ```
>
> and then rebooting the machine, or by directly loading the module with kldload(8).

Occasionally, `Device not configured` will be displayed when trying to mount a CD-ROM. This usually means that the CD-ROM drive thinks that there is no disk in the tray, or that the drive is not visible on the bus. It can take a couple of seconds for a CD-ROM drive to realize that a media is present, so be patient.

Sometimes, a SCSI CD-ROM may be missed because it did not have enough time to answer the bus reset. To resolve this,add the following option to the kernel configuration and rebuild the kernel.

```
options SCSI_DELAY=15000
```

This tells the SCSI bus to pause 15 seconds during boot, to give the CD-ROM drive every possible chance to answer the bus reset.

19.5.8 Burning Raw Data CDs

It is possible to burn a file directly to CD, without creating an ISO 9660 file system. Some people do this for backup purposes. This command runs more quickly than burning a standard CD:

```
# burncd -f /dev/acd1 -s 12 data archive.tar.gz fixate
```

In order to retrieve the data burned to such a CD, the data must be read from the raw device node:

```
# tar xzvf /dev/acd1
```

This type of disk can not be mounted as a normal CD-ROM and the data cannot be read under any operating system except FreeBSD. In order to mount the CD, or to share the data with another operating system, mkisofs(8) must be used as described above.

19.5.9 Using the ATAPI/CAM Driver

Contributed by Marc Fonvieille.

This driver allows ATAPI devices, such as CD/DVD drives, to be accessed through the SCSI subsystem, and so allows the use of applications like `sysutils/cdrdao` or cdrecord(1).

To use this driver, add the following line to `/boot/loader.conf`:

```
atapicam_load="YES"
```

then, reboot the system.

Note: Users who prefer to statically compile atapicam(4) support into the kernel, should add this line to the kernel configuration file:

```
device atapicam
```

Ensure the following lines are still in the kernel configuration file:

```
device ata
device scbus
device cd
device pass
```

Then rebuild, install the new kernel, and reboot the machine.

During the boot process, the burner should show up, like so:

```
acd0: CD-RW <MATSHITA CD-RW/DVD-ROM UJDA740> at ata1-master PIO4
cd0 at ata1 bus 0 target 0 lun 0
cd0: <MATSHITA CDRW/DVD UJDA740 1.00> Removable CD-ROM SCSI-0 device
cd0: 16.000MB/s transfers
cd0: Attempt to query device size failed: NOT READY, Medium not present - tray closed
```

The drive can now be accessed via the `/dev/cd0` device name. For example, to mount a CD-ROM on `/mnt`, type the following:

```
# mount -t cd9660 /dev/cd0 /mnt
```

As `root`, run the following command to get the SCSI address of the burner:

```
# camcontrol devlist
<MATSHITA CDRW/DVD UJDA740 1.00>   at scbus1 target 0 lun 0 (pass0,cd0)
```

In this example, `1,0,0` is the SCSI address to use with cdrecord(1) and other SCSI applications.

For more information about ATAPI/CAM and SCSI system, refer to atapicam(4) and cam(4).

19.6 Creating and Using DVD Media

Contributed by Marc Fonvieille. With inputs from Andy Polyakov.

19.6.1 Introduction

Compared to the CD, the DVD is the next generation of optical media storage technology. The DVD can hold more data than any CD and is the standard for video publishing.

Five physical recordable formats can be defined for a recordable DVD:

- DVD-R: This was the first DVD recordable format available. The DVD-R standard is defined by the DVD Forum (http://www.dvdforum.com/forum.shtml). This format is write once.
- DVD-RW: This is the rewritable version of the DVD-R standard. A DVD-RW can be rewritten about 1000 times.

- DVD-RAM: This is a rewritable format which can be seen as a removable hard drive. However, this media is not compatible with most DVD-ROM drives and DVD-Video players as only a few DVD writers support the DVD-RAM format. Refer to Section 19.6.9 for more information on DVD-RAM use.
- DVD+RW: This is a rewritable format defined by the DVD+RW Alliance (http://www.dvdrw.com/). A DVD+RW can be rewritten about 1000 times.
- DVD+R: This format is the write once variation of the DVD+RW format.

A single layer recordable DVD can hold up to 4,700,000,000 bytes which is actually 4.38 GB or 4485 MB as 1 kilobyte is 1024 bytes.

> **Note:** A distinction must be made between the physical media and the application. For example, a DVD-Video is a specific file layout that can be written on any recordable DVD physical media such as DVD-R, DVD+R, or DVD-RW. Before choosing the type of media, ensure that both the burner and the DVD-Video player are compatible with the media under consideration.

19.6.2 Configuration

To perform DVD recording, use growisofs(1). This command is part of the `sysutils/dvd+rw-tools` utilities which support all DVD media types.

These tools use the SCSI subsystem to access the devices, therefore ATAPI/CAM support must be loaded or statically compiled into the kernel. This support is not needed if the burner uses the USB interface. Refer to Section 19.4 for more details on USB device configuration.

DMA access must also be enabled for ATAPI devices, by adding the following line to `/boot/loader.conf`:

```
hw.ata.atapi_dma="1"
```

Before attempting to use **dvd+rw-tools**, consult the Hardware Compatibility Notes (http://fy.chalmers.se/~appro/linux/DVD+RW/hcn.html).

> **Note:** For a graphical user interface, consider using `sysutils/k3b` which provides a user friendly interface to growisofs(1) and many other burning tools.

19.6.3 Burning Data DVDs

Since growisofs(1) is a front-end to mkisofs, it will invoke mkisofs(8) to create the file system layout and perform the write on the DVD. This means that an image of the data does not need to be created before the burning process.

To burn to a DVD+R or a DVD-R the data in `/path/to/data`, use the following command:

```
# growisofs -dvd-compat -Z /dev/cd0 -J -R /path/to/data
```

In this example, `-J -R` is passed to mkisofs(8) to create an ISO 9660 file system with Joliet and Rock Ridge extensions. Refer to mkisofs(8) for more details.

For the initial session recording, `-Z` is used for both single and multiple sessions. Replace */dev/cd0*, with the name of the DVD device. Using `-dvd-compat` indicates that the disk will be closed and that the recording will be unappendable. This should also provide better media compatibility with DVD-ROM drives.

To burn a pre-mastered image, such as *imagefile.iso*, use:

```
# growisofs -dvd-compat -Z /dev/cd0=imagefile.iso
```

The write speed should be detected and automatically set according to the media and the drive being used. To force the write speed, use `-speed=`. Refer to growisofs(1) for example usage.

> **Note:** In order to support working files larger than 4.38GB, an UDF/ISO-9660 hybrid filesystem must be created by passing `-udf -iso-level 3` to mkisofs(8) and all related programs, such as growisofs(1). This is required only when creating an ISO image file or when writing files directly to a disk. Since a disk created this way must be mounted as an UDF filesystem with mount_udf(8), it will be usable only on an UDF aware operating system. Otherwise it will look as if it contains corrupted files.
>
> To create this type of ISO file:
>
> ```
> % mkisofs -R -J -udf -iso-level 3 -o imagefile.iso /path/to/data
> ```
>
> To burn files directly to a disk:
>
> ```
> # growisofs -dvd-compat -udf -iso-level 3 -Z /dev/cd0 -J -R /path/to/data
> ```
>
> When an ISO image already contains large files, no additional options are required for growisofs(1) to burn that image on a disk.
>
> Be sure to use an up-to-date version of `sysutils/cdrtools`, which contains mkisofs(8), as an older version may not contain large files support. If the latest version does not work, install `sysutils/cdrtools-devel` and read its mkisofs(8).

19.6.4 Burning a DVD-Video

A DVD-Video is a specific file layout based on the ISO 9660 and micro-UDF (M-UDF) specifications. Since DVD-Video presents a specific data structure hierarchy, a particular program such as `multimedia/dvdauthor` is needed to author the DVD.

If an image of the DVD-Video file system already exists, it can be burned in the same way as any other image. If `dvdauthor` was used to make the DVD and the result is in `/path/to/video`, the following command should be used to burn the DVD-Video:

```
# growisofs -Z /dev/cd0 -dvd-video /path/to/video
```

`-dvd-video` is passed to mkisofs(8) to instruct it to create a DVD-Video file system layout. This option implies the `-dvd-compat` growisofs(1) option.

19.6.5 Using a DVD+RW

Unlike CD-RW, a virgin DVD+RW needs to be formatted before first use. It is *recommended* to let growisofs(1) take care of this automatically whenever appropriate. However, it is possible to use `dvd+rw-format` to format the DVD+RW:

```
# dvd+rw-format /dev/cd0
```

Only perform this operation once and keep in mind that only virgin DVD+RW medias need to be formatted. Once formatted, the DVD+RW can be burned as usual.

To burn a totally new file system and not just append some data onto a DVD+RW, the media does not need to be blanked first. Instead, write over the previous recording like this:

```
# growisofs -Z /dev/cd0 -J -R /path/to/newdata
```

The DVD+RW format supports appending data to a previous recording. This operation consists of merging a new session to the existing one as it is not considered to be multi-session writing. growisofs(1) will *grow* the ISO 9660 file system present on the media.

For example, to append data to a DVD+RW, use the following:

```
# growisofs -M /dev/cd0 -J -R /path/to/nextdata
```

The same mkisofs(8) options used to burn the initial session should be used during next writes.

> **Note:** Use `-dvd-compat` for better media compatibility with DVD-ROM drives. When using DVD+RW, this option will not prevent the addition of data.

To blank the media, use:

```
# growisofs -Z /dev/cd0=/dev/zero
```

19.6.6 Using a DVD-RW

A DVD-RW accepts two disc formats: incremental sequential and restricted overwrite. By default, DVD-RW discs are in sequential format.

A virgin DVD-RW can be directly written without being formatted. However, a non-virgin DVD-RW in sequential format needs to be blanked before writing a new initial session.

To blank a DVD-RW in sequential mode:

```
# dvd+rw-format -blank=full /dev/cd0
```

> **Note:** A full blanking using `-blank=full` will take about one hour on a 1x media. A fast blanking can be performed using `-blank`, if the DVD-RW will be recorded in Disk-At-Once (DAO) mode. To burn the DVD-RW in DAO mode, use the command:
>
> ```
> # growisofs -use-the-force-luke=dao -Z /dev/cd0=imagefile.iso
> ```

Since growisofs(1) automatically attempts to detect fast blanked media and engage DAO write, `-use-the-force-luke=dao` should not be required.

One should instead use restricted overwrite mode with any DVD-RW as this format is more flexible than the default of incremental sequential.

To write data on a sequential DVD-RW, use the same instructions as for the other DVD formats:

```
# growisofs -Z /dev/cd0 -J -R /path/to/data
```

To append some data to a previous recording, use `-M` with growisofs(1). However, if data is appended on a DVD-RW in incremental sequential mode, a new session will be created on the disc and the result will be a multi-session disc.

A DVD-RW in restricted overwrite format does not need to be blanked before a new initial session. Instead, overwrite the disc with `-Z`. It is also possible to grow an existing ISO 9660 file system written on the disc with `-M`. The result will be a one-session DVD.

To put a DVD-RW in restricted overwrite format, the following command must be used:

```
# dvd+rw-format /dev/cd0
```

To change back to sequential format, use:

```
# dvd+rw-format -blank=full /dev/cd0
```

19.6.7 Multi-Session

Few DVD-ROM drives support multi-session DVDs and most of the time only read the first session. DVD+R, DVD-R and DVD-RW in sequential format can accept multiple sessions. The notion of multiple sessions does not exist for the DVD+RW and the DVD-RW restricted overwrite formats.

Using the following command after an initial non-closed session on a DVD+R, DVD-R, or DVD-RW in sequential format, will add a new session to the disc:

```
# growisofs -M /dev/cd0 -J -R /path/to/nextdata
```

Using this command with a DVD+RW or a DVD-RW in restricted overwrite mode will append data while merging the new session to the existing one. The result will be a single-session disc. Use this method to add data after an initial write on these types of media.

Note: Since some space on the media is used between each session to mark the end and start of sessions, one should add sessions with a large amount of data to optimize media space. The number of sessions is limited to 154 for a DVD+R, about 2000 for a DVD-R, and 127 for a DVD+R Double Layer.

19.6.8 For More Information

To obtain more information about a DVD, use `dvd+rw-mediainfo /dev/cd0` while the disc in the specified drive.

More information about **dvd+rw-tools** can be found in growisofs(1), on the dvd+rw-tools web site (http://fy.chalmers.se/~appro/linux/DVD+RW/), and in the cdwrite mailing list (http://lists.debian.org/cdwrite/) archives.

> **Note:** When creating a problem report related to the use of **dvd+rw-tools**, always include the output of `dvd+rw-mediainfo`.

19.6.9 Using a DVD-RAM

19.6.9.1 Configuration

DVD-RAM writers can use either a SCSI or ATAPI interface. For ATAPI devices, DMA access has to be enabled by adding the following line to `/boot/loader.conf`:

```
hw.ata.atapi_dma="1"
```

19.6.9.2 Preparing the Media

A DVD-RAM can be seen as a removable hard drive. Like any other hard drive, the DVD-RAM must be formatted before it can be used. In this example, the whole disk space will be formatted with a standard UFS2 file system:

```
# dd if=/dev/zero of=/dev/acd0 bs=2k count=1
# bsdlabel -Bw acd0
# newfs /dev/acd0
```

The DVD device, `acd0`, must be changed according to the configuration.

19.6.9.3 Using the Media

Once the DVD-RAM has been formatted, it can be mounted as a normal hard drive:

```
# mount /dev/acd0 /mnt
```

Once mounted, the DVD-RAM will be both readable and writeable.

19.7 Creating and Using Floppy Disks

Original work by Julio Merino. Rewritten by Martin Karlsson.

Storing data on floppy disks is sometimes useful, for example when one does not have any other removable storage media or when one needs to transfer small amounts of data to another computer.

This section explains how to use floppy disks in FreeBSD. It covers formatting and usage of 3.5inch DOS floppies, but the concepts are similar for other floppy disk formats.

19.7.1 Formatting Floppies

19.7.1.1 The Device

Floppy disks are accessed through entries in `/dev`, just like other devices. To access the raw floppy disk, simply use `/dev/fdN`.

19.7.1.2 Formatting

A floppy disk needs to be low-level formatted before it can be used. This is usually done by the vendor, but formatting is a good way to check media integrity. Although it is possible to force other disk sizes, 1440kB is what most floppy disks are designed for.

To low-level format the floppy disk, use fdformat(1). This utility expects the device name as an argument.

Make note of any error messages, as these can help determine if the disk is good or bad.

19.7.1.2.1 Formatting Floppy Disks

To format the floppy, insert a new 3.5inch floppy disk into the first floppy drive and issue:

```
# /usr/sbin/fdformat -f 1440 /dev/fd0
```

19.7.2 The Disk Label

After low-level formatting the disk, a disk label needs to placed on it. This disk label will be destroyed later, but it is needed by the system to determine the size of the disk and its geometry.

The new disk label will take over the whole disk and will contain all the proper information about the geometry of the floppy. The geometry values for the disk label are listed in `/etc/disktab`.

To write the disk label, use bsdlabel(8):

```
# /sbin/bsdlabel -B -w /dev/fd0 fd1440
```

19.7.3 The File System

The floppy is now ready to be high-level formatted. This will place a new file system on it so that FreeBSD can read and write to the disk. Since creating the new file system destroys the disk label, the disk label needs to be recreated whenever the disk is reformatted.

The floppy's file system can be either UFS or FAT. FAT is generally a better choice for floppies.

To put a new file system on the floppy, issue:

```
# /sbin/newfs_msdos /dev/fd0
```

The disk is now ready for use.

19.7.4 Using the Floppy

To use the floppy, mount it with mount_msdosfs(8). One can also use `emulators/mtools` from the Ports Collection.

19.8 Creating and Using Data Tapes

Tape technology has continued to evolve but is less likely to be used in a modern system. Modern backup systems tend to use off site combined with local removable disk drive technologies. Still, FreeBSD will support any tape drive that uses SCSI, such as LTO and older devices such as DAT. There is limited support for SATA and USB tape drives.

19.8.1 Serial Access with sa(4)

FreeBSD uses the sa(4) driver, providing `/dev/sa0`, `/dev/nsa0`, and `/dev/esa0`. In normal use, only `/dev/sa0` is needed. `/dev/nsa0` is the same physical drive as `/dev/sa0` but does not rewind the tape after writing a file. This allows writing more than one file to a tape. Using `/dev/esa0` ejects the tape after the device is closed, if applicable.

19.8.2 Controlling the Tape Drive with mt(1)

mt(1) is the FreeBSD utility for controlling other operations of the tape drive, such as seeking through files on a tape or writing tape control marks to the tape.

For example, the first three files on a tape can be preserved by skipping past them before writing a new file:

```
# mt -f /dev/nsa0 fsf 3
```

19.8.3 Using tar(1) to Read and Write Tape Backups

An example of writing a single file to tape using tar(1):

```
# tar cvf /dev/sa0 file
```

Recovering files from a tar(1) archive on tape into the current directory:

```
# tar xvf /dev/sa0
```

19.8.4 Using dump(8) and restore(8) to Create and Restore Backups

A simple backup of `/usr` with dump(8):

```
# dump -0aL -b64 -f /dev/nsa0 /usr
```

Interactively restoring files from a dump(8) file on tape into the current directory:

```
# restore -i -f /dev/nsa0
```

19.8.5 Other Tape Software

Higher-level programs are available to simplify tape backup. The most popular are **Amanda** and **Bacula**. These programs aim to make backups easier and more convenient, or to automate complex backups of multiple machines. The Ports Collection contains both these and other tape utility applications.

19.9 Backup Strategies

Original work by Lowell Gilbert.

The first requirement in devising a backup plan is to make sure that all of the following problems are covered:

- Disk failure.
- Accidental file deletion.
- Random file corruption.
- Complete machine destruction, say by fire, including destruction of any on-site backups.

Some systems will be best served by having each of these problems covered by a completely different technique. Except for strictly personal systems with low-value data, it is unlikely that one technique will cover all of them.

Some possible techniques include:

- Archives of the whole system, backed up onto permanent, off-site media. This provides protection against all of the problems listed above, but is slow and inconvenient to restore from. Copies of the backups can be stored on site or online, but there will still be inconveniences in restoring files, especially for non-privileged users.
- Filesystem snapshots, which are really only helpful in the accidental file deletion scenario, but can be *very* helpful in that case, as well as quick and easy to deal with.
- Copies of whole file systems or disks which can be created with a periodic `net/rsync` of the whole machine. This is generally most useful in networks with unique requirements. For general protection against disk failure, this is usually inferior to RAID. For restoring accidentally deleted files, it can be comparable to UFS snapshots.
- RAID, which minimizes or avoids downtime when a disk fails at the expense of having to deal with disk failures more often, because there are more disks, albeit at a much lower urgency.
- Checking fingerprints of files using mtree(8). Although this is not a backup, this technique indicates when one needs to resort to backups. This is particularly important for offline backups, and should be checked periodically.

It is quite easy to come up with more techniques, many of them variations on the ones listed above. Specialized requirements usually lead to specialized techniques. For example, backing up a live database usually requires a method particular to the database software as an intermediate step. The important thing is to know which dangers should be protected against, and how each will be handled.

19.10 Backup Basics

The major backup programs built into FreeBSD are dump(8), tar(1), cpio(1), and pax(1).

19.10.1 Dump and Restore

The traditional UNIX backup programs are `dump` and `restore`. They operate on the drive as a collection of disk blocks, below the abstractions of files, links and directories that are created by the file systems. Unlike other backup software, `dump` backs up an entire file system on a device. It is unable to backup only part of a file system or a directory tree that spans more than one file system. `dump` does not write files and directories, but rather writes the raw data blocks that comprise files and directories. When used to extract data, `restore` stores temporary files in `/tmp/` by default. When using a recovery disk with a small `/tmp`, set `TMPDIR` to a directory with more free space in order for the restore to succeed.

> **Note:** If `dump` is used on the root directory, it will not back up `/home`, `/usr` or many other directories since these are typically mount points for other file systems or symbolic links into those file systems.

`dump` has quirks that remain from its early days in Version 6 of AT&T UNIX,circa 1975. The default parameters are suitable for 9-track tapes (6250 bpi), not the high-density media available today (up to 62,182 ftpi). These defaults must be overridden on the command line to utilize the capacity of current tape drives.

It is also possible to backup data across the network to a tape drive attached to another computer with `rdump` and `rrestore`. Both programs rely upon rcmd(3) and ruserok(3) to access the remote tape drive. Therefore, the user performing the backup must be listed in `.rhosts` on the remote computer. The arguments to `rdump` and `rrestore` must be suitable to use on the remote computer. For example, to `rdump` from a FreeBSD computer to an Exabyte tape drive connected to a host called `komodo`, use:

```
# /sbin/rdump 0dsbfu 54000 13000 126 komodo:/dev/nsa8 /dev/da0a 2>&1
```

There are security implications to allowing `.rhosts` authentication, so use with caution.

It is also possible to use `dump` and `restore` in a more secure fashion over `ssh`.

Example 19-1. Using dump over ssh

```
# /sbin/dump -0uan -f - /usr | gzip -2 | ssh -c blowfish \
          targetuser@targetmachine.example.com dd of=/mybigfiles/dump-usr-l0.gz
```

Or, use the built-in `RSH`:

Example 19-2. Using dump over ssh with `RSH` Set

```
# env RSH=/usr/bin/ssh /sbin/dump -0uan -f targetuser@targetmachine.example.com:/dev/sa0 /usr
```

19.10.2 `tar`

tar(1) also dates back to Version 6 of AT&T UNIX, circa 1975. `tar` operates in cooperation with the file system and writes files and directories to tape. `tar` does not support the full range of options that are available from cpio(1), but it does not require the unusual command pipeline that `cpio` uses.

To `tar` to an Exabyte tape drive connected to a host called `komodo`:

```
# tar cf - . | rsh komodo dd of=tape-device obs=20b
```

When backing up over an insecure network, instead use `ssh`.

19.10.3 `cpio`

cpio(1) is the original UNIX file interchange tape program for magnetic media. `cpio` includes options to perform byte-swapping, write a number of different archive formats, and pipe the data to other programs. This last feature makes `cpio` an excellent choice for installation media. `cpio` does not know how to walk the directory tree and a list of files must be provided through `stdin`.

Since `cpio` does not support backups across the network, use a pipeline and `ssh` to send the data to a remote tape drive.

```
# for f in directory_list; do
find $f >> backup.list
done
# cpio -v -o --format=newc < backup.list | ssh user@host "cat > backup_device"
```

Where *`directory_list`* is the list of directories to back up, *`user@host`* is the user/hostname combination that will be performing the backups, and *`backup_device`* is where the backups should be written to, such as `/dev/nsa0`).

19.10.4 `pax`

pax(1) is the IEEE/POSIX answer to `tar` and `cpio`. Over the years the various versions of `tar` and `cpio` have become slightly incompatible. So rather than fight it out to fully standardize them, POSIX created a new archive utility. `pax` attempts to read and write many of the various `cpio` and `tar` formats, plus new formats of its own. Its command set more resembles `cpio` than `tar`.

19.10.5 Amanda

Amanda (Advanced Maryland Network Disk Archiver) is a client/server backup system, rather than a single program. An **Amanda** server will backup to a single tape drive any number of computers that have **Amanda** clients and a network connection to the **Amanda** server. A common problem at sites with a number of large disks is that the length of time required to backup to data directly to tape exceeds the amount of time available for the task. **Amanda** solves this problem by using a “holding disk” to backup several file systems at the same time. **Amanda** creates “archive sets”: a group of tapes used over a period of time to create full backups of all the file systems listed in **Amanda**’s configuration file. The “archive set” also contains nightly incremental, or differential, backups of all the file systems. Restoring a damaged file system requires the most recent full backup and the incremental backups.

The configuration file provides fine grained control of backups and the network traffic that **Amanda** generates. **Amanda** will use any of the above backup programs to write the data to tape. **Amanda** is not installed by but is available as either a port or package.

19.10.6 Do Nothing

“Do nothing” is not a computer program, but it is the most widely used backup strategy. There are no initial costs. There is no backup schedule to follow. Just say no. If something happens to your data, grin and bear it!

If your time and data is worth little to nothing, then “Do nothing” is the most suitable backup program for the computer. But beware, FreeBSD is a useful tool and over time it can be used to create a valuable collection of files.

“Do nothing” is the correct backup method for `/usr/obj` and other directory trees that can be exactly recreated by the computer. An example is the files that comprise the HTML or PostScript version of this Handbook. These document formats have been created from XML input files. Creating backups of the HTML or PostScript files is not necessary if the XML files are backed up regularly.

19.10.7 Which Backup Program Is Best?

dump(8) *Period.* Elizabeth D. Zwicky torture tested all the backup programs discussed here. The clear choice for preserving all your data and all the peculiarities of UNIX file systems is `dump`. Elizabeth created file systems containing a large variety of unusual conditions (and some not so unusual ones) and tested each program by doing a backup and restore of those file systems. The peculiarities included: files with holes, files with holes and a block of nulls, files with funny characters in their names, unreadable and unwritable files, devices, files that change size during the backup, files that are created/deleted during the backup and more. She presented the results at LISA V in Oct. 1991. See torture-testing Backup and Archive Programs (http://www.coredumps.de/doc/dump/zwicky/testdump.doc.html).

19.10.8 Emergency Restore Procedure

19.10.8.1 Before the Disaster

There are four steps which should be performed in preparation for any disaster that may occur.

First, print the bsdlabel of each disk using a command such as `bsdlabel da0 | lpr`. Also print a copy of `/etc/fstab` and all boot messages.

Second, burn a “livefs” CD. This CD contains support for booting into a FreeBSD “livefs” rescue mode, allowing the user to perform many tasks like running dump(8), restore(8), fdisk(8), bsdlabel(8), newfs(8), mount(8), and more. The livefs CD image for FreeBSD/i386 8.4-RELEASE is available from ftp://ftp.FreeBSD.org/pub/FreeBSD/releases/i386/ISO-IMAGES/8.4/FreeBSD-8.4-RELEASE-i386-livefs.iso.

> **Note:** Livefs CD images are not available for FreeBSD 9.1-RELEASE and later. In addition to the CD-ROM installation images, flash drive installation images may be used to recover a system. The “memstick” image for FreeBSD/i386 9.1-RELEASE is available from ftp://ftp.FreeBSD.org/pub/FreeBSD/releases/i386/i386/ISO-IMAGES/9.1/FreeBSD-9.1-RELEASE-i386-memstick.img.

Third, create backup tapes regularly. Any changes that made after the last backup may be irretrievably lost. Write-protect the backup media.

Fourth, test the “livefs” CD and the backups. Make notes of the procedure. Store these notes with the CD, the printouts, and the backups. These notes may prevent the inadvertent destruction of the backups while under the stress of performing an emergency recovery.

For an added measure of security, store an extra “livefs” CD and the latest backup at a remote location, where a remote location is *not* the basement of the same building. A remote location should be physically separated from the computers and disk drives by a significant distance.

19.10.8.2 After the Disaster

First, determine if the hardware survived. Thanks to regular, off-site backups, there is no need to worry about the software.

If the hardware has been damaged, the parts should be replaced before attempting to use the computer.

If the hardware is okay, insert the “livefs” CD and boot the computer. The original install menu will be displayed on the screen. Select the correct country, then choose Fixit -- Repair mode with CD-ROM/DVD/floppy or start a shell. then select CD-ROM/DVD -- Use the live filesystem CD-ROM/DVD. `restore` and the other needed programs are located in `/mnt2/rescue`.

Recover each file system separately.

Try to `mount` the root partition of the first disk using `mount /dev/da0a /mnt`. If the bsdlabel was damaged, use `bsdlabel` to re-partition and label the disk to match the label that was printed and saved. Use `newfs` to re-create the file systems. Re-mount the root partition of the disk read-write using `mount -u -o rw /mnt`. Use the backups to recover the data for this file system. Unmount the file system with `umount /mnt`. Repeat for each file system that was damaged.

Once the system is running, backup the data onto new media as whatever caused the crash or data loss may strike again. Another hour spent now may save further distress later.

19.11 Network, Memory, and File-Backed File Systems

Reorganized and enhanced by Marc Fonvieille.

In addition to physical disks such as floppies, CDs, and hard drives, FreeBSD also supports *virtual disks*.

These include network file systems such as the Network File System and Coda, memory-based file systems, and file-backed file systems.

According to the FreeBSD version, the tools used for the creation and use of file-backed and memory-based file systems differ.

Note: Use devfs(5) to allocate device nodes transparently for the user.

19.11.1 File-Backed File System

mdconfig(8) is used to configure and enable memory disks, md(4), under FreeBSD. To use mdconfig(8), md(4) must be first loaded. When using a custom kernel configuration file, ensure it includes this line:

```
device md
```

mdconfig(8) supports several types of memory backed virtual disks: memory disks allocated with malloc(9) and memory disks using a file or swap space as backing. One possible use is the mounting of CD images.

To mount an existing file system image:

Example 19-3. Using `mdconfig` to Mount an Existing File System Image

```
# mdconfig -a -t vnode -f diskimage -u 0
# mount /dev/md0 /mnt
```

To create a new file system image with mdconfig(8):

Example 19-4. Creating a New File-Backed Disk with `mdconfig`

```
# dd if=/dev/zero of=newimage bs=1k count=5k
5120+0 records in
5120+0 records out
# mdconfig -a -t vnode -f newimage -u 0
# bsdlabel -w md0 auto
# newfs md0a
/dev/md0a: 5.0MB (10224 sectors) block size 16384, fragment size 2048
        using 4 cylinder groups of 1.25MB, 80 blks, 192 inodes.
super-block backups (for fsck -b #) at:
 160, 2720, 5280, 7840
# mount /dev/md0a /mnt
# df /mnt
Filesystem 1K-blocks Used Avail Capacity  Mounted on
/dev/md0a       4710    4  4330     0%    /mnt
```

If unit number is not specified with `-u`, mdconfig(8) uses the md(4) automatic allocation to select an unused device. The name of the allocated unit will be output to stdout, such as `md4`. Refer to mdconfig(8) for more details about.

While mdconfig(8) is useful, it takes several command lines to create a file-backed file system. FreeBSD also comes with mdmfs(8) which automatically configures a md(4) disk using mdconfig(8), puts a UFS file system on it using newfs(8), and mounts it using mount(8). For example, to create and mount the same file system image as above, type the following:

Example 19-5. Configure and Mount a File-Backed Disk with `mdmfs`

```
# dd if=/dev/zero of=newimage bs=1k count=5k
5120+0 records in
5120+0 records out
# mdmfs -F newimage -s 5m md0 /mnt
# df /mnt
Filesystem 1K-blocks Used Avail Capacity  Mounted on
/dev/md0        4718    4  4338     0%    /mnt
```

When `md` is used without a unit number, mdmfs(8) uses the md(4) auto-unit feature to automatically select an unused device. For more details about mdmfs(8), refer to its manual page.

19.11.2 Memory-Based File System

For a memory-based file system, “swap backing” should normally be used. This does not mean that the memory disk will be swapped out to disk by default, but rather that the memory disk will be allocated from a memory pool which can be swapped out to disk if needed. It is also possible to create memory-based disks which are malloc(9) backed, but using large malloc backed memory disks can result in a system panic if the kernel runs out of memory.

Example 19-6. Creating a New Memory-Based Disk with `mdconfig`

```
# mdconfig -a -t swap -s 5m -u 1
# newfs -U md1
/dev/md1: 5.0MB (10240 sectors) block size 16384, fragment size 2048
        using 4 cylinder groups of 1.27MB, 81 blks, 192 inodes.
        with soft updates
super-block backups (for fsck -b #) at:
 160, 2752, 5344, 7936
# mount /dev/md1 /mnt
# df /mnt
Filesystem 1K-blocks Used Avail Capacity  Mounted on
/dev/md1        4718    4  4338     0%    /mnt
```

Example 19-7. Creating a New Memory-Based Disk with `mdmfs`

```
# mdmfs -s 5m md2 /mnt
# df /mnt
Filesystem 1K-blocks Used Avail Capacity  Mounted on
/dev/md2        4846    2  4458     0%    /mnt
```

19.11.3 Detaching a Memory Disk from the System

When a memory-based or file-based file system is no longer in use, its resources should be released back to the system. First, unmount the file system, then use mdconfig(8) to detach the disk from the system and release the resources.

For example, to detach and free all resources used by `/dev/md4`:

```
# mdconfig -d -u 4
```

It is possible to list information about configured md(4) devices by running `mdconfig -l`.

19.12 File System Snapshots

Contributed by Tom Rhodes.

FreeBSD offers a feature in conjunction with Soft Updates: file system snapshots.

UFS snapshots allow a user to create images of specified file systems, and treat them as a file. Snapshot files must be created in the file system that the action is performed on, and a user may create no more than 20 snapshots per file system. Active snapshots are recorded in the superblock so they are persistent across unmount and remount operations along with system reboots. When a snapshot is no longer required, it can be removed using rm(1). While snapshots may be removed in any order, all the used space may not be acquired because another snapshot will possibly claim some of the released blocks.

The un-alterable `snapshot` file flag is set by mksnap_ffs(8) after initial creation of a snapshot file. unlink(1) makes an exception for snapshot files since it allows them to be removed.

Snapshots are created using mount(8). To place a snapshot of `/var` in the file `/var/snapshot/snap`, use the following command:

```
# mount -u -o snapshot /var/snapshot/snap /var
```

Alternatively, use mksnap_ffs(8) to create the snapshot:

```
# mksnap_ffs /var /var/snapshot/snap
```

One can find snapshot files on a file system, such as `/var`, using find(1):

```
# find /var -flags snapshot
```

Once a snapshot has been created, it has several uses:

- Some administrators will use a snapshot file for backup purposes, because the snapshot can be transferred to CDs or tape.
- The file system integrity checker, fsck(8), may be run on the snapshot. Assuming that the file system was clean when it was mounted, this should always provide a clean and unchanging result.
- Running dump(8) on the snapshot will produce a dump file that is consistent with the file system and the timestamp of the snapshot. dump(8) can also take a snapshot, create a dump image, and then remove the snapshot in one command by using `-L`.
- The snapshot can be mounted as a frozen image of the file system. To mount(8) the snapshot `/var/snapshot/snap` run:

  ```
  # mdconfig -a -t vnode -f /var/snapshot/snap -u 4
  # mount -r /dev/md4 /mnt
  ```

The frozen `/var` is now available through `/mnt`. Everything will initially be in the same state it was during the snapshot creation time. The only exception is that any earlier snapshots will appear as zero length files. To unmount the snapshot, use:

```
# umount /mnt
# mdconfig -d -u 4
```

For more information about `softupdates` and file system snapshots, including technical papers, visit Marshall Kirk McKusick's website at http://www.mckusick.com/.

19.13 File System Quotas

Quotas are an optional feature of the operating system that can be used to limit the amount of disk space or the number of files a user or members of a group may allocate on a per-file system basis. This is used most often on timesharing systems where it is desirable to limit the amount of resources any one user or group of users may allocate. This prevents one user or group of users from consuming all of the available disk space.

19.13.1 Configuring the System to Enable Disk Quotas

Before using disk quotas, quota support must be added to the kernel by adding the following line to the kernel configuration file:

```
options QUOTA
```

The `GENERIC` kernel does not have this enabled by default, so a custom kernel must be compiled in order to use disk quotas. Refer to Chapter 9 for more information on kernel configuration.

Next, enable disk quotas in `/etc/rc.conf`:

```
quota_enable="YES"
```

For finer control over quota startup, an additional configuration variable is available. Normally on bootup, the quota integrity of each file system is checked by quotacheck(8). This program insures that the data in the quota database properly reflects the data on the file system. This is a time consuming process that will significantly affect the time the system takes to boot. To skip this step, add this variable to `/etc/rc.conf`:

```
check_quotas="NO"
```

Finally, edit `/etc/fstab` to enable disk quotas on a per-file system basis. This is when user or group quotas can be enabled on the file systems.

To enable per-user quotas on a file system, add `userquota` to the options field in the `/etc/fstab` entry for the file system to enable quotas on. For example:

```
/dev/da1s2g   /home    ufs rw,userquota 1 2
```

To enable group quotas, instead use `groupquota`. To enable both user and group quotas, change the entry as follows:

```
/dev/da1s2g    /home    ufs rw,userquota,groupquota 1 2
```

By default, the quota files are stored in the root directory of the file system as `quota.user` and `quota.group`. Refer to fstab(5) for more information. Even though an alternate location for the quota files can be specified, this is not recommended because the various quota utilities do not seem to handle this properly.

Once the configuration is complete, reboot the system with the new kernel. `/etc/rc` will automatically run the appropriate commands to create the initial quota files for all of the quotas enabled in `/etc/fstab`. There is no need to manually create any zero length quota files.

In the normal course of operations, there should be no need to manually run quotacheck(8), quotaon(8), or quotaoff(8). However, one should read their manual pages to be familiar with their operation.

19.13.2 Setting Quota Limits

Once the system has been configured to enable quotas, verify they really are enabled by running:

```
# quota -v
```

There should be a one line summary of disk usage and current quota limits for each file system that quotas are enabled on.

The system is now ready to be assigned quota limits with edquota(8).

Several options are available to enforce limits on the amount of disk space a user or group may allocate, and how many files they may create. Allocations can be limited based on disk space (block quotas), number of files (inode quotas), or a combination of both. Each limits is further broken down into two categories: hard and soft limits.

A hard limit may not be exceeded. Once a user reaches a hard limit, no further allocations can be made on that file system by that user. For example, if the user has a hard limit of 500 kbytes on a file system and is currently using 490 kbytes, the user can only allocate an additional 10 kbytes. Attempting to allocate an additional 11 kbytes will fail.

Soft limits can be exceeded for a limited amount of time, known as the grace period, which is one week by default. If a user stays over their limit longer than the grace period, the soft limit turns into a hard limit and no further allocations are allowed. When the user drops back below the soft limit, the grace period is reset.

The following is an example output from edquota(8). When edquota(8) is invoked, the editor specified by `EDITOR` is opened in order to edit the quota limits. The default editor is set to **vi**.

```
# edquota -u test

Quotas for user test:
/usr: kbytes in use: 65, limits (soft = 50, hard = 75)
        inodes in use: 7, limits (soft = 50, hard = 60)
/usr/var: kbytes in use: 0, limits (soft = 50, hard = 75)
        inodes in use: 0, limits (soft = 50, hard = 60)
```

There are normally two lines for each file system that has quotas enabled. One line represents the block limits and the other represents the inode limits. Change the value to modify the quota limit. For example, to raise this user's block limit from a soft limit of 50 and a hard limit of 75 to a soft limit of 500 and a hard limit of 600, change:

```
/usr: kbytes in use: 65, limits (soft = 50, hard = 75)
```

to:

```
/usr: kbytes in use: 65, limits (soft = 500, hard = 600)
```

The new quota limits take affect upon exiting the editor.

Sometimes it is desirable to set quota limits on a range of UIDs. This can be done by passing `-p` to edquota(8). First, assign the desired quota limit to a user, then run `edquota -p protouser startuid-enduid`. For example, if `test` has the desired quota limits, the following command will duplicate those quota limits for UIDs 10,000 through 19,999:

```
# edquota -p test 10000-19999
```

For more information, refer to edquota(8).

19.13.3 Checking Quota Limits and Disk Usage

Either quota(1) or repquota(8) can be used to check quota limits and disk usage. To check individual user or group quotas and disk usage, use quota(1). A user may only examine their own quota and the quota of a group they are a member of. Only the superuser may view all user and group quotas. To get a summary of all quotas and disk usage for file systems with quotas enabled, use repquota(8).

The following is sample output from `quota -v` for a user that has quota limits on two file systems.

```
Disk quotas for user test (uid 1002):
     Filesystem  usage    quota   limit   grace   files   quota   limit   grace
           /usr      65*     50      75   5days       7      50      60
       /usr/var       0      50      75               0      50      60
```

In this example, the user is currently 15 kbytes over the soft limit of 50 kbytes on `/usr` and has 5 days of grace period left. The asterisk `*` indicates that the user is currently over the quota limit.

Normally, file systems that the user is not using any disk space on will not show in the output of quota(1), even if the user has a quota limit assigned for that file system. Use `-v` to display those file systems, such as `/usr/var` in the above example.

19.13.4 Quotas over NFS

Quotas are enforced by the quota subsystem on the NFS server. The rpc.rquotad(8) daemon makes quota information available to quota(1) on NFS clients, allowing users on those machines to see their quota statistics.

Enable `rpc.rquotad` in `/etc/inetd.conf` like so:

```
rquotad/1      dgram rpc/udp wait root /usr/libexec/rpc.rquotad rpc.rquotad
```

Now restart `inetd`:

```
# service inetd restart
```

19.14 Encrypting Disk Partitions

Contributed by Lucky Green.

FreeBSD offers excellent online protections against unauthorized data access. File permissions and Mandatory Access Control (MAC) help prevent unauthorized users from accessing data while the operating system is active and the computer is powered up. However, the permissions enforced by the operating system are irrelevant if an attacker has physical access to a computer and can move the computer's hard drive to another system to copy and analyze the data.

Regardless of how an attacker may have come into possession of a hard drive or powered-down computer, both the GEOM Based Disk Encryption (`gbde`) and `geli` cryptographic subsystems in FreeBSD are able to protect the data on the computer's file systems against even highly-motivated attackers with significant resources. Unlike cumbersome encryption methods that encrypt only individual files, `gbde` and `geli` transparently encrypt entire file systems. No cleartext ever touches the hard drive's platter.

19.14.1 Disk Encryption with gbde

1. Configuring **gbde** requires superuser privileges.

   ```
   % su -
   Password:
   ```

2. If using a custom kernel configuration file, ensure it contains this line:

   ```
   options GEOM_BDE
   ```

   If the kernel already contains this support, use `kldload` to load gbde(4):

   ```
   # kldload geom_bde
   ```

19.14.1.1 Preparing the Encrypted Hard Drive

The following example demonstrates adding a new hard drive to a system that will hold a single encrypted partition. This partition will be mounted as `/private`. **gbde** can also be used to encrypt `/home` and `/var/mail`, but this requires more complex instructions which exceed the scope of this introduction.

1. Add the New Hard Drive

   Install the new drive to the system as explained in Section 19.3. For the purposes of this example, a new hard drive partition has been added as `/dev/ad4s1c` and `/dev/ad0s1*` represents the existing standard FreeBSD partitions.

   ```
   # ls /dev/ad*
   /dev/ad0        /dev/ad0s1b     /dev/ad0s1e     /dev/ad4s1
   /dev/ad0s1      /dev/ad0s1c     /dev/ad0s1f     /dev/ad4s1c
   /dev/ad0s1a     /dev/ad0s1d     /dev/ad4
   ```

2. Create a Directory to Hold `gbde` Lock Files

   ```
   # mkdir /etc/gbde
   ```

   The **gbde** lock file contains information that **gbde** requires to access encrypted partitions. Without access to the lock file, **gbde** will not be able to decrypt the data contained in the encrypted partition without significant manual intervention which is not supported by the software. Each encrypted partition uses a separate lock file.

3. Initialize the `gbde` Partition

   A **gbde** partition must be initialized before it can be used. This initialization needs to be performed only once:

   ```
   # gbde init /dev/ad4s1c -i -L /etc/gbde/ad4s1c.lock
   ```

   gbde(8) will open the default editor, in order to set various configuration options in a template. For use with UFS1 or UFS2, set the sector_size to 2048:

   ```
   # $FreeBSD: src/sbin/gbde/template.txt,v 1.1.36.1 2009/08/03 08:13:06 kensmith Exp $
   #
   # Sector size is the smallest unit of data which can be read or written.
   # Making it too small decreases performance and decreases available space.
   # Making it too large may prevent filesystems from working.  512 is the
   # minimum and always safe.  For UFS, use the fragment size
   #
   sector_size    =       2048
   [...]
   ```

   gbde(8) will ask the user twice to type the passphrase used to secure the data. The passphrase must be the same both times. The ability of **gbde** to protect data depends entirely on the quality of the passphrase. For tips on how to select a secure passphrase that is easy to remember, see the Diceware Passphrase (http://world.std.com/~reinhold/diceware.html) website.

   `gbde init`creates a lock file for the **gbde** partition. In this example, it is stored as `/etc/gbde/ad4s1c.lock`. **gbde** lock files must end in ".lock" in order to be correctly detected by the `/etc/rc.d/gbde` start up script.

   > **Caution: gbde** lock files *must* be backed up together with the contents of any encrypted partitions. While deleting a lock file alone cannot prevent a determined attacker from decrypting a **gbde** partition, without the lock file, the legitimate owner will be unable to access the data on the encrypted partition without a significant amount of work that is totally unsupported by gbde(8).

4. Attach the Encrypted Partition to the Kernel

   ```
   # gbde attach /dev/ad4s1c -l /etc/gbde/ad4s1c.lock
   ```

   This command will prompt to input the passphrase that was selected during the initialization of the encrypted partition. The new encrypted device will appear in `/dev` as `/dev/device_name.bde`:

   ```
   # ls /dev/ad*
   /dev/ad0        /dev/ad0s1b     /dev/ad0s1e     /dev/ad4s1
   /dev/ad0s1      /dev/ad0s1c     /dev/ad0s1f     /dev/ad4s1c
   /dev/ad0s1a     /dev/ad0s1d     /dev/ad4        /dev/ad4s1c.bde
   ```

5. Create a File System on the Encrypted Device

   Once the encrypted device has been attached to the kernel, a file system can be created on the device using newfs(8). This example creates a UFS2 file system with soft updates enabled.

   ```
   # newfs -U /dev/ad4s1c.bde
   ```

   > **Note:** newfs(8) must be performed on an attached **gbde** partition which is identified by a `*.bde` extension to the device name.

6. Mount the Encrypted Partition

   Create a mount point for the encrypted file system:

   ```
   # mkdir /private
   ```

   Mount the encrypted file system:

   ```
   # mount /dev/ad4s1c.bde /private
   ```

7. Verify That the Encrypted File System is Available

   The encrypted file system should now be visible to df(1) and be available for use.

   ```
   % df -H
   Filesystem        Size   Used  Avail Capacity  Mounted on
   /dev/ad0s1a      1037M    72M   883M     8%    /
   /devfs            1.0K   1.0K     0B   100%    /dev
   /dev/ad0s1f       8.1G    55K   7.5G     0%    /home
   /dev/ad0s1e      1037M   1.1M   953M     0%    /tmp
   /dev/ad0s1d       6.1G   1.9G   3.7G    35%    /usr
   /dev/ad4s1c.bde   150G   4.1K   138G     0%    /private
   ```

19.14.1.2 Mounting Existing Encrypted File Systems

After each boot, any encrypted file systems must be re-attached to the kernel, checked for errors, and mounted, before the file systems can be used. The required commands must be executed as `root`.

1. Attach the `gbde` Partition to the Kernel

   ```
   # gbde attach /dev/ad4s1c -l /etc/gbde/ad4s1c.lock
   ```

   This command will prompt for the passphrase that was selected during initialization of the encrypted **gbde** partition.

2. Check the File System for Errors

   Since encrypted file systems cannot yet be listed in `/etc/fstab` for automatic mounting, the file systems must be checked for errors by running fsck(8) manually before mounting:

   ```
   # fsck -p -t ffs /dev/ad4s1c.bde
   ```

3. Mount the Encrypted File System

   ```
   # mount /dev/ad4s1c.bde /private
   ```

   The encrypted file system is now available for use.

19.14.1.2.1 Automatically Mounting Encrypted Partitions

It is possible to create a script to automatically attach, check, and mount an encrypted partition, but for security reasons the script should not contain the gbde(8) password. Instead, it is recommended that such scripts be run manually while providing the password via the console or ssh(1).

As an alternative, an `rc.d` script is provided. Arguments for this script can be passed via rc.conf(5):

```
gbde_autoattach_all="YES"
gbde_devices="ad4s1c"
gbde_lockdir="/etc/gbde"
```

This requires that the **gbde** passphrase be entered at boot time. After typing the correct passphrase, the **gbde** encrypted partition will be mounted automatically. This can be useful when using **gbde** on laptops.

19.14.1.3 Cryptographic Protections Employed by `gbde`

gbde(8) encrypts the sector payload using 128-bit AES in CBC mode. Each sector on the disk is encrypted with a different AES key. For more information on the cryptographic design, including how the sector keys are derived from the user-supplied passphrase, refer to gbde(4).

19.14.1.4 Compatibility Issues

sysinstall(8) is incompatible with **gbde**-encrypted devices. All `*.bde` devices must be detached from the kernel before starting sysinstall(8) or it will crash during its initial probing for devices. To detach the encrypted device used in the example, use the following command:

```
# gbde detach /dev/ad4s1c
```

Also, since vinum(4) does not use the geom(4) subsystem, **gbde** can not be used with **vinum** volumes.

19.14.2 Disk Encryption with `geli`

Contributed by Daniel Gerzo.

An alternative cryptographic GEOM class is available through geli(8). `geli` differs from `gbde`; offers different features, and uses a different scheme for doing cryptographic work.

geli(8) provides the following features:

- Utilizes the crypto(9) framework and, when cryptographic hardware is available, `geli` uses it automatically.
- Supports multiple cryptographic algorithms such as AES, Blowfish, and 3DES.
- Allows the root partition to be encrypted. The passphrase used to access the encrypted root partition will be requested during system boot.
- Allows the use of two independent keys such as a “key” and a “company key”.
- `geli` is fast as it performs simple sector-to-sector encryption.
- Allows backup and restore of master keys. If a user destroys their keys, it is still possible to get access to the data by restoring keys from the backup.
- Allows a disk to attach with a random, one-time key which is useful for swap partitions and temporary file systems.

More `geli` features can be found in geli(8).

This section describes how to enable support for `geli` in the FreeBSD kernel and explains how to create and use a `geli` encryption provider.

Superuser privileges are required since modifications to the kernel are necessary.

1. Adding `geli` Support to the Kernel

   For a custom kernel, ensure the kernel configuration file contains these lines:

   ```
   options GEOM_ELI
   device crypto
   ```

   Alternatively, the `geli` module can be loaded at boot time by adding the following line to `/boot/loader.conf`:

   ```
   geom_eli_load="YES"
   ```

   geli(8) should now be supported by the kernel.

2. Generating the Master Key

   The following example describes how to generate a key file which will be used as part of the master key for the encrypted provider mounted under `/private`. The key file will provide some random data used to encrypt the master key. The master key will also be protected by a passphrase. The provider’s sector size will be 4kB. The example will describe how to attach to the `geli` provider, create a file system on it, mount it, work with it, and finally, how to detach it.

   It is recommended to use a bigger sector size, such as 4kB, for better performance.

   The master key will be protected with a passphrase and the data source for the key file will be `/dev/random`. The sector size of the provider `/dev/da2.eli` will be 4kB.

   ```
   # dd if=/dev/random of=/root/da2.key bs=64 count=1
   # geli init -s 4096 -K /root/da2.key /dev/da2
   Enter new passphrase:
   Reenter new passphrase:
   ```

   It is not mandatory to use both a passphrase and a key file as either method of securing the master key can be used in isolation.

If the key file is given as "-", standard input will be used. This example shows how more than one key file can be used:

```
# cat keyfile1 keyfile2 keyfile3 | geli init -K - /dev/da2
```

3. Attaching the Provider with the Generated Key

```
# geli attach -k /root/da2.key /dev/da2
Enter passphrase:
```

The new plaintext device will be named `/dev/da2.eli`.

```
# ls /dev/da2*
/dev/da2  /dev/da2.eli
```

4. Creating the New File System

```
# dd if=/dev/random of=/dev/da2.eli bs=1m
# newfs /dev/da2.eli
# mount /dev/da2.eli /private
```

The encrypted file system should now be visible to df(1) and be available for use:

```
# df -H
Filesystem     Size   Used  Avail Capacity  Mounted on
/dev/ad0s1a    248M    89M   139M    38%    /
/devfs         1.0K   1.0K     0B   100%    /dev
/dev/ad0s1f    7.7G   2.3G   4.9G    32%    /usr
/dev/ad0s1d    989M   1.5M   909M     0%    /tmp
/dev/ad0s1e    3.9G   1.3G   2.3G    35%    /var
/dev/da2.eli   150G   4.1K   138G     0%    /private
```

5. Unmounting and Detaching the Provider

Once the work on the encrypted partition is done, and the `/private` partition is no longer needed, it is prudent to consider unmounting and detaching the `geli` encrypted partition from the kernel:

```
# umount /private
# geli detach da2.eli
```

More information about the use of geli(8) can be found in its manual page.

19.14.2.1 Using the `geli rc.d` Script

`geli` comes with a `rc.d` script which can be used to simplify the usage of `geli`. An example of configuring `geli` through rc.conf(5) follows:

```
geli_devices="da2"
geli_da2_flags="-p -k /root/da2.key"
```

This configures `/dev/da2` as a `geli` provider of which the master key file is located in `/root/da2.key`. `geli` will not use a passphrase when attaching to the provider if `-P` was given during the `geli init` phase. The system will detach the `geli` provider from the kernel before the system shuts down.

More information about configuring `rc.d` is provided in the rc.d section of the Handbook.

19.15 Encrypting Swap Space

Written by Christian Brüffer.

Like the encryption of disk partitions, encryption of swap space is used to protect sensitive information. Consider an application that deals with passwords. As long as these passwords stay in physical memory, these passwords will not be written to disk and be cleared after a reboot. If FreeBSD starts swapping out memory pages to free space for other applications, the passwords may be written to the disk platters unencrypted. Encrypting swap space can be a solution for this scenario.

The gbde(8) or geli(8) encryption systems may be used for swap encryption. Both systems use the `encswap` rc.d script.

Note: For the remainder of this section, `ad0s1b` will be the swap partition.

Swap partitions are not encrypted by default and should be cleared of any sensitive data before continuing. To overwrite the current swap parition with random garbage, execute the following command:

```
# dd if=/dev/random of=/dev/ad0s1b bs=1m
```

19.15.1 Swap Encryption with gbde(8)

The `.bde` suffix should be added to the device in the respective `/etc/fstab` swap line:

```
# Device                Mountpoint      FStype  Options         Dump    Pass#
/dev/ad0s1b.bde               none            swap    sw              0       0
```

19.15.2 Swap Encryption with geli(8)

The procedure for instead using geli(8) for swap encryption is similar to that of using gbde(8). The `.eli` suffix should be added to the device in the respective `/etc/fstab` swap line:

```
# Device                Mountpoint      FStype  Options         Dump    Pass#
/dev/ad0s1b.eli               none            swap    sw              0       0
```

geli(8) uses the AES algorithm with a key length of 128 bit by default. These defaults can be altered by using `geli_swap_flags` in `/etc/rc.conf`. The following line tells the `encswap` rc.d script to create geli(8) swap partitions using the Blowfish algorithm with a key length of 128 bits and a sectorsize of 4 kilobytes, and sets “detach on last close”:

```
geli_swap_flags="-e blowfish -l 128 -s 4096 -d"
```

Refer to the description of `onetime` in geli(8) for a list of possible options.

19.15.3 Encrypted Swap Verification

Once the system has rebooted, proper operation of the encrypted swap can be verified using `swapinfo`.

If gbde(8) is being used:

```
% swapinfo
Device          1K-blocks     Used    Avail Capacity
/dev/ad0s1b.bde    542720        0   542720     0%
```

If geli(8) is being used:

```
% swapinfo
Device          1K-blocks     Used    Avail Capacity
/dev/ad0s1b.eli    542720        0   542720     0%
```

19.16 Highly Available Storage (HAST)

Contributed by Daniel Gerzo. With inputs from Freddie Cash, Pawel Jakub Dawidek, Michael W. Lucas, and Viktor Petersson.

19.16.1 Synopsis

High availability is one of the main requirements in serious business applications and highly-available storage is a key component in such environments. Highly Available STorage, or HAST, was developed by Pawel Jakub Dawidek <pjd@FreeBSD.org> as a framework which allows transparent storage of the same data across several physically separated machines connected by a TCP/IP network. HAST can be understood as a network-based RAID1 (mirror), and is similar to the DRBD® storage system known from the GNU/Linux platform. In combination with other high-availability features of FreeBSD like CARP, HAST makes it possible to build a highly-available storage cluster that is resistant to hardware failures.

After reading this section, you will know:

- What HAST is, how it works and which features it provides.
- How to set up and use HAST on FreeBSD.
- How to integrate CARP and devd(8) to build a robust storage system.

Before reading this section, you should:

- Understand UNIX and FreeBSD basics.
- Know how to configure network interfaces and other core FreeBSD subsystems.
- Have a good understanding of FreeBSD networking.

The HAST project was sponsored by The FreeBSD Foundation with support from OMCnet Internet Service GmbH (http://www.omc.net/) and TransIP BV (http://www.transip.nl/).

19.16.2 HAST Features

The main features of the HAST system are:

- Can be used to mask I/O errors on local hard drives.

- File system agnostic as it works with any file system supported by FreeBSD.
- Efficient and quick resynchronization, synchronizing only blocks that were modified during the downtime of a node.
- Can be used in an already deployed environment to add additional redundancy.
- Together with CARP, **Heartbeat**, or other tools, it can be used to build a robust and durable storage system.

19.16.3 HAST Operation

As HAST provides a synchronous block-level replication of any storage media to several machines, it requires at least two physical machines: the `primary`, also known as the `master` node, and the `secondary` or `slave` node. These two machines together are referred to as a cluster.

> **Note:** HAST is currently limited to two cluster nodes in total.

Since HAST works in a primary-secondary configuration, it allows only one of the cluster nodes to be active at any given time. The `primary` node, also called `active`, is the one which will handle all the I/O requests to HAST-managed devices. The `secondary` node is automatically synchronized from the `primary` node.

The physical components of the HAST system are:

- local disk on primary node, and
- disk on remote, secondary node.

HAST operates synchronously on a block level, making it transparent to file systems and applications. HAST provides regular GEOM providers in `/dev/hast/` for use by other tools or applications, thus there is no difference between using HAST-provided devices and raw disks or partitions.

Each write, delete, or flush operation is sent to the local disk and to the remote disk over TCP/IP. Each read operation is served from the local disk, unless the local disk is not up-to-date or an I/O error occurs. In such case, the read operation is sent to the secondary node.

19.16.3.1 Synchronization and Replication Modes

HAST tries to provide fast failure recovery. For this reason, it is very important to reduce synchronization time after a node's outage. To provide fast synchronization, HAST manages an on-disk bitmap of dirty extents and only synchronizes those during a regular synchronization, with an exception of the initial sync.

There are many ways to handle synchronization. HAST implements several replication modes to handle different synchronization methods:

- *memsync*: report write operation as completed when the local write operation is finished and when the remote node acknowledges data arrival, but before actually storing the data. The data on the remote node will be stored directly after sending the acknowledgement. This mode is intended to reduce latency, but still provides very good reliability.
- *fullsync*: report write operation as completed when local write completes and when remote write completes. This is the safest and the slowest replication mode. This mode is the default.

- *async*: report write operation as completed when local write completes. This is the fastest and the most dangerous replication mode. It should be used when replicating to a distant node where latency is too high for other modes.

19.16.4 HAST Configuration

HAST requires `GEOM_GATE` support which is not present in the default `GENERIC` kernel. However, the `geom_gate.ko` loadable module is available in the default FreeBSD installation. Alternatively, to build `GEOM_GATE` support into the kernel statically, add this line to the custom kernel configuration file:

```
options GEOM_GATE
```

The HAST framework consists of several parts from the operating system's point of view:

- the hastd(8) daemon responsible for data synchronization,
- the hastctl(8) userland management utility,
- and the hast.conf(5) configuration file.

The following example describes how to configure two nodes in `master-slave` / `primary-secondary` operation using HAST to replicate the data between the two. The nodes will be called *`hasta`* with an IP address of *`172.16.0.1`* and *`hastb`* with an IP of address *`172.16.0.2`*. Both nodes will have a dedicated hard drive `/dev/`*`ad6`* of the same size for HAST operation. The HAST pool, sometimes also referred to as a resource or the GEOM provider in `/dev/hast/`, will be called *`test`*.

Configuration of HAST is done using `/etc/hast.conf`. This file should be the same on both nodes. The simplest configuration possible is:

```
resource test {
       on hasta {
               local /dev/ad6
               remote 172.16.0.2
       }
       on hastb {
               local /dev/ad6
               remote 172.16.0.1
       }
}
```

For more advanced configuration, refer to hast.conf(5).

> **Tip:** It is also possible to use host names in the `remote` statements. In such a case, make sure that these hosts are resolvable and are defined in `/etc/hosts` or in the local DNS.

Now that the configuration exists on both nodes, the HAST pool can be created. Run these commands on both nodes to place the initial metadata onto the local disk and to start hastd(8):

```
# hastctl create test
# service hastd onestart
```

> **Note:** It is *not* possible to use GEOM providers with an existing file system or to convert an existing storage to a HAST-managed pool. This procedure needs to store some metadata on the provider and there will not be enough required space available on an existing provider.

A HAST node's `primary` or `secondary` role is selected by an administrator, or software like **Heartbeat**, using hastctl(8). On the primary node, *`hasta`*, issue this command:

```
# hastctl role primary test
```

Similarly, run this command on the secondary node, *`hastb`*:

```
# hastctl role secondary test
```

> **Caution:** When the nodes are unable to communicate with each other, and both are configured as primary nodes, the condition is called `split-brain`. To troubleshoot this situation, follow the steps described in Section 19.16.5.2.

Verify the result by running hastctl(8) on each node:

```
# hastctl status test
```

The important text is the `status` line, which should say `complete` on each of the nodes. If it says `degraded`, something went wrong. At this point, the synchronization between the nodes has already started. The synchronization completes when `hastctl status` reports 0 bytes of `dirty` extents.

The next step is to create a filesystem on the `/dev/hast/`*`test`* GEOM provider and mount it. This must be done on the `primary` node, as `/dev/hast/`*`test`* appears only on the `primary` node. Creating the filesystem can take a few minutes, depending on the size of the hard drive:

```
# newfs -U /dev/hast/test
# mkdir /hast/test
# mount /dev/hast/test /hast/test
```

Once the HAST framework is configured properly, the final step is to make sure that HAST is started automatically during system boot. Add this line to `/etc/rc.conf`:

```
hastd_enable="YES"
```

19.16.4.1 Failover Configuration

The goal of this example is to build a robust storage system which is resistant to the failure of any given node. The scenario is that a `primary` node of the cluster fails. If this happens, the `secondary` node is there to take over seamlessly, check and mount the file system, and continue to work without missing a single bit of data.

To accomplish this task, another FreeBSD feature, CARP, provides for automatic failover on the IP layer. CARP (Common Address Redundancy Protocol) allows multiple hosts on the same network segment to share an IP address. Set up CARP on both nodes of the cluster according to the documentation available in Section 32.12. After setup, each node will have its own `carp0` interface with a shared IP address of *`172.16.0.254`*. The primary HAST node of the cluster must be the master CARP node.

The HAST pool created in the previous section is now ready to be exported to the other hosts on the network. This can be accomplished by exporting it through NFS or **Samba**, using the shared IP address *172.16.0.254*. The only problem which remains unresolved is an automatic failover should the primary node fail.

In the event of CARP interfaces going up or down, the FreeBSD operating system generates a devd(8) event, making it possible to watch for state changes on the CARP interfaces. A state change on the CARP interface is an indication that one of the nodes failed or came back online. These state change events make it possible to run a script which will automatically handle the HAST failover.

To be able to catch state changes on the CARP interfaces, add this configuration to `/etc/devd.conf` on each node:

```
notify 30 {
       match "system" "IFNET";
       match "subsystem" "carp0";
       match "type" "LINK_UP";
       action "/usr/local/sbin/carp-hast-switch master";
};

notify 30 {
       match "system" "IFNET";
       match "subsystem" "carp0";
       match "type" "LINK_DOWN";
       action "/usr/local/sbin/carp-hast-switch slave";
};
```

Restart devd(8) on both nodes to put the new configuration into effect:

```
# service devd restart
```

When the `carp0` interface state changes by going up or down , the system generates a notification, allowing the devd(8) subsystem to run an arbitrary script, in this case `/usr/local/sbin/carp-hast-switch`. This script handles the automatic failover. For further clarification about the above devd(8) configuration, refer to devd.conf(5).

An example of such a script could be:

```
#!/bin/sh

# Original script by Freddie Cash <fjwcash@gmail.com>
# Modified by Michael W. Lucas <mwlucas@BlackHelicopters.org>
# and Viktor Petersson <vpetersson@wireload.net>

# The names of the HAST resources, as listed in /etc/hast.conf
resources="test"

# delay in mounting HAST resource after becoming master
# make your best guess
delay=3

# logging
log="local0.debug"
name="carp-hast"

# end of user configurable stuff
```

```
case "$1" in
        master)
                logger -p $log -t $name "Switching to primary provider for ${resources}."
                sleep ${delay}

                # Wait for any "hastd secondary" processes to stop
                for disk in ${resources}; do
                        while $( pgrep -lf "hastd: ${disk} \(secondary\)" > /dev/null 2>&1 ); do
                                sleep 1
                        done

                        # Switch role for each disk
                        hastctl role primary ${disk}
                        if [ $? -ne 0 ]; then
                                logger -p $log -t $name "Unable to change role to primary for reso
                                exit 1
                        fi
                done

                # Wait for the /dev/hast/* devices to appear
                for disk in ${resources}; do
                        for I in $( jot 60 ); do
                                [ -c "/dev/hast/${disk}" ] && break
                                sleep 0.5
                        done

                        if [ ! -c "/dev/hast/${disk}" ]; then
                                logger -p $log -t $name "GEOM provider /dev/hast/${disk} did not a
                                exit 1
                        fi
                done

                logger -p $log -t $name "Role for HAST resources ${resources} switched to primary.


                logger -p $log -t $name "Mounting disks."
                for disk in ${resources}; do
                        mkdir -p /hast/${disk}
                        fsck -p -y -t ufs /dev/hast/${disk}
                        mount /dev/hast/${disk} /hast/${disk}
                done

        ;;

        slave)
                logger -p $log -t $name "Switching to secondary provider for ${resources}."

                # Switch roles for the HAST resources
                for disk in ${resources}; do
                        if ! mount | grep -q "^/dev/hast/${disk} on "
                        then
                        else
                                umount -f /hast/${disk}
```

```
                        fi
                        sleep $delay
                        hastctl role secondary ${disk} 2>&1
                        if [ $? -ne 0 ]; then
                                logger -p $log -t $name "Unable to switch role to secondary for res
                                exit 1
                        fi
                        logger -p $log -t $name "Role switched to secondary for resource ${disk}."
                done
        ;;
esac
```

In a nutshell, the script takes these actions when a node becomes `master` / `primary`:

- Promotes the HAST pools to primary on a given node.
- Checks the file system under the HAST pool.
- Mounts the pools at an appropriate place.

When a node becomes `backup` / `secondary`:

- Unmounts the HAST pools.
- Degrades the HAST pools to secondary.

> **Caution:** Keep in mind that this is just an example script which serves as a proof of concept. It does not handle all the possible scenarios and can be extended or altered in any way, for example, to start/stop required services.

> **Tip:** For this example, a standard UFS file system was used. To reduce the time needed for recovery, a journal-enabled UFS or ZFS file system can be used instead.

More detailed information with additional examples can be found in the HAST Wiki (http://wiki.FreeBSD.org/HAST) page.

19.16.5 Troubleshooting

19.16.5.1 General Troubleshooting Tips

HAST should generally work without issues. However, as with any other software product, there may be times when it does not work as supposed. The sources of the problems may be different, but the rule of thumb is to ensure that the time is synchronized between all nodes of the cluster.

When troubleshooting HAST problems, the debugging level of hastd(8) should be increased by starting hastd(8) with `-d`. This argument may be specified multiple times to further increase the debugging level. A lot of useful information may be obtained this way. Consider also using `-F`, which starts hastd(8) in the foreground.

19.16.5.2 Recovering from the Split-brain Condition

`Split-brain` is when the nodes of the cluster are unable to communicate with each other, and both are configured as primary. This is a dangerous condition because it allows both nodes to make incompatible changes to the data. This problem must be corrected manually by the system administrator.

The administrator must decide which node has more important changes (or merge them manually) and let HAST perform full synchronization of the node which has the broken data. To do this, issue these commands on the node which needs to be resynchronized:

```
# hastctl role init <resource>
# hastctl create <resource>
# hastctl role secondary <resource>
```

Chapter 20 GEOM: Modular Disk Transformation Framework

Written by Tom Rhodes.

20.1 Synopsis

This chapter covers the use of disks under the GEOM framework in FreeBSD. This includes the major RAID control utilities which use the framework for configuration. This chapter will not go into in depth discussion on how GEOM handles or controls I/O, the underlying subsystem, or code. This information is provided in geom(4) and its various `SEE ALSO` references. This chapter is also not a definitive guide to RAID configurations and only GEOM-supported RAID classifications will be discussed.

After reading this chapter, you will know:

- What type of RAID support is available through GEOM.
- How to use the base utilities to configure, maintain, and manipulate the various RAID levels.
- How to mirror, stripe, encrypt, and remotely connect disk devices through GEOM.
- How to troubleshoot disks attached to the GEOM framework.

Before reading this chapter, you should:

- Understand how FreeBSD treats disk devices.
- Know how to configure and install a new FreeBSD kernel.

20.2 GEOM Introduction

GEOM permits access and control to classes, such as Master Boot Records and BSD labels, through the use of providers, or the special files in `/dev`. By supporting various software RAID configurations, GEOM transparently provides access to the operating system and operating system utilities.

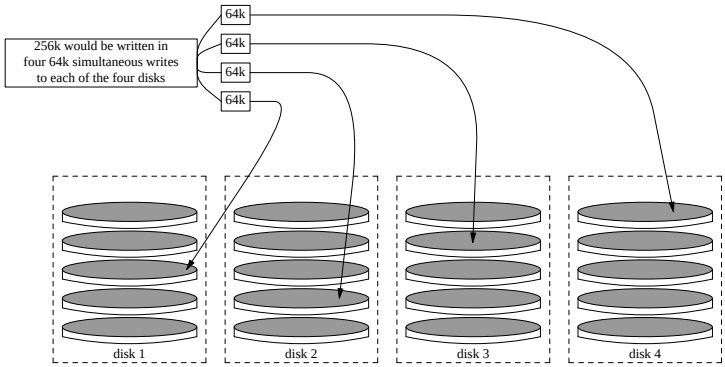
20.3 RAID0 - Striping

Written by Tom Rhodes and Murray Stokely.

Striping combine several disk drives into a single volume. In many cases, this is done through the use of hardware controllers. The GEOM disk subsystem provides software support for RAID0, also known as disk striping.

In a RAID0 system, data is split into blocks that get written across all the drives in the array. Instead of having to wait on the system to write 256k to one disk, a RAID0 system can simultaneously write 64k to each of four different disks, offering superior I/O performance. This performance can be enhanced further by using multiple disk controllers.

Each disk in a RAID0 stripe must be of the same size, since I/O requests are interleaved to read or write to multiple disks in parallel.



Creating a Stripe of Unformatted ATA Disks

1. Load the `geom_stripe.ko` module:

   ```
   # kldload geom_stripe
   ```

2. Ensure that a suitable mount point exists. If this volume will become a root partition, then temporarily use another mount point such as `/mnt`:

   ```
   # mkdir /mnt
   ```

3. Determine the device names for the disks which will be striped, and create the new stripe device. For example, to stripe two unused and unpartitioned ATA disks with device names of `/dev/ad2` and `/dev/ad3`:

   ```
   # gstripe label -v st0 /dev/ad2 /dev/ad3
   Metadata value stored on /dev/ad2.
   Metadata value stored on /dev/ad3.
   Done.
   ```

4. Write a standard label, also known as a partition table, on the new volume and install the default bootstrap code:

   ```
   # bsdlabel -wB /dev/stripe/st0
   ```

5. This process should create two other devices in `/dev/stripe` in addition to `st0`. Those include `st0a` and `st0c`. At this point, a file system may be created on `st0a` using `newfs`:

   ```
   # newfs -U /dev/stripe/st0a
   ```

   Many numbers will glide across the screen, and after a few seconds, the process will be complete. The volume has been created and is ready to be mounted.

To manually mount the created disk stripe:

```
# mount /dev/stripe/st0a /mnt
```

To mount this striped file system automatically during the boot process, place the volume information in `/etc/fstab`. In this example, a permanent mount point, named `stripe`, is created:

```
# mkdir /stripe
# echo "/dev/stripe/st0a /stripe ufs rw 2 2" \
    >> /etc/fstab
```

The `geom_stripe.ko` module must also be automatically loaded during system initialization, by adding a line to `/boot/loader.conf`:

```
# echo 'geom_stripe_load="YES"' >> /boot/loader.conf
```

20.4 RAID1 - Mirroring

RAID1, or *mirroring*, is the technique of writing the same data to more than one disk drive. Mirrors are usually used to guard against data loss due to drive failure. Each drive in a mirror contains an identical copy of the data. When an individual drive fails, the mirror continues to work, providing data from the drives that are still functioning. The computer keeps running, and the administrator has time to replace the failed drive without user interruption.

Two common situations are illustrated in these examples. The first creates a mirror out of two new drives and uses it as a replacement for an existing single drive. The second example creates a mirror on a single new drive, copies the old drive's data to it, then inserts the old drive into the mirror. While this procedure is slightly more complicated, it only requires one new drive.

Traditionally, the two drives in a mirror are identical in model and capacity, but gmirror(8) does not require that. Mirrors created with dissimilar drives will have a capacity equal to that of the smallest drive in the mirror. Extra space on larger drives will be unused. Drives inserted into the mirror later must have at least as much capacity as the smallest drive already in the mirror.

> **Warning:** The mirroring procedures shown here are non-destructive, but as with any major disk operation, make a full backup first.

20.4.1 Metadata Issues

Many disk systems store metadata at the end of each disk. Old metadata should be erased before reusing the disk for a mirror. Most problems are caused by two particular types of leftover metadata: GPT partition tables, and old gmirror(8) metadata from a previous mirror.

GPT metadata can be erased with gpart(8). This example erases both primary and backup GPT partition tables from disk `ada8`:

```
# gpart destroy -F ada8
```

gmirror(8) can remove a disk from an active mirror and erase the metadata in one step. Here, the example disk `ada8` is removed from the active mirror `gm4`:

```
# gmirror remove gm4 ada8
```

If the mirror is not running but old mirror metadata is still on the disk, use `gmirror clear` to remove it:

```
# gmirror clear ada8
```

gmirror(8) stores one block of metadata at the end of the disk. Because GPT partition schemes also store metadata at the end of the disk, mirroring full GPT disks with gmirror(8) is not recommended. MBR partitioning is used here because it only stores a partition table at the start of the disk and does not conflict with gmirror(8).

20.4.2 Creating a Mirror with Two New Disks

In this example, FreeBSD has already been installed on a single disk, `ada0`. Two new disks, `ada1` and `ada2`, have been connected to the system. A new mirror will be created on these two disks and used to replace the old single disk.

gmirror(8) requires a kernel module, `geom_mirror.ko`, either built into the kernel or loaded at boot- or run-time. Manually load the kernel module now:

```
# gmirror load
```

Create the mirror with the two new drives:

```
# gmirror label -v gm0 /dev/ada1 /dev/ada2
```

`gm0` is a user-chosen device name assigned to the new mirror. After the mirror has been started, this device name will appear in `/dev/mirror/`.

MBR and bsdlabel partition tables can now be created on the mirror with gpart(8). Here we show a traditional split-filesystem layout, with partitions for `/`, swap, `/var`, `/tmp`, and `/usr`. A single `/` filesystem and a swap partition will also work.

Partitions on the mirror do not have to be the same size as those on the existing disk, but they must be large enough to hold all the data already present on `ada0`.

```
# gpart create -s MBR mirror/gm0
# gpart add -t freebsd -a 4k mirror/gm0
# gpart show mirror/gm0
=>       63  156301423  mirror/gm0  MBR  (74G)
         63         63                    - free -  (31k)
        126  156301299                1  freebsd  (74G)
  156301425         61                    - free -  (30k)

# gpart create -s BSD mirror/gm0s1
# gpart add -t freebsd-ufs  -a 4k -s 2g mirror/gm0s1
# gpart add -t freebsd-swap -a 4k -s 4g mirror/gm0s1
# gpart add -t freebsd-ufs  -a 4k -s 2g mirror/gm0s1
# gpart add -t freebsd-ufs  -a 4k -s 1g mirror/gm0s1
# gpart add -t freebsd-ufs  -a 4k       mirror/gm0s1
# gpart show mirror/gm0s1
=>        0  156301299  mirror/gm0s1  BSD  (74G)
          0          2                     - free -  (1.0k)
          2    4194304                   1  freebsd-ufs  (2.0G)
    4194306    8388608                   2  freebsd-swap  (4.0G)
   12582914    4194304                   4  freebsd-ufs  (2.0G)
   16777218    2097152                   5  freebsd-ufs  (1.0G)
   18874370  137426928                   6  freebsd-ufs  (65G)
  156301298          1                     - free -  (512B)
```

Make the mirror bootable by installing bootcode in the MBR and bsdlabel and setting the active slice:

```
# gpart bootcode -b /boot/mbr mirror/gm0
# gpart set -a active -i 1 mirror/gm0
# gpart bootcode -b /boot/boot mirror/gm0s1
```

Format the filesystems on the new mirror, enabling soft-updates.

```
# newfs -U /dev/mirror/gm0s1a
# newfs -U /dev/mirror/gm0s1d
# newfs -U /dev/mirror/gm0s1e
# newfs -U /dev/mirror/gm0s1f
```

Filesystems from the original `ada0` disk can now be copied onto the mirror with dump(8) and restore(8).

```
# mount /dev/mirror/gm0s1a /mnt
# dump -C16 -b64 -0aL -f - / | (cd /mnt && restore -rf -)
# mount /dev/mirror/gm0s1d /mnt/var
# mount /dev/mirror/gm0s1e /mnt/tmp
# mount /dev/mirror/gm0s1f /mnt/usr
# dump -C16 -b64 -0aL -f - /var | (cd /mnt/var && restore -rf -)
# dump -C16 -b64 -0aL -f - /tmp | (cd /mnt/tmp && restore -rf -)
# dump -C16 -b64 -0aL -f - /usr | (cd /mnt/usr && restore -rf -)
```

`/mnt/etc/fstab` must be edited to point to the new mirror filesystems:

```
# Device                Mountpoint      FStype  Options Dump    Pass#
/dev/mirror/gm0s1a     /               ufs     rw      1       1
/dev/mirror/gm0s1b     none            swap    sw      0       0
/dev/mirror/gm0s1d     /var            ufs     rw      2       2
/dev/mirror/gm0s1e     /tmp            ufs     rw      2       2
/dev/mirror/gm0s1f     /usr            ufs     rw      2       2
```

If the gmirror(8) kernel module has not been built into the kernel, `/mnt/boot/loader.conf` is edited to load the module at boot:

```
geom_mirror_load="YES"
```

Reboot the system to test the new mirror and verify that all data has been copied. The BIOS will see the mirror as two individual drives rather than a mirror. Because the drives are identical, it does not matter which is selected to boot.

See the Troubleshooting section if there are problems booting. Powering down and disconnecting the original `ada0` disk will allow it to be kept as an offline backup.

In use, the mirror will behave just like the original single drive.

20.4.3 Creating a Mirror with an Existing Drive

In this example, FreeBSD has already been installed on a single disk, `ada0`. A new disk, `ada1`, has been connected to the system. A one-disk mirror will be created on the new disk, the existing system copied onto it, and then the old disk will be inserted into the mirror. This slightly complex procedure is required because gmirror(8) needs to put a 512-byte block of metadata at the end of each disk, and the existing `ada0` has usually had all of its space already allocated.

Load the gmirror(8) kernel module:

```
# gmirror load
```

Check the media size of the original disk with diskinfo(8):

```
# diskinfo -v ada0 | head -n3
```

```
/dev/ada0
       512             # sectorsize
       1000204821504   # mediasize in bytes (931G)
```

Create a mirror on the new disk. To make certain that the mirror capacity is not any larger than the original drive, gnop(8) is used to create a fake drive of the exact same size. This drive does not store any data, but is used only to limit the size of the mirror. When gmirror(8) creates the mirror, it will restrict the capacity to the size of `gzero.nop`, even if the new drive (`ada1`) has more space. Note that the *`1000204821504`* in the second line should be equal to `ada0`'s media size as shown by diskinfo(8) above.

```
# geom zero load
# gnop create -s 1000204821504 gzero
# gmirror label -v gm0 gzero.nop ada1
# gmirror forget gm0
```

`gzero.nop` does not store any data, so the mirror does not see it as connected. The mirror is told to "forget" unconnected components, removing references to `gzero.nop`. The result is a mirror device containing only a single disk, `ada1`.

After creating `gm0`, view the partition table on `ada0`.

This output is from a 1 TB drive. If there is some unallocated space at the end of the drive, the contents may be copied directly from `ada0` to the new mirror.

However, if the output shows that all of the space on the disk is allocated like the following listing, there is no space available for the 512-byte gmirror(8) metadata at the end of the disk.

```
# gpart show ada0
=>        63  1953525105        ada0  MBR  (931G)
          63  1953525105           1  freebsd  [active]  (931G)
```

In this case, the partition table must be edited to reduce the capacity by one sector on `mirror/gm0`. The procedure will be explained later.

In either case, partition tables on the primary disk should be copied first with the gpart(8) `backup` and `restore` subcommands.

```
# gpart backup ada0 > table.ada0
# gpart backup ada0s1 > table.ada0s1
```

These commands create two files, `table.ada0` and `table.ada0s1`. This example is from a 1 TB drive:

```
# cat table.ada0
MBR 4
1 freebsd         63 1953525105   [active]

# cat table.ada0s1
BSD 8
1  freebsd-ufs          0    4194304
2 freebsd-swap    4194304   33554432
4  freebsd-ufs   37748736   50331648
5  freebsd-ufs   88080384   41943040
6  freebsd-ufs  130023424  838860800
7  freebsd-ufs  968884224  984640881
```

If the output of `gpart show` shows no free space at the end of the disk, the size of both the slice and the last partition must be reduced by one sector. Edit the two files, reducing the size of both the slice and last partition by one. These are the last numbers in each listing.

```
# cat table.ada0
MBR 4
1 freebsd         63 1953525104   [active]

# cat table.ada0s1
BSD 8
1  freebsd-ufs          0    4194304
2 freebsd-swap    4194304   33554432
4  freebsd-ufs   37748736   50331648
5  freebsd-ufs   88080384   41943040
6  freebsd-ufs  130023424  838860800
7  freebsd-ufs  968884224  984640880
```

If at least one sector was unallocated at the end of the disk, these two files can be used without modification.

Now restore the partition table into `mirror/gm0`:

```
# gpart restore mirror/gm0 < table.ada0
# gpart restore mirror/gm0s1 < table.ada0s1
```

Check the partition table with `gpart show`. This example has `gm0s1a` for `/`, `gm0s1d` for `/var`, `gm0s1e` for `/usr`, `gm0s1f` for `/data1`, and `gm0s1g` for `/data2`.

```
# gpart show mirror/gm0
=>        63  1953525104  mirror/gm0  MBR  (931G)
          63  1953525042           1  freebsd  [active]  (931G)
  1953525105          62              - free -  (31k)

# gpart show mirror/gm0s1
=>         0  1953525042  mirror/gm0s1  BSD  (931G)
           0     2097152             1  freebsd-ufs  (1.0G)
     2097152    16777216             2  freebsd-swap  (8.0G)
    18874368    41943040             4  freebsd-ufs  (20G)
    60817408    20971520             5  freebsd-ufs  (10G)
    81788928   629145600             6  freebsd-ufs  (300G)
   710934528  1242590514             7  freebsd-ufs  (592G)
  1953525042          63                - free -  (31k)
```

Both the slice and the last partition should have some free space at the end of each disk.

Create filesystems on these new partitions. The number of partitions will vary, matching the partitions on the original disk, `ada0`.

```
# newfs -U /dev/mirror/gm0s1a
# newfs -U /dev/mirror/gm0s1d
# newfs -U /dev/mirror/gm0s1e
# newfs -U /dev/mirror/gm0s1f
# newfs -U /dev/mirror/gm0s1g
```

Make the mirror bootable by installing bootcode in the MBR and bsdlabel and setting the active slice:

```
# gpart bootcode -b /boot/mbr mirror/gm0
# gpart set -a active -i 1 mirror/gm0
# gpart bootcode -b /boot/boot mirror/gm0s1
```

Adjust `/etc/fstab` to use the new partitions on the mirror. Back up this file first by copying it to `/etc/fstab.orig`.

```
# cp /etc/fstab /etc/fstab.orig
```

Edit `/etc/fstab`, replacing `/dev/ada0` with `mirror/gm0`.

```
# Device                Mountpoint      FStype  Options Dump    Pass#
/dev/mirror/gm0s1a      /               ufs     rw      1       1
/dev/mirror/gm0s1b      none            swap    sw      0       0
/dev/mirror/gm0s1d      /var            ufs     rw      2       2
/dev/mirror/gm0s1e      /usr            ufs     rw      2       2
/dev/mirror/gm0s1f      /data1          ufs     rw      2       2
/dev/mirror/gm0s1g      /data2          ufs     rw      2       2
```

If the gmirror(8) kernel module has not been built into the kernel, edit `/boot/loader.conf` to load it:

```
geom_mirror_load="YES"
```

Filesystems from the original disk can now be copied onto the mirror with dump(8) and restore(8). Note that it may take some time to create a snapshot for each filesystem dumped with `dump -L`.

```
# mount /dev/mirror/gm0s1a /mnt
# dump -C16 -b64 -0aL -f - /    | (cd /mnt && restore -rf -)
# mount /dev/mirror/gm0s1d /mnt/var
# mount /dev/mirror/gm0s1e /mnt/usr
# mount /dev/mirror/gm0s1f /mnt/data1
# mount /dev/mirror/gm0s1g /mnt/data2
# dump -C16 -b64 -0aL -f - /usr | (cd /mnt/usr && restore -rf -)
# dump -C16 -b64 -0aL -f - /var | (cd /mnt/var && restore -rf -)
# dump -C16 -b64 -0aL -f - /data1 | (cd /mnt/data1 && restore -rf -)
# dump -C16 -b64 -0aL -f - /data2 | (cd /mnt/data2 && restore -rf -)
```

Restart the system, booting from `ada1`. If everything is working, the system will boot from `mirror/gm0`, which now contains the same data as `ada0` had previously. See the Troubleshooting section if there are problems booting.

At this point, the mirror still consists of only the single `ada1` disk.

After booting from `mirror/gm0` successfully, the final step is inserting `ada0` into the mirror.

> **Important:** When `ada0` is inserted into the mirror, its former contents will be overwritten by data on the mirror. Make certain that `mirror/gm0` has the same contents as `ada0` before adding `ada0` to the mirror. If there is something wrong with the contents copied by dump(8) and restore(8), revert `/etc/fstab` to mount the filesystems on `ada0`, reboot, and try the whole procedure again.

```
# gmirror insert gm0 ada0
GEOM_MIRROR: Device gm0: rebuilding provider ada0
```

Synchronization between the two disks will start immediately. gmirror(8) `status` shows the progress.

```
# gmirror status
      Name    Status  Components
mirror/gm0  DEGRADED  ada1 (ACTIVE)
                      ada0 (SYNCHRONIZING, 64%)
```

After a while, synchronization will finish.

```
GEOM_MIRROR: Device gm0: rebuilding provider ada0 finished.
# gmirror status
      Name    Status  Components
mirror/gm0  COMPLETE  ada1 (ACTIVE)
                      ada0 (ACTIVE)
```

`mirror/gm0` now consists of the two disks `ada0` and `ada1`, and the contents are automatically synchronized with each other. In use, `mirror/gm0` will behave just like the original single drive.

20.4.4 Troubleshooting

20.4.4.1 Problems with Booting

20.4.4.1.1 BIOS Settings

BIOS settings may have to be changed to boot from one of the new mirrored drives. Either mirror drive can be used for booting, as they contain identical data.

20.4.4.1.2 Boot Problems

If the boot stopped with this message, something is wrong with the mirror device:

```
Mounting from ufs:/dev/mirror/gm0s1a failed with error 19.

Loader variables:
  vfs.root.mountfrom=ufs:/dev/mirror/gm0s1a
  vfs.root.mountfrom.options=rw

Manual root filesystem specification:
  <fstype>:<device> [options]
      Mount <device> using filesystem <fstype>
      and with the specified (optional) option list.

    eg. ufs:/dev/da0s1a
        zfs:tank
        cd9660:/dev/acd0 ro
          (which is equivalent to: mount -t cd9660 -o ro /dev/acd0 /)

  ?               List valid disk boot devices
  .               Yield 1 second (for background tasks)
  <empty line>    Abort manual input

mountroot>
```

Forgetting to load the `geom_mirror` module in `/boot/loader.conf` can cause this problem. To fix it, boot from a FreeBSD 9.0 or later installation media and choose `Shell` at the first prompt. Then load the mirror module and mount the mirror device:

```
# gmirror load
# mount /dev/mirror/gm0s1a /mnt
```

Edit `/mnt/boot/loader.conf`, adding a line to load the mirror module:

```
geom_mirror_load="YES"
```

Save the file and reboot.

Other problems that cause `error 19` require more effort to fix. Enter `ufs:/dev/ada0s1a` at the boot loader prompt. Although the system should boot from `ada0`, another prompt to select a shell appears because `/etc/fstab` is incorrect. Press the Enter key at the prompt. Undo the modifications so far by reverting `/etc/fstab`, mounting filesystems from the original disk (`ada0`) instead of the mirror. Reboot the system and try the procedure again.

```
Enter full pathname of shell or RETURN for /bin/sh:
# cp /etc/fstab.orig /etc/fstab
# reboot
```

20.4.5 Recovering from Disk Failure

The benefit of disk mirroring is that an individual disk can fail without causing the mirror to lose any data. In the above example, if `ada0` fails, the mirror will continue to work, providing data from the remaining working drive, `ada1`.

To replace the failed drive, shut down the system and physically replace the failed drive with a new drive of equal or greater capacity. Manufacturers use somewhat arbitrary values when rating drives in gigabytes, and the only way to really be sure is to compare the total count of sectors shown by `diskinfo -v`. A drive with larger capacity than the mirror will work, although the extra space on the new drive will not be used.

After the computer is powered back up, the mirror will be running in a “degraded” mode with only one drive. The mirror is told to forget drives that are not currently connected:

```
# gmirror forget gm0
```

Any old metadata should be cleared from the replacement disk. Then the disk, `ada4` for this example, is inserted into the mirror:

```
# gmirror insert gm0 /dev/ada4
```

Resynchronization begins when the new drive is inserted into the mirror. This process of copying mirror data to a new drive can take a while. Performance of the mirror will be greatly reduced during the copy, so inserting new drives is best done when there is low demand on the computer.

Progress can be monitored with `gmirror status`, which shows drives that are being synchronized and the percentage of completion. During resynchronization, the status will be `DEGRADED`, changing to `COMPLETE` when the process is finished.

20.5 RAID3 - Byte-level Striping with Dedicated Parity

Written by Mark Gladman and Daniel Gerzo. Based on documentation by Tom Rhodes and Murray Stokely.

RAID3 is a method used to combine several disk drives into a single volume with a dedicated parity disk. In a RAID3 system, data is split up into a number of bytes that are written across all the drives in the array except for one disk which acts as a dedicated parity disk. This means that reading 1024KB from a RAID3 implementation will access all disks in the array. Performance can be enhanced by using multiple disk controllers. The RAID3 array provides a fault tolerance of 1 drive, while providing a capacity of 1 - 1/n times the total capacity of all drives in the array, where n is the number of hard drives in the array. Such a configuration is mostly suitable for storing data of larger sizes such as multimedia files.

At least 3 physical hard drives are required to build a RAID3 array. Each disk must be of the same size, since I/O requests are interleaved to read or write to multiple disks in parallel. Also, due to the nature of RAID3, the number of drives must be equal to 3, 5, 9, 17, and so on, or 2^n + 1.

20.5.1 Creating a Dedicated RAID3 Array

In FreeBSD, support for RAID3 is implemented by the graid3(8) GEOM class. Creating a dedicated RAID3 array on FreeBSD requires the following steps.

> **Note:** While it is theoretically possible to boot from a RAID3 array on FreeBSD, that configuration is uncommon and is not advised.

1. First, load the `geom_raid3.ko` kernel module by issuing the following command:

   ```
   # graid3 load
   ```

   Alternatively, it is possible to manually load the `geom_raid3.ko` module:

   ```
   # kldload geom_raid3.ko
   ```

2. Create or ensure that a suitable mount point exists:

   ```
   # mkdir /multimedia/
   ```

3. Determine the device names for the disks which will be added to the array, and create the new RAID3 device. The final device listed will act as the dedicated parity disk. This example uses three unpartitioned ATA drives: *ada1* and *ada2* for data, and *ada3* for parity.

   ```
   # graid3 label -v gr0 /dev/ada1 /dev/ada2 /dev/ada3
   Metadata value stored on /dev/ada1.
   Metadata value stored on /dev/ada2.
   Metadata value stored on /dev/ada3.
   Done.
   ```

4. Partition the newly created `gr0` device and put a UFS file system on it:

   ```
   # gpart create -s GPT /dev/raid3/gr0
   # gpart add -t freebsd-ufs /dev/raid3/gr0
   # newfs -j /dev/raid3/gr0p1
   ```

   Many numbers will glide across the screen, and after a bit of time, the process will be complete. The volume has been created and is ready to be mounted:

```
# mount /dev/raid3/gr0p1 /multimedia/
```

The RAID3 array is now ready to use.

Additional configuration is needed to retain the above setup across system reboots.

1. The `geom_raid3.ko` module must be loaded before the array can be mounted. To automatically load the kernel module during system initialization, add the following line to `/boot/loader.conf`:

   ```
   geom_raid3_load="YES"
   ```

2. The following volume information must be added to `/etc/fstab` in order to automatically mount the array's file system during the system boot process:

   ```
   /dev/raid3/gr0p1        /multimedia     ufs     rw      2       2
   ```

20.6 GEOM Gate Network Devices

GEOM supports the remote use of devices, such as disks, CD-ROMs, and files through the use of the gate utilities. This is similar to NFS.

To begin, an exports file must be created. This file specifies who is permitted to access the exported resources and what level of access they are offered. For example, to export the fourth slice on the first SCSI disk, the following `/etc/gg.exports` is more than adequate:

```
192.168.1.0/24 RW /dev/da0s4d
```

This allows all hosts inside the specified private network access to the file system on the `da0s4d` partition.

To export this device, ensure it is not currently mounted, and start the ggated(8) server daemon:

```
# ggated
```

To `mount` the device on the client machine, issue the following commands:

```
# ggatec create -o rw 192.168.1.1 /dev/da0s4d
ggate0
# mount /dev/ggate0 /mnt
```

The device may now be accessed through the `/mnt` mount point.

> **Note:** However, this will fail if the device is currently mounted on either the server machine or any other machine on the network.

When the device is no longer needed, unmount it with umount(8), similar to any other disk device.

20.7 Labeling Disk Devices

During system initialization, the FreeBSD kernel creates device nodes as devices are found. This method of probing for devices raises some issues. For instance, what if a new disk device is added via USB? It is likely that a flash device may be handed the device name of `da0` and the original `da0` shifted to `da1`. This will cause issues mounting file systems if they are listed in `/etc/fstab` which may also prevent the system from booting.

One solution is to chain SCSI devices in order so a new device added to the SCSI card will be issued unused device numbers. But what about USB devices which may replace the primary SCSI disk? This happens because USB devices are usually probed before the SCSI card. One solution is to only insert these devices after the system has been booted. Another method is to use only a single ATA drive and never list the SCSI devices in `/etc/fstab`.

A better solution is to use `glabel` to label the disk devices and use the labels in `/etc/fstab`. Because `glabel` stores the label in the last sector of a given provider, the label will remain persistent across reboots. By using this label as a device, the file system may always be mounted regardless of what device node it is accessed through.

> **Note:** `glabel` can create both transient and permanent labels. Only permanent labels are consistent across reboots. Refer to glabel(8) for more information on the differences between labels.

20.7.1 Label Types and Examples

Permanent labels can be a generic or a file system label. Permanent file system labels can be created with tunefs(8) or newfs(8). These types of labels are created in a sub-directory of `/dev`, and will be named according to the file system type. For example, UFS2 file system labels will be created in `/dev/ufs`. Generic permanent labels can be created with `glabel label`. These are not file system specific and will be created in `/dev/label`.

Temporary labels are destroyed at the next reboot. These labels are created in `/dev/label` and are suited to experimentation. A temporary label can be created using `glabel create`.

To create a permanent label for a UFS2 file system without destroying any data, issue the following command:

```
# tunefs -L home /dev/da3
```

> **Warning:** If the file system is full, this may cause data corruption.

A label should now exist in `/dev/ufs` which may be added to `/etc/fstab`:

```
/dev/ufs/home           /home           ufs     rw              2       2
```

> **Note:** The file system must not be mounted while attempting to run `tunefs`.

Now the file system may be mounted:

```
# mount /home
```

From this point on, so long as the `geom_label.ko` kernel module is loaded at boot with `/boot/loader.conf` or the `GEOM_LABEL` kernel option is present, the device node may change without any ill effect on the system.

File systems may also be created with a default label by using the `-L` flag with `newfs`. Refer to newfs(8) for more information.

The following command can be used to destroy the label:

```
# glabel destroy home
```

The following example shows how to label the partitions of a boot disk.

Example 20-1. Labeling Partitions on the Boot Disk

By permanently labeling the partitions on the boot disk, the system should be able to continue to boot normally, even if the disk is moved to another controller or transferred to a different system. For this example, it is assumed that a single ATA disk is used, which is currently recognized by the system as `ad0`. It is also assumed that the standard FreeBSD partition scheme is used, with `/`, `/var`, `/usr` and `/tmp`, as well as a swap partition.

Reboot the system, and at the loader(8) prompt, press **4** to boot into single user mode. Then enter the following commands:

```
# glabel label rootfs /dev/ad0s1a
GEOM_LABEL: Label for provider /dev/ad0s1a is label/rootfs
# glabel label var /dev/ad0s1d
GEOM_LABEL: Label for provider /dev/ad0s1d is label/var
# glabel label usr /dev/ad0s1f
GEOM_LABEL: Label for provider /dev/ad0s1f is label/usr
# glabel label tmp /dev/ad0s1e
GEOM_LABEL: Label for provider /dev/ad0s1e is label/tmp
# glabel label swap /dev/ad0s1b
GEOM_LABEL: Label for provider /dev/ad0s1b is label/swap
# exit
```

The system will continue with multi-user boot. After the boot completes, edit `/etc/fstab` and replace the conventional device names, with their respective labels. The final `/etc/fstab` will look like this:

```
# Device                Mountpoint      FStype  Options         Dump    Pass#
/dev/label/swap         none            swap    sw              0       0
/dev/label/rootfs       /               ufs     rw              1       1
/dev/label/tmp          /tmp            ufs     rw              2       2
/dev/label/usr          /usr            ufs     rw              2       2
/dev/label/var          /var            ufs     rw              2       2
```

The system can now be rebooted. If everything went well, it will come up normally and `mount` will show:

```
# mount
/dev/label/rootfs on / (ufs, local)
devfs on /dev (devfs, local)
/dev/label/tmp on /tmp (ufs, local, soft-updates)
/dev/label/usr on /usr (ufs, local, soft-updates)
/dev/label/var on /var (ufs, local, soft-updates)
```

Starting with FreeBSD 7.2, the glabel(8) class supports a new label type for UFS file systems, based on the unique file system id, `ufsid`. These labels may be found in `/dev/ufsid` and are created automatically during system startup. It is possible to use `ufsid` labels to mount partitions using `/etc/fstab`. Use `glabel status` to receive a list of file systems and their corresponding `ufsid` labels:

```
% glabel status
```

```
                  Name  Status  Components
ufsid/486b6fc38d330916     N/A  ad4s1d
ufsid/486b6fc16926168e     N/A  ad4s1f
```

In the above example, `ad4s1d` represents `/var`, while `ad4s1f` represents `/usr`. Using the `ufsid` values shown, these partitions may now be mounted with the following entries in `/etc/fstab`:

```
/dev/ufsid/486b6fc38d330916        /var        ufs        rw        2      2
/dev/ufsid/486b6fc16926168e        /usr        ufs        rw        2      2
```

Any partitions with `ufsid` labels can be mounted in this way, eliminating the need to manually create permanent labels, while still enjoying the benefits of device name independent mounting.

20.8 UFS Journaling Through GEOM

Beginning with FreeBSD 7.0, support for UFS journals is available. The implementation is provided through the GEOM subsystem and is configured using gjournal(8).

Journaling stores a log of file system transactions, such as changes that make up a complete disk write operation, before meta-data and file writes are committed to the disk. This transaction log can later be replayed to redo file system transactions, preventing file system inconsistencies.

This method provides another mechanism to protect against data loss and inconsistencies of the file system. Unlike Soft Updates, which tracks and enforces meta-data updates, and snapshots, which create an image of the file system, a log is stored in disk space specifically for this task, and in some cases, may be stored on another disk entirely.

Unlike other file system journaling implementations, the `gjournal` method is block based and not implemented as part of the file system. It is a GEOM extension.

To enable support for `gjournal`, the FreeBSD kernel must have the following option which is the default on FreeBSD 7.0 and later:

```
options UFS_GJOURNAL
```

If journaled volumes need to be mounted during startup, the `geom_journal.ko` kernel module needs to be loaded, by adding the following line to `/boot/loader.conf`:

```
geom_journal_load="YES"
```

Alternatively, this function can be built into a custom kernel, by adding the following line in the kernel configuration file:

```
options GEOM_JOURNAL
```

Creating a journal on a free file system may now be done using the following steps. In this example, `da4` is a new SCSI disk:

```
# gjournal load
# gjournal label /dev/da4
```

At this point, there should be a `/dev/da4` device node and a `/dev/da4.journal` device node. A file system may now be created on this device:

```
# newfs -O 2 -J /dev/da4.journal
```

This command will create a UFS2 file system on the journaled device.

`mount` the device at the desired point with:

```
# mount /dev/da4.journal /mnt
```

> **Note:** In the case of several slices, a journal will be created for each individual slice. For instance, if `ad4s1` and `ad4s2` are both slices, then `gjournal` will create `ad4s1.journal` and `ad4s2.journal`.

For better performance, the journal may be kept on another disk. In this configuration, the journal provider or storage device should be listed after the device to enable journaling on. Journaling may also be enabled on current file systems by using `tunefs`. However, *always* make a backup before attempting to alter a file system. In most cases, `gjournal` will fail if it is unable to create the journal, but this does not protect against data loss incurred as a result of misusing `tunefs`.

It is also possible to journal the boot disk of a FreeBSD system. Refer to the article Implementing UFS Journaling on a Desktop PC (http://www.FreeBSD.org/doc/en_US.ISO8859-1/articles/gjournal-desktop) for detailed instructions.

Chapter 21 File Systems Support

Written by Tom Rhodes.

21.1 Synopsis

File systems are an integral part of any operating system. They allow users to upload and store files, provide access to data, and make hard drives useful. Different operating systems differ in their native file system. Traditionally, the native FreeBSD file system has been the Unix File System UFS which has been modernized as UFS2. Since FreeBSD 7.0, the Z File System ZFS is also available as a native file system.

In addition to its native file systems, FreeBSD supports a multitude of other file systems so that data from other operating systems can be accessed locally, such as data stored on locally attached USB storage devices, flash drives, and hard disks. This includes support for the Linux Extended File System (EXT) and the Microsoft New Technology File System (NTFS).

There are different levels of FreeBSD support for the various file systems. Some require a kernel module to be loaded and others may require a toolset to be installed. Some non-native file system support is full read-write while others are read-only.

After reading this chapter, you will know:

- The difference between native and supported file systems.
- Which file systems are supported by FreeBSD.
- How to enable, configure, access, and make use of non-native file systems.

Before reading this chapter, you should:

- Understand UNIX and FreeBSD basics.
- Be familiar with the basics of kernel configuration and compilation.
- Feel comfortable installing software in FreeBSD.
- Have some familiarity with disks, storage, and device names in FreeBSD.

21.2 The Z File System (ZFS)

The Z file system, originally developed by Sun, is designed to use a pooled storage method in that space is only used as it is needed for data storage. It is also designed for maximum data integrity, supporting data snapshots, multiple copies, and data checksums. It uses a software data replication model, known as RAID-Z. RAID-Z provides redundancy similar to hardware RAID, but is designed to prevent data write corruption and to overcome some of the limitations of hardware RAID.

21.2.1 ZFS Tuning

Some of the features provided by ZFS are RAM-intensive, so some tuning may be required to provide maximum efficiency on systems with limited RAM.

21.2.1.1 Memory

At a bare minimum, the total system memory should be at least one gigabyte. The amount of recommended RAM depends upon the size of the pool and the ZFS features which are used. A general rule of thumb is 1GB of RAM for every 1TB of storage. If the deduplication feature is used, a general rule of thumb is 5GB of RAM per TB of storage to be deduplicated. While some users successfully use ZFS with less RAM, it is possible that when the system is under heavy load, it may panic due to memory exhaustion. Further tuning may be required for systems with less than the recommended RAM requirements.

21.2.1.2 Kernel Configuration

Due to the RAM limitations of the i386 platform, users using ZFS on the i386 architecture should add the following option to a custom kernel configuration file, rebuild the kernel, and reboot:

```
options         KVA_PAGES=512
```

This option expands the kernel address space, allowing the `vm.kvm_size` tunable to be pushed beyond the currently imposed limit of 1 GB, or the limit of 2 GB for PAE. To find the most suitable value for this option, divide the desired address space in megabytes by four (4). In this example, it is `512` for 2 GB.

21.2.1.3 Loader Tunables

The `kmem` address space can be increased on all FreeBSD architectures. On a test system with one gigabyte of physical memory, success was achieved with the following options added to `/boot/loader.conf`, and the system restarted:

```
vm.kmem_size="330M"
vm.kmem_size_max="330M"
vfs.zfs.arc_max="40M"
vfs.zfs.vdev.cache.size="5M"
```

For a more detailed list of recommendations for ZFS-related tuning, see http://wiki.freebsd.org/ZFSTuningGuide.

21.2.2 Using ZFS

There is a start up mechanism that allows FreeBSD to mount ZFS pools during system initialization. To set it, issue the following commands:

```
# echo 'zfs_enable="YES"' >> /etc/rc.conf
# service zfs start
```

The examples in this section assume three SCSI disks with the device names *`da0`*, *`da1`*, and *`da2`*. Users of IDE hardware should instead use *`ad`* device names.

21.2.2.1 Single Disk Pool

To create a simple, non-redundant ZFS pool using a single disk device, use `zpool`:

```
# zpool create example /dev/da0
```

To view the new pool, review the output of `df`:

```
# df
Filesystem  1K-blocks    Used    Avail Capacity  Mounted on
/dev/ad0s1a   2026030  235230  1628718    13%    /
devfs               1       1        0   100%    /dev
/dev/ad0s1d  54098308 1032846 48737598     2%    /usr
example      17547136       0 17547136     0%    /example
```

This output shows that the `example` pool has been created and *mounted*. It is now accessible as a file system. Files may be created on it and users can browse it, as seen in the following example:

```
# cd /example
# ls
# touch testfile
# ls -al
total 4
drwxr-xr-x   2 root  wheel    3 Aug 29 23:15 .
drwxr-xr-x  21 root  wheel  512 Aug 29 23:12 ..
-rw-r--r--   1 root  wheel    0 Aug 29 23:15 testfile
```

However, this pool is not taking advantage of any ZFS features. To create a dataset on this pool with compression enabled:

```
# zfs create example/compressed
# zfs set compression=gzip example/compressed
```

The `example/compressed` dataset is now a ZFS compressed file system. Try copying some large files to `/example/compressed`.

Compression can be disabled with:

```
# zfs set compression=off example/compressed
```

To unmount a file system, issue the following command and then verify by using `df`:

```
# zfs umount example/compressed
# df
Filesystem  1K-blocks    Used    Avail Capacity  Mounted on
/dev/ad0s1a   2026030  235232  1628716    13%    /
devfs               1       1        0   100%    /dev
/dev/ad0s1d  54098308 1032864 48737580     2%    /usr
example      17547008       0 17547008     0%    /example
```

To re-mount the file system to make it accessible again, and verify with `df`:

```
# zfs mount example/compressed
# df
Filesystem         1K-blocks    Used    Avail Capacity  Mounted on
/dev/ad0s1a          2026030  235234  1628714    13%    /
devfs                      1       1        0   100%    /dev
/dev/ad0s1d         54098308 1032864 48737580     2%    /usr
example             17547008       0 17547008     0%    /example
example/compressed  17547008       0 17547008     0%    /example/compressed
```

The pool and file system may also be observed by viewing the output from `mount`:

```
# mount
/dev/ad0s1a on / (ufs, local)
devfs on /dev (devfs, local)
/dev/ad0s1d on /usr (ufs, local, soft-updates)
example on /example (zfs, local)
example/data on /example/data (zfs, local)
example/compressed on /example/compressed (zfs, local)
```

ZFS datasets, after creation, may be used like any file systems. However, many other features are available which can be set on a per-dataset basis. In the following example, a new file system, `data` is created. Important files will be stored here, the file system is set to keep two copies of each data block:

```
# zfs create example/data
# zfs set copies=2 example/data
```

It is now possible to see the data and space utilization by issuing `df`:

```
# df
Filesystem         1K-blocks    Used    Avail Capacity  Mounted on
/dev/ad0s1a          2026030  235234  1628714    13%    /
devfs                      1       1        0   100%    /dev
/dev/ad0s1d         54098308 1032864 48737580     2%    /usr
example             17547008       0 17547008     0%    /example
example/compressed  17547008       0 17547008     0%    /example/compressed
example/data        17547008       0 17547008     0%    /example/data
```

Notice that each file system on the pool has the same amount of available space. This is the reason for using `df` in these examples, to show that the file systems use only the amount of space they need and all draw from the same pool. The ZFS file system does away with concepts such as volumes and partitions, and allows for several file systems to occupy the same pool.

To destroy the file systems and then destroy the pool as they are no longer needed:

```
# zfs destroy example/compressed
# zfs destroy example/data
# zpool destroy example
```

21.2.2.2 ZFS RAID-Z

There is no way to prevent a disk from failing. One method of avoiding data loss due to a failed hard disk is to implement RAID. ZFS supports this feature in its pool design.

To create a RAID-Z pool, issue the following command and specify the disks to add to the pool:

```
# zpool create storage raidz da0 da1 da2
```

> **Note:** Sun recommends that the amount of devices used in a RAID-Z configuration is between three and nine. For environments requiring a single pool consisting of 10 disks or more, consider breaking it up into smaller RAID-Z groups. If only two disks are available and redundancy is a requirement, consider using a ZFS mirror. Refer to zpool(8) for more details.

This command creates the `storage` zpool. This may be verified using mount(8) and df(1). This command makes a new file system in the pool called `home`:

```
# zfs create storage/home
```

It is now possible to enable compression and keep extra copies of directories and files using the following commands:

```
# zfs set copies=2 storage/home
# zfs set compression=gzip storage/home
```

To make this the new home directory for users, copy the user data to this directory, and create the appropriate symbolic links:

```
# cp -rp /home/* /storage/home
# rm -rf /home /usr/home
# ln -s /storage/home /home
# ln -s /storage/home /usr/home
```

Users should now have their data stored on the freshly created `/storage/home`. Test by adding a new user and logging in as that user.

Try creating a snapshot which may be rolled back later:

```
# zfs snapshot storage/home@08-30-08
```

Note that the snapshot option will only capture a real file system, not a home directory or a file. The `@` character is a delimiter used between the file system name or the volume name. When a user's home directory gets trashed, restore it with:

```
# zfs rollback storage/home@08-30-08
```

To get a list of all available snapshots, run `ls` in the file system's `.zfs/snapshot` directory. For example, to see the previously taken snapshot:

```
# ls /storage/home/.zfs/snapshot
```

It is possible to write a script to perform regular snapshots on user data. However, over time, snapshots may consume a great deal of disk space. The previous snapshot may be removed using the following command:

```
# zfs destroy storage/home@08-30-08
```

After testing, `/storage/home` can be made the real `/home` using this command:

```
# zfs set mountpoint=/home storage/home
```

Run `df` and `mount` to confirm that the system now treats the file system as the real `/home`:

```
# mount
/dev/ad0s1a on / (ufs, local)
devfs on /dev (devfs, local)
/dev/ad0s1d on /usr (ufs, local, soft-updates)
storage on /storage (zfs, local)
```

```
storage/home on /home (zfs, local)
# df
Filesystem   1K-blocks    Used    Avail Capacity  Mounted on
/dev/ad0s1a    2026030  235240  1628708    13%    /
devfs                1       1        0   100%    /dev
/dev/ad0s1d   54098308 1032826 48737618     2%    /usr
storage       26320512       0 26320512     0%    /storage
storage/home  26320512       0 26320512     0%    /home
```

This completes the RAID-Z configuration. To get status updates about the file systems created during the nightly periodic(8) runs, issue the following command:

```
# echo 'daily_status_zfs_enable="YES"' >> /etc/periodic.conf
```

21.2.2.3 Recovering RAID-Z

Every software RAID has a method of monitoring its `state`. The status of RAID-Z devices may be viewed with the following command:

```
# zpool status -x
```

If all pools are healthy and everything is normal, the following message will be returned:

```
all pools are healthy
```

If there is an issue, perhaps a disk has gone offline, the pool state will look similar to:

```
  pool: storage
 state: DEGRADED
status: One or more devices has been taken offline by the administrator.
       Sufficient replicas exist for the pool to continue functioning in a
       degraded state.
action: Online the device using 'zpool online' or replace the device with
       'zpool replace'.
 scrub: none requested
config:

       NAME        STATE     READ WRITE CKSUM
       storage     DEGRADED     0     0     0
         raidz1    DEGRADED     0     0     0
           da0     ONLINE       0     0     0
           da1     OFFLINE      0     0     0
           da2     ONLINE       0     0     0

errors: No known data errors
```

This indicates that the device was previously taken offline by the administrator using the following command:

```
# zpool offline storage da1
```

It is now possible to replace `da1` after the system has been powered down. When the system is back online, the following command may issued to replace the disk:

```
# zpool replace storage da1
```

From here, the status may be checked again, this time without the `-x` flag to get state information:

```
# zpool status storage
 pool: storage
 state: ONLINE
 scrub: resilver completed with 0 errors on Sat Aug 30 19:44:11 2008
config:

       NAME        STATE     READ WRITE CKSUM
       storage     ONLINE       0     0     0
         raidz1    ONLINE       0     0     0
           da0     ONLINE       0     0     0
           da1     ONLINE       0     0     0
           da2     ONLINE       0     0     0

errors: No known data errors
```

As shown from this example, everything appears to be normal.

21.2.2.4 Data Verification

ZFS uses checksums to verify the integrity of stored data. These are enabled automatically upon creation of file systems and may be disabled using the following command:

```
# zfs set checksum=off storage/home
```

Doing so is *not* recommended as checksums take very little storage space and are used to check data integrity using checksum verification in a process is known as "scrubbing." To verify the data integrity of the `storage` pool, issue this command:

```
# zpool scrub storage
```

This process may take considerable time depending on the amount of data stored. It is also very I/O intensive, so much so that only one scrub may be run at any given time. After the scrub has completed, the status is updated and may be viewed by issuing a status request:

```
# zpool status storage
 pool: storage
 state: ONLINE
 scrub: scrub completed with 0 errors on Sat Jan 26 19:57:37 2013
config:

       NAME        STATE     READ WRITE CKSUM
       storage     ONLINE       0     0     0
         raidz1    ONLINE       0     0     0
           da0     ONLINE       0     0     0
           da1     ONLINE       0     0     0
           da2     ONLINE       0     0     0

errors: No known data errors
```

The completion time is displayed and helps to ensure data integrity over a long period of time.

Refer to zfs(8) and zpool(8) for other ZFS options.

21.2.2.5 ZFS Quotas

ZFS supports different types of quotas: the refquota, the general quota, the user quota, and the group quota. This section explains the basics of each type and includes some usage instructions.

Quotas limit the amount of space that a dataset and its descendants can consume, and enforce a limit on the amount of space used by filesystems and snapshots for the descendants. Quotas are useful to limit the amount of space a particular user can use.

> **Note:** Quotas cannot be set on volumes, as the `volsize` property acts as an implicit quota.

The `refquota=`*`size`* limits the amount of space a dataset can consume by enforcing a hard limit on the space used. However, this hard limit does not include space used by descendants, such as file systems or snapshots.

To enforce a general quota of 10 GB for `storage/home/bob`, use the following:

```
# zfs set quota=10G storage/home/bob
```

User quotas limit the amount of space that can be used by the specified user. The general format is `userquota@`*`user=size`*, and the user's name must be in one of the following formats:

- POSIX compatible name such as *`joe`*.
- POSIX numeric ID such as *`789`*.
- SID name such as *`joe.bloggs@example.com`*.
- SID numeric ID such as *`S-1-123-456-789`*.

For example, to enforce a quota of 50 GB for a user named *`joe`*, use the following:

```
# zfs set userquota@joe=50G
```

To remove the quota or make sure that one is not set, instead use:

```
# zfs set userquota@joe=none
```

User quota properties are not displayed by `zfs get all`. Non-`root` users can only see their own quotas unless they have been granted the `userquota` privilege. Users with this privilege are able to view and set everyone's quota.

The group quota limits the amount of space that a specified group can consume. The general format is `groupquota@`*`group=size`*.

To set the quota for the group *`firstgroup`* to 50 GB, use:

```
# zfs set groupquota@firstgroup=50G
```

To remove the quota for the group *`firstgroup`*, or to make sure that one is not set, instead use:

```
# zfs set groupquota@firstgroup=none
```

As with the user quota property, non-`root` users can only see the quotas associated with the groups that they belong to. However, `root` or a user with the `groupquota` privilege can view and set all quotas for all groups.

To display the amount of space consumed by each user on the specified filesystem or snapshot, along with any specified quotas, use `zfs userspace`. For group information, use `zfs groupspace`. For more information about supported options or how to display only specific options, refer to zfs(1).

Users with sufficient privileges and `root` can list the quota for `storage/home/bob` using:

```
# zfs get quota storage/home/bob
```

21.2.2.6 ZFS Reservations

ZFS supports two types of space reservations. This section explains the basics of each and includes some usage instructions.

The `reservation` property makes it possible to reserve a minimum amount of space guaranteed for a dataset and its descendants. This means that if a 10 GB reservation is set on `storage/home/bob`, if disk space gets low, at least 10 GB of space is reserved for this dataset. The `refreservation` property sets or indicates the minimum amount of space guaranteed to a dataset excluding descendants, such as snapshots. As an example, if a snapshot was taken of `storage/home/bob`, enough disk space would have to exist outside of the `refreservation` amount for the operation to succeed because descendants of the main data set are not counted by the `refreservation` amount and so do not encroach on the space set.

Reservations of any sort are useful in many situations, such as planning and testing the suitability of disk space allocation in a new system, or ensuring that enough space is available on file systems for system recovery procedures and files.

The general format of the `reservation` property is `reservation=`*`size`*, so to set a reservation of 10 GB on `storage/home/bob`, use:

```
# zfs set reservation=10G storage/home/bob
```

To make sure that no reservation is set, or to remove a reservation, use:

```
# zfs set reservation=none storage/home/bob
```

The same principle can be applied to the `refreservation` property for setting a refreservation, with the general format `refreservation=`*`size`*.

To check if any reservations or refreservations exist on `storage/home/bob`, execute one of the following commands:

```
# zfs get reservation storage/home/bob
# zfs get refreservation storage/home/bob
```

21.3 Linux Filesystems

This section describes some of the Linux filesystems supported by FreeBSD.

21.3.1 ext2

The ext2fs(5) file system kernel implementation has been available since FreeBSD 2.2. In FreeBSD 8.x and earlier, the code is licensed under the GPL. Since FreeBSD 9.0, the code has been rewritten and is now BSD licensed.

The ext2fs(5) driver allows the FreeBSD kernel to both read and write to ext2 file systems.

To access an ext2 file system, first load the kernel loadable module:

```
# kldload ext2fs
```

Then, to mount an ext2fs(5) volume located on `/dev/ad1s1`:

```
# mount -t ext2fs /dev/ad1s1 /mnt
```

21.3.2 XFS

XFS was originally written by SGI for the IRIX operating system and was then ported to Linux and released under the GPL. See this page (http://oss.sgi.com/projects/xfs) for more details. The FreeBSD port was started by Russel Cattelan, Alexander Kabaev <`kan@FreeBSD.org`>, and Craig Rodrigues <`rodrigc@FreeBSD.org`>.

To load XFS as a kernel-loadable module:

```
# kldload xfs
```

The xfs(5) driver lets the FreeBSD kernel access XFS filesystems. However, only read-only access is supported and writing to a volume is not possible.

To mount a xfs(5) volume located on `/dev/ad1s1`:

```
# mount -t xfs /dev/ad1s1 /mnt
```

The `sysutils/xfsprogs` port includes the `mkfs.xfs` which enables the creation of XFS filesystems, plus utilities for analyzing and repairing them.

The `-p` flag to `mkfs.xfs` can be used to create an xfs(5) filesystem which is populated with files and other metadata. This can be used to quickly create a read-only filesystem which can be tested on FreeBSD.

21.3.3 ReiserFS

The Reiser file system, ReiserFS, was ported to FreeBSD by Jean-Sébastien Pédron <`dumbbell@FreeBSD.org`>, and has been released under the GPL .

The ReiserFS driver permits the FreeBSD kernel to access ReiserFS file systems and read their contents, but not write to them.

First, the kernel-loadable module needs to be loaded:

```
# kldload reiserfs
```

Then, to mount a ReiserFS volume located on `/dev/ad1s1`:

```
# mount -t reiserfs /dev/ad1s1 /mnt
```

Chapter 22 The `vinum` Volume Manager

Originally written by Greg Lehey.

22.1 Synopsis

No matter the type of disks, there are always potential problems. The disks can be too small, too slow, or too unreliable to meet the system's requirements. While disks are getting bigger, so are data storage requirements. Often a file system is needed that is bigger than a disk's capacity. Various solutions to these problems have been proposed and implemented.

One method is through the use of multiple, and sometimes redundant, disks. In addition to supporting various cards and controllers for hardware Redundant Array of Independent Disks RAID systems, the base FreeBSD system includes the `vinum` volume manager, a block device driver that implements virtual disk drives and addresses these three problems. `vinum` provides more flexibility, performance, and reliability than traditional disk storage and implements RAID-0, RAID-1, and RAID-5 models, both individually and in combination.

This chapter provides an overview of potential problems with traditional disk storage, and an introduction to the `vinum` volume manager.

> **Note:** Starting with FreeBSD 5, `vinum` has been rewritten in order to fit into the GEOM architecture, while retaining the original ideas, terminology, and on-disk metadata. This rewrite is called *gvinum* (for *GEOM vinum*). While this chapter uses the term `vinum`, any command invocations should be performed with `gvinum`. The name of the kernel module has changed from the original `vinum.ko` to `geom_vinum.ko`, and all device nodes reside under `/dev/gvinum` instead of `/dev/vinum`. As of FreeBSD 6, the original `vinum` implementation is no longer available in the code base.

22.2 Access Bottlenecks

Modern systems frequently need to access data in a highly concurrent manner. For example, large FTP or HTTP servers can maintain thousands of concurrent sessions and have multiple 100 Mbit/s connections to the outside world, well beyond the sustained transfer rate of most disks.

Current disk drives can transfer data sequentially at up to 70 MB/s, but this value is of little importance in an environment where many independent processes access a drive, and where they may achieve only a fraction of these values. In such cases, it is more interesting to view the problem from the viewpoint of the disk subsystem. The important parameter is the load that a transfer places on the subsystem, or the time for which a transfer occupies the drives involved in the transfer.

In any disk transfer, the drive must first position the heads, wait for the first sector to pass under the read head, and then perform the transfer. These actions can be considered to be atomic as it does not make any sense to interrupt them.

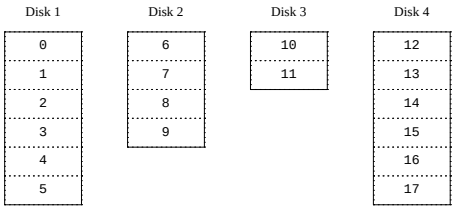
Consider a typical transfer of about 10 kB: the current generation of high-performance disks can position the heads in an average of 3.5 ms. The fastest drives spin at 15,000 rpm, so the average rotational latency (half a revolution) is 2 ms. At 70 MB/s, the transfer itself takes about 150 μs, almost nothing compared to the positioning time. In such a case, the effective transfer rate drops to a little over 1 MB/s and is clearly highly dependent on the transfer size.

The traditional and obvious solution to this bottleneck is “more spindles”: rather than using one large disk, use several smaller disks with the same aggregate storage space. Each disk is capable of positioning and transferring independently, so the effective throughput increases by a factor close to the number of disks used.

The actual throughput improvement is smaller than the number of disks involved. Although each drive is capable of transferring in parallel, there is no way to ensure that the requests are evenly distributed across the drives. Inevitably the load on one drive will be higher than on another.

The evenness of the load on the disks is strongly dependent on the way the data is shared across the drives. In the following discussion, it is convenient to think of the disk storage as a large number of data sectors which are addressable by number, rather like the pages in a book. The most obvious method is to divide the virtual disk into groups of consecutive sectors the size of the individual physical disks and store them in this manner, rather like taking a large book and tearing it into smaller sections. This method is called *concatenation* and has the advantage that the disks are not required to have any specific size relationships. It works well when the access to the virtual disk is spread evenly about its address space. When access is concentrated on a smaller area, the improvement is less marked. Figure 22-1 illustrates the sequence in which storage units are allocated in a concatenated organization.

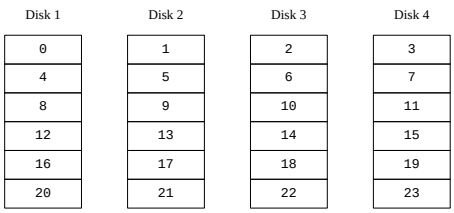
Figure 22-1. Concatenated Organization



An alternative mapping is to divide the address space into smaller, equal-sized components and store them sequentially on different devices. For example, the first 256 sectors may be stored on the first disk, the next 256 sectors on the next disk and so on. After filling the last disk, the process repeats until the disks are full. This mapping is called *striping* or RAID-0.

RAID offers various forms of fault tolerance, though RAID-0 is somewhat misleading as it provides no redundancy. Striping requires somewhat more effort to locate the data, and it can cause additional I/O load where a transfer is spread over multiple disks, but it can also provide a more constant load across the disks. Figure 22-2 illustrates the sequence in which storage units are allocated in a striped organization.

Figure 22-2. Striped Organization



22.3 Data Integrity

The final problem with disks is that they are unreliable. Although reliability has increased tremendously over the last few years, disk drives are still the most likely core component of a server to fail. When they do, the results can be catastrophic and replacing a failed disk drive and restoring data can result in server downtime.

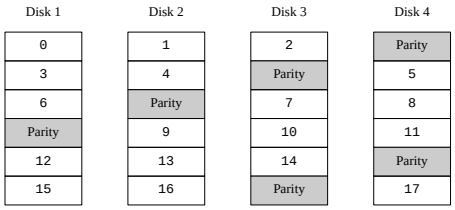
One approach to this problem is *mirroring*, or RAID-1, which keeps two copies of the data on different physical hardware. Any write to the volume writes to both disks; a read can be satisfied from either, so if one drive fails, the data is still available on the other drive.

Mirroring has two problems:

- It requires twice as much disk storage as a non-redundant solution.
- Writes must be performed to both drives, so they take up twice the bandwidth of a non-mirrored volume. Reads do not suffer from a performance penalty and can even be faster.

An alternative solution is *parity*, implemented in RAID levels 2, 3, 4 and 5. Of these, RAID-5 is the most interesting. As implemented in `vinum`, it is a variant on a striped organization which dedicates one block of each stripe to parity one of the other blocks. As implemented by `vinum`, a RAID-5 plex is similar to a striped plex, except that it implements RAID-5 by including a parity block in each stripe. As required by RAID-5, the location of this parity block changes from one stripe to the next. The numbers in the data blocks indicate the relative block numbers.

Figure 22-3. RAID-5 Organization



Compared to mirroring, RAID-5 has the advantage of requiring significantly less storage space. Read access is similar to that of striped organizations, but write access is significantly slower, approximately 25% of the read performance. If one drive fails, the array can continue to operate in degraded mode where a read from one of the remaining accessible drives continues normally, but a read from the failed drive is recalculated from the corresponding block from all the remaining drives.

22.4 vinum Objects

In order to address these problems, vinum implements a four-level hierarchy of objects:

- The most visible object is the virtual disk, called a *volume*. Volumes have essentially the same properties as a UNIX disk drive, though there are some minor differences. For one, they have no size limitations.
- Volumes are composed of *plexes*, each of which represent the total address space of a volume. This level in the hierarchy provides redundancy. Think of plexes as individual disks in a mirrored array, each containing the same data.
- Since vinum exists within the UNIX disk storage framework, it would be possible to use UNIX partitions as the building block for multi-disk plexes. In fact, this turns out to be too inflexible as UNIX disks can have only a limited number of partitions. Instead, vinum subdivides a single UNIX partition, the *drive*, into contiguous areas called *subdisks*, which are used as building blocks for plexes.
- Subdisks reside on vinum *drives*, currently UNIX partitions. vinum drives can contain any number of subdisks. With the exception of a small area at the beginning of the drive, which is used for storing configuration and state information, the entire drive is available for data storage.

The following sections describe the way these objects provide the functionality required of vinum.

22.4.1 Volume Size Considerations

Plexes can include multiple subdisks spread over all drives in the vinum configuration. As a result, the size of an individual drive does not limit the size of a plex or a volume.

22.4.2 Redundant Data Storage

vinum implements mirroring by attaching multiple plexes to a volume. Each plex is a representation of the data in a volume. A volume may contain between one and eight plexes.

Although a plex represents the complete data of a volume, it is possible for parts of the representation to be physically missing, either by design (by not defining a subdisk for parts of the plex) or by accident (as a result of the failure of a drive). As long as at least one plex can provide the data for the complete address range of the volume, the volume is fully functional.

22.4.3 Which Plex Organization?

vinum implements both concatenation and striping at the plex level:

- A *concatenated plex* uses the address space of each subdisk in turn. Concatenated plexes are the most flexible as they can contain any number of subdisks, and the subdisks may be of different length. The plex may be extended by adding additional subdisks. They require less CPU time than striped plexes, though the difference in CPU overhead is not measurable. On the other hand, they are most susceptible to hot spots, where one disk is very active and others are idle.
- A *striped plex* stripes the data across each subdisk. The subdisks must all be the same size and there must be at least two subdisks in order to distinguish it from a concatenated plex. The greatest advantage of striped plexes is

that they reduce hot spots. By choosing an optimum sized stripe, about 256 kB, the load can be evened out on the component drives. Extending a plex by adding new subdisks is so complicated that `vinum` does not implement it.

Table 22-1 summarizes the advantages and disadvantages of each plex organization.

Table 22-1. `vinum` Plex Organizations

Plex type	Minimum subdisks	Can add subdisks	Must be equal size	Application
concatenated	1	yes	no	Large data storage with maximum placement flexibility and moderate performance
striped	2	no	yes	High performance in combination with highly concurrent access

22.5 Some Examples

`vinum` maintains a *configuration database* which describes the objects known to an individual system. Initially, the user creates the configuration database from one or more configuration files using gvinum(8). `vinum` stores a copy of its configuration database on each disk *device* under its control. This database is updated on each state change, so that a restart accurately restores the state of each `vinum` object.

22.5.1 The Configuration File

The configuration file describes individual `vinum` objects. The definition of a simple volume might be:

```
drive a device /dev/da3h
volume myvol
  plex org concat
    sd length 512m drive a
```

This file describes four `vinum` objects:

- The *drive* line describes a disk partition (*drive*) and its location relative to the underlying hardware. It is given the symbolic name *a*. This separation of symbolic names from device names allows disks to be moved from one location to another without confusion.
- The *volume* line describes a volume. The only required attribute is the name, in this case *myvol*.
- The *plex* line defines a plex. The only required parameter is the organization, in this case *concat*. No name is necessary as the system automatically generates a name from the volume name by adding the suffix *.px*, where *x* is the number of the plex in the volume. Thus this plex will be called *myvol.p0*.
- The *sd* line describes a subdisk. The minimum specifications are the name of a drive on which to store it, and the length of the subdisk. No name is necessary as the system automatically assigns names derived from the plex

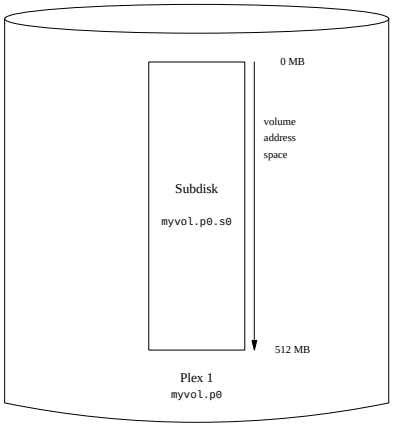
name by adding the suffix *.sx*, where *x* is the number of the subdisk in the plex. Thus `vinum` gives this subdisk the name *myvol.p0.s0*.

After processing this file, gvinum(8) produces the following output:

```
# gvinum -> create config1
Configuration summary
Drives:         1 (4 configured)
Volumes:        1 (4 configured)
Plexes:         1 (8 configured)
Subdisks:       1 (16 configured)

 D a                     State: up       Device /dev/da3h      Avail: 2061/2573 MB (80%)

 V myvol                 State: up       Plexes:       1 Size:      512 MB

 P myvol.p0            C State: up       Subdisks:     1 Size:      512 MB

 S myvol.p0.s0           State: up       PO:        0  B Size:      512 MB
```

This output shows the brief listing format of gvinum(8). It is represented graphically in Figure 22-4.

Figure 22-4. A Simple `vinum` Volume



This figure, and the ones which follow, represent a volume, which contains the plexes, which in turn contains the subdisks. In this example, the volume contains one plex, and the plex contains one subdisk.

This particular volume has no specific advantage over a conventional disk partition. It contains a single plex, so it is not redundant. The plex contains a single subdisk, so there is no difference in storage allocation from a conventional disk partition. The following sections illustrate various more interesting configuration methods.

22.5.2 Increased Resilience: Mirroring

The resilience of a volume can be increased by mirroring. When laying out a mirrored volume, it is important to ensure that the subdisks of each plex are on different drives, so that a drive failure will not take down both plexes. The following configuration mirrors a volume:

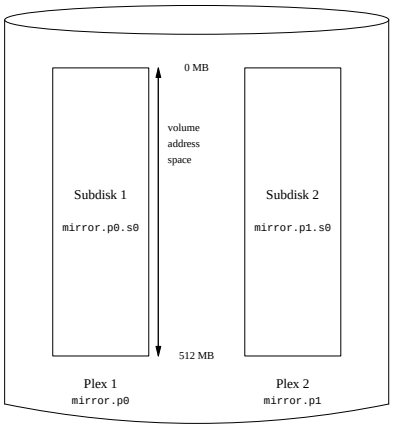
```
drive b device /dev/da4h
volume mirror
plex org concat
  sd length 512m drive a
   plex org concat
     sd length 512m drive b
```

In this example, it was not necessary to specify a definition of drive *a* again, since `vinum` keeps track of all objects in its configuration database. After processing this definition, the configuration looks like:

```
Drives:         2 (4 configured)
Volumes:        2 (4 configured)
Plexes:         3 (8 configured)
Subdisks:       3 (16 configured)

D a                     State: up       Device /dev/da3h       Avail: 1549/2573 MB (60%)
D b                     State: up       Device /dev/da4h       Avail: 2061/2573 MB (80%)

V myvol                 State: up       Plexes:       1 Size:        512 MB
V mirror                State: up       Plexes:       2 Size:        512 MB

P myvol.p0            C State: up       Subdisks:     1 Size:        512 MB
P mirror.p0           C State: up       Subdisks:     1 Size:        512 MB
P mirror.p1           C State: initializing     Subdisks:     1 Size:        512 MB

S myvol.p0.s0           State: up       PO:        0  B Size:        512 MB
   S mirror.p0.s0          State: up       PO:        0  B Size:        512 MB
   S mirror.p1.s0          State: empty    PO:        0  B Size:        512 MB
```

Figure 22-5 shows the structure graphically.

Figure 22-5. A Mirrored `vinum` Volume



In this example, each plex contains the full 512 MB of address space. As in the previous example, each plex contains only a single subdisk.

22.5.3 Optimizing Performance

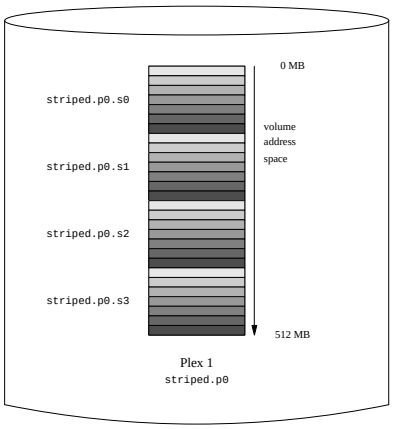
The mirrored volume in the previous example is more resistant to failure than an unmirrored volume, but its performance is less as each write to the volume requires a write to both drives, using up a greater proportion of the total disk bandwidth. Performance considerations demand a different approach: instead of mirroring, the data is striped across as many disk drives as possible. The following configuration shows a volume with a plex striped across four disk drives:

```
 drive c device /dev/da5h
drive d device /dev/da6h
volume stripe
plex org striped 512k
  sd length 128m drive a
  sd length 128m drive b
  sd length 128m drive c
  sd length 128m drive d
```

As before, it is not necessary to define the drives which are already known to `vinum`. After processing this definition, the configuration looks like:

```
   Drives:         4 (4 configured)
   Volumes:        3 (4 configured)
   Plexes:         4 (8 configured)
   Subdisks:       7 (16 configured)

D a                     State: up       Device /dev/da3h        Avail: 1421/2573 MB (55%)
D b                     State: up       Device /dev/da4h        Avail: 1933/2573 MB (75%)
D c                     State: up       Device /dev/da5h        Avail: 2445/2573 MB (95%)
D d                     State: up       Device /dev/da6h        Avail: 2445/2573 MB (95%)

V myvol                 State: up       Plexes:       1 Size:        512 MB
V mirror                State: up       Plexes:       2 Size:        512 MB
V striped               State: up       Plexes:       1 Size:        512 MB

P myvol.p0            C State: up       Subdisks:     1 Size:        512 MB
P mirror.p0           C State: up       Subdisks:     1 Size:        512 MB
P mirror.p1           C State: initializing     Subdisks:     1 Size:        512 MB
P striped.p1            State: up       Subdisks:     1 Size:        512 MB

S myvol.p0.s0           State: up       PO:        0  B Size:        512 MB
S mirror.p0.s0          State: up       PO:        0  B Size:        512 MB
S mirror.p1.s0          State: empty    PO:        0  B Size:        512 MB
S striped.p0.s0         State: up       PO:        0  B Size:        128 MB
S striped.p0.s1         State: up       PO:      512 kB Size:        128 MB
S striped.p0.s2         State: up       PO:     1024 kB Size:        128 MB
S striped.p0.s3         State: up       PO:     1536 kB Size:        128 MB
```

Figure 22-6. A Striped vinum Volume



This volume is represented in Figure 22-6. The darkness of the stripes indicates the position within the plex address space, where the lightest stripes come first and the darkest last.

22.5.4 Resilience and Performance

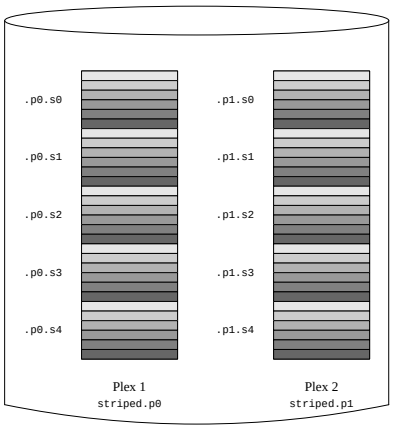
With sufficient hardware, it is possible to build volumes which show both increased resilience and increased performance compared to standard UNIX partitions. A typical configuration file might be:

```
  volume raid10
plex org striped 512k
  sd length 102480k drive a
  sd length 102480k drive b
  sd length 102480k drive c
  sd length 102480k drive d
  sd length 102480k drive e
plex org striped 512k
  sd length 102480k drive c
  sd length 102480k drive d
  sd length 102480k drive e
  sd length 102480k drive a
  sd length 102480k drive b
```

The subdisks of the second plex are offset by two drives from those of the first plex. This helps to ensure that writes do not go to the same subdisks even if a transfer goes over two drives.

Figure 22-7 represents the structure of this volume.

Figure 22-7. A Mirrored, Striped vinum Volume



22.6 Object Naming

vinum assigns default names to plexes and subdisks, although they may be overridden. Overriding the default names is not recommended as it does not bring a significant advantage and it can cause confusion.

Names may contain any non-blank character, but it is recommended to restrict them to letters, digits and the underscore characters. The names of volumes, plexes, and subdisks may be up to 64 characters long, and the names of drives may be up to 32 characters long.

vinum objects are assigned device nodes in the hierarchy /dev/gvinum. The configuration shown above would cause vinum to create the following device nodes:

- Device entries for each volume. These are the main devices used by vinum. The configuration above would include the devices /dev/gvinum/myvol, /dev/gvinum/mirror, /dev/gvinum/striped, /dev/gvinum/raid5 and /dev/gvinum/raid10.
- All volumes get direct entries under /dev/gvinum/.
- The directories /dev/gvinum/plex, and /dev/gvinum/sd, which contain device nodes for each plex and for each subdisk, respectively.

For example, consider the following configuration file:

```
      drive drive1 device /dev/sd1h
     drive drive2 device /dev/sd2h
     drive drive3 device /dev/sd3h
     drive drive4 device /dev/sd4h
volume s64 setupstate
  plex org striped 64k
    sd length 100m drive drive1
```

```
        sd length 100m drive drive2
        sd length 100m drive drive3
        sd length 100m drive drive4
```

After processing this file, gvinum(8) creates the following structure in `/dev/gvinum`:

```
        drwxr-xr-x  2 root  wheel       512 Apr 13
16:46 plex
       crwxr-xr--  1 root  wheel   91,   2 Apr 13 16:46 s64
       drwxr-xr-x  2 root  wheel       512 Apr 13 16:46 sd

    /dev/vinum/plex:
    total 0
    crwxr-xr--  1 root  wheel   25, 0x10000002 Apr 13 16:46 s64.p0

    /dev/vinum/sd:
    total 0
    crwxr-xr--  1 root  wheel   91, 0x20000002 Apr 13 16:46 s64.p0.s0
    crwxr-xr--  1 root  wheel   91, 0x20100002 Apr 13 16:46 s64.p0.s1
    crwxr-xr--  1 root  wheel   91, 0x20200002 Apr 13 16:46 s64.p0.s2
    crwxr-xr--  1 root  wheel   91, 0x20300002 Apr 13 16:46 s64.p0.s3
```

Although it is recommended that plexes and subdisks should not be allocated specific names, `vinum` drives must be named. This makes it possible to move a drive to a different location and still recognize it automatically. Drive names may be up to 32 characters long.

22.6.1 Creating File Systems

Volumes appear to the system to be identical to disks, with one exception. Unlike UNIX drives, `vinum` does not partition volumes, which thus do not contain a partition table. This has required modification to some disk utilities, notably newfs(8), so that it does not try to interpret the last letter of a `vinum` volume name as a partition identifier. For example, a disk drive may have a name like `/dev/ad0a` or `/dev/da2h`. These names represent the first partition (`a`) on the first (0) IDE disk (`ad`) and the eighth partition (`h`) on the third (2) SCSI disk (`da`) respectively. By contrast, a `vinum` volume might be called `/dev/gvinum/concat`, which has no relationship with a partition name.

In order to create a file system on this volume, use newfs(8):

```
# newfs /dev/gvinum/concat
```

22.7 Configuring `vinum`

The `GENERIC` kernel does not contain `vinum`. It is possible to build a custom kernel which includes `vinum`, but this is not recommended. The standard way to start `vinum` is as a kernel module. kldload(8) is not needed because when gvinum(8) starts, it checks whether the module has been loaded, and if it is not, it loads it automatically.

22.7.1 Startup

vinum stores configuration information on the disk slices in essentially the same form as in the configuration files. When reading from the configuration database, vinum recognizes a number of keywords which are not allowed in the configuration files. For example, a disk configuration might contain the following text:

```
volume myvol state up
volume bigraid state down
plex name myvol.p0 state up org concat vol myvol
plex name myvol.p1 state up org concat vol myvol
plex name myvol.p2 state init org striped 512b vol myvol
plex name bigraid.p0 state initializing org raid5 512b vol bigraid
sd name myvol.p0.s0 drive a plex myvol.p0 state up len 1048576b driveoffset 265b plexoffset 0b
sd name myvol.p0.s1 drive b plex myvol.p0 state up len 1048576b driveoffset 265b plexoffset 1048576b
sd name myvol.p1.s0 drive c plex myvol.p1 state up len 1048576b driveoffset 265b plexoffset 0b
sd name myvol.p1.s1 drive d plex myvol.p1 state up len 1048576b driveoffset 265b plexoffset 1048576b
sd name myvol.p2.s0 drive a plex myvol.p2 state init len 524288b driveoffset 1048841b plexoffset 0b
sd name myvol.p2.s1 drive b plex myvol.p2 state init len 524288b driveoffset 1048841b plexoffset 524288b
sd name myvol.p2.s2 drive c plex myvol.p2 state init len 524288b driveoffset 1048841b plexoffset 1048576b
sd name myvol.p2.s3 drive d plex myvol.p2 state init len 524288b driveoffset 1048841b plexoffset 1572864b
sd name bigraid.p0.s0 drive a plex bigraid.p0 state initializing len 4194304b driveoff set 1573129b plexoffset 0b
sd name bigraid.p0.s1 drive b plex bigraid.p0 state initializing len 4194304b driveoff set 1573129b plexoffset 4194304b
sd name bigraid.p0.s2 drive c plex bigraid.p0 state initializing len 4194304b driveoff set 1573129b plexoffset 8388608b
sd name bigraid.p0.s3 drive d plex bigraid.p0 state initializing len 4194304b driveoff set 1573129b plexoffset 12582912b
sd name bigraid.p0.s4 drive e plex bigraid.p0 state initializing len 4194304b driveoff set 1573129b plexoffset 16777216b
```

The obvious differences here are the presence of explicit location information and naming, both of which are allowed but discouraged, and the information on the states. vinum does not store information about drives in the configuration information. It finds the drives by scanning the configured disk drives for partitions with a vinum label. This enables vinum to identify drives correctly even if they have been assigned different UNIX drive IDs.

22.7.1.1 Automatic Startup

Gvinum always features an automatic startup once the kernel module is loaded, via loader.conf(5). To load the *Gvinum* module at boot time, add `geom_vinum_load="YES"` to `/boot/loader.conf`.

When vinum is started with `gvinum start`, vinum reads the configuration database from one of the vinum drives. Under normal circumstances, each drive contains an identical copy of the configuration database, so it does not matter which drive is read. After a crash, however, vinum must determine which drive was updated most recently and read the configuration from this drive. It then updates the configuration, if necessary, from progressively older drives.

22.8 Using vinum for the Root File System

For a machine that has fully-mirrored file systems using vinum, it is desirable to also mirror the root file system. Setting up such a configuration is less trivial than mirroring an arbitrary file system because:

- The root file system must be available very early during the boot process, so the vinum infrastructure must already be available at this time.
- The volume containing the root file system also contains the system bootstrap and the kernel. These must be read using the host system's native utilities, such as the BIOS, which often cannot be taught about the details of vinum.

In the following sections, the term "root volume" is generally used to describe the vinum volume that contains the root file system.

22.8.1 Starting up vinum Early Enough for the Root File System

vinum must be available early in the system boot as loader(8) must be able to load the vinum kernel module before starting the kernel. This can be accomplished by putting this line in /boot/loader.conf:

```
geom_vinum_load="YES"
```

22.8.2 Making a vinum-based Root Volume Accessible to the Bootstrap

The current FreeBSD bootstrap is only 7.5 KB of code and does not understand the internal vinum structures. This means that it cannot parse the vinum configuration data or figure out the elements of a boot volume. Thus, some workarounds are necessary to provide the bootstrap code with the illusion of a standard a partition that contains the root file system.

For this to be possible, the following requirements must be met for the root volume:

- The root volume must not be a stripe or RAID-5.
- The root volume must not contain more than one concatenated subdisk per plex.

Note that it is desirable and possible to use multiple plexes, each containing one replica of the root file system. The bootstrap process will only use one replica for finding the bootstrap and all boot files, until the kernel mounts the root file system. Each single subdisk within these plexes needs its own a partition illusion, for the respective device to be bootable. It is not strictly needed that each of these faked a partitions is located at the same offset within its device, compared with other devices containing plexes of the root volume. However, it is probably a good idea to create the vinum volumes that way so the resulting mirrored devices are symmetric, to avoid confusion.

In order to set up these a partitions for each device containing part of the root volume, the following is required:

1. The location, offset from the beginning of the device, and size of this device's subdisk that is part of the root volume needs to be examined, using the command:

   ```
   # gvinum l -rv root
   ```

   vinum offsets and sizes are measured in bytes. They must be divided by 512 in order to obtain the block numbers that are to be used by bsdlabel.

2. Run this command for each device that participates in the root volume:

   ```
   # bsdlabel -e devname
   ```

   devname must be either the name of the disk, like da0 for disks without a slice table, or the name of the slice, like ad0s1.

   If there is already an a partition on the device from a pre-vinum root file system, it should be renamed to something else so that it remains accessible (just in case), but will no longer be used by default to bootstrap the system. A currently mounted root file system cannot be renamed, so this must be executed either when being booted from a "Fixit" media, or in a two-step process where, in a mirror, the disk that is not been currently booted is manipulated first.

   The offset of the vinum partition on this device (if any) must be added to the offset of the respective root volume subdisk on this device. The resulting value will become the offset value for the new a partition. The size value for this partition can be taken verbatim from the calculation above. The fstype should be 4.2BSD. The

`fsize`, `bsize`, and `cpg` values should be chosen to match the actual file system, though they are fairly unimportant within this context.

That way, a new `a` partition will be established that overlaps the `vinum` partition on this device. `bsdlabel` will only allow for this overlap if the `vinum` partition has properly been marked using the `vinum` fstype.

3. A faked `a` partition now exists on each device that has one replica of the root volume. It is highly recommendable to verify the result using a command like:

   ```
   # fsck -n /dev/devnamea
   ```

It should be remembered that all files containing control information must be relative to the root file system in the `vinum` volume which, when setting up a new `vinum` root volume, might not match the root file system that is currently active. So in particular, `/etc/fstab` and `/boot/loader.conf` need to be taken care of.

At next reboot, the bootstrap should figure out the appropriate control information from the new `vinum`-based root file system, and act accordingly. At the end of the kernel initialization process, after all devices have been announced, the prominent notice that shows the success of this setup is a message like:

```
Mounting root from ufs:/dev/gvinum/root
```

22.8.3 Example of a `vinum`-based Root Setup

After the `vinum` root volume has been set up, the output of `gvinum l -rv root` could look like:

```
...
Subdisk root.p0.s0:
               Size:        125829120 bytes (120 MB)
               State: up
               Plex root.p0 at offset 0 (0  B)
               Drive disk0 (/dev/da0h) at offset 135680 (132 kB)

Subdisk root.p1.s0:
               Size:        125829120 bytes (120 MB)
               State: up
               Plex root.p1 at offset 0 (0  B)
               Drive disk1 (/dev/da1h) at offset 135680 (132 kB)
```

The values to note are `135680` for the offset, relative to partition `/dev/da0h`. This translates to 265 512-byte disk blocks in `bsdlabel`'s terms. Likewise, the size of this root volume is 245760 512-byte blocks. `/dev/da1h`, containing the second replica of this root volume, has a symmetric setup.

The bsdlabel for these devices might look like:

```
...
8 partitions:
#        size   offset    fstype   [fsize bsize bps/cpg]
  a:   245760      281    4.2BSD     2048 16384     0   # (Cyl.    0*- 15*)
  c: 71771688        0    unused        0     0         # (Cyl.    0 - 4467*)
  h: 71771672       16     vinum                        # (Cyl.    0*- 4467*)
```

It can be observed that the `size` parameter for the faked `a` partition matches the value outlined above, while the `offset` parameter is the sum of the offset within the `vinum` partition `h`, and the offset of this partition within the

device or slice. This is a typical setup that is necessary to avoid the problem described in Section 22.8.4.3. The entire `a` partition is completely within the `h` partition containing all the `vinum` data for this device.

In the above example, the entire device is dedicated to `vinum` and there is no leftover pre-`vinum` root partition.

22.8.4 Troubleshooting

The following list contains a few known pitfalls and solutions.

22.8.4.1 System Bootstrap Loads, but System Does Not Boot

If for any reason the system does not continue to boot, the bootstrap can be interrupted by pressing **space** at the 10-seconds warning. The loader variable `vinum.autostart` can be examined by typing `show` and manipulated using `set` or `unset`.

If the `vinum` kernel module was not yet in the list of modules to load automatically, type `load geom_vinum`.

When ready, the boot process can be continued by typing `boot -as` which `-as` requests the kernel to ask for the root file system to mount (`-a`) and make the boot process stop in single-user mode (`-s`), where the root file system is mounted read-only. That way, even if only one plex of a multi-plex volume has been mounted, no data inconsistency between plexes is being risked.

At the prompt asking for a root file system to mount, any device that contains a valid root file system can be entered. If `/etc/fstab` is set up correctly, the default should be something like `ufs:/dev/gvinum/root`. A typical alternate choice would be something like `ufs:da0d` which could be a hypothetical partition containing the pre-`vinum` root file system. Care should be taken if one of the alias `a` partitions is entered here, that it actually references the subdisks of the `vinum` root device, because in a mirrored setup, this would only mount one piece of a mirrored root device. If this file system is to be mounted read-write later on, it is necessary to remove the other plex(es) of the `vinum` root volume since these plexes would otherwise carry inconsistent data.

22.8.4.2 Only Primary Bootstrap Loads

If `/boot/loader` fails to load, but the primary bootstrap still loads (visible by a single dash in the left column of the screen right after the boot process starts), an attempt can be made to interrupt the primary bootstrap by pressing **space**. This will make the bootstrap stop in stage two. An attempt can be made here to boot off an alternate partition, like the partition containing the previous root file system that has been moved away from `a`.

22.8.4.3 Nothing Boots, the Bootstrap Panics

This situation will happen if the bootstrap had been destroyed by the `vinum` installation. Unfortunately, `vinum` accidentally leaves only 4 KB at the beginning of its partition free before starting to write its `vinum` header information. However, the stage one and two bootstraps plus the bsdlabel require 8 KB. So if a `vinum` partition was started at offset 0 within a slice or disk that was meant to be bootable, the `vinum` setup will trash the bootstrap.

Similarly, if the above situation has been recovered, by booting from a “Fixit” media, and the bootstrap has been re-installed using `bsdlabel -B` as described in Section 13.3.2, the bootstrap will trash the `vinum` header, and `vinum` will no longer find its disk(s). Though no actual `vinum` configuration data or data in `vinum` volumes will be trashed, and it would be possible to recover all the data by entering exactly the same `vinum` configuration data again, the situation is hard to fix. It is necessary to move the entire `vinum` partition by at least 4 KB, in order to have the `vinum` header and the system bootstrap no longer collide.

Chapter 23 Virtualization

Contributed by Murray Stokely.

23.1 Synopsis

Virtualization software allows multiple operating systems to run simultaneously on the same computer. Such software systems for PCs often involve a host operating system which runs the virtualization software and supports any number of guest operating systems.

After reading this chapter, you will know:

- The difference between a host operating system and a guest operating system.
- How to install FreeBSD on an Intel-based Apple Macintosh computer.
- How to install FreeBSD on Microsoft Windows with **Virtual PC**.
- How to tune a FreeBSD system for best performance under virtualization.

Before reading this chapter, you should:

- Understand the basics of UNIX and FreeBSD.
- Know how to install FreeBSD.
- Know how to set up a network connection.
- Know how to install additional third-party software.

23.2 FreeBSD as a Guest OS

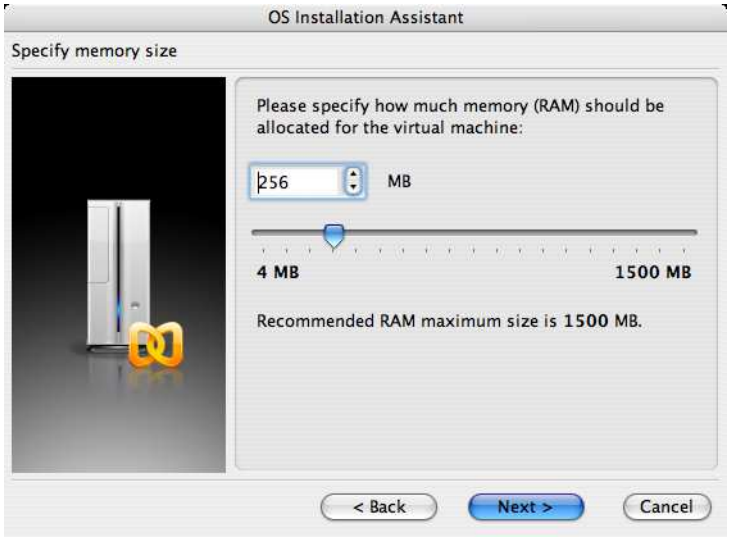
23.2.1 Parallels on Mac OS® X

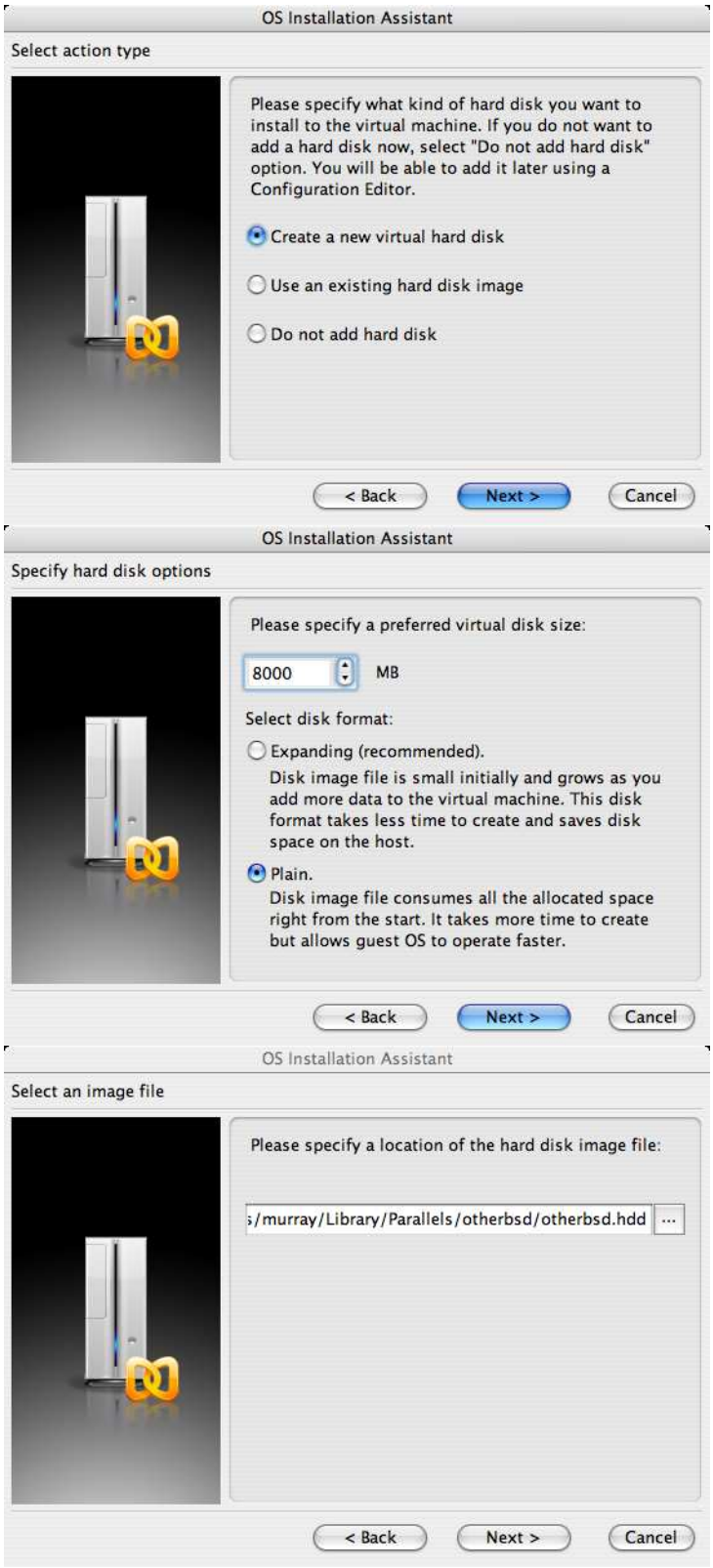
Parallels Desktop for Mac is a commercial software product available for Intel based Apple Mac computers running Mac OS 10.4.6 or higher. FreeBSD is a fully supported guest operating system. Once **Parallels** has been installed on Mac OS X, the user must configure a virtual machine and then install the desired guest operating system.

23.2.1.1 Installing FreeBSD on Parallels/Mac OS X

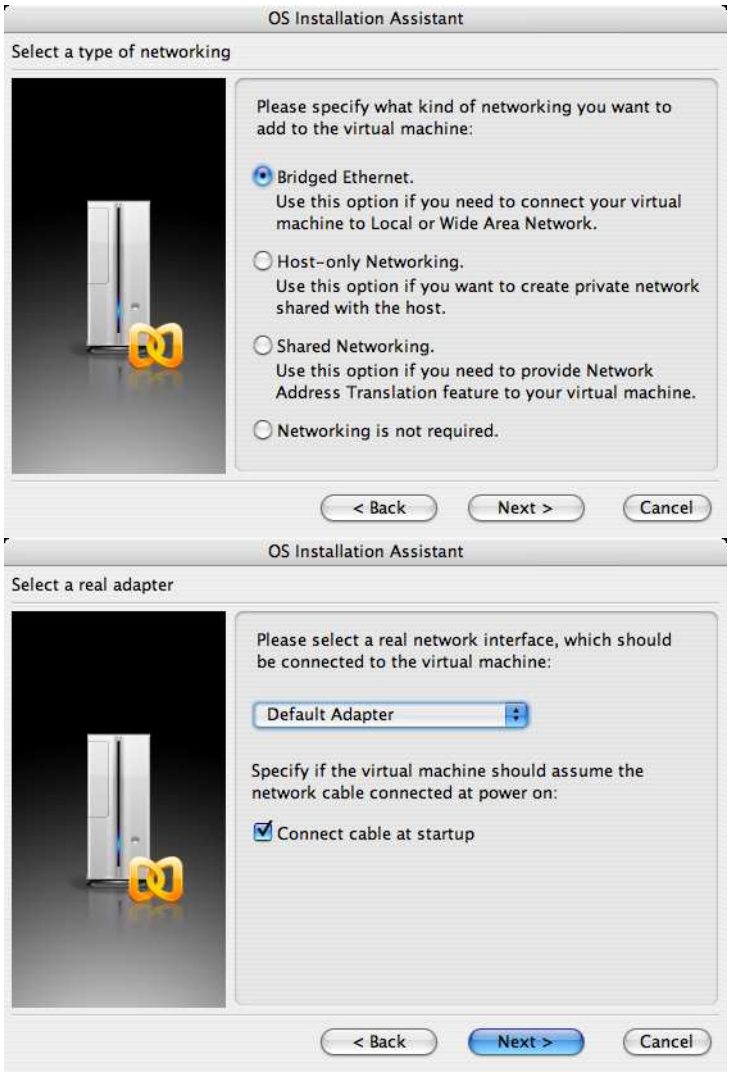
The first step in installing FreeBSD on **Parallels** is to create a new virtual machine for installing FreeBSD. Select FreeBSD as the Guest OS Type when prompted:

Choose a reasonable amount of disk and memory depending on the plans for this virtual FreeBSD instance. 4GB of disk space and 512MB of RAM work well for most uses of FreeBSD under **Parallels**:

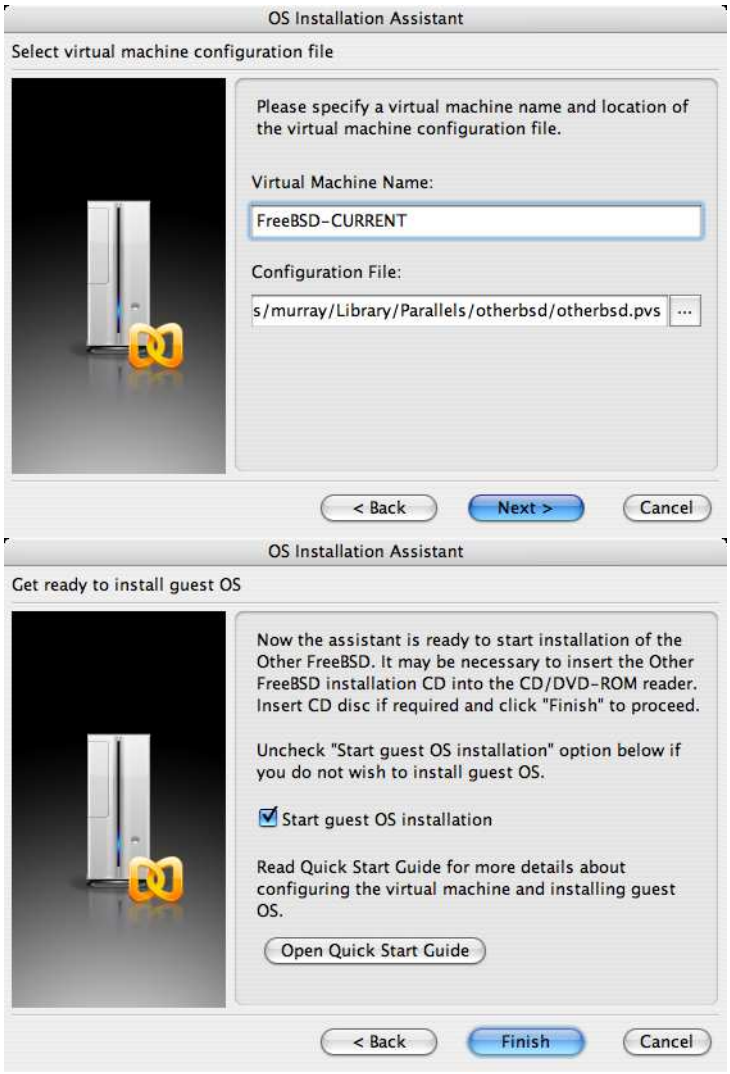




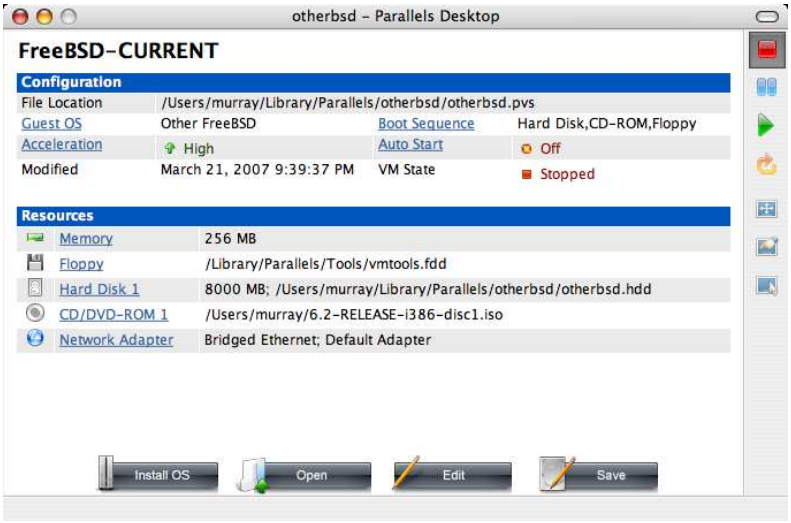
Select the type of networking and a network interface:



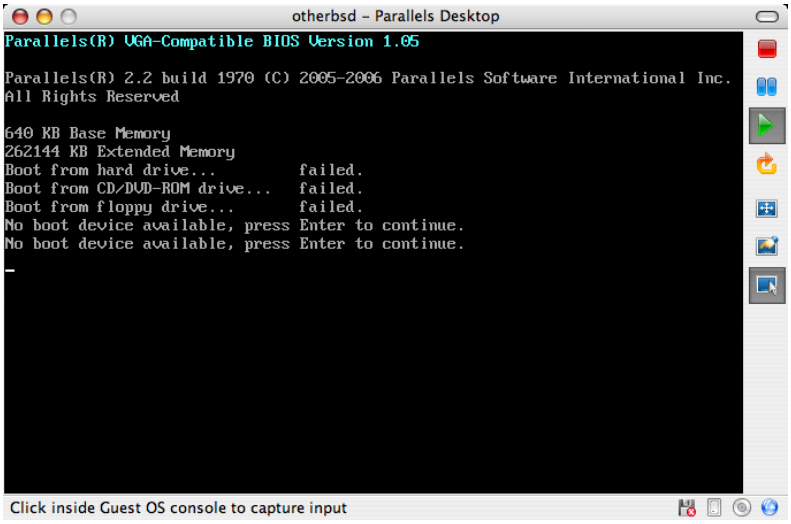
Save and finish the configuration:



After the FreeBSD virtual machine has been created, FreeBSD can be installed on it. This is best done with an official FreeBSD CD/DVD or with an ISO image downloaded from an official FTP site. Copy the appropriate ISO image to the local Mac filesystem or insert a CD/DVD in the Mac's CD drive. Click on the disc icon in the bottom right corner of the FreeBSD **Parallels** window. This will bring up a window that can be used to associate the CDROM drive in the virtual machine with the ISO file on disk or with the real CDROM drive.

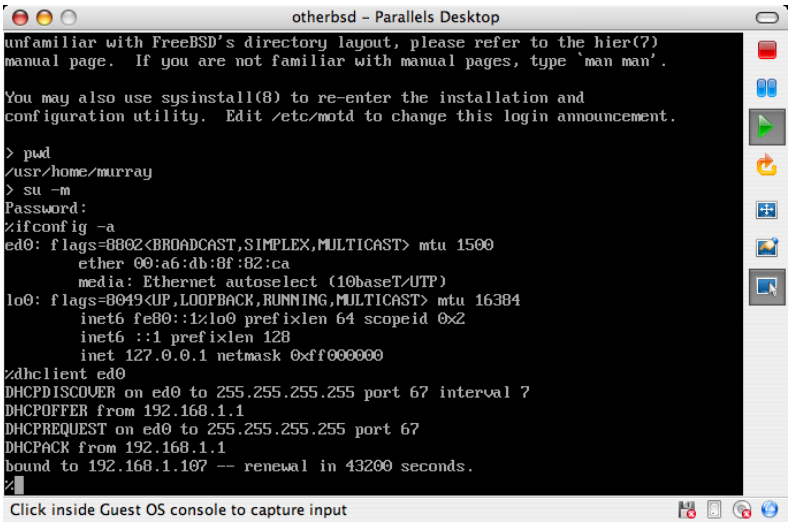


Once this association with the CDROM source has been made, reboot the FreeBSD virtual machine by clicking the reboot icon. **Parallels** will reboot with a special BIOS that first checks if there is a CDROM.



In this case it will find the FreeBSD installation media and begin a normal FreeBSD installation. Perform the installation, but do not attempt to configure **Xorg** at this time.

When the installation is finished, reboot into the newly installed FreeBSD virtual machine.



23.2.1.2 Configuring FreeBSD on Parallels

After FreeBSD has been successfully installed on Mac OS X with **Parallels**, there are a number of configuration steps that can be taken to optimize the system for virtualized operation.

1. Set Boot Loader Variables

   The most important step is to reduce the `kern.hz` tunable to reduce the CPU utilization of FreeBSD under the **Parallels** environment. This is accomplished by adding the following line to `/boot/loader.conf`:

   ```
   kern.hz=100
   ```

   Without this setting, an idle FreeBSD **Parallels** guest will use roughly 15% of the CPU of a single processor iMac®. After this change the usage will be closer to 5%.

2. Create a New Kernel Configuration File

All of the SCSI, FireWire, and USB device drivers can be removed from a custom kernel configuration file. **Parallels** provides a virtual network adapter used by the ed(4) driver, so all network devices except for ed(4) and miibus(4) can be removed from the kernel.

3. Configure Networking

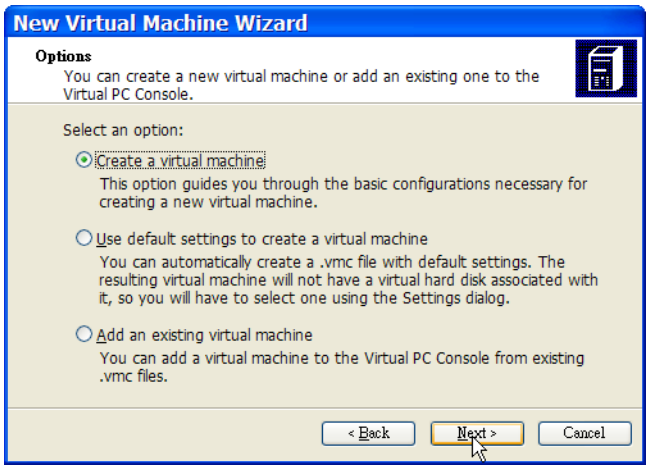
   The most basic networking setup uses DHCP to connect the virtual machine to the same local area network as the host Mac. This can be accomplished by adding `ifconfig_ed0="DHCP"` to `/etc/rc.conf`. More advanced networking setups are described in Chapter 32.

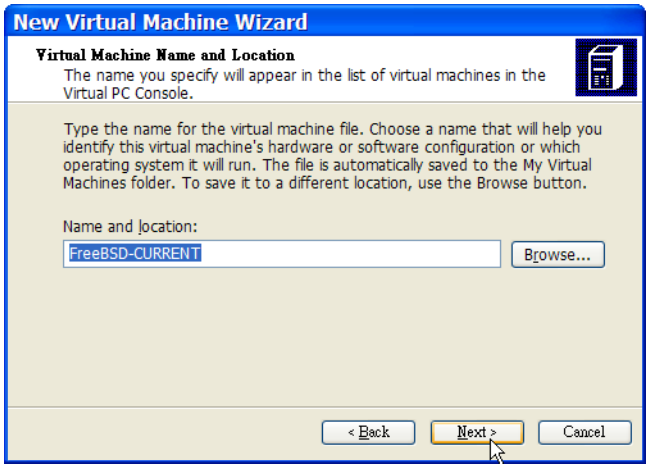
23.2.2 Virtual PC on Windows

Virtual PC for Windows is a Microsoft software product available for free download. See this website for the system requirements (http://www.microsoft.com/windows/downloads/virtualpc/sysreq.mspx). Once **Virtual PC** has been installed on Microsoft Windows, the user can configure a virtual machine and then install the desired guest operating system.

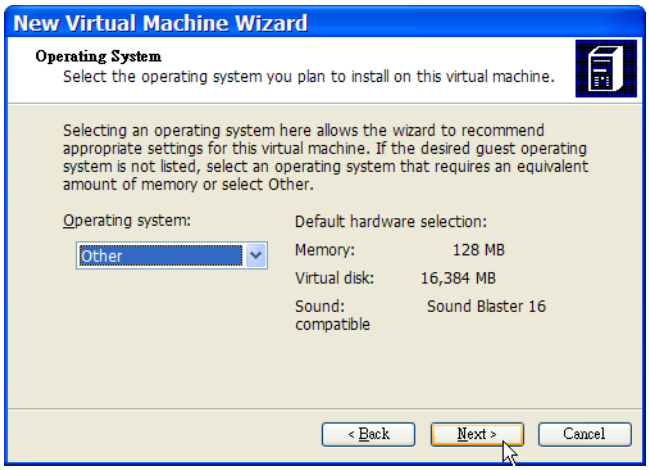
23.2.2.1 Installing FreeBSD on Virtual PC

The first step in installing FreeBSD on **Virtual PC** is to create a new virtual machine for installing FreeBSD. Select Create a virtual machine when prompted:

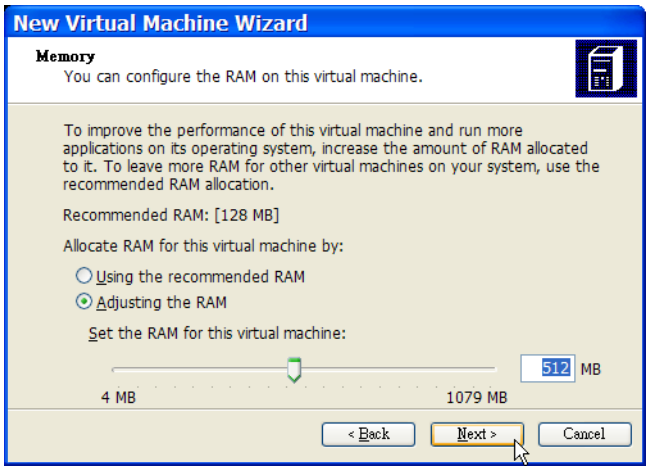


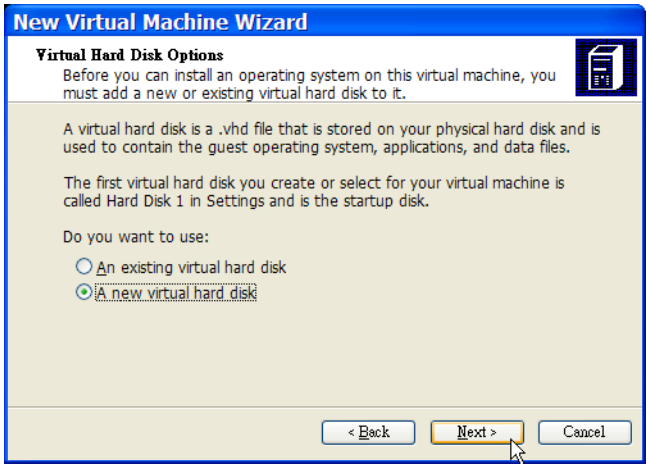


Select Other as the Operating system when prompted:

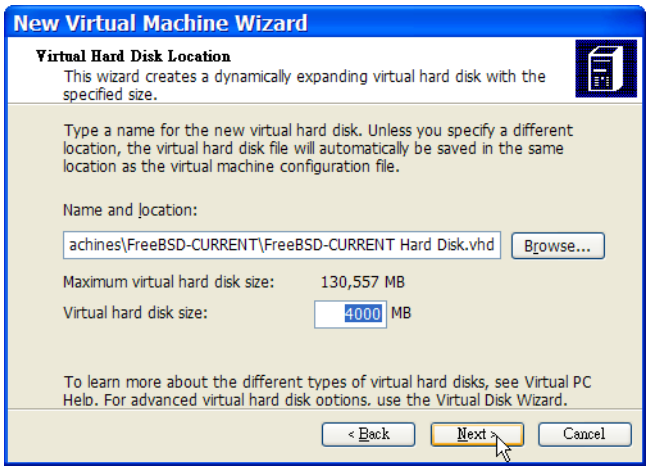


Then, choose a reasonable amount of disk and memory depending on the plans for this virtual FreeBSD instance. 4GB of disk space and 512MB of RAM work well for most uses of FreeBSD under **Virtual PC**:

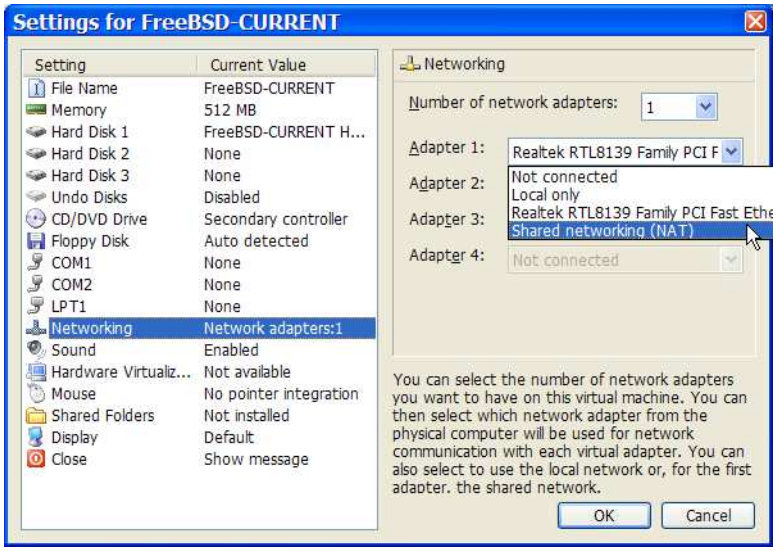




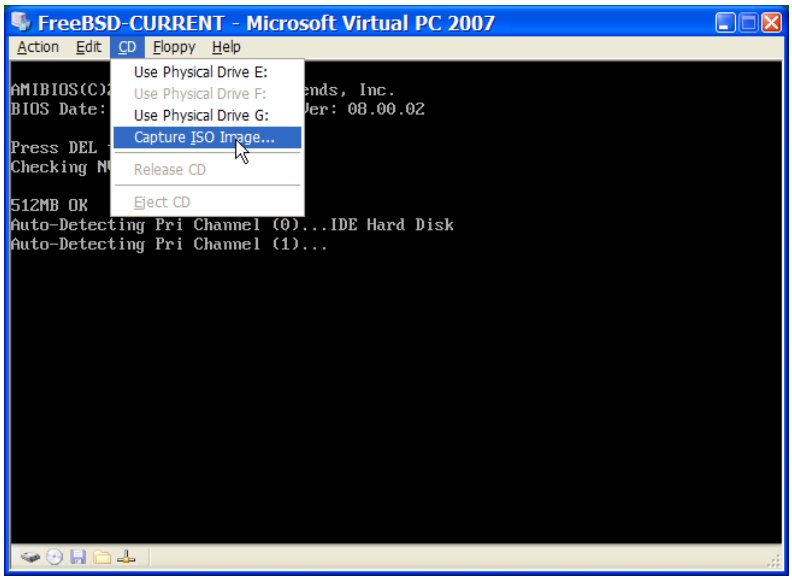
Save and finish the configuration:



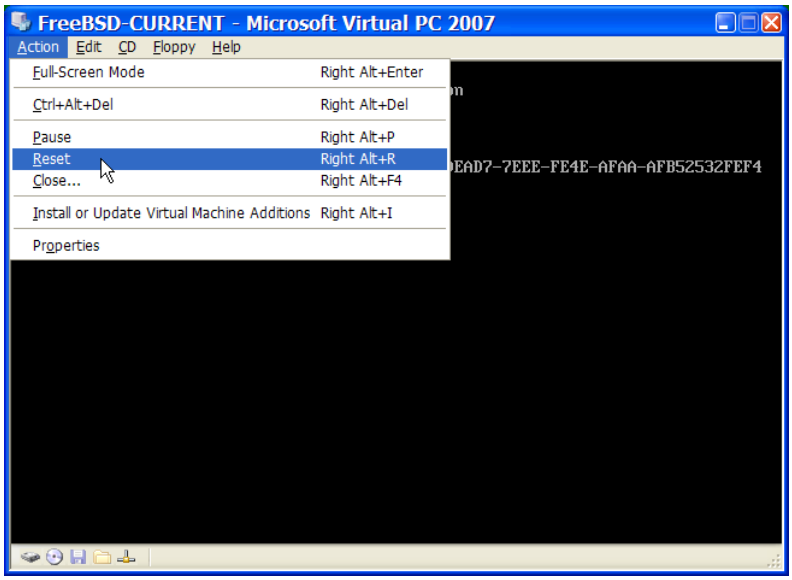
Select the FreeBSD virtual machine and click Settings, then set the type of networking and a network interface:



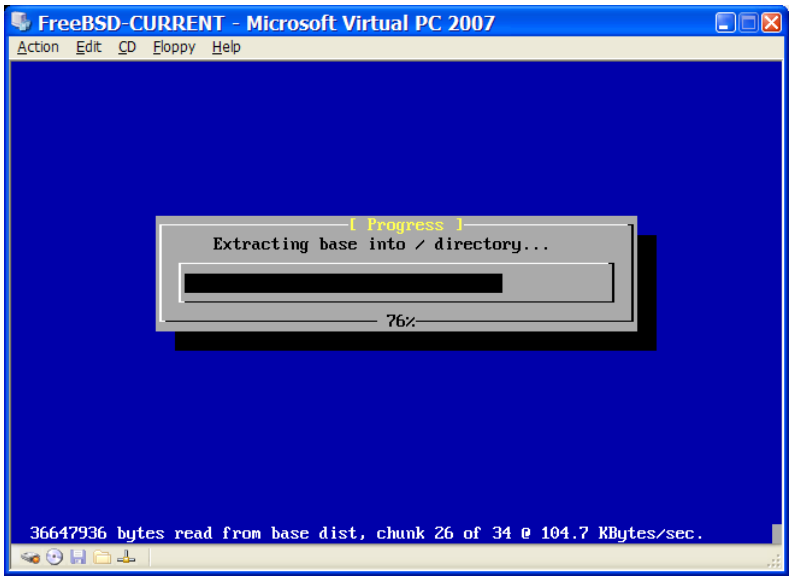
After the FreeBSD virtual machine has been created, FreeBSD can be installed on it. This is best done with an official FreeBSD CD/DVD or with an ISO image downloaded from an official FTP site. Copy the appropriate ISO image to the local Windows filesystem or insert a CD/DVD in the CD drive, then double click on the FreeBSD virtual machine to boot. Then, click CD and choose Capture ISO Image... on the **Virtual PC** window. This will bring up a window where the CDROM drive in the virtual machine can be associated with an ISO file on disk or with the real CDROM drive.



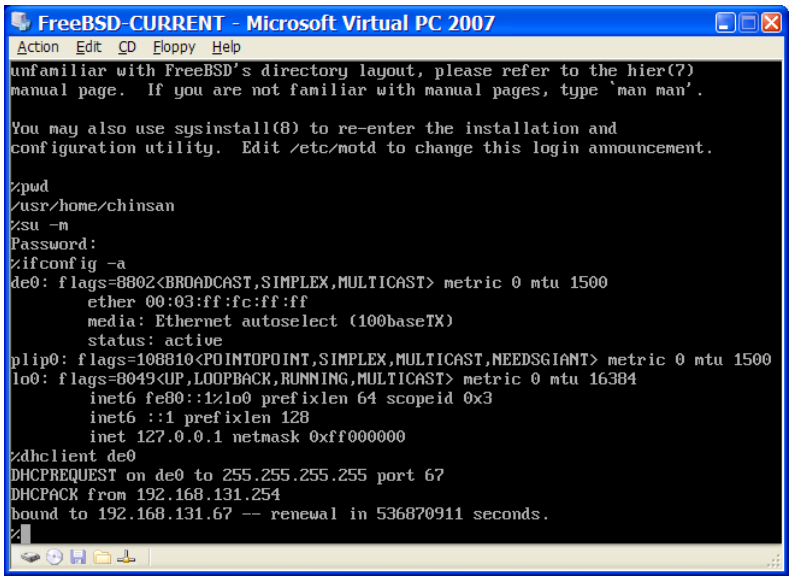
Once this association with the CDROM source has been made, reboot the FreeBSD virtual machine by clicking Action and Reset. **Virtual PC** will reboot with a special BIOS that first checks for a CDROM.



In this case it will find the FreeBSD installation media and begin a normal FreeBSD installation. Continue with the installation, but do not attempt to configure **Xorg** at this time.



When the installation is finished, remember to eject the CD/DVD or release the ISO image. Finally, reboot into the newly installed FreeBSD virtual machine.



23.2.2.2 Configuring FreeBSD on Virtual PC

After FreeBSD has been successfully installed on Microsoft Windows with **Virtual PC** , there are a number of configuration steps that can be taken to optimize the system for virtualized operation.

1. Set Boot Loader Variables

   The most important step is to reduce the `kern.hz` tunable to reduce the CPU utilization of FreeBSD under the **Virtual PC** environment. This is accomplished by adding the following line to `/boot/loader.conf`:

   ```
   kern.hz=100
   ```

Without this setting, an idle FreeBSD **Virtual PC** guest OS will use roughly 40% of the CPU of a single processor computer. After this change, the usage will be closer to 3%.

2. Create a New Kernel Configuration File

   All of the SCSI, FireWire, and USB device drivers can be removed from a custom kernel configuration file. **Virtual PC** provides a virtual network adapter used by the de(4) driver, so all network devices except for de(4) and miibus(4) can be removed from the kernel.

3. Configure Networking

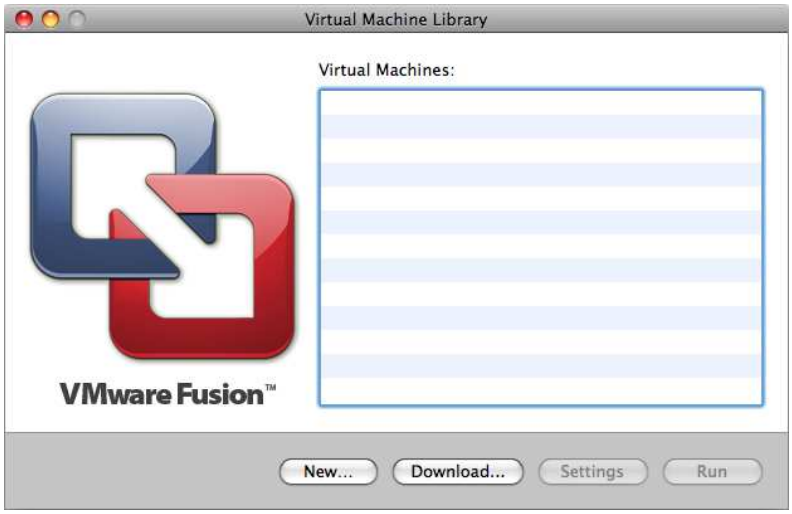
   The most basic networking setup uses DHCP to connect the virtual machine to the same local area network as the Microsoft Windows host. This can be accomplished by adding `ifconfig_de0="DHCP"` to `/etc/rc.conf`. More advanced networking setups are described in Chapter 32.

23.2.3 VMware Fusion on Mac OS

VMware Fusion for Mac is a commercial software product available for Intel based Apple Mac computers running Mac OS 10.4.9 or higher. FreeBSD is a fully supported guest operating system. Once **VMware Fusion** has been installed on Mac OS X, the user can configure a virtual machine and then install the desired guest operating system.

23.2.3.1 Installing FreeBSD on VMware Fusion

The first step is to start **VMware Fusion** which will load the Virtual Machine Library. Click New to create the virtual machine:



This will load the New Virtual Machine Assistant. Click Continue to proceed:

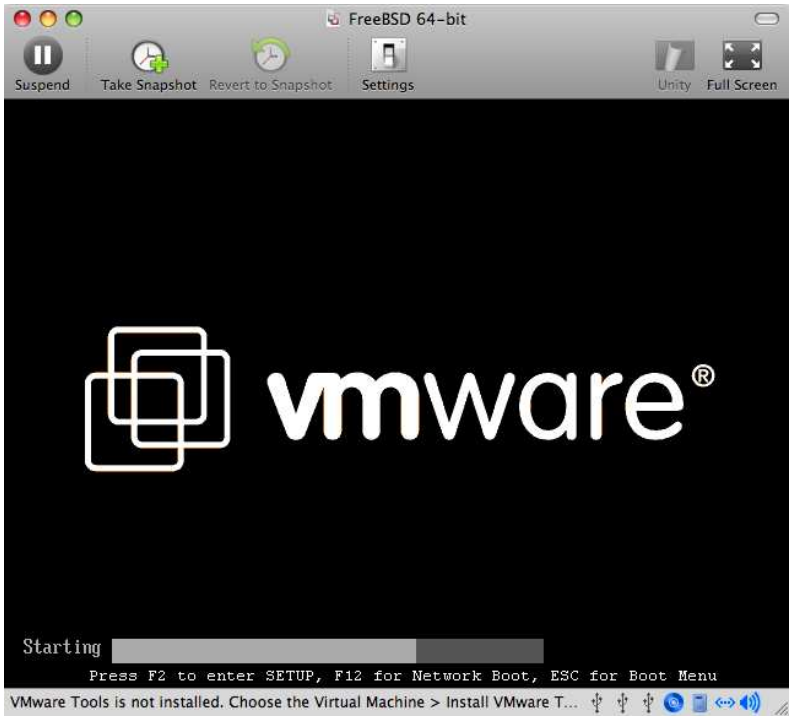
Select Other as the Operating System and either FreeBSD or FreeBSD 64-bit, as the Version when prompted:

Choose the name of the virtual machine and the directory where it should be saved:

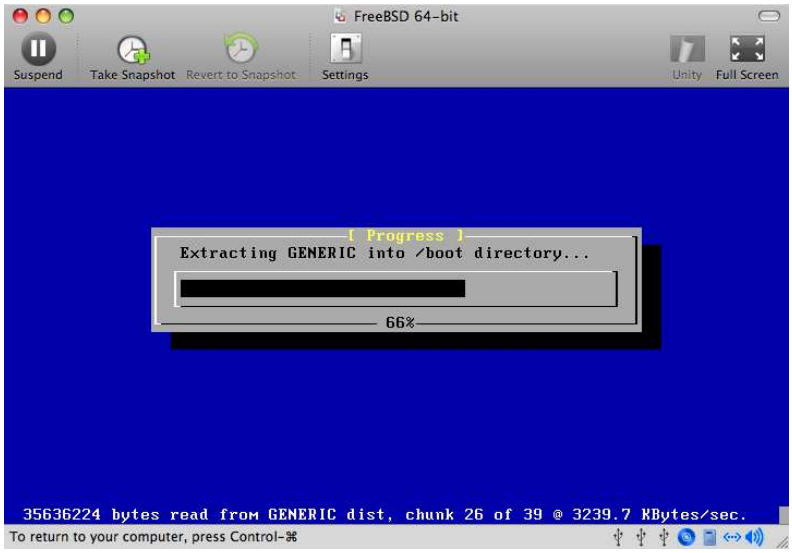
Choose the size of the Virtual Hard Disk for the virtual machine:

Choose the method to install the virtual machine, either from an ISO image or from a CD/DVD:

Click Finish and the virtual machine will boot:

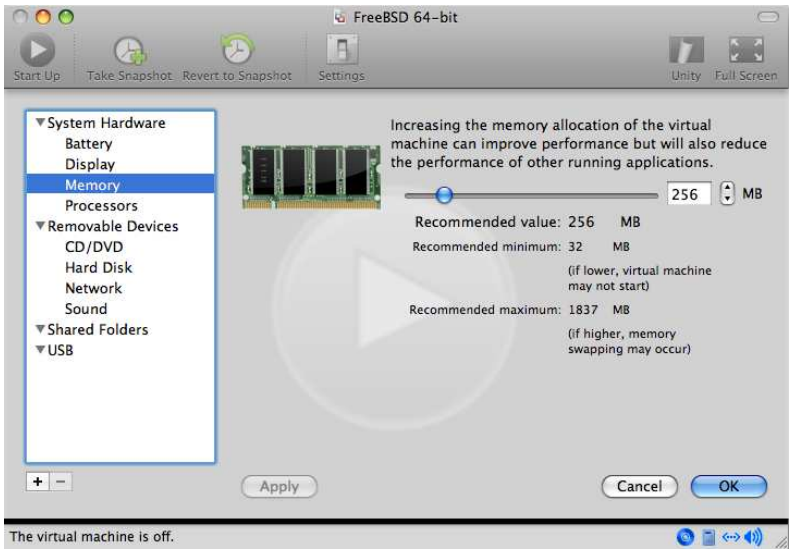


Install FreeBSD as usual:



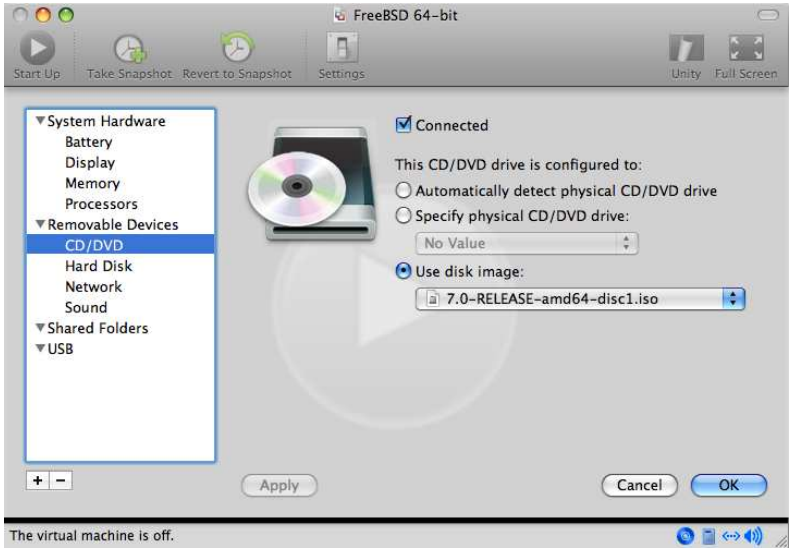
Once the install is complete, the settings of the virtual machine can be modified, such as memory usage:

> **Note:** The System Hardware settings of the virtual machine cannot be modified while the virtual machine is running.

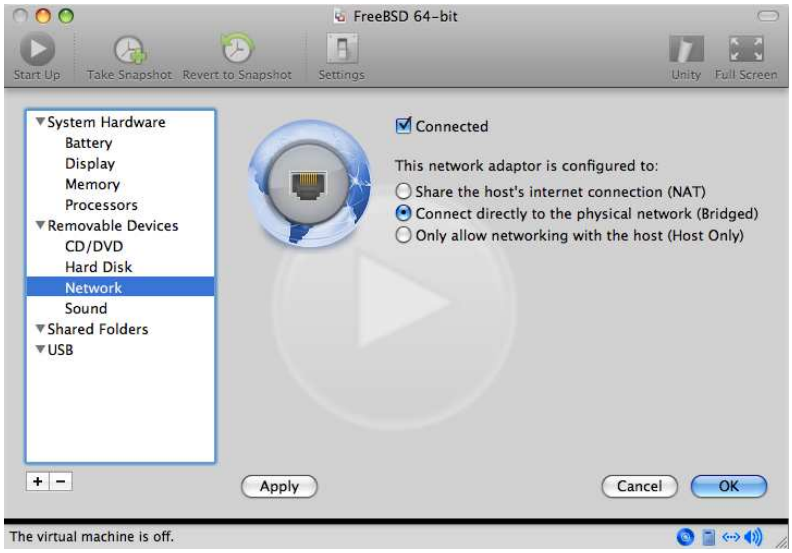


The number of CPUs the virtual machine will have access to:

The status of the CDROM device. Normally the CD/DVD/ISO is disconnected from the virtual machine when it is no longer needed.



The last thing to change is how the virtual machine will connect to the network. To allow connections to the virtual machine from other machines besides the host, choose Connect directly to the physical network (Bridged). Otherwise, Share the host's internet connection (NAT) is preferred so that the virtual machine can have access to the Internet, but the network cannot access the virtual machine.



After modifying the settings, boot the newly installed FreeBSD virtual machine.

23.2.3.2 Configuring FreeBSD on VMware Fusion

After FreeBSD has been successfully installed on Mac OS X with **VMware Fusion**, there are a number of configuration steps that can be taken to optimize the system for virtualized operation.

1. Set Boot Loader Variables

   The most important step is to reduce the `kern.hz` tunable to reduce the CPU utilization of FreeBSD under the **VMware Fusion** environment. This is accomplished by adding the following line to `/boot/loader.conf`:

   ```
   kern.hz=100
   ```

   Without this setting, an idle FreeBSD **VMware Fusion** guest will use roughly 15% of the CPU of a single processor iMac. After this change, the usage will be closer to 5%.

2. Create a New Kernel Configuration File

   All of the FireWire, and USB device drivers can be removed from a custom kernel configuration file. **VMware Fusion** provides a virtual network adapter used by the em(4) driver, so all network devices except for em(4) can be removed from the kernel.

3. Configure Networking

   The most basic networking setup uses DHCP to connect the virtual machine to the same local area network as the host Mac. This can be accomplished by adding `ifconfig_em0="DHCP"` to `/etc/rc.conf`. More advanced networking setups are described in Chapter 32.

23.2.4 VirtualBox™ Guest Additions on a FreeBSD Guest

The **VirtualBox™** guest additions provide support for:

- Clipboard sharing.
- Mouse pointer integration.
- Host time synchronization.
- Window scaling.
- Seamless mode.

Note: The following commands are run in the FreeBSD guest.

First, install the `emulators/virtualbox-ose-additions` package or port in the FreeBSD guest. This will install the port:

```
# cd /usr/ports/emulators/virtualbox-ose-additions && make install clean
```

Add these lines to `/etc/rc.conf`:

```
vboxguest_enable="YES"
vboxservice_enable="YES"
```

If ntpd(8) or ntpdate(8) is used, host time synchronization should be disabled:

```
vboxservice_flags="--disable-timesync"
```

The `vboxvideo` driver should be automatically recognized by `Xorg -configure`. If not, modify `/etc/X11/xorg.conf` for the **VirtualBox** video card:

```
Section "Device"
       ### Available Driver options are:-
       ### Values: <i>: integer, <f>: float, <bool>: "True"/"False",
       ### <string>: "String", <freq>: "<f> Hz/kHz/MHz"
       ### [arg]: arg optional
       Identifier "Card0"
       Driver "vboxvideo"
       VendorName "InnoTek Systemberatung GmbH"
       BoardName "VirtualBox Graphics Adapter"
       BusID "PCI:0:2:0"
EndSection
```

To use the `vboxmouse` driver, adjust the mouse section in `/etc/X11/xorg.conf`:

```
Section "InputDevice"
       Identifier "Mouse0"
       Driver "vboxmouse"
EndSection
```

HAL users should create the following `/usr/local/etc/hal/fdi/policy/90-vboxguest.fdi` or copy it from `/usr/local/share/hal/fdi/policy/10osvendor/90-vboxguest.fdi`:

```
<?xml version="1.0" encoding="utf-8"?>
<!--
# Sun VirtualBox
```

```
# Hal driver description for the vboxmouse driver
# $Id: chapter.xml,v 1.33 2012-03-17 04:53:52 eadler Exp $

       Copyright (C) 2008-2009 Sun Microsystems, Inc.

       This file is part of VirtualBox Open Source Edition (OSE, as
       available from http://www.virtualbox.org. This file is free software;
       you can redistribute it and/or modify it under the terms of the GNU
       General Public License (GPL) as published by the Free Software
       Foundation, in version 2 as it comes in the "COPYING" file of the
       VirtualBox OSE distribution. VirtualBox OSE is distributed in the
       hope that it will be useful, but WITHOUT ANY WARRANTY of any kind.

       Please contact Sun Microsystems, Inc., 4150 Network Circle, Santa
       Clara, CA 95054 USA or visit http://www.sun.com if you need
       additional information or have any questions.
-->
<deviceinfo version="0.2">
  <device>
    <match key="info.subsystem" string="pci">
      <match key="info.product" string="VirtualBox guest Service">
        <append key="info.capabilities" type="strlist">input</append>
       <append key="info.capabilities" type="strlist">input.mouse</append>
        <merge key="input.x11_driver" type="string">vboxmouse</merge>
       <merge key="input.device" type="string">/dev/vboxguest</merge>
      </match>
    </match>
  </device>
</deviceinfo>
```

23.3 FreeBSD as a Host

VirtualBox is an actively developed, complete virtualization package, that is available for most operating systems including Windows, Mac OS, Linux and FreeBSD. It is equally capable of running Windows or UNIX-like guests. It is released as open source software, but with closed-source components available in a separate extension pack. These components include support for USB 2.0 devices. More information may be found on the "Downloads" page of the **VirtualBox** wiki (http://www.virtualbox.org/wiki/Downloads). Currently, these extensions are not available for FreeBSD.

23.3.1 Installing VirtualBox

VirtualBox is available as a FreeBSD package or port in `emulators/virtualbox-ose`. The port can be installed using these commands:

```
# cd /usr/ports/emulators/virtualbox-ose
# make install clean
```

One useful option in the port's configuration menu is the `GuestAdditions` suite of programs. These provide a number of useful features in guest operating systems, like mouse pointer integration (allowing the mouse to be

shared between host and guest without the need to press a special keyboard shortcut to switch) and faster video rendering, especially in Windows guests. The guest additions are available in the Devices menu, after the installation of the guest is finished.

A few configuration changes are needed before **VirtualBox** is started for the first time. The port installs a kernel module in `/boot/modules` which must be loaded into the running kernel:

```
# kldload vboxdrv
```

To ensure the module always gets loaded after a reboot, add the following line to `/boot/loader.conf`:

```
vboxdrv_load="YES"
```

To use the kernel modules that allow bridged or host-only networking, add the following to `/etc/rc.conf` and reboot the computer:

```
vboxnet_enable="YES"
```

The `vboxusers` group is created during installation of **VirtualBox**. All users that need access to **VirtualBox** will have to be added as members of this group. `pw` can be used to add new members:

```
# pw groupmod vboxusers -m yourusername
```

The default permissions for `/dev/vboxnetctl` are restrictive and need to be changed for bridged networking:

```
# chown root:vboxusers /dev/vboxnetctl
# chmod 0660 /dev/vboxnetctl
```

To make this permissions change permanent, add these lines to `/etc/devfs.conf`:

```
own     vboxnetctl root:vboxusers
perm    vboxnetctl 0660
```

To launch **VirtualBox**, type from a **Xorg** session:

```
% VirtualBox
```

For more information on configuring and using **VirtualBox**, refer to the official website (http://www.virtualbox.org). For FreeBSD-specific information and troubleshooting instructions, refer to the relevant page in the FreeBSD wiki (http://wiki.FreeBSD.org/VirtualBox).

23.3.2 VirtualBox USB Support

In order to be able to read and write to USB devices, users need to be members of `operator`:

```
# pw groupmod operator -m jerry
```

Then, add the following to `/etc/devfs.rules`, or create this file if it does not exist yet:

```
[system=10]
add path 'usb/*' mode 0660 group operator
```

To load these new rules, add the following to `/etc/rc.conf`:

```
devfs_system_ruleset="system"
```

Then, restart devfs:

```
# service devfs restart
```

USB can now be enabled in the guest operating system. USB devices should be visible in the VirtualBox preferences.

23.3.3 VirtualBox Host DVD/CD Access

Access to the host DVD/CD drives from guests is achieved through the sharing of the physical drives. Within VirtualBox, this is set up from the Storage window in the Settings of the virtual machine. If needed, create an empty IDE CD/DVD device first. Then choose the Host Drive from the popup menu for the virtual CD/DVD drive selection. A checkbox labeled `Passthrough` will appear. This allows the virtual machine to use the hardware directly. For example, audio CDs or the burner will only function if this option is selected.

HAL needs to run for **VirtualBox** DVD/CD functions to work, so enable it in `/etc/rc.conf` and start it if it is not already running:

```
hald_enable="YES"
```

```
# service hald start
```

In order for users to be able to use **VirtualBox** DVD/CD functions, they need access to `/dev/xpt0`, `/dev/cd`*N*, and `/dev/pass`*N*. This is usually achieved by making the user a member of `operator`. Permissions to these devices have to be corrected by adding the following lines to `/etc/devfs.conf`:

```
perm cd* 0600
perm xpt0 0660
perm pass* 0660
```

```
# service devfs restart
```

Chapter 24 Localization - i18n/L10n Usage and Setup

Contributed by Andrey Chernov. Rewritten by Michael C. Wu.

24.1 Synopsis

FreeBSD is a distributed project with users and contributors located all over the world. This chapter discusses the internationalization and localization features of FreeBSD that allow non-English speaking users to get real work done. Since there are many aspects of the i18n implementation in both the system and application levels, more specific sources of documentation are referred to, where applicable.

After reading this chapter, you will know:

- How different languages and locales are encoded on modern operating systems.
- How to set the locale for a login shell.
- How to configure the console for non-English languages.
- How to use **Xorg**effectively with different languages.
- Where to find more information about writing i18n-compliant applications.

Before reading this chapter, you should:

- Know how to install additional third-party applications.

24.2 The Basics

24.2.1 What Is i18n/L10n?

The term internationalization has been shortened to i18n, which represents the number of letters between the first and the last letters of internationalization. L10n uses the same naming scheme, coming from “localization”. Combined together, i18n/L10n methods, protocols, and applications allow users to use languages of their choice.

i18n applications are programmed using i18n kits under libraries. These allow developers to write a simple file and translate displayed menus and texts to each language.

24.2.2 Why Use i18n/L10n?

Using i18n/L10n allows a user to view, input, or process data in non-English languages.

24.2.3 Which Languages Are Supported?

i18n and L10n are not FreeBSD specific. Currently, one can choose from most of the major languages, including but not limited to: Chinese, German, Japanese, Korean, French, Russian, and Vietnamese.

24.3 Using Localization

Localization settings are based on three main terms: Language Code, Country Code, and Encoding. Locale names are constructed from these parts as follows:

```
LanguageCode_CountryCode.Encoding
```

24.3.1 Language and Country Codes

In order to localize a FreeBSD system to a specific language, the user needs to determine the codes for the specific country and language as the country code tells applications which variation of the given language to use. The following are examples of language/country codes:

Language/Country Code	Description
en_US	English - United States
ru_RU	Russian for Russia
zh_TW	Traditional Chinese for Taiwan

A complete listing of available locales can be found by typing:

```
% locale -a
```

24.3.2 Encodings

Some languages use non-ASCII encodings that are 8-bit, wide, or multibyte characters. For more information on these encodings, refer to multibyte(3). Older applications do not recognize these encodings and mistake them for control characters. Newer applications usually recognize 8-bit characters. Depending on the implementation, users may be required to compile an application with wide or multibyte character support, or configure it correctly. To provide application support for wide or multibyte characters, the FreeBSD Ports Collection (http://www.FreeBSD.org/ports/index.html) contains programs for several languages. Refer to the i18n documentation in the respective FreeBSD port.

Specifically, the user needs to look at the application documentation to decide how to configure it correctly or to determine which compile options to use when building the port.

Some things to keep in mind are:

- Language specific single C chars character sets such as ISO8859-1, ISO8859-15, KOI8-R, and CP437. These are described in multibyte(3).
- Wide or multibyte encodings such as EUC and Big5.

The active list of character sets can be found at the IANA Registry (http://www.iana.org/assignments/character-sets).

> **Note:** FreeBSD uses Xorg-compatible locale encodings instead.

In the FreeBSD Ports Collection, i18n applications include `i18n` in their names for easy identification. However, they do not always support the language needed.

24.3.3 Setting Locale

Usually it is sufficient to export the value of the locale name as `LANG` in the login shell. This could be done in the user's `~/.login_conf` or in the startup file of the user's shell: (`~/.profile`, `~/.bashrc`, or `~/.cshrc`). There is no need to set the locale subsets such as `LC_CTYPE` or `LC_CTIME`. Refer to language-specific FreeBSD documentation for more information.

Each user should set the following two environment variables in their configuration files:

- `LANG` for POSIX setlocale(3) family functions
- `MM_CHARSET` for applications' MIME character set

These should be set in the user's shell configuration, the specific application configuration, and the **Xorg** configuration.

24.3.3.1 Setting Locale Methods

This section describes the two methods for setting locale. The first is recommended and assigns the environment variables in the login class. The second method adds the environment variable assignments to the system's shell startup file.

24.3.3.1.1 Login Classes Method

This method allows environment variables needed for locale name and MIME character sets to be assigned once for every possible shell instead of adding specific shell assignments to each shell's startup file. User Level Setup can be performed by each user while Administrator Level Setup requires superuser privileges.

24.3.3.1.1.1 User Level Setup

This provides a minimal example of a `.login_conf` located in a user's home directory which has both variables set for the Latin-1 encoding:

```
me:\
       :charset=ISO-8859-1:\
       :lang=de_DE.ISO8859-1:
```

Here is an example of a user's `.login_conf` that sets the variables for Traditional Chinese in BIG-5 encoding. More variables are set because some applications do not correctly respect locale variables for Chinese, Japanese, and Korean.

```
#Users who do not wish to use monetary units or time formats
#of Taiwan can manually change each variable
me:\
       :lang=zh_TW.Big5:\
       :setenv=LC_ALL=zh_TW.Big5:\
       :setenv=LC_COLLATE=zh_TW.Big5:\
       :setenv=LC_CTYPE=zh_TW.Big5:\
       :setenv=LC_MESSAGES=zh_TW.Big5:\
       :setenv=LC_MONETARY=zh_TW.Big5:\
       :setenv=LC_NUMERIC=zh_TW.Big5:\
```

```
        :setenv=LC_TIME=zh_TW.Big5:\
        :charset=big5:\
        :xmodifiers="@im=gcin": #Set gcin as the XIM Input Server
```

See Administrator Level Setup and login.conf(5) for more details.

24.3.3.1.1.2 Administrator Level Setup

Verify that the user's login class in `/etc/login.conf` sets the correct language:

```
language_name|Account Type Description:\
        :charset=MIME_charset:\
        :lang=locale_name:\
        :tc=default:
```

The previous Latin-1 example would look like this:

```
german|German Users Accounts:\
        :charset=ISO-8859-1:\
        :lang=de_DE.ISO8859-1:\
        :tc=default:
```

Whenever this file is edited, execute the following command to update the capability database:

```
# cap_mkdb /etc/login.conf
```

Changing Login Classes with vipw(8)

When using `vipw` to add new users, use *`language`* to set the language:

```
user:password:1111:11:language:0:0:User Name:/home/user:/bin/sh
```

Changing Login Classes with adduser(8)

When using `adduser` to add new users, configure the language as follows:

- If all new users use the same language, set `defaultclass = language` in `/etc/adduser.conf`.
- Alternatively, input the specified language at this prompt:

  ```
  Enter login class: default []:
  ```

  when creating a new user using adduser(8).
- Another alternative is to use the following when creating a user that uses a different language than the one set in `/etc/adduser.conf`:

  ```
  # adduser -class language
  ```

Changing Login Classes with pw(8)

If pw(8) is used to add new users, call it in this form:

```
# pw useradd user_name -L language
```

24.3.3.1.2 Shell Startup File Method

> **Note:** This method is not recommended because it requires a different setup for each shell. Use the Login Class Method instead.

To add the locale name and MIME character set, set the two environment variables shown below in the `/etc/profile` or `/etc/csh.login` shell startup files. This example sets the German language:

In `/etc/profile`:

```
LANG=de_DE.ISO8859-1; export LANG
MM_CHARSET=ISO-8859-1; export MM_CHARSET
```

Or in `/etc/csh.login`:

```
setenv LANG de_DE.ISO8859-1
setenv MM_CHARSET ISO-8859-1
```

Alternatively, add the above settings to `/usr/share/skel/dot.profile` or `/usr/share/skel/dot.login`.

To configure **Xorg**, add *one* of the following to `~/.xinitrc`, depending upon the shell:

```
LANG=de_DE.ISO8859-1; export LANG
```

```
setenv LANG de_DE.ISO8859-1
```

24.3.4 Console Setup

For all single C chars character sets, set the correct console fonts in `/etc/rc.conf` for the language in question with:

```
font8x16=font_name
font8x14=font_name
font8x8=font_name
```

The *`font_name`* is taken from `/usr/share/syscons/fonts`, without the `.fnt` suffix.

The keymap and screenmap for the single C chars character set can be set using `sysinstall`. Once inside **sysinstall**, choose Configure, then Console. Alternatively, add the following to `/etc/rc.conf`:

```
scrnmap=screenmap_name
keymap=keymap_name
keychange="fkey_number sequence"
```

The *`screenmap_name`* is taken from `/usr/share/syscons/scrnmaps`, without the `.scm` suffix. A screenmap with a corresponding mapped font is usually needed as a workaround for expanding bit 8 to bit 9 on a VGA adapter's font character matrix. This will move letters out of the pseudographics area if the screen font uses a bit 8 column.

If **moused** is enabled in `/etc/rc.conf`, review the mouse cursor information in the next paragraph.

By default, the mouse cursor of the syscons(4) driver occupies the 0xd0-0xd3 range in the character set. If the language uses this range, move the cursor's range. To enable this workaround for FreeBSD, add the following line to `/etc/rc.conf`:

```
mousechar_start=3
```

The *`keymap_name`* in the above example is taken from `/usr/share/syscons/keymaps`, without the `.kbd` suffix. When uncertain as to which keymap to use, kbdmap(1) can be used to test keymaps without rebooting.

The `keychange` is usually needed to program function keys to match the selected terminal type because function key sequences cannot be defined in the key map.

Be sure to set the correct console terminal type in `/etc/ttys` for all virtual terminal entries. Current pre-defined correspondences are:

Character Set	Terminal Type
ISO8859-1 or ISO8859-15	`cons25l1`
ISO8859-2	`cons25l2`
ISO8859-7	`cons25l7`
KOI8-R	`cons25r`
KOI8-U	`cons25u`
CP437 (VGA default)	`cons25`
US-ASCII	`cons25w`

For languages with wide or multibyte characters, use the correct FreeBSD port in `/usr/ports/`*`language`*. Some applications appear as serial terminals to the system. Reserve enough terminals in `/etc/ttys` for both **Xorg** and the pseudo-serial console. Here is a partial list of applications for using other languages in the console:

Language	Location
Traditional Chinese (BIG-5)	`chinese/big5con`
Japanese	`japanese/kon2-16dot` or `japanese/mule-freewnn`
Korean	`korean/han`

24.3.5 Xorg Setup

Although **Xorg** is not installed with FreeBSD, it can be installed from the Ports Collection. Refer to Chapter 6 for more information on how to do this. This section discusses how to localize **Xorg** once it is installed.

Application specific i18n settings such as fonts and menus can be tuned in `~/.Xresources`.

24.3.5.1 Displaying Fonts

After installing `x11-servers/xorg-server`, install the language's TrueType fonts. Setting the correct locale should allow users to view their selected language in graphical application menus.

24.3.5.2 Inputting Non-English Characters

The X Input Method (XIM) protocol is an input standard for **Xorg** clients. All **Xorg** applications should be written as XIM clients that take input from XIM input servers. There are several XIM servers available for different languages.

24.3.6 Printer Setup

Some single C chars character sets are hardware coded into printers. Wide or multibyte character sets require special setup using a utility such as **apsfilter**. Documents can be converted to PostScript or PDF formats using language specific converters.

24.3.7 Kernel and File Systems

The FreeBSD fast filesystem (FFS) is 8-bit clean, so it can be used with any single C chars character set. However, character set names are not stored in the filesystem as it is raw 8-bit and does not understand encoding order. Officially, FFS does not support any form of wide or multibyte character sets. However, some wide or multibyte character sets have independent patches for enabling support on FFS. Refer to the respective languages' web sites for more information and the patch files.

FreeBSD's support for the MS-DOS filesystem has the configurable ability to convert between MS-DOS, Unicode character sets, and chosen FreeBSD filesystem character sets. Refer to mount_msdosfs(8) for details.

24.4 Compiling i18n Programs

Many applications in the FreeBSD Ports Collection have been ported with i18n support. Some of these include `-i18n` in the port name. These and many other programs have built in support for i18n and need no special consideration.

However, some applications such as **MySQL** need to have their `Makefile` configured with the specific charset. This is usually done in the port's `Makefile` or by passing a value to **configure** in the source.

24.5 Localizing FreeBSD to Specific Languages

24.5.1 Russian Language (KOI8-R Encoding)

Originally contributed by Andrey Chernov.

For more information about KOI8-R encoding, refer to KOI8-R References (Russian Net Character Set) (http://koi8.pp.ru/).

24.5.1.1 Locale Setup

To set this locale, put the following lines into each user's `~/.login_conf`:

```
me:My Account:\
       :charset=KOI8-R:\
```

```
        :lang=ru_RU.KOI8-R:
```

24.5.1.2 Console Setup

- Add the following lines to `/etc/rc.conf`:

  ```
  keymap="ru.koi8-r"
  scrnmap="koi8-r2cp866"
  font8x16="cp866b-8x16"
  font8x14="cp866-8x14"
  font8x8="cp866-8x8"
  mousechar_start=3
  ```

- For each `ttyv` entry in `/etc/ttys`, use `cons25r` as the terminal type.

24.5.1.3 Printer Setup

Since most printers with Russian characters come with hardware code page CP866, a special output filter is needed to convert from KOI8-R to CP866. FreeBSD installs a default filter as `/usr/libexec/lpr/ru/koi2alt`. A Russian printer `/etc/printcap` entry should look like:

```
lp|Russian local line printer:\
        :sh:of=/usr/libexec/lpr/ru/koi2alt:\
        :lp=/dev/lpt0:sd=/var/spool/output/lpd:lf=/var/log/lpd-errs:
```

Refer to printcap(5) for a more detailed description.

24.5.1.4 MS-DOS® and Russian Filenames

The following example fstab(5) entry enables support for Russian filenames in mounted MS-DOS filesystems:

```
/dev/ad0s2      /dos/c  msdos   rw,-Lru_RU.KOI8-R 0 0
```

`-L` selects the locale name. Refer to mount_msdosfs(8) for more details.

24.5.1.5 Xorg Setup

1. First, configure the non-X locale setup.
2. When using **Xorg**, install the `x11-fonts/xorg-fonts-cyrillic` package.

   Check the `"Files"` section in `/etc/X11/xorg.conf`. The following line must be added *before* any other `FontPath` entries:

   ```
   FontPath   "/usr/local/lib/X11/fonts/cyrillic"
   ```

   Note: Search the Ports Collection for more Cyrillic fonts.

3. To activate a Russian keyboard, add the following to the "Keyboard" section of /etc/xorg.conf:

```
Option "XkbLayout"   "us,ru"
Option "XkbOptions"  "grp:toggle"
```

Make sure that XkbDisable is commented out in that file.

For grp:toggle use **Right Alt**, for grp:ctrl_shift_toggle use **Ctrl**+**Shift**. For grp:caps_toggle use **CapsLock**. The old **CapsLock** function is still available in LAT mode only using **Shift**+**CapsLock**. grp:caps_toggle does not work in **Xorg** for some unknown reason.

If the keyboard has “Windows” keys, and some non-alphabetical keys are mapped incorrectly, add the following line to /etc/xorg.conf:

```
Option "XkbVariant" ",winkeys"
```

Note: The Russian XKB keyboard may not work with non-localized applications.

Note: Minimally localized applications should call a XtSetLanguageProc (NULL, NULL, NULL); function early in the program.

See KOI8-R for X Window (http://koi8.pp.ru/xwin.html) for more instructions on localizing **Xorg** applications.

24.5.2 Traditional Chinese Localization for Taiwan

The FreeBSD-Taiwan Project has a Chinese HOWTO for FreeBSD at http://netlab.cse.yzu.edu.tw/~statue/freebsd/zh-tut/ using many Chinese ports. The current editor for the FreeBSD Chinese HOWTO is Shen Chuan-Hsing <statue@freebsd.sinica.edu.tw>.

24.5.3 German Language Localization for All ISO 8859-1 Languages

Slaven Rezic <eserte@cs.tu-berlin.de> wrote a tutorial on using umlauts on FreeBSD. The tutorial is written in German and is available at http://user.cs.tu-berlin.de/~eserte/FreeBSD/doc/umlaute/umlaute.html.

24.5.4 Greek Language Localization

Nikos Kokkalis <nickkokkalis@gmail.com> has written a complete article on Greek support in FreeBSD. It is available here (http://www.FreeBSD.org/doc/el_GR.ISO8859-7/articles/greek-language- support/ index.html), in Greek only, as part of the official FreeBSD Greek documentation.

24.5.5 Japanese and Korean Language Localization

For Japanese, refer to http://www.jp.FreeBSD.org/, and for Korean, refer to http://www.kr.FreeBSD.org/.

24.5.6 Non-English FreeBSD Documentation

Some FreeBSD contributors have translated parts of the FreeBSD documentation to other languages. They are available through links on the main site (http://www.FreeBSD.org/index.html) or in `/usr/share/doc`.

Chapter 25 Updating and Upgrading FreeBSD

Restructured, reorganized, and parts updated by Jim Mock. Original work by Jordan Hubbard, Poul-Henning Kamp, John Polstra, and Nik Clayton.

25.1 Synopsis

FreeBSD is under constant development between releases. Some people prefer to use the officially released versions, while others prefer to keep in sync with the latest developments. However, even official releases are often updated with security and other critical fixes. Regardless of the version used, FreeBSD provides all the necessary tools to keep the system updated, and allows for easy upgrades between versions. This chapter describes how to track the development system and the basic tools for keeping a FreeBSD system up-to-date.

After reading this chapter, you will know:

- Which utilities are available to update the system and the Ports Collection.
- How to keep a FreeBSD system up-to-date with **freebsd-update**, **Subversion**, or **CTM**.
- How to compare the state of an installed system against a known pristine copy.
- How to keep the installed documentation up-to-date with **Subversion** or documentation ports.
- The difference between the two development branches: FreeBSD-STABLE and FreeBSD-CURRENT.
- How to rebuild and reinstall the entire base system.

Before reading this chapter, you should:

- Properly set up the network connection (Chapter 32).
- Know how to install additional third-party software (Chapter 5).

> **Note:** Throughout this chapter, `svn` is used to obtain and update FreeBSD sources. To use it, first install the `devel/subversion` port or package.

25.2 FreeBSD Update

Written by Tom Rhodes. Based on notes provided by Colin Percival.

Applying security patches is an important part of maintaining computer software, especially the operating system. For the longest time on FreeBSD, this process was not an easy one. Patches had to be applied to the source code, the code rebuilt into binaries, and then the binaries had to be re-installed.

This is no longer the case as FreeBSD now includes a utility called `freebsd-update`. This utility provides two separate functions. First, it allows for binary security and errata updates to be applied to the FreeBSD base system without the build and install requirements. Second, the utility supports minor and major release upgrades.

> **Note:** Binary updates are available for all architectures and releases currently supported by the security team. Before updating to a new release, its release announcement should be reviewed as it contains important information pertinent to the release. Release announcements are available from http://www.FreeBSD.org/releases/.

If a `crontab` utilizing the features of freebsd-update(8) exists, it must be disabled before the following operation is started.

25.2.1 The Configuration File

Some users may wish to tweak the default configuration in `/etc/freebsd-update.conf`, allowing better control of the process. The options are well documented, but the following may require a bit more explanation:

```
# Components of the base system which should be kept updated.
Components src world kernel
```

This parameter controls which parts of FreeBSD will be kept up-to-date. The default is to update the source code, the entire base system, and the kernel. Components are the same as those available during installation. For instance, adding `world/games` would allow game patches to be applied. Using `src/bin` would allow the source code in `src/bin` to be updated.

The best option is to leave this at the default as changing it to include specific items requires the user to list every item to be updated. This could have disastrous consequences as source code and binaries may become out of sync.

```
# Paths which start with anything matching an entry in an IgnorePaths
# statement will be ignored.
IgnorePaths
```

To leave specified directories, such as `/bin` or `/sbin`, untouched during the update process, add their paths to this statement. This option may be used to prevent `freebsd-update` from overwriting local modifications.

```
# Paths which start with anything matching an entry in an UpdateIfUnmodified
# statement will only be updated if the contents of the file have not been
# modified by the user (unless changes are merged; see below).
UpdateIfUnmodified /etc/ /var/ /root/ /.cshrc /.profile
```

This option will only update unmodified configuration files in the specified directories. Any changes made by the user will invalidate the automatic updating of these files. There is another option, `KeepModifiedMetadata`, which will instruct `freebsd-update` to save the changes during the merge.

```
# When upgrading to a new FreeBSD release, files which match MergeChanges
# will have any local changes merged into the version from the new release.
MergeChanges /etc/ /var/named/etc/
```

List of directories with configuration files that `freebsd-update` should attempt to merge. The file merge process is a series of diff(1) patches similar to mergemaster(8), but with fewer options. Merges are either accepted, open an editor, or `freebsd-update` will abort. When in doubt, backup `/etc` and just accept the merges. See Section 25.7.12.1 for more information about `mergemaster`.

```
# Directory in which to store downloaded updates and temporary
# files used by FreeBSD Update.
```

```
# WorkDir /var/db/freebsd-update
```

This directory is where all patches and temporary files are placed. In cases where the user is doing a version upgrade, this location should have a least a gigabyte of disk space available.

```
# When upgrading between releases, should the list of Components be
# read strictly (StrictComponents yes) or merely as a list of components
# which *might* be installed of which FreeBSD Update should figure out
# which actually are installed and upgrade those (StrictComponents no)?
# StrictComponents no
```

When this option is set to `yes`, `freebsd-update` will assume that the `Components` list is complete and will not attempt to make changes outside of the list. Effectively, `freebsd-update` will attempt to update every file which belongs to the `Components` list.

25.2.2 Security Patches

FreeBSD security patches may be downloaded and installed using the following command:

```
# freebsd-update fetch
# freebsd-update install
```

If the update applied any kernel patches, the system will need a reboot in order to boot into the patched kernel. Otherwise, the system should be patched and `freebsd-update` may be run as a nightly cron(8) job by adding this entry to `/etc/crontab`:

```
@daily                                  root    freebsd-update cron
```

This entry states that `freebsd-update` will run once every day. When run with `cron`, `freebsd-update` will only check if updates exist. If patches exist, they will automatically be downloaded to the local disk but will not be applied. The `root` user will be sent an email so that they may be reviewed and manually installed.

If anything goes wrong, `freebsd-update` has the ability to roll back the last set of changes with the following command:

```
# freebsd-update rollback
```

Once complete, the system should be restarted if the kernel or any kernel modules were modified. This will allow FreeBSD to load the new binaries into memory.

Only the `GENERIC` kernel can be automatically updated by `freebsd-update`. If a custom kernel is installed, it will have to be rebuilt and reinstalled after `freebsd-update` finishes installing the rest of the updates. However, `freebsd-update` will detect and update the `GENERIC` kernel if `/boot/GENERIC` exists, even if it is not the current running kernel of the system.

> **Note:** It is a good idea to always keep a copy of the `GENERIC` kernel in `/boot/GENERIC`. It will be helpful in diagnosing a variety of problems, and in performing version upgrades using `freebsd-update` as described in Section 25.2.3.

Unless the default configuration in `/etc/freebsd-update.conf` has been changed, `freebsd-update` will install the updated kernel sources along with the rest of the updates. Rebuilding and reinstalling a new custom kernel can then be performed in the usual way.

> **Note:** The updates distributed by `freebsd-update` do not always involve the kernel. It is not necessary to rebuild a custom kernel if the kernel sources have not been modified by the execution of `freebsd-update install`. However, `freebsd-update` will always update `/usr/src/sys/conf/newvers.sh`. The current patch level, as indicated by the `-p` number reported by `uname -r`, is obtained from this file. Rebuilding a custom kernel, even if nothing else changed, allows uname(1) to accurately report the current patch level of the system. This is particularly helpful when maintaining multiple systems, as it allows for a quick assessment of the updates installed in each one.

25.2.3 Major and Minor Version Upgrades

Upgrades from one minor version of FreeBSD to another, like from FreeBSD 9.0 to FreeBSD 9.1, are called *minor version* upgrades. Generally, installed applications will continue to work without problems after minor version upgrades.

Major version upgrades occur when FreeBSD is upgraded from one major version to another, like from FreeBSD 8.X to FreeBSD 9.X. Major version upgrades remove old object files and libraries which will break most third party applications. It is recommended that all installed ports either be removed and re-installed or upgraded after a major version upgrade using a utility such as `ports-mgmt/portmaster`. A brute-force rebuild of all installed applications can be accomplished with this command:

```
# portmaster -af
```

This will ensure everything will be re-installed correctly. Note that setting the `BATCH` environment variable to `yes` will answer `yes` to any prompts during this process, removing the need for manual intervention during the build process.

25.2.3.1 Dealing with Custom Kernels

If a custom kernel is in use, the upgrade process is slightly more involved, and the procedure varies depending on the version of FreeBSD.

25.2.3.1.1 Custom Kernels with FreeBSD 8.X

A copy of the `GENERIC` kernel is needed, and should be placed in `/boot/GENERIC`. If the `GENERIC` kernel is not present in the system, it may be obtained using one of the following methods:

- If a custom kernel has only been built once, the kernel in `/boot/kernel.old` is actually `GENERIC`. Rename this directory to `/boot/GENERIC`.
- Assuming physical access to the machine is possible, a copy of the `GENERIC` kernel can be installed from the installation media using the following commands:

```
# mount /cdrom
# cd /cdrom/X.Y-RELEASE/kernels
# ./install.sh GENERIC
```

Replace *X.Y-RELEASE* with the actual version of the release being used. The GENERIC kernel will be installed in /boot/GENERIC by default.

- Failing all the above, the GENERIC kernel may be rebuilt and installed from source:

```
# cd /usr/src
# env DESTDIR=/boot/GENERIC make kernel __MAKE_CONF=/dev/null SRCCONF=/dev/null
# mv /boot/GENERIC/boot/kernel/* /boot/GENERIC
# rm -rf /boot/GENERIC/boot
```

For this kernel to be picked up as GENERIC by freebsd-update, the GENERIC configuration file must not have been modified in any way. It is also suggested that it is built without any other special options.

Rebooting to the GENERIC kernel is not required at this stage.

25.2.3.1.2 Custom Kernels with FreeBSD 9.X and Later

- If a custom kernel has only been built once, the kernel in /boot/kernel.old is actually the GENERIC kernel. Rename this directory to /boot/kernel.
- If physical access to the machine is available, a copy of the GENERIC kernel can be installed from the installation media using these commands:

```
# mount /cdrom
# cd /cdrom/usr/freebsd-dist
# tar -C/ -xvf kernel.txz boot/kernel/kernel
```

- If the options above cannot be used, the GENERIC kernel may be rebuilt and installed from source:

```
# cd /usr/src
# make kernel __MAKE_CONF=/dev/null SRCCONF=/dev/null
```

For this kernel to be identified as the GENERIC kernel by freebsd-update, the GENERIC configuration file must not have been modified in any way. It is also suggested that the kernel is built without any other special options.

Rebooting to the GENERIC kernel is not required at this stage.

25.2.3.2 Performing the Upgrade

Major and minor version upgrades may be performed by providing freebsd-update with a release version target. The following command will update to FreeBSD 9.1:

```
# freebsd-update -r 9.1-RELEASE upgrade
```

After the command has been received, freebsd-update will evaluate the configuration file and current system in an attempt to gather the information necessary to perform the upgrade. A screen listing will display which components have and have not been detected. For example:

```
Looking up update.FreeBSD.org mirrors... 1 mirrors found.
Fetching metadata signature for 9.0-RELEASE from update1.FreeBSD.org... done.
Fetching metadata index... done.
Inspecting system... done.

The following components of FreeBSD seem to be installed:
```

```
kernel/smp src/base src/bin src/contrib src/crypto src/etc src/games
src/gnu src/include src/krb5 src/lib src/libexec src/release src/rescue
src/sbin src/secure src/share src/sys src/tools src/ubin src/usbin
world/base world/info world/lib32 world/manpages

The following components of FreeBSD do not seem to be installed:
kernel/generic world/catpages world/dict world/doc world/games
world/proflibs

Does this look reasonable (y/n)? y
```

At this point, `freebsd-update` will attempt to download all files required for the upgrade. In some cases, the user may be prompted with questions regarding what to install or how to proceed.

When using a custom kernel, the above step will produce a warning similar to the following:

```
WARNING: This system is running a "MYKERNEL" kernel, which is not a
kernel configuration distributed as part of FreeBSD 9.0-RELEASE.
This kernel will not be updated: you MUST update the kernel manually
before running "/usr/sbin/freebsd-update install"
```

This warning may be safely ignored at this point. The updated `GENERIC` kernel will be used as an intermediate step in the upgrade process.

Once all the patches have been downloaded to the local system, they will be applied. This process may take a while, depending on the speed and workload of the machine. Configuration files will then be merged. The merging process requires some user intervention as a file may be merged or an editor may appear on screen for a manual merge. The results of every successful merge will be shown to the user as the process continues. A failed or ignored merge will cause the process to abort. Users may wish to make a backup of `/etc` and manually merge important files, such as `master.passwd` or `group` at a later time.

> **Note:** The system is not being altered yet as all patching and merging is happening in another directory. Once all patches have been applied successfully, all configuration files have been merged and it seems the process will go smoothly, the changes can be committed to disk by the user using the following command:
>
> ```
> # freebsd-update install
> ```

The kernel and kernel modules will be patched first. At this point, the machine must be rebooted. If the system is running with a custom kernel, use nextboot(8) to set the kernel for the next boot to the updated `/boot/GENERIC`:

```
# nextboot -k GENERIC
```

> **Warning:** Before rebooting with the `GENERIC` kernel, make sure it contains all the drivers required for the system to boot properly and connect to the network, if the machine being updated is accessed remotely. In particular, if the running custom kernel contains built-in functionality usually provided by kernel modules, make sure to temporarily load these modules into the `GENERIC` kernel using the `/boot/loader.conf` facility. It is recommended to disable non-essential services as well as any disk and network mounts until the upgrade process is complete.

The machine should now be restarted with the updated kernel:

```
# shutdown -r now
```

Once the system has come back online, restart `freebsd-update` using the following command. The state of the process has been saved and thus, `freebsd-update` will not start from the beginning, but will remove all old shared libraries and object files.

```
# freebsd-update install
```

> **Note:** Depending upon whether any library version numbers were bumped, there may only be two install phases instead of three.

25.2.3.3 Rebuilding Ports After a Major Version Upgrade

After a major version upgrade, all third party software needs to be rebuilt and re-installed. This is required as installed software may depend on libraries which have been removed during the upgrade process. This process can be automated using `ports-mgmt/portmaster`:

```
# portmaster -f
```

Once this has completed, finish the upgrade process with a final call to `freebsd-update` in order to tie up all the loose ends in the upgrade process:

```
# freebsd-update install
```

If the `GENERIC` kernel was temporarily used, this is the time to build and install a new custom kernel in the usual way.

Reboot the machine into the new FreeBSD version. The process is complete.

25.2.4 System State Comparison

`freebsd-update` can be used to test the state of the installed FreeBSD version against a known good copy. This option evaluates the current version of system utilities, libraries, and configuration files. To begin the comparison, issue the following command:

```
# freebsd-update IDS >> outfile.ids
```

> **Warning:** While the command name is IDS it is not a replacement for a real intrusion detection system such as `security/snort`. As `freebsd-update` stores data on disk, the possibility of tampering is evident. While this possibility may be reduced using `kern.securelevel` and by storing the `freebsd-update` data on a read only file system when not in use, a better solution would be to compare the system against a secure disk, such as a DVD or securely stored external USB disk device.

The system will now be inspected, and a lengthy listing of files, along with the sha256(1) hash values for both the known value in the release and the current installation, will be sent to the specified `outfile.ids` file.

The entries in the listing are extremely long, but the output format may be easily parsed. For instance, to obtain a list of all files which differ from those in the release, issue the following command:

```
# cat outfile.ids | awk '{ print $1 }' | more
/etc/master.passwd
/etc/motd
/etc/passwd
/etc/pf.conf
```

This sample output has been truncated as many more files exist. Some files have natural modifications. For example, `/etc/passwd` has been modified because users have been added to the system. Other files, such as kernel modules, may differ as `freebsd-update` may have updated them. To exclude specific files or directories, add them to the `IDSIgnorePaths` option in `/etc/freebsd-update.conf`.

This system may be used as part of an elaborate upgrade method, aside from the previously discussed version.

25.3 Portsnap: a Ports Collection Update Tool

Written by Tom Rhodes. Based on notes provided by Colin Percival.

The base system of FreeBSD includes portsnap(8) for updating the Ports Collection. This utility connects to a FreeBSD site, verifies the secure key, and downloads a new copy of the Ports Collection. The key is used to verify the integrity of all downloaded files. To download the latest Ports Collection files, issue the following command:

```
# portsnap fetch
Looking up portsnap.FreeBSD.org mirrors... 9 mirrors found.
Fetching snapshot tag from geodns-1.portsnap.freebsd.org... done.
Fetching snapshot metadata... done.
Updating from Tue May 22 02:12:15 CEST 2012 to Wed May 23 16:28:31 CEST 2012.
Fetching 3 metadata patches.. done.
Applying metadata patches... done.
Fetching 3 metadata files... done.
Fetching 90 patches.....10....20....30....40....50....60....70....80....90. done.
Applying patches... done.
Fetching 133 new ports or files... done.
```

What this example shows is that portsnap(8) has found and verified several patches to the current ports data. This also indicates that the utility was run previously; if it was a first time run, the collection would have simply been downloaded.

When portsnap(8) successfully completes a `fetch` operation, the Ports Collection and subsequent patches which exist on the local system have passed verification. The first time `portsnap` is executed, use `extract` to install the downloaded files:

```
# portsnap extract
/usr/ports/.cvsignore
/usr/ports/CHANGES
/usr/ports/COPYRIGHT
/usr/ports/GIDs
/usr/ports/KNOBS
/usr/ports/LEGAL
```

```
/usr/ports/MOVED
/usr/ports/Makefile
/usr/ports/Mk/bsd.apache.mk
/usr/ports/Mk/bsd.autotools.mk
/usr/ports/Mk/bsd.cmake.mk
...
```

To update an already installed Ports Collection, use `portsnap update`:

```
# portsnap update
```

The process is now complete, and applications may be installed or upgraded using the updated Ports Collection.

When using `fetch`, the `extract` or the `update` operation may be run consecutively:

```
# portsnap fetch update
```

This command downloads the latest version of the Ports Collection and updates the local version under `/usr/ports`.

25.4 Updating the Documentation Set

Documentation is an integral part of the FreeBSD operating system. While an up-to-date version of the FreeBSD Documentation Set is always available on the FreeBSD web site (http://www.freebsd.org/doc/), some users might have slow or no permanent network connectivity. There are several ways to update the local copy of documentation with the latest FreeBSD Documentation Set.

25.4.1 Using Subversion to Update the Documentation

The FreeBSD documentation sources can be obtained with **svn**. This section describes how to:

- Install the documentation toolchain, the tools that are required to rebuild the FreeBSD documentation from its source.
- Download a copy of the documentation source at `/usr/doc`, using **svn**.
- Rebuild the FreeBSD documentation from its source, and install it under `/usr/share/doc`.
- Recognize some of the build options that are supported by the build system of the documentation, such as the options that build only some of the different language translations of the documentation or the options that select a specific output format.

25.4.2 Installing svn and the Documentation Toolchain

Rebuilding the FreeBSD documentation from source requires a collection of tools which are not part of the FreeBSD base system due to the amount of disk space these tools use. They are also not useful to all FreeBSD users, only those users that are actively writing new documentation for FreeBSD or are frequently updating their documentation from source.

The required tools, including **svn**, are available in the `textproc/docproj` meta-port developed by the FreeBSD Documentation Project.

Note: When no PostScript or PDF documentation required, one might consider installing the `textproc/docproj-nojadetex` port instead. This version of the documentation toolchain includes everything except the **teTeX** typesetting engine. **teTeX** is a very large collection of tools, so it may be quite sensible to omit its installation if PDF output is not really necessary.

25.4.3 Updating the Documentation Sources

In this example, **svn** is used to fetch a clean copy of the documentation sources from the western US mirror using the HTTPS protocol:

```
# svn checkout https://svn0.us-west.FreeBSD.org/doc/head /usr/doc
```

Select the closest mirror from the available Subversion mirror sites.

The initial download of the documentation sources may take a while. Let it run until it completes.

Future updates of the documentation sources may be fetched by running:

```
# svn update /usr/doc
```

After checking out the sources, an alternative way of updating the documentation is supported by the `/usr/doc/Makefile` by running the following commands:

```
# cd /usr/doc
# make update
```

25.4.4 Tunable Options of the Documentation Sources

The updating and build system of the FreeBSD documentation set supports a few options that ease the process of updating only parts of the documentation, or the build of specific translations. These options can be set either as system-wide options in `/etc/make.conf`, or as command-line options passed to make(1).

The options include:

`DOC_LANG`

The list of languages and encodings to build and install, such as `en_US.ISO8859-1` for English documentation.

`FORMATS`

A single format or a list of output formats to be built. Currently, `html`, `html-split`, `txt`, `ps`, `pdf`, and `rtf` are supported.

`DOCDIR`

Where to install the documentation. It defaults to `/usr/share/doc`.

For more `make` variables supported as system-wide options in FreeBSD, refer to make.conf(5).

For more `make` variables supported by the build system of the FreeBSD documentation, refer to the FreeBSD Documentation Project Primer for New Contributors (http://www.FreeBSD.org/doc/en_US.ISO8859-1/books/fdp-primer).

25.4.5 Installing the FreeBSD Documentation from Source

Once an up-to-date snapshot of the documentation sources has been fetched to `/usr/doc`, everything is ready for an update of the installed documentation.

A full update of all the languages defined in `DOC_LANG` may be performed by typing:

```
# cd /usr/doc
# make install clean
```

If an update of only a specific language is desired, make(1) can be invoked in a language specific subdirectory of `/usr/doc`:

```
# cd /usr/doc/en_US.ISO8859-1
# make update install clean
```

The output formats that will be installed may be specified by setting `FORMATS`:

```
# cd /usr/doc
# make FORMATS='html html-split' install clean
```

For information on editing and submitting corrections to the documentation, refer to the FreeBSD Documentation Project Primer for New Contributors (http://www.FreeBSD.org/doc/en_US.ISO8859-1/books/fdp-primer).

25.4.6 Using Documentation Ports

Based on the work of Marc Fonvieille.

The previous section presented a method for updating the FreeBSD documentation from sources. Source based updates may not be feasible or practical for all FreeBSD systems as building the documentation sources requires the *documentation toolchain*, a certain level of familiarity with **svn** and source checkouts from a repository, and a few manual steps to build the checked out sources. This section describes an alternative method which uses the Ports Collection and makes it possible to:

- Download and install pre-built snapshots of the documentation, without having to locally build anything or install the documentation toolchain.
- Download the documentation sources and build them through the ports framework, making the checkout and build steps a bit easier.

These two methods of updating the FreeBSD documentation are supported by a set of *documentation ports*, updated by the Documentation Engineering Team <`doceng@FreeBSD.org`> on a monthly basis. These are listed in the FreeBSD Ports Collection, under the docs (http://www.freshports.org/docs/) category.

25.4.6.1 Building and Installing Documentation Ports

The documentation ports use the ports building framework to make documentation builds easier. They automate the process of checking out the documentation source, running make(1) with the appropriate environment settings and command-line options, and they make the installation or deinstallation of documentation as easy as the installation of any other FreeBSD port or package.

> **Note:** As an extra feature, when the documentation ports are built locally, they record a dependency to the *documentation toolchain* ports, so that they are also automatically installed.

Organization of the documentation ports is as follows:

- The “master port”, `misc/freebsd-doc-en`, which installs all of the English documentation ports.
- The “all in one port”, `misc/freebsd-doc-all`, builds and installs all documentation in all available languages.
- There is a “slave port” for each translation, such as `misc/freebsd-doc-hu` for the Hungarian-language documents.

For example, to build and install the English documentation in split HTML format, similar to the format used on http://www.FreeBSD.org, to `/usr/local/share/doc/freebsd`, install the following port

```
# cd /usr/ports/misc/freebsd-doc-en
# make install clean
```

25.4.6.1.1 Common Knobs and Options

There are many options for modifying the default behavior of the documentation ports, including:

`WITH_HTML`

Builds the HTML format with a single HTML file per document. The formatted documentation is saved to a file called `article.html`, or `book.html`, as appropriate, plus images.

`WITH_PDF`

Builds the Adobe Portable Document Format (PDF). The formatted documentation is saved to a file called `article.pdf` or `book.pdf`, as appropriate.

`DOCBASE`

Specifies where to install the documentation. It defaults to `/usr/local/share/doc/freebsd`.

> **Note:** The default target directory differs from the directory used **svn**. This is because ports are usually installed within `/usr/local`. This can be overridden by using `PREFIX`.

This example uses variables to install the Hungarian documentation as a PDF:

```
# cd /usr/ports/misc/freebsd-doc-hu
# make -DWITH_PDF DOCBASE=share/doc/freebsd/hu install clean
```

25.4.6.2 Using Documentation Packages

Building the documentation ports from source, as described in the previous section, requires a local installation of the documentation toolchain and a bit of disk space for the build of the ports. When resources are not available to install the documentation toolchain, or because the build from sources would take too much disk space, it is still possible to install pre-built snapshots of the documentation ports.

The Documentation Engineering Team <doceng@FreeBSD.org> prepares monthly snapshots of the FreeBSD documentation packages. These binary packages can be used with any of the bundled package tools, like pkg_add(1), pkg_delete(1), and so on.

> **Note:** When binary packages are used, the FreeBSD documentation will be installed in *all* available formats for the given language.

For example, the following command will install the latest pre-built package of the Hungarian documentation:

```
# pkg_add -r hu-freebsd-doc
```

> **Note:** Packages use a format that differs from the corresponding port's name: *lang*-freebsd-doc, where *lang* is the short format of the language code, such as hu for Hungarian, or zh_cn for Simplified Chinese.

25.4.6.3 Updating Documentation Ports

Documentation ports can be updated like any other port. For example, the following command updates the installed Hungarian documentation using ports-mgmt/portmaster by using packages only:

```
# portmaster -PP hu-freebsd-doc
```

25.5 Tracking a Development Branch

There are two development branches to FreeBSD: FreeBSD-CURRENT and FreeBSD-STABLE. This section provides an explanation of each and describes how to keep a system up-to-date with each respective tree. FreeBSD-CURRENT will be discussed first, then FreeBSD-STABLE.

25.5.1 Staying Current with FreeBSD

FreeBSD-CURRENT is the "bleeding edge" of FreeBSD development. FreeBSD-CURRENT users are expected to have a high degree of technical skill and should be capable of solving difficult system problems on their own. If you are new to FreeBSD, track FreeBSD-STABLE instead.

25.5.1.1 What Is FreeBSD-CURRENT?

FreeBSD-CURRENT is the latest working sources for FreeBSD. This includes work in progress, experimental changes, and transitional mechanisms that might or might not be present in the next official release of the software. While many FreeBSD developers compile the FreeBSD-CURRENT source code daily, there are periods of time when the sources are not buildable. These problems are resolved as quickly as possible, but whether or not FreeBSD-CURRENT brings disaster or greatly desired functionality can be a matter of when the source code was synced

25.5.1.2 Who Needs FreeBSD-CURRENT?

FreeBSD-CURRENT is made available for three primary interest groups:

1. Members of the FreeBSD community who are actively working on some part of the source tree and for whom keeping “current” is an absolute requirement.
2. Members of the FreeBSD community who are active testers, willing to spend time solving problems in order to ensure that FreeBSD-CURRENT remains as sane as possible. These testers wish to make topical suggestions on changes and the general direction of FreeBSD, and submit patches to implement them.
3. Those who merely wish to keep an eye on things, or to use the current sources for reference purposes. These people also make the occasional comment or contribute code.

25.5.1.3 What Is FreeBSD-CURRENT *Not*?

1. A fast-track to getting new features before the next release. Pre-release features are not yet fully tested and most likely contain bugs.
2. A quick way of getting bug fixes, though the fix is just as likely to introduce new bugs as to fix existing ones.
3. In no way “officially supported”.

25.5.1.4 Using FreeBSD-CURRENT

1. Join the freebsd-current (http://lists.FreeBSD.org/mailman/listinfo/freebsd-current) and the svn-src-head (http://lists.FreeBSD.org/mailman/listinfo/svn-src-head) lists. This is *essential* in order to see the comments that people are making about the current state of the system and to receive important bulletins which may be critical to the system’s continued health.

   The svn-src-head (http://lists.FreeBSD.org/mailman/listinfo/svn-src-head) list records the commit log entry for each change as it is made, along with any pertinent information on possible side-effects.

   To join these lists, go to http://lists.FreeBSD.org/mailman/listinfo, click on the list to subscribe to, and follow the instructions. In order to track changes to the whole source tree, subscribe to the svn-src-all (http://lists.FreeBSD.org/mailman/listinfo/svn-src-all) list.
2. Grab the sources from a FreeBSD mirror site using one of the following methods:
   a. Use svn to check out the desired development or release branch. This is the recommended method, providing access to FreeBSD development as it occurs. Checkout the -CURRENT code from the `head`

branch of one of the Subversion mirror sites. Due to the size of the repository, it is recommended that only desired subtrees be checked out.

b.

Use the **CTM** facility. If you have bad connectivity such as high price connections or only email access, **CTM** is an option, but it is not as reliable as **Subversion**. For this reason, **Subversion** is the recommended method for any system with Internet connectivity.

3. If you plan to run, and not just look at the sources, download *all* of FreeBSD-CURRENT, not just selected portions. Various parts of the source depend on updates elsewhere, and trying to compile just a subset is almost guaranteed to cause problems.

   Before compiling FreeBSD-CURRENT, read `/usr/src/Makefile` very carefully.
   Install a new kernel and rebuild the world the first time through as part of the upgrading process. Read the FreeBSD-CURRENT mailing list (http://lists.FreeBSD.org/mailman/listinfo/freebsd-current) and `/usr/src/UPDATING` to stay up-to-date on other bootstrapping procedures that sometimes become necessary on the road to the next release.

4. Be active! FreeBSD-CURRENT users are encouraged to submit their suggestions for enhancements or bug fixes. Suggestions with accompanying code are received most enthusiastically!

25.5.2 Staying Stable with FreeBSD

25.5.2.1 What Is FreeBSD-STABLE?

FreeBSD-STABLE is the development branch from which major releases are made. Changes go into this branch at a different pace, and with the general assumption that they have first gone into FreeBSD-CURRENT for testing. This is *still* a development branch, however, and this means that at any given time, the sources for FreeBSD-STABLE may or may not be suitable for any particular purpose. It is simply another engineering development track, not a resource for end-users.

25.5.2.2 Who Needs FreeBSD-STABLE?

Those interested in tracking or contributing to the FreeBSD development process, especially as it relates to the next “point” release of FreeBSD, should consider following FreeBSD-STABLE.

While security fixes go into the FreeBSD-STABLE branch, one does not *need* to track FreeBSD-STABLE to receive security fixes. Every security advisory for FreeBSD explains how to fix the problem for the releases it affects which are not yet EOL. [1].

While the FreeBSD-STABLE branch should compile and run at all times, this cannot be guaranteed. While code is developed in FreeBSD-CURRENT before including it in FreeBSD-STABLE, more people run FreeBSD-STABLE than FreeBSD-CURRENT, so it is inevitable that bugs and corner cases will sometimes be found in FreeBSD-STABLE that were not apparent in FreeBSD-CURRENT.

For these reasons, one should *not* blindly track FreeBSD-STABLE. It is particularly important not to update any production servers to FreeBSD-STABLE without first thoroughly testing the code in that development environment.

Except for those users who have the resources to perform testing, it is recommended that users instead run the most recent release of FreeBSD, and use the binary update mechanism to move from release to release.

25.5.2.3 Using FreeBSD-STABLE

1. Join the freebsd-stable (http://lists.FreeBSD.org/mailman/listinfo/freebsd-stable) list in order to stay informed of build-dependencies that may appear in FreeBSD-STABLE or any other issues requiring special attention. Developers will also make announcements in this mailing list when they are contemplating some controversial fix or update, giving the users a chance to respond if they have any issues to raise concerning the proposed change.

   Join the relevant **SVN** list for the branch being tracked. For example, users tracking the 9-STABLE branch should join the svn-src-stable-9 (http://lists.FreeBSD.org/mailman/listinfo/svn-src-stable-9) list. This list records the commit log entry for each change as it is made, along with any pertinent information on possible side-effects.

   To join these lists, go to http://lists.FreeBSD.org/mailman/listinfo, click on the list to subscribe to, and follow the instructions. In order to track changes for the whole source tree, subscribe to svn-src-all (http://lists.FreeBSD.org/mailman/listinfo/svn-src-all).

2. To install a new system in order to run monthly snapshots built from FreeBSD-STABLE, refer to Snapshots (http://www.FreeBSD.org/snapshots/)for more information. Alternatively, it is possible to install the most recent FreeBSD-STABLE release from the mirror sites and follow the instructions below to upgrade the system to the most up-to-date FreeBSD-STABLE source code.

   Several methods are available to upgrade from a FreeBSD mirror site on a system already running a previous release of FreeBSD:

   a. Use svn to check out the desired development or release branch. This is the recommended method, providing access to FreeBSD development as it occurs. Branch names include `head` for the current development head, and branches identified in the release engineering page (http://www.FreeBSD.org/releng/), such as `stable/9` or `releng/9.0`. URL prefixes for **Subversion** checkout of the base system are shown in Subversion mirror sites. Because of the size of the repository, it is recommended that only desired subtrees be checked out.

   b.

      Consider using **CTM** if you do not have a fast connection to the Internet.

3. If you need rapid on-demand access to the source and communications bandwidth is not a consideration, use **Subversion**. Otherwise, use **CTM**.

4.

   Before compiling FreeBSD-STABLE, read `/usr/src/Makefile` carefully.
   Install a new kernel and rebuild the world the first time through as part of the upgrading process. Read FreeBSD-STABLE mailing list (http://lists.FreeBSD.org/mailman/listinfo/freebsd-stable) and `/usr/src/UPDATING` to keep up-to-date on other bootstrapping procedures that sometimes become necessary on the road to the next release.

25.6 Synchronizing Source

There are various ways of using an Internet or email connection to stay up-to-date with any given area, or all areas, of the FreeBSD project sources. The primary services are Subversion and CTM.

> **Warning:** While it is possible to update only parts of the source tree, the only supported update procedure is to update the entire tree and recompile all the programs that run in user space, such as those in `/bin` and `/sbin`, and kernel sources. Updating only part of the source tree, only the kernel, or only the userland programs will often result in problems ranging from compile errors to kernel panics or data corruption.

Subversion uses the *pull* model of updating sources. The user, or a `cron` script, invokes the `svn` program, and it brings files up-to-date. **Subversion** is the preferred means of updating local source trees. The updates are up-to-the-minute and the user controls when they are downloaded. It is easy to restrict updates to specific files or directories and the requested updates are generated on the fly by the server.

CTM does not interactively compare the local sources with those on the master archive or otherwise pull them across. Instead, a script which identifies changes in files since its previous run is executed several times a day on the master CTM machine. Any detected changes are compressed, stamped with a sequence-number, and encoded for transmission over email in printable ASCII only. Once received, these “CTM deltas” can then be handed to the ctm_rmail(1) utility which will automatically decode, verify, and apply the changes to the user’s copy of the sources. This process is more efficient than **Subversion** and places less strain on server resources since it is a *push* rather than a *pull* model.

There are other trade-offs. If a user inadvertently wipes out portions of the local archive, **Subversion** will detect and rebuild the damaged portions. **CTM** will not do this, and if a user deletes some portion of the source tree and does not have a backup, they will have to start from scratch from the most recent CTM “base delta” and rebuild it all with **CTM**.

25.7 Rebuilding “world”

Once the local source tree is synchronized against a particular version of FreeBSD such as FreeBSD-STABLE or FreeBSD-CURRENT, the source tree can be used to rebuild the system.

> **Make a Backup:** It cannot be stressed enough how important it is to make a backup of the system *before* rebuilding the system. While rebuilding the world is an easy task, there will inevitably be times when mistakes in the source tree render the system unbootable.
>
> Create and verify a backup and have a bootable installation media at hand. You will probably never have to use it, but it is better to be safe than sorry!

> **Subscribe to the Right Mailing List:** The FreeBSD-STABLE and FreeBSD-CURRENT branches are, by their nature, *in development*. People that contribute to FreeBSD are human, and mistakes occasionally happen.
>
> Sometimes these mistakes can be quite harmless, just causing the system to print a new diagnostic warning. Or the change may be catastrophic, and render the system unbootable or destroy file systems.
>
> When problems occur, a “heads up” is posted to the appropriate mailing list, explaining the nature of the problem and which systems it affects. An “all clear” announcement is posted when the problem has been solved.

Users who track FreeBSD-STABLE or FreeBSD-CURRENT and do not read FreeBSD-STABLE mailing list (http://lists.FreeBSD.org/mailman/listinfo/freebsd-stable) or FreeBSD-CURRENT mailing list (http://lists.FreeBSD.org/mailman/listinfo/freebsd-current) respectively, are asking for trouble.

Do Not Use `make world`: Some older documentation recommends using `make world`. However, that command skips some important steps and should only be used by experts. For almost all circumstances `make world` is the wrong thing to do, and the procedure described here should be used instead.

25.7.1 The Canonical Way to Update Your System

Before updating the system, read `/usr/src/UPDATING` for any pre-buildworld steps necessary for that version of the sources. Then, use the procedure outlined here.

These upgrade steps assume an upgrade from an older FreeBSD version, consisting of an old compiler, old kernel, old world, and old configuration files. “World” includes the core system binaries, libraries, and programming files. The compiler is part of “world”, but has a few special concerns.

These steps also assume that the sources to a newer version have already been obtained. If the sources are not up-to-date, refer to Section 25.6 for detailed help about synchronizing to a newer version.

Updating the system from source is a more subtle process than it might initially seem to be, and the FreeBSD developers have found it necessary over the years to change the recommended approach fairly dramatically as new kinds of unavoidable dependencies come to light. The rest of this section describes the rationale behind the currently recommended upgrade sequence.

Any successful update sequence must deal with the following issues:

- The old compiler might have a bug and not be able to compile the new kernel. So, the new kernel should be built with the new compiler, meaning that the new compiler must be built before the new kernel is built. This does not necessarily mean that the new compiler must be *installed* before building the new kernel.
- The new world might rely on new kernel features. So, the new kernel must be installed before the new world is installed.

These first two issues are the basis for the core `buildworld`, `buildkernel`, `installkernel`, `installworld` sequence described in the following paragraphs. Other reasons for using these steps are listed below:

- The old world might not run correctly on the new kernel, so the new world must be installed immediately upon installing the new kernel.
- Some configuration changes must be made before the new world is installed, but others might break the old world. Hence, two different configuration upgrade steps are generally needed.
- For the most part, the update process only replaces or adds files and existing old files are not deleted. In a few cases, this can cause problems. As a result, the update procedure will sometimes specify certain files that should be manually deleted at certain steps. This may or may not be automated in the future.

These concerns have led to the following recommended sequence. Note that the detailed sequence for particular updates may require additional steps, but this core process should remain unchanged for some time:

1. `make buildworld`

   This first compiles the new compiler and a few related tools, then uses the new compiler to compile the rest of the new world. The result ends up in `/usr/obj`.

2. `make buildkernel`

   This uses the *new* compiler residing in `/usr/obj` in order to protect against compiler-kernel mismatches.

3. `make installkernel`

   Place the new kernel and kernel modules onto the disk, making it possible to boot with the newly updated kernel.

4. Reboot into single user mode.

   Single user mode minimizes problems from updating software that is already running. It also minimizes any problems from running the old world on a new kernel.

5. `mergemaster -p`

   This does some initial configuration file updates in preparation for the new world. For instance, it may add new user groups to the system, or new user names to the password database. This is often necessary when new groups or special system-user accounts have been added since the last update, so that the `installworld` step will be able to use the newly installed system user or system group names without problems.

6. `make installworld`

   Copies the world from `/usr/obj`. The new kernel and new world are now installed on disk.

7. `mergemaster`

   Repeated to update the remaining configuration files, now that the new world is on disk.

8. `make delete-old`

   This target deletes old (obsolete) files. This is important because sometimes they cause problems if left on the disk, for example the presence of the old `utmp.h` causes problems in some ports when the new `utmpx.h` is installed.

9. Reboot.

   A full machine reboot is needed now to load the new kernel and new world with new configuration files.

10. `make delete-old-libs`

    Remove any obsolete libraries to avoid conflicts with newer ones. Make sure that all ports have been rebuilt before old libraries are removed.

Upgrades from one release of the same FreeBSD branch to a more recent release of the same branch, such as from 9.0 to 9.1, may not need this procedure since it is less likely to run into serious mismatches between compiler, kernel, userland, and configuration files. The approach of `make world` followed by building and installing a new kernel might work well enough for minor updates.

When upgrading across major releases, people who do not follow this procedure should expect some problems.

It is also worth noting that many upgrades may require specific additional steps such as renaming or deleting specific files prior to installworld. Read `/usr/src/UPDATING` carefully, especially at the end, where the currently recommended upgrade sequence is explicitly spelled out.

This procedure has evolved over time as the developers have found it impossible to completely prevent certain kinds of mismatch problems. Hopefully, the current procedure will remain stable for a long time.

To summarize, the currently recommended way of upgrading FreeBSD from sources is:

```
# cd /usr/src
# make buildworld
# make buildkernel
# make installkernel
# shutdown -r now
```

> **Note:** There are a few rare cases when an extra run of `mergemaster -p` is needed before the `buildworld` step. These are described in `UPDATING`. In general, though, this step can safely be omitted when not updating across one or more major FreeBSD versions.

After `installkernel` finishes successfully, boot into single user mode using `boot -s` from the loader prompt. Then run:

```
# mount -u /
# mount -a -t ufs
# adjkerntz -i
# mergemaster -p
# cd /usr/src
# make installworld
# mergemaster
# make delete-old
# reboot
# make delete-old-libs
```

> **Read Further Explanations:** The following sections clearly describe each step, especially when using a custom kernel configuration.

25.7.2 Read `/usr/src/UPDATING`

Before updating, read `/usr/src/UPDATING`. This file contains important information about potential problems and may specify the order to run certain commands. If `UPDATING` contradicts the procedure in this section, `UPDATING` takes precedence.

> **Important:** Reading `UPDATING` is not an acceptable substitute for subscribing to the correct mailing list. The two requirements are complementary, not exclusive.

25.7.3 Check `/etc/make.conf`

Available make(1) options are shown in make.conf(5) and `/usr/share/examples/etc/make.conf`. These settings can be added to `/etc/make.conf` to control the way make(1) runs and how it builds programs. Changes to some settings can have far-reaching and potentially surprising effects. Read the comments in both locations and keep in mind that the defaults have been chosen for a combination of performance and safety.

Options set in `/etc/make.conf` take effect every time make(1) is used, including compiling applications from the Ports Collection or user-written C programs, or building the FreeBSD operating system.

25.7.4 Check `/etc/src.conf`

`/etc/src.conf` controls the building of the operating system from source code. Unlike `/etc/make.conf`, the contents of `/etc/src.conf` only take effect when the FreeBSD operating system itself is being built. Descriptions of the many options available for this file are shown in src.conf(5). Be cautious about disabling seemingly unneeded kernel modules and build options. Sometimes there are unexpected or subtle interactions.

25.7.5 Update the Files in `/etc`

`/etc` contains a large part of the system's configuration information, as well as scripts that are run at system startup. Some of these scripts change between FreeBSD versions.

Some of the configuration files are used in the day to day running of the system, such as `/etc/group`.

There have been occasions when the installation part of `make installworld` expected certain usernames or groups to exist. When performing an upgrade, it is likely that these users or groups do not yet exist. In some cases `make buildworld` will check to see if these users or groups exist.

The solution is to run mergemaster(8) in pre-buildworld mode with `-p`. This compares only those files that are essential for the success of `buildworld` or `installworld`.

> **Tip:** To check which files are owned by the group being renamed or deleted:
>
> ```
> # find / -group GID -print
> ```
>
> This command will show all files owned by group *GID*, which can be either a group name or a numeric group ID.

25.7.6 Drop to Single User Mode

Consider compiling the system in single user mode. Reinstalling the system touches a lot of important system files, all the standard system binaries, libraries, and include files. Changing these on a running system, particularly one with active users, is asking for trouble.

Another method is to compile the system in multi-user mode, and then drop into single user mode for the installation. With this method, hold off on the following steps until the build has completed. Drop to single user mode in order to run `installkernel` or `installworld`.

To enter single user mode from a running system:

```
# shutdown now
```

Alternatively, reboot the system, and at the boot prompt, select the "single user" option. Once at the single user mode shell prompt, run:

```
# fsck -p
# mount -u /
```

```
# mount -a -t ufs
# swapon -a
```

This checks the file systems, remounts / read/write, mounts all the other UFS file systems referenced in /etc/fstab, and turns swapping on.

> **Note:** If the CMOS clock is set to local time and not to GMT (this is true if the output of date(1) does not show the correct time and zone), run the following command:
>
> ```
> # adjkerntz -i
> ```
>
> This ensures that the local time-zone settings get set up correctly.

25.7.7 Remove /usr/obj

As parts of the system are rebuilt, they are, by default, placed in subdirectories of /usr/obj. The directories shadow those under /usr/src.

To speed up the make buildworld process, and possibly save some dependency headaches, remove this directory if it already exists.

Some files below /usr/obj may have the immutable flag set which must be removed first using chflags(1).

```
# cd /usr/obj
# chflags -R noschg *
# rm -rf *
```

25.7.8 Recompile the Base System

25.7.8.1 Saving the Output

It is a good idea to save the output from running make(1) to a file. If something goes wrong, a copy of the error message can be posted to one of the FreeBSD mailing lists.

The easiest way to do this is to use script(1) with a parameter that specifies the name of the file to save all output to. Run this command immediately before rebuilding the world, and then type **exit** when the process has finished:

```
# script /var/tmp/mw.out
Script started, output file is /var/tmp/mw.out
# make TARGET
... compile, compile, compile ...
# exit
Script done, ...
```

Do not save the output in /tmp as this directory may be cleared at next reboot. A better place to save the file is /var/tmp or in root's home directory.

25.7.8.2 Compile the Base System

While in `/usr/src` type:

```
# cd /usr/src
```

To rebuild the world, use make(1). This command reads instructions from the `Makefile`, which describes how the programs that comprise FreeBSD should be built and the order in which they should be built.

The general format of the command is as follows:

```
# make -x -DVARIABLE target
```

In this example, `-x` is an option passed to make(1). Refer to make(1) for an examples of available options.

`-DVARIABLE` passes a variable to the `Makefile`. The behavior of the `Makefile` is controlled by these variables. These are the same variables as are set in `/etc/make.conf`, and this provides another way of setting them. For example:

```
# make -DNO_PROFILE target
```

is another way of specifying that profiled libraries should not be built, and corresponds with the

```
NO_PROFILE=    true     #    Avoid compiling profiled libraries
```

line in `/etc/make.conf`.

`target` tells make(1) what to do. Each `Makefile` defines a number of different “targets”, and the choice of target determines what happens.

Some targets listed in the `Makefile` are used by the build process to break out the steps necessary to rebuild the system into a number of sub-steps.

Most of the time, no parameters need to be passed to make(1) and the command looks like this:

```
# make target
```

Where `target` is one of many build options. The first target should always be `buildworld`.

As the names imply, `buildworld` builds a complete new tree under `/usr/obj` and `installworld` installs this tree on the current machine.

Having separate options is useful for two reasons. First, it allows for a “self hosted” build that does not affect any components of a running system. Because of this, `buildworld` can be run on a machine running in multi-user mode with no fear of ill-effects. It is still recommended that `installworld` be run in part in single user mode, though.

Secondly, it allows NFS mounts to be used to upgrade multiple machines on a network. If order to upgrade three machines, `A`, `B` and `C`, run `make buildworld` and `make installworld` on `A`. `B` and `C` should then NFS mount `/usr/src` and `/usr/obj` from `A`, and run `make installworld` to install the results of the build on `B` and `C`.

Although the `world` target still exists, users are strongly encouraged not to use it.

Instead, run:

```
# make buildworld
```

It is possible to specify `-j` which will cause `make` to spawn several simultaneous processes. This is most useful on multi-CPU machines. However, since much of the compiling process is I/O bound rather than CPU bound, it is also useful on single CPU machines.

On a typical single-CPU machine, run:

```
# make -j4 buildworld
```

make(1) will then have up to 4 processes running at any one time. Empirical evidence posted to the mailing lists shows this generally gives the best performance benefit.

On a multi-CPU machine using an SMP configured kernel, try values between 6 and 10 and see how they speed things up.

25.7.8.3 Timings

Many factors influence the build time, but fairly recent machines may only take a one or two hours to build the FreeBSD-STABLE tree, with no tricks or shortcuts used during the process. A FreeBSD-CURRENT tree will take somewhat longer.

25.7.9 Compile and Install a New Kernel

To take full advantage of the new system, recompile the kernel. This is practically a necessity, as certain memory structures may have changed, and programs like ps(1) and top(1) will fail to work until the kernel and source code versions are the same.

The simplest, safest way to do this is to build and install a kernel based on `GENERIC`. While `GENERIC` may not have all the necessary devices for the system, it should contain everything necessary to boot the system back to single user mode. This is a good test that the new system works properly. After booting from `GENERIC` and verifying that the system works, a new kernel can be built based on a custom kernel configuration file.

On FreeBSD it is important to build world before building a new kernel.

> **Note:** To build a custom kernel with an existing customized configuration file, use `KERNCONF=`*`MYKERNEL`*:
>
> ```
> # cd /usr/src
> # make buildkernel KERNCONF=MYKERNEL
> # make installkernel KERNCONF=MYKERNEL
> ```

If `kern.securelevel` has been raised above 1 *and* `noschg` or similar flags have been set on the kernel binary, drop into single user mode to use `installkernel`. Otherwise, both these commands can be run from multi user mode without problems. See init(8) for details about `kern.securelevel` and chflags(1) for details about the various file flags.

25.7.10 Reboot into Single User Mode

Reboot into single user mode to test that the new kernel works using the instructions in Section 25.7.6.

25.7.11 Install the New System Binaries

Next, use `installworld` to install the new system binaries:

```
# cd /usr/src
# make installworld
```

> **Note:** If variables were specified to `make buildworld`, specify the same variables to `make installworld`. However, `-j` must never be used with `installworld`.
>
> For example, if you ran:
>
> ```
> # make -DNO_PROFILE buildworld
> ```
>
> install the results with:
>
> ```
> # make -DNO_PROFILE installworld
> ```
>
> otherwise, the command will try to install profiled libraries that were not built during the `make buildworld` phase.

25.7.12 Update Files Not Updated by `make installworld`

Remaking the world will not update certain directories, such as `/etc`, `/var` and `/usr`, with new or changed configuration files.

The simplest way to update the files in these directories is to use mergemaster(8). Be sure to first make a backup of `/etc` in case anything goes wrong.

25.7.12.1 `mergemaster`

Contributed by Tom Rhodes.

mergemaster(8) is a Bourne script to aid in determining the differences between the configuration files in `/etc`, and the configuration files in the source tree `/usr/src/etc`. This is the recommended solution for keeping the system configuration files up to date with those located in the source tree.

To begin, type `mergemaster` and it will build a temporary root environment, from `/` down, and populate it with various system configuration files. Those files are then compared to the ones currently installed in the system. Files that differ will be shown in diff(1) format, with the + sign representing added or modified lines, and `-` representing lines that will be either removed completely, or replaced with a new file. Refer to diff(1) for more information about the diff(1) syntax and how file differences are shown.

mergemaster(8) will then display each file that differs, and present the options of either deleting the new file, referred to as the temporary file, installing the temporary file in its unmodified state, merging the temporary file with the currently installed file, or viewing the diff(1) results again.

Choosing to delete the temporary file will tell mergemaster(8) to keep the current file unchanged and to delete the new version. This option is not recommended, unless there is no reason to change the current file. To get help at any time, type **?** at the mergemaster(8) prompt. If the user chooses to skip a file, it will be presented again after all other files have been dealt with.

Choosing to install the unmodified temporary file will replace the current file with the new one. For most unmodified files, this is the best option.

Choosing to merge the file will present a text editor, and the contents of both files. The files can be merged by reviewing both files side by side on the screen, and choosing parts from both to create a finished product. When the files are compared side by side, **l** selects the left contents and **r** selects contents from the right. The final output will be a file consisting of both parts, which can then be installed. This option is customarily used for files where settings have been modified by the user.

Choosing to view the diff(1) results again will display the file differences just like mergemaster(8) did before prompting an option.

After mergemaster(8) is done with the system files, it will prompt for other options. mergemaster(8) may prompt to rebuild the password file and will finish up with an option to remove left-over temporary files.

25.7.12.2 Manual Update

To perform the update manually instead, do not just copy over the files from `/usr/src/etc` to `/etc` and expect it to work. Some files must be "installed" first as `/usr/src/etc` *is not* a copy of what `/etc` should look like. In addition, some files that should be in `/etc` are not in `/usr/src/etc`.

If you are using mergemaster(8) (as recommended), you can skip forward to the next section.

The simplest way to merge files by hand is to install the files into a new directory, and then work through them looking for differences.

> **Backup Your Existing `/etc`:** It is recommended to first copy the existing `/etc` somewhere safe, like so:
>
> ```
> # cp -Rp /etc /etc.old
> ```
>
> where `-R` does a recursive copy and `-p` preserves times and the ownerships on files.

Build a temporary set of directories into which the new `/etc` and other files can be installed:

```
# mkdir /var/tmp/root
# cd /usr/src/etc
# make DESTDIR=/var/tmp/root distrib-dirs distribution
```

This will build the necessary directory structure and install the files. A lot of the subdirectories that have been created under `/var/tmp/root` are empty and should be deleted. The simplest way to do this is to:

```
# cd /var/tmp/root
# find -d . -type d | xargs rmdir 2>/dev/null
```

This will remove all empty directories while redirecting standard error to `/dev/null` to prevent the warnings about the directories that are not empty.

`/var/tmp/root` now contains all the files that should be placed in appropriate locations below `/`. Go through each of these files, determining how they differ from the system's existing files.

Some of the files installed into `/var/tmp/root` have a leading ".". Make sure to use `ls -a` in order to catch them.

The simplest way to compare files is to use diff(1):

```
# diff /etc/shells /var/tmp/root/etc/shells
```

This command will show the differences between the existing `/etc/shells` and the new `/var/tmp/root/etc/shells`. Review the differences to decide whether to merge in custom changes or to replace the existing file with the new one.

Name the New Root Directory (`/var/tmp/root`) with a Time Stamp, so You Can Easily Compare Differences Between Versions: Frequently rebuilding world entails frequently updating `/etc` as well, which can be a bit of a chore.

To speed up this process, use the following procedure to keep a copy of the last set of changed files that were merged into `/etc`.

1. Make the world as normal. When updating `/etc` and the other directories, give the target directory a name based on the current date:

   ```
   # mkdir /var/tmp/root-20130214
   # cd /usr/src/etc
   # make DESTDIR=/var/tmp/root-20130214 \
       distrib-dirs distribution
   ```

2. Merge in the changes from this directory as outlined above. *Do not* remove the `/var/tmp/root-20130214` directory when you have finished.
3. After downloading the latest version of the source and remaking it, follow step 1. Create a new directory, which reflects the new date. This example uses `/var/tmp/root-20130221`.
4. Use diff(1) to see the differences that have been made in the intervening week by creating a recursive diff between the two directories:

   ```
   # cd /var/tmp
   # diff -r root-20130214 root-20130221
   ```

   Typically, this will be a much smaller set of differences than those between `/var/tmp/root-20130221/etc` and `/etc`. Because the set of differences is smaller, it is easier to migrate those changes across into `/etc`.
5. When finished, remove the older of the two `/var/tmp/root-*` directories:

   ```
   # rm -rf /var/tmp/root-20130214
   ```

6. Repeat this process whenever merging in changes to `/etc`.

Use date(1) to automate the generation of the directory names:

```
# mkdir /var/tmp/root-‘date "+%Y%m%d"‘
```

25.7.13 Deleting Obsolete Files and Directories

Based on notes provided by Anton Shterenlikht.

As a part of the FreeBSD development lifecycle, files and their contents occasionally become obsolete. This may be because functionality is implemented elsewhere, the version number of the library has changed, or it was removed from the system entirely. This includes old files, libraries, and directories, which should be removed when updating the system. The benefit is that the system is not cluttered with old files which take up unnecessary space on the storage and backup media. Additionally, if the old library has a security or stability issue, the system should be

updated to the newer library to keep it safe and to prevent crashes caused by the old library. Files, directories, and libraries which are considered obsolete are listed in `/usr/src/ObsoleteFiles.inc`. The following instructions should be used to remove obsolete files during the system upgrade process.

After the `make installworld` and the subsequent `mergemaster` have finished successfully, check for obsolete files and libraries as follows:

```
# cd /usr/src
# make check-old
```

If any obsolete files are found, they can be deleted using the following command:

```
# make delete-old
```

> **Tip:** Refer to `/usr/src/Makefile` for more targets of interest.

A prompt is displayed before deleting each obsolete file. To skip the prompt and let the system remove these files automatically, use `BATCH_DELETE_OLD_FILES`:

```
# make -DBATCH_DELETE_OLD_FILES delete-old
```

The same goal can be achieved by piping these commands through `yes`:

```
# yes|make delete-old
```

25.7.14 Rebooting

Verify that everything appears to be in the right place, then reboot the system using shutdown(8):

```
# shutdown -r now
```

25.7.15 Deleting obsolete libraries

> **Warning:** Deleting obsolete files will break applications that still depend on those obsolete files. This is especially true for old libraries. In most cases, the programs, ports, or libraries that used the old library need to be recompiled before `make delete-old-libs` is executed.

Utilities for checking shared library dependencies are available from the Ports Collection in `sysutils/libchk` or `sysutils/bsdadminscripts`.

Obsolete shared libraries can conflict with newer libraries, causing messages like these:

```
/usr/bin/ld: warning: libz.so.4, needed by /usr/local/lib/libtiff.so, may conflict with libz.so.5
/usr/bin/ld: warning: librpcsvc.so.4, needed by /usr/local/lib/libXext.so, may conflict with libr
```

To solve these problems, determine which port installed the library:

```
# pkg_info -W  /usr/local/lib/libtiff.so
  /usr/local/lib/libtiff.so was installed by package tiff-3.9.4
  # pkg_info -W /usr/local/lib/libXext.so
  /usr/local/lib/libXext.so was installed by package libXext-1.1.1,1
```

Then deinstall, rebuild and reinstall the port. `ports-mgmt/portmaster` can be used to automate this process. After all ports are rebuilt and no longer use the old libraries, delete the old libraries using the following command:

```
# make delete-old-libs
```

You should now have successfully upgraded the FreeBSD system. Congratulations.

If things went slightly wrong, it is easy to rebuild a particular piece of the system. For example, if `/etc/magic` was accidentally deleted as part of the upgrade or merge of `/etc`, file(1) will stop working. To fix this, run:

```
# cd /usr/src/usr.bin/file
# make all install
```

25.7.16 Questions

1. Do I need to re-make the world for every change?

There is no easy answer, as it depends on the nature of the change. For example, if running **svn** only shows the following files as being updated:

```
src/games/cribbage/instr.c
src/games/sail/pl_main.c
src/release/sysinstall/config.c
src/release/sysinstall/media.c
src/share/mk/bsd.port.mk
```

it probably is not worth rebuilding the entire world. Instead, go into the appropriate sub-directories and run `make all install`. But if something major changed, such as `src/lib/libc/stdlib`, either re-make world, or at least those parts of it that are statically linked.

At the end of the day, it is your call. Some users re-make the world every fortnight and let changes accumulate over that fortnight. Others only re-make those things that have changed and are careful to spot all the dependencies.

It all depends on how often a user wants to upgrade and whether they are tracking FreeBSD-STABLE or FreeBSD-CURRENT.

2. My compile failed with lots of signal 11 (or other signal number) errors. What happened?

This normally indicates hardware problems. (Re)making world is an effective way to stress test hardware, and will frequently throw up memory problems which normally manifest themselves as the compiler mysteriously aborts.

A sure indicator of this occurs when **make** is restarted and it dies at a different point in the process.

To resolve this error, start swapping around the components in the machine to determine which one is failing.

3. Can `/usr/obj` be removed when finished?

The short answer is yes.

`/usr/obj` contains all the object files that were produced during the compilation phase. Normally, one of the first steps in the `make buildworld` process is to remove this directory and start afresh. Keeping `/usr/obj` around when finished makes little sense, and its removal frees up a approximately 2 GB of disk space.

Advances users can instruct `make buildworld` to skip this step. This speeds up subsequent builds, since most of the sources will not need to be recompiled. The flip side is that subtle dependency problems can creep in, causing the build to fail in odd ways. This frequently generates noise on the FreeBSD mailing lists, when one person complains that their build has failed, not realizing that it is because they have tried to cut corners.

4. Can interrupted builds be resumed?

This depends on how far into the process the problem occurs.

In general, `make buildworld` builds new copies of essential tools, such as gcc(1) and make(1), and the system libraries. These tools and libraries are then installed, used to rebuild themselves, and are installed again. The entire system, including regular user programs such as ls(1) or grep(1), is then rebuilt with the new system files.

During the last stage, it is fairly safe to:

```
... fix the problem ...
# cd /usr/src
# make -DNO_CLEAN all
```

This will not undo the work of the previous `make buildworld`.

If you see the message:

```
--------------------------------------------------------------
Building everything..
--------------------------------------------------------------
```

in the `make buildworld` output, it is probably fairly safe to do so.

If that message is not displayed, or you are not sure, it is always better to be safe than sorry, and restart the build from scratch.

5. How can I speed up making the world?

- Run it in single user mode.
- Put `/usr/src` and `/usr/obj` on separate file systems held on separate disks. If possible, put these disks on separate disk controllers.
- Alternately, put these file systems across multiple disks using ccd(4).
- Turn off profiling by setting “NO_PROFILE=true” in `/etc/make.conf`.
- Pass `-j`*n* to make(1) to run multiple processes in parallel. This usually helps on both single and multi processor machines.

- The file system holding `/usr/src` can be mounted or remounted with `noatime`. This prevents the file system from recording the file access time which is probably not needed.

  ```
  # mount -u -o noatime /usr/src
  ```

  Warning: This example assumes `/usr/src` is on its own file system. If it is part of `/usr`, then use that file system mount point instead.

- The file system holding `/usr/obj` can be mounted or remounted with `async` so that disk writes happen asynchronously. The write completes immediately, and the data is written to the disk a few seconds later. This allows writes to be clustered together, and can provide a dramatic performance boost.

  Warning: Keep in mind that this option makes the file system more fragile. With this option, there is an increased chance that, should power fail, the file system will be in an unrecoverable state when the machine restarts.

  If `/usr/obj` is the only directory on this file system, this is not a problem. If you have other, valuable data on the same file system, ensure that there are verified backups before enabling this option.

  ```
  # mount -u -o async /usr/obj
  ```

  Warning: If `/usr/obj` is not on its own file system, replace it in the example with the name of the appropriate mount point.

6. What do I do if something goes wrong?

Make absolutely sure that the environment has no extraneous cruft from earlier builds:

```
# chflags -R noschg /usr/obj/usr
# rm -rf /usr/obj/usr
# cd /usr/src
# make cleandir
# make cleandir
```

Yes, `make cleandir` really should be run twice.

Then, restart the whole process, starting with `make buildworld`.

If problems persist, send the error and the output of `uname -a` to FreeBSD general questions mailing list (http://lists.FreeBSD.org/mailman/listinfo/freebsd-questions). Be prepared to answer other questions about the setup!

25.8 Tracking for Multiple Machines

Contributed by Mike Meyer.

When multiple machines need to track the same source tree, it is a waste of disk space, network bandwidth, and CPU cycles to have each system download the sources and rebuild everything. The solution is to have one machine do most of the work, while the rest of the machines mount that work via NFS. This section outlines a method of doing so.

25.8.1 Preliminaries

First, identify a set of machines which will run the same set of binaries, known as a *build set*. Each machine can have a custom kernel, but will run the same userland binaries. From that set, choose a machine to be the *build machine* that the world and kernel are built on. Ideally, this is a fast machine that has sufficient spare CPU to run `make buildworld` and `make buildkernel`. Select a machine to be the *test machine*, which will test software updates before they are put into production. This *must* be a machine that can afford to be down for an extended period of time. It can be the build machine, but need not be.

All the machines in this build set need to mount `/usr/obj` and `/usr/src` from the same machine, and at the same point. Ideally, those directories are on two different drives on the build machine, but they can be NFS mounted on that machine as well. For multiple build sets, `/usr/src` should be on one build machine, and NFS mounted on the rest.

Finally, ensure that `/etc/make.conf` and `/etc/src.conf` on all the machines in the build set agree with the build machine. That means that the build machine must build all the parts of the base system that any machine in the build set is going to install. Also, each build machine should have its kernel name set with `KERNCONF` in `/etc/make.conf`, and the build machine should list them all in `KERNCONF`, listing its own kernel first. The build machine must have the kernel configuration files for each machine in `/usr/src/sys/`*`arch`*`/conf` if it is going to build their kernels.

25.8.2 The Base System

On the build machine, build the kernel and world as described in Section 25.7.8.2, but do not install anything. After the build has finished, go to the test machine, and install the built kernel. If this machine mounts `/usr/src` and `/usr/obj` via NFS, enable the network and mount these directories after rebooting to single user mode. The easiest way to do this is to boot to multi-user, then run `shutdown now` to go to single user mode. Once there, install the new kernel and world and run `mergemaster` as usual. When done, reboot to return to normal multi-user operations for this machine.

After verifying that everything on the test machine is working properly, use the same procedure to install the new software on each of the other machines in the build set.

25.8.3 Ports

The same ideas can be used for the ports tree. The first critical step is to mount `/usr/ports` from the same machine to all the machines in the build set. Then, configure `/etc/make.conf` properly to share distfiles. Set `DISTDIR` to a common shared directory that is writable by whichever user `root` is mapped to by the NFS mounts. Each machine should set `WRKDIRPREFIX` to a local build directory. Finally, if the system is to build and distribute packages, set `PACKAGES` to a directory similar to `DISTDIR`.

Notes

1. For a complete description of the current security policy for old releases of FreeBSD, refer to http://www.FreeBSD.org/security/.

Chapter 26 DTrace

Written by Tom Rhodes.

26.1 Synopsis

DTrace, also known as Dynamic Tracing, was developed by Sun as a tool for locating performance bottlenecks in production and pre-production systems. It is not, in any way, a debugging tool, but a tool for real time system analysis to locate performance and other issues.

DTrace is a remarkable profiling tool, with an impressive array of features for diagnosing system issues. It may also be used to run pre-written scripts to take advantage of its capabilities. Users may even author their own utilities using the DTrace D Language, allowing them to customize their profiling based on specific needs.

After reading this chapter, you will know:

- What DTrace is and what features it provides.
- Differences between the Solaris DTrace implementation and the one provided by FreeBSD.
- How to enable and use DTrace on FreeBSD.

Before reading this chapter, you should:

- Understand UNIX and FreeBSD basics (Chapter 4).
- Be familiar with the basics of kernel configuration/compilation (Chapter 9).
- Have some familiarity with security and how it pertains to FreeBSD (Chapter 15).
- Understand how to obtain and rebuild the FreeBSD sources (Chapter 25).

> **Warning:** This feature is considered experimental. Some options may be lacking in functionality, other parts may not work at all. In time, this feature will be considered production ready and this documentation will be altered to fit that situation.

26.2 Implementation Differences

While the DTrace in FreeBSD is very similar to that found in Solaris, differences exist that should be explained before continuing. The primary difference users will notice is that on FreeBSD, DTrace needs to be specifically enabled. There are kernel options and modules which must be enabled for DTrace to work properly. These will be explained later.

There is a `DDB_CTF` kernel option which is used to enable support for loading the CTF data from kernel modules and the kernel itself. CTF is the Solaris Compact C Type Format which encapsulates a reduced form of debugging information similar to DWARF and the venerable stabs. This CTF data is added to the binaries by the `ctfconvert` and `ctfmerge` build tools. The `ctfconvert` utility parses DWARF ELF debug sections created by the compiler and `ctfmerge` merges CTF ELF sections from objects into either executables or shared libraries. More on how to enable this for the kernel and FreeBSD build is forthcoming.

Some different providers exist for FreeBSD than for Solaris. Most notable is the `dtmalloc` provider, which allows tracing `malloc()` by type in the FreeBSD kernel.

Only `root` may use DTrace on FreeBSD. This is related to security differences, Solaris has a few low level security checks which do not yet exist in FreeBSD. As such, the `/dev/dtrace/dtrace` is strictly limited to `root` users only.

Finally, the DTrace software falls under Sun's CDDL license. The `Common Development and Distribution License` comes with FreeBSD, see the `/usr/src/cddl/contrib/opensolaris/OPENSOLARIS.LICENSE` or view it online at http://www.opensolaris.org/os/licensing.

This license means that a FreeBSD kernel with the DTrace options is still BSD licensed; however the CDDL kicks in when the modules are distributed in binary form, or the binaries are loaded.

26.3 Enabling DTrace Support

To enable support for DTrace, add the following lines to the kernel configuration file:

```
options         KDTRACE_HOOKS
options         DDB_CTF
```

> **Note:** Users of the AMD64 architecture will want to add the following line to their kernel configuration file:
>
> ```
> options         KDTRACE_FRAME
> ```
>
> This option provides support for the FBT feature. DTrace will work without this option; however, there will be limited support for function boundary tracing.

All sources must be rebuilt and installed with CTF options. To accomplish this task, rebuild the FreeBSD sources using:

```
# cd /usr/src

# make WITH_CTF=1 kernel
```

The system will need to be restarted.

After rebooting and allowing the new kernel to be loaded into memory, support for the Korn shell should be added. This is needed as the DTraceToolkit has several utilities written in `ksh`. Install the `shells/ksh93`. It is also possible to run these tools under `shells/pdksh` or `shells/mksh`.

Finally, obtain the current DTraceToolkit. If you are running FreeBSD 10, you will find the DTraceToolkit in `/usr/share/dtrace`. Otherwise, you can install the DTraceToolkit using the `sysutils/DTraceToolkit` port.

26.4 Using DTrace

Before making use of DTrace functionality, the DTrace device must exist. To load the device, issue the following command:

```
# kldload dtraceall
```

DTrace support should now be available. To view all probes the administrator may now execute the following command:

```
# dtrace -l | more
```

All output is passed to the `more` utility as it will quickly overflow the screen buffer. At this point, DTrace should be considered working. It is now time to review the toolkit.

The toolkit is a collection of ready-made scripts to run with DTrace to collect system information. There are scripts to check open files, memory, CPU usage and a lot more. Extract the scripts with the following command:

```
# gunzip -c DTraceToolkit* | tar xvf -
```

Change into that directory with the `cd` and change the execution permissions on all files, designated as those files with lower case names, to `755`.

All of these scripts will need modifications to their contents. The ones which refer to `/usr/bin/ksh` need that changed to `/usr/local/bin/ksh`, the others which use `/usr/bin/sh` need to be altered to use `/bin/sh`, and finally the ones which use `/usr/bin/perl` will need altered to use `/usr/local/bin/perl`.

> **Important:** At this point it is prudent to remind the reader that DTrace support in FreeBSD is *incomplete* and *experimental*. Many of these scripts will not work as they are either too Solaris-specific or use probes which are unsupported at this time.

At the time of this writing only two of the scripts of the DTrace Toolkit are fully supported in FreeBSD: the `hotkernel` and `procsystime` scripts. These are the two we will explore in the following parts of this section.

The `hotkernel` is designed to identify which function is using the most kernel time. Run normally, it will produce output similar to the following:

```
# cd /usr/share/dtrace/toolkit
# ./hotkernel
Sampling... Hit Ctrl-C to end.
```

The system administrator must use the **Ctrl**+**C** key combination to stop the process. Upon termination, the script will display a list of kernel functions and timing information, sorting the output in increasing order of time:

```
kernel‘_thread_lock_flags                                2   0.0%
0xc1097063                                               2   0.0%
kernel‘sched_userret                                     2   0.0%
kernel‘kern_select                                       2   0.0%
kernel‘generic_copyin                                    3   0.0%
kernel‘_mtx_assert                                       3   0.0%
kernel‘vm_fault                                          3   0.0%
kernel‘sopoll_generic                                    3   0.0%
kernel‘fixup_filename                                    4   0.0%
kernel‘_isitmyx                                          4   0.0%
kernel‘find_instance                                     4   0.0%
kernel‘_mtx_unlock_flags                                 5   0.0%
kernel‘syscall                                           5   0.0%
kernel‘DELAY                                             5   0.0%
0xc108a253                                               6   0.0%
```

```
kernel‘witness_lock                                        7   0.0%
kernel‘read_aux_data_no_wait                               7   0.0%
kernel‘Xint0x80_syscall                                    7   0.0%
kernel‘witness_checkorder                                  7   0.0%
kernel‘sse2_pagezero                                       8   0.0%
kernel‘strncmp                                             9   0.0%
kernel‘spinlock_exit                                      10   0.0%
kernel‘_mtx_lock_flags                                    11   0.0%
kernel‘witness_unlock                                     15   0.0%
kernel‘sched_idletd                                      137   0.3%
0xc10981a5                                             42139  99.3%
```

This script will also work with kernel modules. To use this feature, run the script with the `-m` flag:

```
# ./hotkernel -m
Sampling... Hit Ctrl-C to end.
^C
MODULE                                                 COUNT   PCNT
0xc107882e                                                 1   0.0%
0xc10e6aa4                                                 1   0.0%
0xc1076983                                                 1   0.0%
0xc109708a                                                 1   0.0%
0xc1075a5d                                                 1   0.0%
0xc1077325                                                 1   0.0%
0xc108a245                                                 1   0.0%
0xc107730d                                                 1   0.0%
0xc1097063                                                 2   0.0%
0xc108a253                                                73   0.0%
kernel                                                   874   0.4%
0xc10981a5                                            213781  99.6%
```

The `procsystime` script captures and prints the system call time usage for a given PID or process name. In the following example, a new instance of `/bin/csh` was spawned. The `procsystime` was executed and remained waiting while a few commands were typed on the other incarnation of `csh`. These are the results of this test:

```
# ./procsystime -n csh
Tracing... Hit Ctrl-C to end...
^C

Elapsed Times for processes csh,

         SYSCALL          TIME (ns)
          getpid               6131
       sigreturn               8121
           close              19127
           fcntl              19959
             dup              26955
         setpgid              28070
            stat              31899
       setitimer              40938
           wait4              62717
       sigaction              67372
     sigprocmask             119091
```

```
  gettimeofday             183710
         write             263242
        execve             492547
         ioctl             770073
         vfork            3258923
    sigsuspend            6985124
          read         3988049784
```

As shown, the `read()` system call seems to use the most time in nanoseconds with the `getpid()` system call used the least amount of time.

26.5 The D Language

The DTrace Toolkit includes many scripts in the special language of DTrace. This language is called “the D language” by Sun documentation, and it is very similar to C++. An in depth discussion of the language is beyond the scope of this document. It is extensively discussed at http://wikis.oracle.com/display/DTrace/Documentation.

IV. Network Communication

FreeBSD is one of the most widely deployed operating systems for high performance network servers. The chapters in this part cover:

- Serial communication
- PPP and PPP over Ethernet
- Electronic Mail
- Running Network Servers
- Firewalls
- Other Advanced Networking Topics

These chapters are designed to be read when you need the information. You do not have to read them in any particular order, nor do you need to read all of them before you can begin using FreeBSD in a network environment.

Chapter 27 Serial Communications

27.1 Synopsis

UNIX has always had support for serial communications as the very first UNIX machines relied on serial lines for user input and output. Things have changed a lot from the days when the average terminal consisted of a 10-character-per-second serial printer and a keyboard. This chapter covers some of the ways serial communications can be used on FreeBSD.

After reading this chapter, you will know:

- How to connect terminals to a FreeBSD system.
- How to use a modem to dial out to remote hosts.
- How to allow remote users to login to a FreeBSD system with a modem.
- How to boot a FreeBSD system from a serial console.

Before reading this chapter, you should:

- Know how to configure and install a custom kernel.
- Understand FreeBSD permissions and processes.
- Have access to the technical manual for the serial hardware to be used with FreeBSD.

27.2 Introduction

27.2.1 Terminology

bps

Bits per Second (bps) is the rate at which data is transmitted.

DTE

An example of a Data Terminal Equipment (DTE) is a computer.

DCE

An example of a Data Communications Equipment (DTE) is a modem.

RS-232

The original standard for hardware serial communications. It is now usually referred to as TIA-232

When talking about communications data rates, this section does not use the term “baud”. Baud refers to the number of electrical state transitions that may be made in a period of time, while bps is the *correct* term to use.

27.2.2 Cables and Ports

To connect a modem or serial terminal to a FreeBSD system, a serial port on the computer and the proper cable to connect to the serial device are needed. Users who are already familiar with serial hardware and cabling can safely skip this section.

27.2.2.1 Cables

There are several different kinds of serial cables. The two most common types are null-modem cables and standard RS-232 cables. The documentation for the hardware should describe the type of cable required.

27.2.2.1.1 Null-modem Cables

A null-modem cable passes some signals, such as “Signal Ground”, straight through, but switches other signals. For example, the “Transmitted Data” pin on one end goes to the “Received Data” pin on the other end.

A null-modem cable can be constructed for use with terminals. The following table shows the RS-232C signal names and the pin numbers on a DB-25 connector. While the standard calls for a straight-through pin 1 to pin 1 *Protective Ground* line, it is often omitted. Some terminals work using only pins 2, 3, and 7, while others require different configurations than the examples shown below.

Table 27-1. DB-25 to DB-25 Null-Modem Cable

Signal	Pin #		Pin #	Signal
SG	7	connects to	7	SG
TD	2	connects to	3	RD
RD	3	connects to	2	TD
RTS	4	connects to	5	CTS
CTS	5	connects to	4	RTS
DTR	20	connects to	6	DSR
DTR	20	connects to	8	DCD
DSR	6	connects to	20	DTR
DCD	8	connects to	20	DTR

The next two tables show two other common schemes.

Table 27-2. DB-9 to DB-9 Null-Modem Cable

Signal	Pin #		Pin #	Signal
RD	2	connects to	3	TD
TD	3	connects to	2	RD
DTR	4	connects to	6	DSR
DTR	4	connects to	1	DCD

Signal	Pin #		Pin #	Signal
SG	5	connects to	5	SG
DSR	6	connects to	4	DTR
DCD	1	connects to	4	DTR
RTS	7	connects to	8	CTS
CTS	8	connects to	7	RTS

Table 27-3. DB-9 to DB-25 Null-Modem Cable

Signal	Pin #		Pin #	Signal
RD	2	connects to	2	TD
TD	3	connects to	3	RD
DTR	4	connects to	6	DSR
DTR	4	connects to	8	DCD
SG	5	connects to	7	SG
DSR	6	connects to	20	DTR
DCD	1	connects to	20	DTR
RTS	7	connects to	5	CTS
CTS	8	connects to	4	RTS

Note: When one pin at one end connects to a pair of pins at the other end, it is usually implemented with one short wire between the pair of pins in their connector and a long wire to the other single pin.

The above designs seem to be the most popular. In another variation, SG connects to SG, TD connects to RD, RTS and CTS connect to DCD, DTR connects to DSR, and vice-versa.

27.2.2.1.2 Standard RS-232C Cables

A standard serial cable passes all of the RS-232C signals straight through. The “Transmitted Data” pin on one end of the cable goes to the “Transmitted Data” pin on the other end. This is the type of cable used to connect a modem to the FreeBSD system, and is also appropriate for some terminals.

27.2.2.2 Ports

Serial ports are the devices through which data is transferred between the FreeBSD host computer and the terminal. This section describes the kinds of ports that exist and how they are addressed in FreeBSD.

27.2.2.2.1 Kinds of Ports

Several kinds of serial ports exist. Before purchasing or constructing a cable, make sure it will fit the ports on the terminal and on the FreeBSD system.

Most terminals have DB-25 ports. Personal computers may have DB-25 or DB-9 ports. A multiport serial card may have RJ-12 or RJ-45 ports.

See the documentation that accompanied the hardware for specifications on the kind of port or visually verify the type of port.

27.2.2.2.2 Port Names

In FreeBSD, each serial port is accessed through an entry in `/dev`. There are two different kinds of entries:

- Call-in ports are named `/dev/ttyu`*`N`* where *`N`* is the port number, starting from zero. Generally, the call-in port is used for terminals. Call-in ports require that the serial line assert the Data Carrier Detect (DCD) signal to work correctly.
- Call-out ports are named `/dev/cuau`*`N`*. Call-out ports are usually not used for terminals, but are used for modems. The call-out port can be used if the serial cable or the terminal does not support the carrier detect signal.

If a terminal is connected to the first serial port(`COM1`), use `/dev/ttyu0` to refer to the terminal. If the terminal is on the second serial port (`COM2`), use `/dev/ttyu1`, and so forth.

27.2.3 Kernel Configuration

FreeBSD supports four serial ports by default. In the MS-DOS world, these are known as `COM1`, `COM2`, `COM3`, and `COM4`. FreeBSD currently supports "dumb" multiport serial interface cards, such as the BocaBoard 1008 and 2016, as well as more intelligent multi-port cards such as those made by Digiboard and Stallion Technologies. However, the default kernel only looks for the standard COM ports.

To see if the kernel recognizes the serial ports, watch for messages while the kernel is booting, or use `/sbin/dmesg` to replay the kernel's boot messages. Look for messages that start with the characters `uart`:

```
# /sbin/dmesg | grep 'uart'
```

If the kernel does not recognize all of the serial ports, configure `/boot/device.hints`. When editing this file, one can comment out or completely remove lines for devices that do not exist on the system.

> **Note:** `port IO_COM1` is a substitution for `port 0x3f8`, `IO_COM2` is `0x2f8`, `IO_COM3` is `0x3e8`, and `IO_COM4` is `0x2e8`. These are fairly common port addresses for their respective serial ports and interrupts 4, 3, 5, and 9 are fairly common interrupt request lines. Regular serial ports *cannot* share interrupts on ISA-bus PCs. Multiport boards have on-board electronics that allow all the 16550A's on the board to share one or two interrupt request lines.

27.2.4 Device Special Files

Most devices in the kernel are accessed through "device special files" which are located in `/dev`. The `sio` devices are accessed through the `/dev/ttyu`*`N`* (dial-in) and `/dev/cuau`*`N`* (call-out) devices. FreeBSD also provides initialization devices (`/dev/ttyu`*`N`*`.init` and `/dev/cuau`*`N`*`.init`) and locking devices (`/dev/ttyu`*`N`*`.lock` and

/dev/cuauN.lock). The initialization devices are used to initialize communications port parameters each time a port is opened, such as crtscts for modems which use RTS/CTS signaling for flow control. The locking devices are used to lock flags on ports to prevent users or programs changing certain parameters. Refer to termios(4), sio(4), and stty(1) for information on terminal settings, locking and initializing devices, and setting terminal options, respectively.

27.2.5 Serial Port Configuration

The ttyuN (or cuauN) is the regular device to open for applications. When a process opens the device, it will have a default set of terminal I/O settings. These settings can be viewed with the command:

```
# stty -a -f /dev/ttyu1
```

When the settings are changed for a device, the settings are in effect until the device is closed. When the device is reopened, it goes back to the default set. To permanently change the default set, open and adjust the settings of the "initial state" device. For example, to turn on CLOCAL mode, 8 bit communication, and XON/XOFF flow control for ttyu5, type:

```
# stty -f /dev/ttyu5.init clocal cs8 ixon ixoff
```

System-wide initialization of serial devices is controlled by /etc/rc.d/serial. This file affects the default settings of serial devices.

To prevent certain settings from being changed by an application, make adjustments to the "lock state" device. For example, to lock the speed of ttyu5 to 57600 bps, type:

```
# stty -f /dev/ttyu5.lock 57600
```

Now, an application that opens ttyu5 and tries to change the speed of the port will be stuck with 57600 bps.

The initial state and lock state devices should only be writable by root.

27.3 Terminals

Contributed by Sean Kelly.

Terminals provide a convenient and low-cost way to access a FreeBSD system when not at the computer's console or on a connected network. This section describes how to use terminals with FreeBSD.

27.3.1 Uses and Types of Terminals

The original UNIX systems did not have consoles. Instead, users logged in and ran programs through terminals that were connected to the computer's serial ports.

The ability to establish a login session on a serial port still exists in nearly every UNIX-like operating system today, including FreeBSD. By using a terminal attached to an unused serial port, a user can log in and run any text program that can normally be run on the console or in an xterm window.

Many terminals can be attached to a FreeBSD system. An older spare computer can be used as a terminal wired into a more powerful computer running FreeBSD. This can turn what might otherwise be a single-user computer into a powerful multiple user system.

This section describes three kinds of terminals supported by FreeBSD: dumb terminals, computers acting as terminals, and X terminals.

27.3.1.1 Dumb Terminals

Dumb terminals are specialized hardware that connect to computers over serial lines. They are called “dumb” because they have only enough computational power to display, send, and receive text. No programs can be run on these devices. Dumb terminals connect to a computer that has all the power to run text editors, compilers, email, games, and so forth.

There are hundreds of kinds of dumb terminals made by many manufacturers, and just about any kind will work with FreeBSD. Some high-end terminals can even display graphics, but only certain software packages can take advantage of these advanced features.

Dumb terminals are popular in work environments where workers do not need access to graphical applications.

27.3.1.2 Computers Acting as Terminals

If a dumb terminal has just enough ability to display, send, and receive text, any spare computer can be a dumb terminal. All that is needed is the proper cable and some *terminal emulation* software to run on the computer.

This configuration can be useful. For example, if one user is busy working at the FreeBSD system’s console, another user can do some text-only work at the same time from a less powerful personal computer hooked up as a terminal to the FreeBSD system.

There are at least two utilities in the base-system of FreeBSD that can be used to work through a serial connection: cu(1) and tip(1).

To connect from a client system that runs FreeBSD to the serial connection of another system, use:

```
# cu -l serial-port-device
```

Where “serial-port-device” is the name of a special device file denoting a serial port on the system. These device files are called `/dev/cuauN`.

The “N”-part of a device name is the serial port number.

> **Note:** Note that device numbers in FreeBSD start from zero and not one. This means that `COM1` is `/dev/cuau0` in FreeBSD.

> **Note:** Some people prefer to use other programs available through the Ports Collection, such as `comms/minicom`.

27.3.1.3 X Terminals

X terminals are the most sophisticated kind of terminal available. Instead of connecting to a serial port, they usually connect to a network like Ethernet. Instead of being relegated to text-only applications, they can display any X application.

This chapter does *not* cover the setup, configuration, or use of X terminals.

27.3.2 Configuration

This section describes how to configure a FreeBSD system to enable a login session on a terminal. It assumes that the kernel is configured to support the serial port to which the terminal is connected and that the terminal is connected.

The `init` process is responsible for all process control and initialization at system startup. One of the tasks performed by `init` is to read `/etc/ttys` and start a `getty` process on the available terminals. The `getty` process is responsible for reading a login name and starting the `login` program.

To configure terminals for a FreeBSD system, the following steps should be taken as `root`:

1. Add a line to `/etc/ttys` for the entry in `/dev` for the serial port if it is not already there.
2. Specify that `/usr/libexec/getty` be run on the port, and specify the appropriate *`getty`* type from `/etc/gettytab`.
3. Specify the default terminal type.
4. Set the port to “on.”
5. Specify whether the port should be “secure.”
6. Force `init` to reread `/etc/ttys`.

As an optional step, create a custom *`getty`* type for use in step 2 by making an entry in `/etc/gettytab`. For more information, refer to gettytab(5) and getty(8).

27.3.2.1 Adding an Entry to `/etc/ttys`

`/etc/ttys` lists all of the ports on the FreeBSD system which allow logins. For example, the first virtual console, `ttyv0`, has an entry in this file, allowing logins on the console. This file also contains entries for the other virtual consoles, serial ports, and pseudo-ttys. For a hardwired terminal, list the serial port’s `/dev` entry without the `/dev` part. For example, `/dev/ttyv0` would be listed as `ttyv0`.

A default FreeBSD install includes an `/etc/ttys` with support for the first four serial ports: `ttyu0` through `ttyu3`. When attaching a terminal to one of those ports, this file does not need to be edited.

Example 27-1. Adding Terminal Entries to `/etc/ttys`

This example configures two terminals: a Wyse-50 and an old 286 IBM PC running **Procomm** terminal software emulating a VT-100 terminal. The Wyse is connected to the second serial port and the 286 to the sixth serial port on a multiport serial card. The corresponding entries in `/etc/ttys` would look like this:

```
ttyu1❶  "/usr/libexec/getty std.38400"❷  wy50❸  on❹  insecure❺
ttyu5   "/usr/libexec/getty std.19200"  vt100  on insecure
```

❶ The first field normally specifies the name of the terminal special file as it is found in `/dev`.

❷ The second field is the command to execute for this line, which is usually getty(8). `getty` initializes and opens the line, sets the speed, prompts for a user name, and then executes login(1).

The `getty` program accepts one (optional) parameter on its command line, the *`getty`* type. A *`getty`* type configures characteristics on the terminal line, like bps rate and parity. `getty` reads these characteristics from `/etc/gettytab`.

`/etc/gettytab` contains many entries for terminal lines, both old and new. In almost all cases, the entries that start with the text `std` will work for hardwired terminals as these entries ignore parity. There is a `std` entry for each bps rate from 110 to 115200. gettytab(5) provides more information.

When setting the *`getty`* type in `/etc/ttys`, make sure that the communications settings on the terminal match.

For this example, the Wyse-50 uses no parity and connects at 38400 bps. The 286 PC uses no parity and connects at 19200 bps.

❸ The third field is the type of terminal usually connected to that terminal line. For dial-up ports, `unknown` or `dialup` is typically used since users may dial up with practically any type of terminal or software. Since the terminal type does not change for hardwired terminals, a real terminal type from termcap(5) can be used in this field.

For this example, the Wyse-50 uses the real terminal type while the 286 PC running **Procomm** will be set to emulate at VT-100.

❹ The fourth field specifies if the port should be enabled. If set to `on`, the `init` process will start the program in the second field, `getty`. If this field is set to `off`, there will be no `getty`, and hence no logins on the port.

❺ The final field is used to specify whether the port is secure. Marking a port as `secure` means that it is trusted enough to allow `root`, or any account with a UID of 0, to login from that port. Insecure ports do not allow `root` logins. On an insecure port, users must login from unprivileged accounts and then use su(1) or a similar mechanism to gain superuser privileges.

It is highly recommended to use `insecure`, even for terminals that are behind locked doors. It is quite easy to login and use `su` when superuser privileges are needed.

27.3.2.2 Force `init` to Reread `/etc/ttys`

After making any changes to `/etc/ttys`, send a SIGHUP (hangup) signal to the `init` process to force it to re-read its configuration file:

```
# kill -HUP 1
```

Note: `init` is always the first process run on a system, therefore it will always have a process ID of 1.

If everything is set up correctly, all cables are in place, and the terminals are powered up, then a `getty` process should be running on each terminal and login prompts should be available on each terminal.

27.3.3 Troubleshooting the Connection

Even with the most meticulous attention to detail, something could still go wrong while setting up a terminal. Here is a list of common symptoms and some suggested fixes.

27.3.3.1 No Login Prompt Appears

Make sure the terminal is plugged in and powered up. If it is a personal computer acting as a terminal, make sure it is running terminal emulation software on the correct serial port.

Make sure the cable is connected firmly to both the terminal and the FreeBSD computer. Make sure it is the right kind of cable.

Make sure the terminal and FreeBSD agree on the bps rate and parity settings. For a video display terminal, make sure the contrast and brightness controls are turned up. If it is a printing terminal, make sure paper and ink are in good supply.

Make sure that a `getty` process is running and serving the terminal. For example, to get a list of running `getty` processes with `ps`, type:

```
# ps -axww|grep getty
```

There should be an entry for the terminal. For example, the following display shows that a `getty` is running on the second serial port, `ttyu1`, and is using the `std.38400` entry in `/etc/gettytab`:

```
22189  d1  Is+    0:00.03 /usr/libexec/getty std.38400 ttyu1
```

If no `getty` process is running, make sure the port is enabled in `/etc/ttys`. Remember to run `kill -HUP 1` after modifying `/etc/ttys`.

If the `getty` process is running but the terminal still does not display a login prompt, or if it displays a prompt but will not accept typed input, the terminal or cable may not support hardware handshaking. Try changing the entry in `/etc/ttys` from `std.38400` to `3wire.38400`, then run `kill -HUP 1` after modifying `/etc/ttys`. The `3wire` entry is similar to `std`, but ignores hardware handshaking. The baud rate may need to be reduced or software flow control enabled when using `3wire` to prevent buffer overflows.

27.3.3.2 If Garbage Appears Instead of a Login Prompt

Make sure the terminal and FreeBSD agree on the bps rate and parity settings. Check the `getty` processes to make sure the correct *`getty`* type is in use. If not, edit `/etc/ttys` and run `kill -HUP 1`.

27.3.3.3 Characters Appear Doubled and the Password Appears When Typed

Switch the terminal, or the terminal emulation software, from “half duplex” or “local echo” to “full duplex.”

27.4 Dial-in Service

Contributed by Guy Helmer. Additions by Sean Kelly.

Configuring a FreeBSD system for dial-in service is similar to connecting terminals except that modems are used instead of terminal devices.

27.4.1 External Versus Internal Modems

External modems are more convenient for dial-up because they often can be semi-permanently configured via parameters stored in non-volatile RAM and they usually provide lighted indicators that display the state of important RS-232 signals, indicating whether the modem is operating properly.

Internal modems usually lack non-volatile RAM, so their configuration may be limited to setting DIP switches. If the internal modem has any signal indicator lights, they are difficult to view when the system's cover is in place.

27.4.1.1 Modems and Cables

When using an external modem, a proper cable is needed. A standard RS-232C serial cable should suffice as long as all of the normal signals are wired:

Table 27-4. Signal Names

Acronyms	Names
RD	Received Data
TD	Transmitted Data
DTR	Data Terminal Ready
DSR	Data Set Ready
DCD	Data Carrier Detect (RS-232's Received Line Signal Detector)
SG	Signal Ground
RTS	Request to Send
CTS	Clear to Send

FreeBSD needs the RTS and CTS signals for flow control at speeds above 2400 bps, the CD signal to detect when a call has been answered or the line has been hung up, and the DTR signal to reset the modem after a session is complete. Some cables are wired without all of the needed signals, so if a login session does not go away when the line hangs up, there may be a problem with the cable.

Like other UNIX-like operating systems, FreeBSD uses the hardware signals to find out when a call has been answered or a line has been hung up and to hangup and reset the modem after a call. FreeBSD avoids sending commands to the modem or watching for status reports from the modem.

27.4.2 Serial Interface Considerations

FreeBSD supports the NS8250-, NS16450-, NS16550-, and NS16550A-based EIA RS-232C (CCITT V.24) communications interfaces. The 8250 and 16450 devices have single-character buffers. The 16550 device provides a

16-character buffer, which allows for better system performance. Bugs in plain 16550's prevent the use of the 16-character buffer, so use 16550A's if possible. Because single-character-buffer devices require more work by the operating system than the 16-character-buffer devices, 16550A-based serial interface cards are preferred. If the system has many active serial ports or will have a heavy load, 16550A-based cards are better for low-error-rate communications.

27.4.3 Quick Overview

As with terminals, `init` spawns a `getty` process for each configured serial port for dial-in connections. For example, if a modem is attached to `/dev/ttyu0`, `ps ax` might show this:

```
 4850 ??  I      0:00.09 /usr/libexec/getty V19200 ttyu0
```

When a user dials the modem's line and the modems connect, the Carrier Detect (CD) line is reported by the modem. The kernel notices that the carrier has been detected and instructs `getty` to open the port. `getty` sends a `login:` prompt at the specified initial line speed. `getty` watches to see if legitimate characters are received, and, in a typical configuration, if it finds junk (probably due to the modem's connection speed being different than `getty`'s speed), `getty` tries adjusting the line speeds until it receives reasonable characters.

After the user enters their login name, `getty` executes `/usr/bin/login`, which completes the login by asking for the user's password and then starting the user's shell.

27.4.4 Configuration Files

There are three system configuration files in `/etc` that probably need to be edited to allow dial-up access to the FreeBSD system. `/etc/gettytab` contains configuration information for the `/usr/libexec/getty` daemon. `/etc/ttys` holds information that tells `init` which `tty`s should have `getty` processes running on them. Lastly, port initialization commands can be placed in `/etc/rc.d/serial`.

There are two schools of thought regarding dial-up modems on UNIX. One group likes to configure their modems and systems so that no matter at what speed a remote user dials in, the local computer-to-modem RS-232 interface runs at a locked speed. The benefit of this configuration is that the remote user always sees a system login prompt immediately. The downside is that the system does not know what a user's true data rate is, so full-screen programs like **Emacs** will not adjust their screen-painting methods to make their response better for slower connections.

The other group configures their modems' RS-232 interface to vary its speed based on the remote user's connection speed. For example, V.32bis (14.4 Kbps) connections to the modem might make the modem run its RS-232 interface at 19.2 Kbps, while 2400 bps connections make the modem's RS-232 interface run at 2400 bps. Because `getty` does not understand any particular modem's connection speed reporting, `getty` gives a `login:` message at an initial speed and watches the characters that come back in response. If the user sees junk, it is assumed that they know they should press **Enter** until they see a recognizable prompt. If the data rates do not match, `getty` sees anything the user types as "junk", tries going to the next speed and gives the `login:` prompt again. This procedure normally only takes a keystroke or two before the user sees a good prompt. This login sequence does not look as clean as the "locked-speed" method, but a user on a low-speed connection should receive better interactive response from full-screen programs.

This section will try to give balanced configuration information, but is biased towards having the modem's data rate follow the connection rate.

27.4.4.1 /etc/gettytab

/etc/gettytab is a termcap(5)-style file of configuration information for getty(8). Refer to gettytab(5) for complete information on the format of the file and the list of capabilities.

27.4.4.1.1 Locked-speed Config

When locking a modem's data communications rate at a particular speed, no changes to /etc/gettytab should be needed.

27.4.4.1.2 Matching-speed Config

Set up an entry in /etc/gettytab to give getty information about the speeds to use for the modem. For a 2400 bps modem, use the existing D2400 entry.

```
#
# Fast dialup terminals, 2400/1200/300 rotary (can start either way)
#
D2400|d2400|Fast-Dial-2400:\
        :nx=D1200:tc=2400-baud:
3|D1200|Fast-Dial-1200:\
        :nx=D300:tc=1200-baud:
5|D300|Fast-Dial-300:\
        :nx=D2400:tc=300-baud:
```

For a higher speed modem, add an entry in /etc/gettytab. This entry is for a 14.4 Kbps modem with a top interface speed of 19.2 Kbps:

```
#
# Additions for a V.32bis Modem
#
um|V300|High Speed Modem at 300,8-bit:\
        :nx=V19200:tc=std.300:
un|V1200|High Speed Modem at 1200,8-bit:\
        :nx=V300:tc=std.1200:
uo|V2400|High Speed Modem at 2400,8-bit:\
        :nx=V1200:tc=std.2400:
up|V9600|High Speed Modem at 9600,8-bit:\
        :nx=V2400:tc=std.9600:
uq|V19200|High Speed Modem at 19200,8-bit:\
        :nx=V9600:tc=std.19200:
```

This will result in 8-bit, no parity connections.

The example above starts the communications rate at 19.2 Kbps (for a V.32bis connection), then cycles through 9600 bps (for V.32), 2400 bps, 1200 bps, 300 bps, and back to 19.2 Kbps. Communications rate cycling is implemented with the nx= ("next table") capability. Each of the lines uses a tc= ("table continuation") entry to pick up the rest of the "standard" settings for a particular data rate.

For a 28.8 Kbps modem or to take advantage of compression on a 14.4 Kbps modem, use a higher communications rate than 19.2 Kbps. Here is an example of a gettytab entry starting a 57.6 Kbps:

```
#
```

```
# Additions for a V.32bis or V.34 Modem
# Starting at 57.6 Kbps
#
vm|VH300|Very High Speed Modem at 300,8-bit:\
        :nx=VH57600:tc=std.300:
vn|VH1200|Very High Speed Modem at 1200,8-bit:\
        :nx=VH300:tc=std.1200:
vo|VH2400|Very High Speed Modem at 2400,8-bit:\
        :nx=VH1200:tc=std.2400:
vp|VH9600|Very High Speed Modem at 9600,8-bit:\
        :nx=VH2400:tc=std.9600:
vq|VH57600|Very High Speed Modem at 57600,8-bit:\
        :nx=VH9600:tc=std.57600:
```

For a slow CPU or a heavily loaded system without 16550A-based serial ports, there may be `sio` "silo" errors at 57.6 Kbps.

27.4.4.2 `/etc/ttys`

Configuration of `/etc/ttys` is covered in Example 27-1. Configuration for modems is similar, but a different argument is passed to `getty` and a different terminal type is specified. The general format for both locked-speed and matching-speed configurations is:

```
ttyu0   "/usr/libexec/getty xxx"   dialup on
```

The first item in the above line is the device special file for this entry. `ttyu0` indicates that `getty` is watching `/dev/ttyu0`. The *xxx* will replace the initial `gettytab` capability and is the process `init` will run on the device. The third item, `dialup`, is the default terminal type. The fourth parameter, `on`, indicates to `init` that the line is operational. There can be a fifth parameter, `secure`, but it should only be used for terminals which are physically secure, such as the system console.

The default terminal type, `dialup` in this example, may depend on local preferences. `dialup` is the traditional default terminal type on dial-up lines so that users may customize their login scripts to notice when the terminal is `dialup` and automatically adjust their terminal type. Setting `vt102` as the default terminal type allows users to use VT102 emulation on their remote systems.

After editing `/etc/ttys`, send the `init` process a HUP signal to re-read the file:

```
# kill -HUP 1
```

Wait until the modem is properly configured and connected before signaling `init`.

27.4.4.2.1 Locked-speed Config

For a locked-speed configuration, the `ttys` entry needs to have a fixed-speed entry provided to `getty`. For a modem whose port speed is locked at 19.2 Kbps, the `ttys` entry might look like this:

```
ttyu0   "/usr/libexec/getty std.19200"   dialup on
```

If the modem is locked at a different data rate, substitute the appropriate value for `std.`*`speed`* instead of `std.19200`. Make sure to use a valid type listed in `/etc/gettytab`.

27.4.4.2.2 Matching-speed Config

In a matching-speed configuration, the `ttys` entry needs to reference the appropriate beginning "auto-baud" entry in `/etc/gettytab`. For example, for the above suggested entry for a matching-speed modem that starts at 19.2 Kbps, the `/etc/ttys` entry might look like this:

```
ttyu0   "/usr/libexec/getty V19200"   dialup on
```

27.4.4.3 `/etc/rc.d/serial`

High-speed modems, like V.32, V.32bis, and V.34 modems, need to use hardware (`RTS/CTS`) flow control. `stty` can be used to set the hardware flow control flag in the FreeBSD kernel for the modem ports.

For example, to set the `termios` flag `crtscts` on `COM2`'s dial-in and dial-out initialization devices, the following lines could be added to `/etc/rc.d/serial`:

```
# Serial port initial configuration
stty -f /dev/ttyu1.init crtscts
stty -f /dev/cuau1.init crtscts
```

27.4.5 Modem Settings

For a modem whose parameters may be permanently set in non-volatile RAM, a terminal program such as `tip` can be used to set the parameters. Connect to the modem using the same communications speed as the initial speed `getty` will use and configure the modem's non-volatile RAM to match these requirements:

- CD asserted when connected.
- DTR asserted for operation and dropping DTR hangs up the line and resets the modem.
- CTS transmitted data flow control.
- Disable XON/XOFF flow control.
- RTS received data flow control.
- Quiet mode (no result codes).
- No command echo.

Read the documentation for the modem to find out which commands and/or DIP switch settings are needed.

For example, to set the above parameters on a U.S. Robotics® Sportster® 14,400 external modem, give these commands to the modem:

```
ATZ
AT&C1&D2&H1&I0&R2&W
```

Other settings can be adjusted in the modem, such as whether it will use V.42bis and/or MNP5 compression.

The U.S. Robotics Sportster 14,400 external modem also has some DIP switches that need to be set. Other modems, may need these settings:

- Switch 1: UP — DTR Normal
- Switch 2: N/A (Verbal Result Codes/Numeric Result Codes)
- Switch 3: UP — Suppress Result Codes
- Switch 4: DOWN — No echo, offline commands
- Switch 5: UP — Auto Answer
- Switch 6: UP — Carrier Detect Normal
- Switch 7: UP — Load NVRAM Defaults
- Switch 8: N/A (Smart Mode/Dumb Mode)

Result codes should be disabled/suppressed for dial-up modems to avoid problems that can occur if `getty` mistakenly gives a `login:` prompt to a modem that is in command mode and the modem echoes the command or returns a result code. This sequence can result in an extended, silly conversation between `getty` and the modem.

27.4.5.1 Locked-speed Config

For a locked-speed configuration, configure the modem to maintain a constant modem-to-computer data rate independent of the communications rate. On a U.S. Robotics Sportster 14,400 external modem, these commands will lock the modem-to-computer data rate at the speed used to issue the commands:

```
ATZ
AT&B1&W
```

27.4.5.2 Matching-speed Config

For a variable-speed configuration, configure the modem to adjust its serial port data rate to match the incoming call rate. On a U.S. Robotics Sportster 14,400 external modem, these commands will lock the modem's error-corrected data rate to the speed used to issue the commands, while allowing the serial port rate to vary for non-error-corrected connections:

```
ATZ
AT&B2&W
```

27.4.5.3 Checking the Modem's Configuration

Most high-speed modems provide commands to view the modem's current operating parameters in a somewhat human-readable fashion. On the U.S. Robotics Sportster 14,400 external modem, `ATI5` displays the settings that are stored in the non-volatile RAM. To see the true operating parameters of the modem, as influenced by the modem's DIP switch settings, use `ATZ` and then `ATI4`.

For a different brand of modem, check the modem's manual to see how to double-check the modem's configuration parameters.

27.4.6 Troubleshooting

Here are a few steps for troubleshooting a dial-up modem on a FreeBSD system.

27.4.6.1 Checking Out the FreeBSD System

Hook up the modem to the FreeBSD system, boot the system, and, if the modem has status indication lights, watch to see whether the modem's DTR indicator lights when the `login:` prompt appears on the system's console. If it lights up, that should mean that FreeBSD has started a `getty` process on the appropriate communications port and is waiting for the modem to accept a call.

If the DTR indicator does not light, login to the FreeBSD system through the console and type `ps ax` to see if FreeBSD is trying to run a `getty` process on the correct port:

```
114 ??  I      0:00.10 /usr/libexec/getty V19200 ttyu0
115 ??  I      0:00.10 /usr/libexec/getty V19200 ttyu1
```

If something like this is displayed instead:

```
114 d0  I      0:00.10 /usr/libexec/getty V19200 ttyu0
```

and the modem has not accepted a call yet, this means that `getty` has completed its open on the communications port. This could indicate a problem with the cabling or a misconfigured modem, because `getty` should not be able to open the communications port until carrier detect has been asserted by the modem.

If no `getty` processes are waiting to open the desired `ttyu`*`N`* port, double-check the entries in `/etc/ttys` to see if there are any mistakes. Also, check `/var/log/messages` to see if there are any log messages from `init` or `getty`. If there are any messages, triple-check `/etc/ttys` and `/etc/gettytab`, as well as the appropriate device special files, `/dev/ttyuN`, for any mistakes, missing entries, or missing device special files.

27.4.6.2 Try Dialing In

Try dialing into the system. Be sure to use 8 bits, no parity, and 1 stop bit on the remote system. If a prompt does not appear right away, or the prompt shows garbage, try pressing **Enter** about once per second. If there is still no `login:` prompt after a while, try sending a `BREAK`. When using a high-speed modem, try dialing again after locking the dialing modem's interface speed.

If there is still no `login:` prompt, check `/etc/gettytab` again and double-check that:

- The initial capability name specified in the entry in `/etc/ttys` matches the name of a capability in `/etc/gettytab`.
- Each `nx=` entry matches another `gettytab` capability name.
- Each `tc=` entry matches another `gettytab` capability name.

If the modem on the FreeBSD system will not answer, make sure that the modem is configured to answer the phone when DTR is asserted. If the modem seems to be configured correctly, verify that the DTR line is asserted by checking the modem's indicator lights.

If it still does not work, take a break and come back to it later. If it still does not work, try sending an email message to the FreeBSD general questions mailing list (http://lists.FreeBSD.org/mailman/listinfo/freebsd-questions) describing the modem and the problem.

27.5 Dial-out Service

The following are tips for getting the host to connect over the modem to another computer. This is appropriate for establishing a terminal session with a remote host.

This kind of connection can be helpful to get a file on the Internet if there are problems using PPP. If PPP is not working, use the terminal session to FTP the needed file. Then use zmodem to transfer it to the machine.

27.5.1 Using a Stock Hayes Modem

A generic Hayes dialer is built into `tip`. Use `at=hayes` in `/etc/remote`.

The Hayes driver is not smart enough to recognize some of the advanced features of newer modems messages like `BUSY`, `NO DIALTONE`, or `CONNECT 115200`. Turn those messages off when using `tip` with `ATX0&W`.

The dial timeout for `tip` is 60 seconds. The modem should use something less, or else `tip` will think there is a communication problem. Try `ATS7=45&W`.

27.5.2 Using `AT` Commands

Create a “direct” entry in `/etc/remote`. For example, if the modem is hooked up to the first serial port, `/dev/cuau0`, use the following line:

```
cuau0:dv=/dev/cuau0:br#19200:pa=none
```

Use the highest bps rate the modem supports in the `br` capability. Then, type `tip cuau0` to connect to the modem.

Or, use `cu` as `root` with the following command:

```
# cu -lline -sspeed
```

`line` is the serial port, such as `/dev/cuau0`, and *`speed`* is the speed, such as `57600`. When finished entering the AT commands, type `~.` to exit.

27.5.3 The `@` Sign Does Not Work

The `@` sign in the phone number capability tells `tip` to look in `/etc/phones` for a phone number. But, the `@` sign is also a special character in capability files like `/etc/remote`, so it needs to be escaped with a backslash:

```
pn=\@
```

27.5.4 Dialing from the Command Line

Put a “generic” entry in `/etc/remote`. For example:

```
tip115200|Dial any phone number at 115200 bps:\
        :dv=/dev/cuau0:br#115200:at=hayes:pa=none:du:
tip57600|Dial any phone number at 57600 bps:\
        :dv=/dev/cuau0:br#57600:at=hayes:pa=none:du:
```

This should now work:

```
# tip -115200 5551234
```

Users who prefer `cu` over `tip`, can use a generic `cu` entry:

```
cu115200|Use cu to dial any number at 115200bps:\
        :dv=/dev/cuau1:br#57600:at=hayes:pa=none:du:
```

and type:

```
# cu 5551234 -s 115200
```

27.5.5 Setting the bps Rate

Put in an entry for `tip1200` or `cu1200`, but go ahead and use whatever bps rate is appropriate with the `br` capability. `tip` thinks a good default is 1200 bps which is why it looks for a `tip1200` entry. 1200 bps does not have to be used, though.

27.5.6 Accessing a Number of Hosts Through a Terminal Server

Rather than waiting until connected and typing `CONNECT` *`host`* each time, use `tip`'s `cm` capability. For example, these entries in `/etc/remote` will let you type `tip pain` or `tip muffin` to connect to the hosts `pain` or `muffin`, and `tip deep13` to connect to the terminal server.

```
pain|pain.deep13.com|Forrester's machine:\
        :cm=CONNECT pain\n:tc=deep13:
muffin|muffin.deep13.com|Frank's machine:\
        :cm=CONNECT muffin\n:tc=deep13:
deep13:Gizmonics Institute terminal server:\
        :dv=/dev/cuau2:br#38400:at=hayes:du:pa=none:pn=5551234:
```

27.5.7 Using More Than One Line with `tip`

This is often a problem where a university has several modem lines and several thousand students trying to use them.

Make an entry in `/etc/remote` and use `@` for the `pn` capability:

```
big-university:\
        :pn=\@:tc=dialout
dialout:\
        :dv=/dev/cuau3:br#9600:at=courier:du:pa=none:
```

Then, list the phone numbers in `/etc/phones`:

```
big-university 5551111
big-university 5551112
big-university 5551113
big-university 5551114
```

`tip` will try each number in the listed order, then give up. To keep retrying, run `tip` in a `while` loop.

27.5.8 Using the Force Character

Ctrl+**P** is the default "force" character, used to tell `tip` that the next character is literal data. The force character can be set to any other character with the `~s` escape, which means "set a variable."

Type `~sforce=`*`single-char`* followed by a newline. *`single-char`* is any single character. If *`single-char`* is left out, then the force character is the null character, which is accessed by typing **Ctrl**+**2** or **Ctrl**+**Space**. A pretty good value for *`single-char`* is **Shift**+**Ctrl**+**6**, which is only used on some terminal servers.

To change the force character, specify the following in `~/.tiprc`:

```
force=single-char
```

27.5.9 Upper Case Characters

This happens when **Ctrl**+**A** is pressed, which is `tip`'s "raise character", specially designed for people with broken caps-lock keys. Use `~s` to set `raisechar` to something reasonable. It can be set to be the same as the force character, if neither feature is used.

Here is a sample `~/.tiprc` for **Emacs** users who need to type **Ctrl**+**2** and **Ctrl**+**A**:

```
force=^^
raisechar=^^
```

The `^^` is **Shift**+**Ctrl**+**6**.

27.5.10 File Transfers with `tip`

When talking to another UNIX-like operating system, files can be sent and received using `~p` (put) and `~t` (take). These commands run `cat` and `echo` on the remote system to accept and send files. The syntax is:

`~p` local-file [remote-file]

`~t` remote-file [local-file]

There is no error checking, so another protocol, like zmodem, should probably be used.

27.5.11 Using zmodem with `tip`?

To receive files, start the sending program on the remote end. Then, type `~C rz` to begin receiving them locally.

To send files, start the receiving program on the remote end. Then, type `~C sz` *`files`* to send them to the remote system.

27.6 Setting Up the Serial Console

Contributed by Kazutaka YOKOTA. Based on a document by Bill Paul.

27.6.1 Introduction

FreeBSD has the ability to boot a system with a dumb terminal on a serial port as a console. This configuration is useful for system administrators who wish to install FreeBSD on machines that have no keyboard or monitor attached, and developers who want to debug the kernel or device drivers.

As described in Chapter 13, FreeBSD employs a three stage bootstrap. The first two stages are in the boot block code which is stored at the beginning of the FreeBSD slice on the boot disk. The boot block then loads and runs the boot loader as the third stage code.

In order to set up booting from a serial console, the boot block code, the boot loader code, and the kernel need to be configured.

27.6.2 Quick Serial Console Configuration

This section assumes the default setup and provides a fast overview of setting up the serial console.

1. Connect the serial cable to `COM1` and the controlling terminal.
2. To see all the boot messages on the serial console, issue the following command as the superuser:

   ```
   # echo 'console="comconsole"' >> /boot/loader.conf
   ```
3. Edit `/etc/ttys` and change `off` to `on` and `dialup` to `vt100` for the `ttyu0` entry. Otherwise, a password will not be required to connect via the serial console, resulting in a potential security hole.
4. Reboot the system to see if the changes took effect.

If a different configuration is required, see the next section for a more in-depth configuration explanation.

27.6.3 In-Depth Serial Console Configuration

1. Prepare a serial cable.

   Use either a null-modem cable or a standard serial cable and a null-modem adapter. See Section 27.2.2 for a discussion on serial cables.
2. Unplug the keyboard.

   Many PC systems probe for the keyboard during the Power-On Self-Test (POST) and will generate an error if the keyboard is not detected. Some machines will refuse to boot until the keyboard is plugged in.

   If the computer complains about the error, but boots anyway, no further configuration is needed.

   If the computer refuses to boot without a keyboard attached, the BIOS needs to be configured so that it ignores this error (if it can). Consult the motherboard's manual for details on how to do this.

   > **Tip:** Try setting the keyboard to "Not installed" in the BIOS. The keyboard can still be used as this setting just tells the BIOS not to probe for a keyboard at power-on. The BIOS should not complain if the keyboard is

absent. You can leave the keyboard plugged in even with this flag set to “Not installed” and the keyboard will still work. If the above option is not present in the BIOS, look for an “Halt on Error” option instead. Setting this to “All but Keyboard” or even to “No Errors”, will have the same effect.

Note: If the system has a PS/2® mouse, chances are good that both the mouse and keyboard need to be unplugged. This is because PS/2 mice share some hardware with the keyboard and leaving the mouse plugged in can fool the keyboard probe into thinking the keyboard is still there.

3. Plug a dumb terminal into `COM1` (`sio0`).

   If a dumb terminal is not available, use an old computer with a modem program, or the serial port on another UNIX box. If there is no `COM1` (`sio0`), get one. At this time, there is no way to select a port other than `COM1` for the boot blocks without recompiling the boot blocks. If `COM1` is being used by another device, temporarily remove that device and install a new boot block and kernel once FreeBSD is up and running.

4. Make sure the configuration file of the custom kernel has appropriate flags set for `COM1` (`sio0`).

   Relevant flags are:

   `0x10`

   Enables console support for this unit. The other console flags are ignored unless this is set. Currently, at most one unit can have console support. The first one, in config file order, with this flag set is preferred. This option alone will not make the serial port the console. Set the following flag or use `-h` as described below, together with this flag.

   `0x20`

   Forces this unit to be the console, unless there is another higher priority console, regardless of `-h` as discussed below. The flag `0x20` must be used together with the `0x10` flag.

   `0x40`

   Reserves this unit (in conjunction with `0x10`) and makes the unit unavailable for normal access. This flag should not be set to the serial port to use as the serial console. The only use of this flag is to designate the unit for kernel remote debugging. See The Developer’s Handbook (http://www.FreeBSD.org/doc/en_US.ISO8859-1/books/developers-handbook/index.html) for more information on remote debugging.

   Here is an example setting:

   ```
   device sio0 flags 0x10
   ```

   Refer to sio(4) for more details.

   If the flags were not set, run UserConfig on a different console or recompile the kernel.

5. Create `boot.config` in the root directory of the `a` partition on the boot drive.

   This file instructs the boot block code how to boot the system. In order to activate the serial console, one or more of the following options are needed. When using multiple options, include them all on the same line:

`-h`

Toggles between the internal and serial consoles. Use this to switch console devices. For instance, to boot from the internal (video) console, use `-h` to direct the boot loader and the kernel to use the serial port as its console device. Alternatively, to boot from the serial port, use `-h` to tell the boot loader and the kernel to use the video display as the console instead.

`-D`

Toggles between the single and dual console configurations. In the single configuration, the console will be either the internal console (video display) or the serial port, depending on the state of `-h`. In the dual console configuration, both the video display and the serial port will become the console at the same time, regardless of the state of `-h`. However, the dual console configuration takes effect only while the boot block is running. Once the boot loader gets control, the console specified by `-h` becomes the only console.

`-P`

Makes the boot block probe the keyboard. If no keyboard is found, the `-D` and `-h` options are automatically set.

> **Note:** Due to space constraints in the current version of the boot blocks, `-P` is capable of detecting extended keyboards only. Keyboards with less than 101 keys and without F11 and F12 keys may not be detected. Keyboards on some laptops may not be properly found because of this limitation. If this is the case, do not use `-P`. Unfortunately there is no workaround for this problem.

Use either `-P` to select the console automatically, or `-h` to activate the serial console.

Other options are described in boot(8).

The options, except for `-P`, are passed to the boot loader. The boot loader will determine whether the internal video or the serial port should become the console by examining the state of `-h`. This means that if `-D` is specified but `-h` is not specified in `/boot.config`, the serial port can be used as the console only during the boot block as the boot loader will use the internal video display as the console.

6. Boot the machine.

   When FreeBSD starts, the boot blocks echo the contents of `/boot.config` to the console. For example:

   ```
   /boot.config: -P
   Keyboard: no
   ```

   The second line appears only if `-P` is in `/boot.config` and indicates the presence or absence of the keyboard. These messages go to either the serial or internal console, or both, depending on the option in `/boot.config`.

Options	Message goes to
none	internal console
`-h`	serial console
`-D`	serial and internal consoles
`-Dh`	serial and internal consoles
`-P`, keyboard present	internal console
`-P`, keyboard absent	serial console

After the message, there will be a small pause before the boot blocks continue loading the boot loader and before any further messages are printed to the console. Under normal circumstances, there is no need to interrupt the boot blocks, but one can do so in order to make sure things are set up correctly.

Press any key, other than **Enter**, at the console to interrupt the boot process. The boot blocks will then prompt for further action:

```
>> FreeBSD/i386 BOOT
Default: 0:ad(0,a)/boot/loader
boot:
```

Verify that the above message appears on either the serial or internal console, or both, according to the options in `/boot.config`. If the message appears in the correct console, press **Enter** to continue the boot process.

If there is no prompt on the serial terminal, something is wrong with the settings. Enter `-h` then **Enter** or **Return** to tell the boot block (and then the boot loader and the kernel) to choose the serial port for the console. Once the system is up, go back and check what went wrong.

During the third stage of the boot process, one can still switch between the internal console and the serial console by setting appropriate environment variables in the boot loader. See Section 27.6.6 for more information.

27.6.4 Summary

Here is the summary of the various settings discussed in this section:

27.6.4.1 Case 1: Set the Flags to 0x10 for `sio0`

```
device sio0 flags 0x10
```

Options in /boot.config	Console during boot blocks	Console during boot loader	Console in kernel
nothing	internal	internal	internal
`-h`	serial	serial	serial
`-D`	serial and internal	internal	internal
`-Dh`	serial and internal	serial	serial
`-P`, keyboard present	internal	internal	internal
`-P`, keyboard absent	serial and internal	serial	serial

27.6.4.2 Case 2: Set the Flags to 0x30 for `sio0`

```
device sio0 flags 0x30
```

Options in /boot.config	Console during boot blocks	Console during boot loader	Console in kernel
nothing	internal	internal	serial
`-h`	serial	serial	serial
`-D`	serial and internal	internal	serial

Options in /boot.config	Console during boot blocks	Console during boot loader	Console in kernel
`-Dh`	serial and internal	serial	serial
`-P`, keyboard present	internal	internal	serial
`-P`, keyboard absent	serial and internal	serial	serial

27.6.5 Tips for the Serial Console

27.6.5.1 Setting a Faster Serial Port Speed

By default, the serial port settings are 9600 baud, 8 bits, no parity, and 1 stop bit. To change the default console speed, the following options are available:

- Recompile the boot blocks with `BOOT_COMCONSOLE_SPEED` set to the new console speed. See Section 27.6.5.2 for detailed instructions about building and installing new boot blocks.

  If the serial console is configured in some other way than by booting with `-h`, or if the serial console used by the kernel is different from the one used by the boot blocks, add the following option to a custom kernel configuration file and compile a new kernel:

  ```
  options CONSPEED=19200
  ```

- Add the `-S` boot option to `/boot.config`. See boot(8) for a description of how to add options to `/boot.config` and a list of the supported options.
- Enable `comconsole_speed` in `/boot/loader.conf`. This option depends on `console`, `boot_serial`, and `boot_multicons` being set in `/boot/loader.conf` too. An example of using `comconsole_speed` to change the serial console speed is:

  ```
  boot_multicons="YES"
  boot_serial="YES"
  comconsole_speed="115200"
  console="comconsole,vidconsole"
  ```

27.6.5.2 Using a Serial Port Other Than `sio0` for the Console

Using a port other than `sio0` as the console requires the boot blocks, the boot loader, and the kernel to be recompiled as follows.

1. Get the kernel source as described in Chapter 25.
2. Edit `/etc/make.conf` and set `BOOT_COMCONSOLE_PORT` to the address of the port to use: 0x3F8, 0x2F8, 0x3E8 or 0x2E8. Only `sio0` through `sio3` (`COM1` through `COM4`) can be used as multiport serial cards will not work. No interrupt setting is needed.
3. Create a custom kernel configuration file and add appropriate flags for the serial port to use. For example, to make `sio1` (`COM2`) the console:

   ```
   device sio1 flags 0x10
   ```

   or

```
device sio1 flags 0x30
```

The console flags for the other serial ports should not be set.

4. Recompile and install the boot blocks and the boot loader:

```
# cd /sys/boot
# make clean
# make
# make install
```

5. Rebuild and install the kernel.
6. Write the boot blocks to the boot disk with bsdlabel(8) and boot from the new kernel.

27.6.5.3 Entering the DDB Debugger from the Serial Line

To drop into the kernel debugger from the serial console, compile a custom kernel with the following options. Note that while this is useful for remote diagnostics, it is also dangerous if a spurious BREAK is generated on the serial port.

```
options BREAK_TO_DEBUGGER
options DDB
```

27.6.5.4 Getting a Login Prompt on the Serial Console

While this is not required, it is possible to get a *login* prompt over the serial line. First, make sure that the boot messages are displayed and it is possible to enter the kernel debugging session through the serial console.

Open `/etc/ttys` with a text editor and locate the lines:

```
ttyu0 "/usr/libexec/getty std.9600" unknown off secure
ttyu1 "/usr/libexec/getty std.9600" unknown off secure
ttyu2 "/usr/libexec/getty std.9600" unknown off secure
ttyu3 "/usr/libexec/getty std.9600" unknown off secure
```

`ttyu0` through `ttyu3` correspond to `COM1` through `COM4`. Change `off` to `on` for the desired port. If the speed of the serial port has been changed, change `std.9600` to match the new setting.

The terminal type can also be changed from `unknown` to the actual type of the serial terminal.

After editing the file, type `kill -HUP 1` to make this change take effect.

27.6.6 Changing Console from the Boot Loader

Previous sections described how to set up the serial console by tweaking the boot block. This section shows how to specify the console by entering some commands and environment variables in the boot loader. As the boot loader is invoked at the third stage of the boot process, the settings in the boot loader will override the settings in the boot block.

27.6.6.1 Setting Up the Serial Console

The boot loader and the kernel to use the serial console can be specified by writing one line in `/boot/loader.conf`:

```
console="comconsole"
```

This will take effect regardless of the settings in the boot block discussed in the previous section.

This line should be the first line of `/boot/loader.conf` so as to see boot messages on the serial console as early as possible.

Likewise, to specify the internal console:

```
console="vidconsole"
```

If the boot loader environment variable `console` is not set, the boot loader, and subsequently the kernel, will use whichever console is indicated by `-h` in the boot block.

The console can be specified in `/boot/loader.conf.local` or in `/boot/loader.conf`.

See loader.conf(5) for more information.

> **Note:** At the moment, the boot loader has no option equivalent to `-P` in the boot block, and there is no provision to automatically select the internal console and the serial console based on the presence of the keyboard.

27.6.6.2 Using a Serial Port Other Than `sio0` for the Console

The boot loader needs to be compiled in order to use a serial port other than `sio0` for the serial console. Follow the procedure described in Section 27.6.5.2.

27.6.7 Caveats

While most systems will boot without a keyboard, quite a few will not boot without a graphics adapter. Machines with AMI BIOSes can be configured to boot with no graphics adapter installed by changing the “graphics adapter” setting in the CMOS configuration to “Not installed.”

However, many machines do not support this option and will refuse to boot if there is no display hardware in the system. With these machines, leave some kind of graphics card plugged in, even if it is just a junky mono board. A monitor does not need to be attached. One might also try installing an AMI BIOS.

Chapter 28 PPP and SLIP

Restructured, reorganized, and updated by Jim Mock.

28.1 Synopsis

FreeBSD has a number of ways to link one computer to another. To establish a network or Internet connection through a dial-up modem, or to allow others to do so through you, requires the use of PPP or SLIP. This chapter describes setting up these modem-based communication services in detail.

After reading this chapter, you will know:

- How to set up user PPP.
- How to set up kernel PPP (FreeBSD 7.X only).
- How to set up PPPoE (PPP over Ethernet).
- How to set up PPPoA (PPP over ATM).
- How to configure and set up a SLIP client and server (FreeBSD 7.X only).

Before reading this chapter, you should:

- Be familiar with basic network terminology.
- Understand the basics and purpose of a dialup connection and PPP and/or SLIP.

You may be wondering what the main difference is between user PPP and kernel PPP. The answer is simple: user PPP processes the inbound and outbound data in userland rather than in the kernel. This is expensive in terms of copying the data between the kernel and userland, but allows a far more feature-rich PPP implementation. User PPP uses the `tun` device to communicate with the outside world whereas kernel PPP uses the `ppp` device.

> **Note:** Throughout in this chapter, user PPP will simply be referred to as **ppp** unless a distinction needs to be made between it and any other PPP software such as **pppd** (FreeBSD 7.X only). Unless otherwise stated, all of the commands explained in this chapter should be executed as `root`.

28.2 Using User PPP

Updated and enhanced by Tom Rhodes. Originally contributed by Brian Somers. With input from Nik Clayton, Dirk Frömberg, and Peter Childs.

28.2.1 User PPP

28.2.1.1 Assumptions

This document assumes you have the following:

- An account with an Internet Service Provider (ISP) which you connect to using PPP.
- A modem or other device connected to your system and properly configured to allow you to connect to your ISP.
- The dial-up number(s) of your ISP.
- Your login name and password. (Either a regular UNIX style login and password pair, or a PAP or CHAP login and password pair).
- The IP address of one or more name servers. Normally, you will be given two IP addresses by your ISP to use for this. If they have not given you at least one, then you can use the `enable dns` command in `ppp.conf` and **ppp** will set the name servers for you. This feature depends on your ISPs PPP implementation supporting DNS negotiation.

The following information may be supplied by your ISP, but is not completely necessary:

- The IP address of your ISP's gateway. The gateway is the machine to which you will connect and will be set up as your *default route*. If you do not have this information, we can make one up and your ISP's PPP server will tell us the correct value when we connect.

  This IP number is referred to as `HISADDR` by **ppp**.
- The netmask you should use. If your ISP has not provided you with one, you can safely use `255.255.255.255`.
- If your ISP provides you with a static IP address and hostname, you can enter it. Otherwise, we simply let the peer assign whatever IP address it sees fit.

If you do not have any of the required information, contact your ISP.

> **Note:** Throughout this section, many of the examples showing the contents of configuration files are numbered by line. These numbers serve to aid in the presentation and discussion only and are not meant to be placed in the actual file. Proper indentation with tab and space characters is also important.

28.2.1.2 Automatic PPP Configuration

Both `ppp` and `pppd` (the kernel level implementation of PPP, FreeBSD 7.X only) use the configuration files located in the `/etc/ppp` directory. Examples for user ppp can be found in `/usr/share/examples/ppp/`.

Configuring `ppp` requires that you edit a number of files, depending on your requirements. What you put in them depends to some extent on whether your ISP allocates IP addresses statically (i.e., you get given one IP address, and always use that one) or dynamically (i.e., your IP address changes each time you connect to your ISP).

28.2.1.2.1 PPP and Static IP Addresses

You will need to edit the `/etc/ppp/ppp.conf` configuration file. It should look similar to the example below.

> **Note:** Lines that end in a `:` start in the first column (beginning of the line)— all other lines should be indented as shown using spaces or tabs.

```
1     default:
2       set log Phase Chat LCP IPCP CCP tun command
3       ident user-ppp VERSION (built COMPILATIONDATE)
4       set device /dev/cuau0
5       set speed 115200
6       set dial "ABORT BUSY ABORT NO\\sCARRIER TIMEOUT 5 \
7                 \"\" AT OK-AT-OK ATE1Q0 OK \\dATDT\\T TIMEOUT 40 CONNECT"
8       set timeout 180
9       enable dns
10
11    provider:
12      set phone "(123) 456 7890"
13      set authname foo
14      set authkey bar
15      set login "TIMEOUT 10 \"\" \"\" gin:--gin: \\U word: \\P col: ppp"
16      set timeout 300
17      set ifaddr x.x.x.x y.y.y.y 255.255.255.255 0.0.0.0
18      add default HISADDR
```

Line 1:

Identifies the default entry. Commands in this entry are executed automatically when ppp is run.

Line 2:

Enables logging parameters. When the configuration is working satisfactorily, this line should be reduced to saying:

```
set log phase tun
```

in order to avoid excessive log file sizes.

Line 3:

Tells PPP how to identify itself to the peer. PPP identifies itself to the peer if it has any trouble negotiating and setting up the link, providing information that the peers administrator may find useful when investigating such problems.

Line 4:

Identifies the device to which the modem is connected. `COM1` is `/dev/cuau0` and `COM2` is `/dev/cuau1`.

Line 5:

Sets the speed you want to connect at. If 115200 does not work (it should with any reasonably new modem), try 38400 instead.

Line 6 & 7:

The dial string. User PPP uses an expect-send syntax similar to the chat(8) program. Refer to the manual page for information on the features of this language.

Note that this command continues onto the next line for readability. Any command in `ppp.conf` may do this if the last character on the line is a `\` character.

Line 8:

Sets the idle timeout for the link. 180 seconds is the default, so this line is purely cosmetic.

Line 9:

Tells PPP to ask the peer to confirm the local resolver settings. If you run a local name server, this line should be commented out or removed.

Line 10:

A blank line for readability. Blank lines are ignored by PPP.

Line 11:

Identifies an entry for a provider called “provider”. This could be changed to the name of your ISP so that later you can use the `load ISP` to start the connection.

Line 12:

Sets the phone number for this provider. Multiple phone numbers may be specified using the colon (`:`) or pipe character (`|`) as a separator. The difference between the two separators is described in ppp(8). To summarize, if you want to rotate through the numbers, use a colon. If you want to always attempt to dial the first number first and only use the other numbers if the first number fails, use the pipe character. Always quote the entire set of phone numbers as shown.

You must enclose the phone number in quotation marks (`"`) if there is any intention on using spaces in the phone number. This can cause a simple, yet subtle error.

Line 13 & 14:

Identifies the user name and password. When connecting using a UNIX style login prompt, these values are referred to by the `set login` command using the \U and \P variables. When connecting using PAP or CHAP, these values are used at authentication time.

Line 15:

If you are using PAP or CHAP, there will be no login at this point, and this line should be commented out or removed. See PAP and CHAP authentication for further details.

The login string is of the same chat-like syntax as the dial string. In this example, the string works for a service whose login session looks like this:

```
J. Random Provider
login: foo
password: bar
protocol: ppp
```

You will need to alter this script to suit your own needs. When you write this script for the first time, you should ensure that you have enabled “chat” logging so you can determine if the conversation is going as expected.

Line 16:

Sets the default idle timeout (in seconds) for the connection. Here, the connection will be closed automatically after 300 seconds of inactivity. If you never want to timeout, set this value to zero or use the `-ddial` command line switch.

Line 17:

Sets the interface addresses. The string *`x.x.x.x`* should be replaced by the IP address that your provider has allocated to you. The string *`y.y.y.y`* should be replaced by the IP address that your ISP indicated for their gateway (the machine to which you connect). If your ISP has not given you a gateway address, use `10.0.0.2/0`. If you need to use a "guessed" address, make sure that you create an entry in `/etc/ppp/ppp.linkup` as per the instructions for PPP and Dynamic IP addresses. If this line is omitted, `ppp` cannot run in `-auto` mode.

Line 18:

Adds a default route to your ISP's gateway. The special word `HISADDR` is replaced with the gateway address specified on line 17. It is important that this line appears after line 17, otherwise `HISADDR` will not yet be initialized.

If you do not wish to run ppp in `-auto`, this line should be moved to the `ppp.linkup` file.

It is not necessary to add an entry to `ppp.linkup` when you have a static IP address and are running ppp in `-auto` mode as your routing table entries are already correct before you connect. You may however wish to create an entry to invoke programs after connection. This is explained later with the sendmail example.

Example configuration files can be found in the `/usr/share/examples/ppp/` directory.

28.2.1.2.2 PPP and Dynamic IP Addresses

If your service provider does not assign static IP addresses, `ppp` can be configured to negotiate the local and remote addresses. This is done by "guessing" an IP address and allowing `ppp` to set it up correctly using the IP Configuration Protocol (IPCP) after connecting. The `ppp.conf` configuration is the same as PPP and Static IP Addresses, with the following change:

```
17      set ifaddr 10.0.0.1/0 10.0.0.2/0 255.255.255.255 0.0.0.0
```

Again, do not include the line number, it is just for reference. Indentation of at least one space is required.

Line 17:

The number after the `/` character is the number of bits of the address that ppp will insist on. You may wish to use IP numbers more appropriate to your circumstances, but the above example will always work.

The last argument (`0.0.0.0`) tells PPP to start negotiations using address `0.0.0.0` rather than `10.0.0.1` and is necessary for some ISPs. Do not use `0.0.0.0` as the first argument to `set ifaddr` as it prevents PPP from setting up an initial route in `-auto` mode.

If you are not running in `-auto` mode, you will need to create an entry in `/etc/ppp/ppp.linkup`. `ppp.linkup` is used after a connection has been established. At this point, `ppp` will have assigned the interface addresses and it will now be possible to add the routing table entries:

```
1     provider:
2      add default HISADDR
```

Line 1:

> On establishing a connection, `ppp` will look for an entry in `ppp.linkup` according to the following rules: First, try to match the same label as we used in `ppp.conf`. If that fails, look for an entry for the IP address of our gateway. This entry is a four-octet IP style label. If we still have not found an entry, look for the `MYADDR` entry.

Line 2:

> This line tells `ppp` to add a default route that points to `HISADDR`. `HISADDR` will be replaced with the IP number of the gateway as negotiated by the IPCP.

See the `pmdemand` entry in the files `/usr/share/examples/ppp/ppp.conf.sample` and `/usr/share/examples/ppp/ppp.linkup.sample` for a detailed example.

28.2.1.2.3 Receiving Incoming Calls

When you configure **ppp** to receive incoming calls on a machine connected to a LAN, you must decide if you wish to forward packets to the LAN. If you do, you should allocate the peer an IP number from your LAN's subnet, and use the command `enable proxy` in your `/etc/ppp/ppp.conf` file. You should also confirm that the `/etc/rc.conf` file contains the following:

```
gateway_enable="YES"
```

28.2.1.2.4 Which getty?

Configuring FreeBSD for Dial-up Services provides a good description on enabling dial-up services using getty(8).

An alternative to `getty` is mgetty (http://mgetty.greenie.net/) (from `comms/mgetty+sendfax` port), a smarter version of `getty` designed with dial-up lines in mind.

The advantages of using `mgetty` is that it actively *talks* to modems, meaning if port is turned off in `/etc/ttys` then your modem will not answer the phone.

Later versions of `mgetty` (from 0.99beta onwards) also support the automatic detection of PPP streams, allowing your clients script-less access to your server.

Refer to Mgetty and AutoPPP for more information on `mgetty`.

*28.2.1.2.5 **PPP** Permissions*

The `ppp` command must normally be run as the `root` user. If however, you wish to allow `ppp` to run in server mode as a normal user by executing `ppp` as described below, that user must be given permission to run `ppp` by adding them to the `network` group in `/etc/group`.

You will also need to give them access to one or more sections of the configuration file using the `allow` command:

```
allow users fred mary
```

If this command is used in the `default` section, it gives the specified users access to everything.

28.2.1.2.6 PPP Shells for Dynamic-IP Users

Create a file called `/etc/ppp/ppp-shell` containing the following:

```
#!/bin/sh
IDENT=‘echo $0 | sed -e ’s/^.*-\(.*\)$/\1/’‘
CALLEDAS="$IDENT"
TTY=‘tty‘

if [ x$IDENT = xdialup ]; then
        IDENT=‘basename $TTY‘
fi

echo "PPP for $CALLEDAS on $TTY"
echo "Starting PPP for $IDENT"

exec /usr/sbin/ppp -direct $IDENT
```

This script should be executable. Now make a symbolic link called `ppp-dialup` to this script using the following commands:

```
# ln -s ppp-shell /etc/ppp/ppp-dialup
```

You should use this script as the *shell* for all of your dialup users. This is an example from `/etc/passwd` for a dialup PPP user with username `pchilds` (remember do not directly edit the password file, use vipw(8)).

```
pchilds:*:1011:300:Peter Childs PPP:/home/ppp:/etc/ppp/ppp-dialup
```

Create a `/home/ppp` directory that is world readable containing the following 0 byte files:

```
-r--r--r--   1 root     wheel           0 May 27 02:23 .hushlogin
-r--r--r--   1 root     wheel           0 May 27 02:22 .rhosts
```

which prevents `/etc/motd` from being displayed.

28.2.1.2.7 PPP Shells for Static-IP Users

Create the `ppp-shell` file as above, and for each account with statically assigned IPs create a symbolic link to `ppp-shell`.

For example, if you have three dialup customers, `fred`, `sam`, and `mary`, that you route /24 CIDR networks for, you would type the following:

```
# ln -s /etc/ppp/ppp-shell /etc/ppp/ppp-fred
# ln -s /etc/ppp/ppp-shell /etc/ppp/ppp-sam
# ln -s /etc/ppp/ppp-shell /etc/ppp/ppp-mary
```

Each of these users dialup accounts should have their shell set to the symbolic link created above (for example, `mary`'s shell should be `/etc/ppp/ppp-mary`).

28.2.1.2.8 Setting Up `ppp.conf` for Dynamic-IP Users

The `/etc/ppp/ppp.conf` file should contain something along the lines of:

```
default:
  set debug phase lcp chat
  set timeout 0

ttyu0:
  set ifaddr 203.14.100.1 203.14.100.20 255.255.255.255
  enable proxy

ttyu1:
  set ifaddr 203.14.100.1 203.14.100.21 255.255.255.255
  enable proxy
```

> **Note:** The indenting is important.

The `default:` section is loaded for each session. For each dialup line enabled in `/etc/ttys` create an entry similar to the one for `ttyu0:` above. Each line should get a unique IP address from your pool of IP addresses for dynamic users.

28.2.1.2.9 Setting Up `ppp.conf` for Static-IP Users

Along with the contents of the sample `/usr/share/examples/ppp/ppp.conf` above you should add a section for each of the statically assigned dialup users. We will continue with our `fred`, `sam`, and `mary` example.

```
fred:
  set ifaddr 203.14.100.1 203.14.101.1 255.255.255.255

sam:
  set ifaddr 203.14.100.1 203.14.102.1 255.255.255.255

mary:
  set ifaddr 203.14.100.1 203.14.103.1 255.255.255.255
```

The file `/etc/ppp/ppp.linkup` should also contain routing information for each static IP user if required. The line below would add a route for the `203.14.101.0/24` network via the client's ppp link.

```
fred:
  add 203.14.101.0 netmask 255.255.255.0 HISADDR

sam:
  add 203.14.102.0 netmask 255.255.255.0 HISADDR

mary:
```

```
  add 203.14.103.0 netmask 255.255.255.0 HISADDR
```

28.2.1.2.10 mgetty and AutoPPP

By default the comms/mgetty+sendfax port comes with the AUTO_PPP option enabled allowing mgetty to detect the LCP phase of PPP connections and automatically spawn off a ppp shell. However, since the default login/password sequence does not occur it is necessary to authenticate users using either PAP or CHAP.

This section assumes the user has successfully compiled, and installed the comms/mgetty+sendfax port on his system.

Make sure your /usr/local/etc/mgetty+sendfax/login.config file has the following in it:

```
/AutoPPP/ -     - /etc/ppp/ppp-pap-dialup
```

This will tell mgetty to run the ppp-pap-dialup script for detected PPP connections.

Create a file called /etc/ppp/ppp-pap-dialup containing the following (the file should be executable):

```
#!/bin/sh
exec /usr/sbin/ppp -direct pap$IDENT
```

For each dialup line enabled in /etc/ttys, create a corresponding entry in /etc/ppp/ppp.conf. This will happily co-exist with the definitions we created above.

```
pap:
  enable pap
  set ifaddr 203.14.100.1 203.14.100.20-203.14.100.40
  enable proxy
```

Each user logging in with this method will need to have a username/password in /etc/ppp/ppp.secret file, or alternatively add the following option to authenticate users via PAP from the /etc/passwd file.

```
enable passwdauth
```

If you wish to assign some users a static IP number, you can specify the number as the third argument in /etc/ppp/ppp.secret. See /usr/share/examples/ppp/ppp.secret.sample for examples.

28.2.1.2.11 MS Extensions

It is possible to configure PPP to supply DNS and NetBIOS nameserver addresses on demand.

To enable these extensions with PPP version 1.x, the following lines might be added to the relevant section of /etc/ppp/ppp.conf.

```
enable msext
set ns 203.14.100.1 203.14.100.2
set nbns 203.14.100.5
```

And for PPP version 2 and above:

```
accept dns
set dns 203.14.100.1 203.14.100.2
```

```
set nbns 203.14.100.5
```

This will tell the clients the primary and secondary name server addresses, and a NetBIOS nameserver host.

In version 2 and above, if the `set dns` line is omitted, PPP will use the values found in `/etc/resolv.conf`.

28.2.1.2.12 PAP and CHAP Authentication

Some ISPs set their system up so that the authentication part of your connection is done using either of the PAP or CHAP authentication mechanisms. If this is the case, your ISP will not give a `login:` prompt when you connect, but will start talking PPP immediately.

PAP is less secure than CHAP, but security is not normally an issue here as passwords, although being sent as plain text with PAP, are being transmitted down a serial line only. There is not much room for crackers to “eavesdrop”.

Referring back to the PPP and Static IP addresses or PPP and Dynamic IP addresses sections, the following alterations must be made:

```
13      set authname MyUserName
14      set authkey MyPassword
15      set login
```

Line 13:

This line specifies your PAP/CHAP user name. You will need to insert the correct value for *`MyUserName`*.

Line 14:

This line specifies your PAP/CHAP password. You will need to insert the correct value for *`MyPassword`*. You may want to add an additional line, such as:

```
16      accept PAP
```

or

```
16      accept CHAP
```

to make it obvious that this is the intention, but PAP and CHAP are both accepted by default.

Line 15:

Your ISP will not normally require that you log into the server if you are using PAP or CHAP. You must therefore disable your “set login” string.

28.2.1.2.13 Changing Your ppp Configuration on the Fly

It is possible to talk to the `ppp` program while it is running in the background, but only if a suitable diagnostic port has been set up. To do this, add the following line to your configuration:

```
set server /var/run/ppp-tun%d DiagnosticPassword 0177
```

This will tell PPP to listen to the specified UNIX domain socket, asking clients for the specified password before allowing access. The `%d` in the name is replaced with the `tun` device number that is in use.

Once a socket has been set up, the pppctl(8) program may be used in scripts that wish to manipulate the running program.

28.2.1.3 Using PPP Network Address Translation Capability

PPP has ability to use internal NAT without kernel diverting capabilities. This functionality may be enabled by the following line in `/etc/ppp/ppp.conf`:

```
nat enable yes
```

Alternatively, PPP NAT may be enabled by command-line option `-nat`. There is also `/etc/rc.conf` knob named `ppp_nat`, which is enabled by default.

If you use this feature, you may also find useful the following `/etc/ppp/ppp.conf` options to enable incoming connections forwarding:

```
nat port tcp 10.0.0.2:ftp ftp
nat port tcp 10.0.0.2:http http
```

or do not trust the outside at all

```
nat deny_incoming yes
```

28.2.1.4 Final System Configuration

You now have `ppp` configured, but there are a few more things to do before it is ready to work. They all involve editing the `/etc/rc.conf` file.

Working from the top down in this file, make sure the `hostname=` line is set, e.g.:

```
hostname="foo.example.com"
```

If your ISP has supplied you with a static IP address and name, it is probably best that you use this name as your host name.

Look for the `network_interfaces` variable. If you want to configure your system to dial your ISP on demand, make sure the `tun0` device is added to the list, otherwise remove it.

```
network_interfaces="lo0 tun0"
ifconfig_tun0=
```

> **Note:** The `ifconfig_tun0` variable should be empty, and a file called `/etc/start_if.tun0` should be created. This file should contain the line:
>
> ```
> ppp -auto mysystem
> ```
>
> This script is executed at network configuration time, starting your ppp daemon in automatic mode. If you have a LAN for which this machine is a gateway, you may also wish to use the `-alias` switch. Refer to the manual page for further details.

Make sure that the router program is set to NO with the following line in your /etc/rc.conf:

```
router_enable="NO"
```

It is important that the routed daemon is not started, as routed tends to delete the default routing table entries created by ppp.

It is probably a good idea to ensure that the sendmail_flags line does not include the -q option, otherwise sendmail will attempt to do a network lookup every now and then, possibly causing your machine to dial out. You may try:

```
sendmail_flags="-bd"
```

The downside of this is that you must force sendmail to re-examine the mail queue whenever the ppp link is up by typing:

```
# /usr/sbin/sendmail -q
```

You may wish to use the !bg command in ppp.linkup to do this automatically:

```
1     provider:
2       delete ALL
3       add 0 0 HISADDR
4       !bg sendmail -bd -q30m
```

If you do not like this, it is possible to set up a “dfilter” to block SMTP traffic. Refer to the sample files for further details.

All that is left is to reboot the machine. After rebooting, you can now either type:

```
# ppp
```

and then dial provider to start the PPP session, or, if you want ppp to establish sessions automatically when there is outbound traffic (and you have not created the start_if.tun0 script), type:

```
# ppp -auto provider
```

28.2.1.5 Summary

To recap, the following steps are necessary when setting up ppp for the first time:

Client side:

1. Ensure that the tun device is built into your kernel.
2. Ensure that the tunN device file is available in the /dev directory.
3. Create an entry in /etc/ppp/ppp.conf. The pmdemand example should suffice for most ISPs.
4. If you have a dynamic IP address, create an entry in /etc/ppp/ppp.linkup.
5. Update your /etc/rc.conf file.
6. Create a start_if.tun0 script if you require demand dialing.

Server side:

1. Ensure that the `tun` device is built into your kernel.
2. Ensure that the `tunN` device file is available in the `/dev` directory.
3. Create an entry in `/etc/passwd` (using the vipw(8) program).
4. Create a profile in this users home directory that runs `ppp -direct direct-server` or similar.
5. Create an entry in `/etc/ppp/ppp.conf`. The `direct-server` example should suffice.
6. Create an entry in `/etc/ppp/ppp.linkup`.
7. Update your `/etc/rc.conf` file.

28.3 Using Kernel PPP

Parts originally contributed by Gennady B. Sorokopud and Robert Huff.

Warning: This section applies and is valid only for FreeBSD 7.X.

28.3.1 Setting Up Kernel PPP

Before you start setting up PPP on your machine, make sure that `pppd` is located in `/usr/sbin` and the directory `/etc/ppp` exists.

`pppd` can work in two modes:

1. As a “client” — you want to connect your machine to the outside world via a PPP serial connection or modem line.
2. As a “server” — your machine is located on the network, and is used to connect other computers using PPP.

In both cases you will need to set up an options file (`/etc/ppp/options` or `~/.ppprc` if you have more than one user on your machine that uses PPP).

You will also need some modem/serial software (preferably `comms/kermit`), so you can dial and establish a connection with the remote host.

28.3.2 Using `pppd` as a Client

Based on information provided by Trev Roydhouse.

The following `/etc/ppp/options` might be used to connect to a Cisco terminal server PPP line.

```
crtscts         # enable hardware flow control
modem           # modem control line
```

```
noipdefault     # remote PPP server must supply your IP address
                # if the remote host does not send your IP during IPCP
                # negotiation, remove this option
passive         # wait for LCP packets
domain ppp.foo.com      # put your domain name here

:remote_ip    # put the IP of remote PPP host here
                # it will be used to route packets via PPP link
                # if you didn't specified the noipdefault option
                # change this line to local_ip:remote_ip

defaultroute    # put this if you want that PPP server will be your
                # default router
```

To connect:

1. Dial to the remote host using **Kermit** (or some other modem program), and enter your user name and password (or whatever is needed to enable PPP on the remote host).
2. Exit **Kermit** (without hanging up the line).
3. Enter the following:

   ```
   # /usr/sbin/pppd /dev/tty01 19200
   ```

   Be sure to use the appropriate speed and device name.

Now your computer is connected with PPP. If the connection fails, you can add the `debug` option to the `/etc/ppp/options` file, and check console messages to track the problem.

Following `/etc/ppp/pppup` script will make all 3 stages automatic:

```
#!/bin/sh
pgrep -l pppd
pid=‘pgrep pppd‘
if [ "X${pid}" != "X" ] ; then
        echo ’killing pppd, PID=’ ${pid}
        kill ${pid}
fi
pgrep -l kermit
pid=‘pgrep kermit‘
if [ "X${pid}" != "X" ] ; then
        echo ’killing kermit, PID=’ ${pid}
        kill -9 ${pid}
fi

ifconfig ppp0 down
ifconfig ppp0 delete

kermit -y /etc/ppp/kermit.dial
pppd /dev/tty01 19200
```

`/etc/ppp/kermit.dial` is a **Kermit** script that dials and makes all necessary authorization on the remote host (an example of such a script is attached to the end of this document).

Use the following `/etc/ppp/pppdown` script to disconnect the PPP line:

```
#!/bin/sh
pid=‘pgrep pppd‘
if [ X${pid} != "X" ] ; then
        echo ’killing pppd, PID=’ ${pid}
        kill -TERM ${pid}
fi

pgrep -l kermit
pid=‘pgrep kermit‘
if [ "X${pid}" != "X" ] ; then
        echo ’killing kermit, PID=’ ${pid}
        kill -9 ${pid}
fi

/sbin/ifconfig ppp0 down
/sbin/ifconfig ppp0 delete
kermit -y /etc/ppp/kermit.hup
/etc/ppp/ppptest
```

Check to see if `pppd` is still running by executing `/usr/etc/ppp/ppptest`, which should look like this:

```
#!/bin/sh
pid=‘pgrep pppd‘
if [ X${pid} != "X" ] ; then
        echo ’pppd running: PID=’ ${pid-NONE}
else
        echo ’No pppd running.’
fi
set -x
netstat -n -I ppp0
ifconfig ppp0
```

To hang up the modem, execute `/etc/ppp/kermit.hup`, which should contain:

```
set line /dev/tty01     ; put your modem device here
set speed 19200
set file type binary
set file names literal
set win 8
set rec pack 1024
set send pack 1024
set block 3
set term bytesize 8
set command bytesize 8
set flow none

pau 1
out +++
inp 5 OK
out ATH0\13
echo \13
```

```
exit
```

Here is an alternate method using `chat` instead of `kermit`:

The following two files are sufficient to accomplish a `pppd` connection.

`/etc/ppp/options`:

```
/dev/cuad1 115200

crtscts                # enable hardware flow control
modem          # modem control line
connect "/usr/bin/chat -f /etc/ppp/login.chat.script"
noipdefault    # remote PPP serve must supply your IP address
               # if the remote host doesn't send your IP during
                # IPCP negotiation, remove this option
passive         # wait for LCP packets
domain your.domain     # put your domain name here

:              # put the IP of remote PPP host here
               # it will be used to route packets via PPP link
                # if you didn't specified the noipdefault option
                # change this line to local_ip:remote_ip

defaultroute   # put this if you want that PPP server will be
               # your default router
```

`/etc/ppp/login.chat.script`:

> **Note:** The following should go on a single line.

```
ABORT BUSY ABORT 'NO CARRIER' "" AT OK ATDTphone.number
  CONNECT "" TIMEOUT 10 ogin:-\\r-ogin: login-id
  TIMEOUT 5 sword: password
```

Once these are installed and modified correctly, all you need to do is run `pppd`, like so:

```
# pppd
```

28.3.3 Using `pppd` as a Server

`/etc/ppp/options` should contain something similar to the following:

```
crtscts                          # Hardware flow control
netmask 255.255.255.0            # netmask (not required)
192.114.208.20:192.114.208.165   # IP's of local and remote hosts
                                 # local ip must be different from one
                                 # you assigned to the Ethernet (or other)
                                 # interface on your machine.
                                 # remote IP is IP address that will be
                                 # assigned to the remote machine
```

```
domain ppp.foo.com              # your domain
passive                         # wait for LCP
modem                           # modem line
```

The following /etc/ppp/pppserv script will tell **pppd** to behave as a server:

```
#!/bin/sh
pgrep -l pppd
pid=‘pgrep pppd‘
if [ "X${pid}" != "X" ] ; then
        echo ’killing pppd, PID=’ ${pid}
        kill ${pid}
fi
pgrep -l kermit
pid=‘pgrep kermit‘
if [ "X${pid}" != "X" ] ; then
        echo ’killing kermit, PID=’ ${pid}
        kill -9 ${pid}
fi

# reset ppp interface
ifconfig ppp0 down
ifconfig ppp0 delete

# enable autoanswer mode
kermit -y /etc/ppp/kermit.ans

# run ppp
pppd /dev/tty01 19200
```

Use this /etc/ppp/pppservdown script to stop the server:

```
#!/bin/sh
pgrep -l pppd
pid=‘pgrep pppd‘
if [ "X${pid}" != "X" ] ; then
        echo ’killing pppd, PID=’ ${pid}
        kill ${pid}
fi
pgrep -l kermit
pid=‘pgrep kermit‘
if [ "X${pid}" != "X" ] ; then
        echo ’killing kermit, PID=’ ${pid}
        kill -9 ${pid}
fi
ifconfig ppp0 down
ifconfig ppp0 delete

kermit -y /etc/ppp/kermit.noans
```

The following **Kermit** script (/etc/ppp/kermit.ans) will enable/disable autoanswer mode on your modem. It should look like this:

```
set line /dev/tty01
set speed 19200
set file type binary
set file names literal
set win 8
set rec pack 1024
set send pack 1024
set block 3
set term bytesize 8
set command bytesize 8
set flow none

pau 1
out +++
inp 5 OK
out ATH0\13
inp 5 OK
echo \13
out ATS0=1\13   ; change this to out ATS0=0\13 if you want to disable
                ; autoanswer mode
inp 5 OK
echo \13
exit
```

A script named `/etc/ppp/kermit.dial` is used for dialing and authenticating on the remote host. You will need to customize it for your needs. Put your login and password in this script; you will also need to change the input statement depending on responses from your modem and remote host.

```
;
; put the com line attached to the modem here:
;
set line /dev/tty01
;
; put the modem speed here:
;
set speed 19200
set file type binary            ; full 8 bit file xfer
set file names literal
set win 8
set rec pack 1024
set send pack 1024
set block 3
set term bytesize 8
set command bytesize 8
set flow none
set modem hayes
set dial hangup off
set carrier auto                ; Then SET CARRIER if necessary,
set dial display on             ; Then SET DIAL if necessary,
set input echo on
set input timeout proceed
set input case ignore
def \%x 0                       ; login prompt counter
```

```
goto slhup

:slcmd                          ; put the modem in command mode
echo Put the modem in command mode.
clear                           ; Clear unread characters from input buffer
pause 1
output +++                      ; hayes escape sequence
input 1 OK\13\10                ; wait for OK
if success goto slhup
output \13
pause 1
output at\13
input 1 OK\13\10
if fail goto slcmd              ; if modem doesn't answer OK, try again

:slhup                          ; hang up the phone
clear                           ; Clear unread characters from input buffer
pause 1
echo Hanging up the phone.
output ath0\13                  ; hayes command for on hook
input 2 OK\13\10
if fail goto slcmd              ; if no OK answer, put modem in command mode

:sldial                         ; dial the number
pause 1
echo Dialing.
output atdt9,550311\13\10               ; put phone number here
assign \%x 0                    ; zero the time counter

:look
clear                           ; Clear unread characters from input buffer
increment \%x                   ; Count the seconds
input 1 {CONNECT }
if success goto sllogin
reinput 1 {NO CARRIER\13\10}
if success goto sldial
reinput 1 {NO DIALTONE\13\10}
if success goto slnodial
reinput 1 {\255}
if success goto slhup
reinput 1 {\127}
if success goto slhup
if < \%x 60 goto look
else goto slhup

:sllogin                        ; login
assign \%x 0                    ; zero the time counter
pause 1
echo Looking for login prompt.

:slloop
increment \%x                   ; Count the seconds
clear                           ; Clear unread characters from input buffer
```

```
output \13
;
; put your expected login prompt here:
;
input 1 {Username: }
if success goto sluid
reinput 1 {\255}
if success goto slhup
reinput 1 {\127}
if success goto slhup
if < \%x 10 goto slloop         ; try 10 times to get a login prompt
else goto slhup                 ; hang up and start again if 10 failures

:sluid
;
; put your userid here:
;
output ppp-login\13
input 1 {Password: }
;
; put your password here:
;
output ppp-password\13
input 1 {Entering SLIP mode.}
echo
quit

:slnodial
echo \7No dialtone.  Check the telephone line!\7
exit 1

; local variables:
; mode: csh
; comment-start: "; "
; comment-start-skip: "; "
; end:
```

28.4 Troubleshooting PPP Connections

Contributed by Tom Rhodes.

This section covers a few issues which may arise when using PPP over a modem connection. For instance, perhaps you need to know exactly what prompts the system you are dialing into will present. Some ISPs present the `ssword` prompt, and others will present `password`; if the `ppp` script is not written accordingly, the login attempt will fail. The most common way to debug `ppp` connections is by connecting manually. The following information will walk you through a manual connection step by step.

28.4.1 Check the Device Nodes

When using a custom kernel, make sure to include the following line in your kernel configuration file:

```
device   uart
```

The `uart` device is already included in the `GENERIC` kernel, so no additional steps are necessary in this case. Just check the `dmesg` output for the modem device with:

```
# dmesg | grep uart
```

You should get some pertinent output about the `uart` devices. These are the COM ports we need. If your modem acts like a standard serial port then you should see it listed on `uart1`, or `COM2`. If so, you are not required to rebuild the kernel. When matching up sio modem is on `uart1` or `COM2` if you are in DOS, then your modem device would be `/dev/cuau1`.

28.4.2 Connecting Manually

Connecting to the Internet by manually controlling `ppp` is quick, easy, and a great way to debug a connection or just get information on how your ISP treats `ppp` client connections. Lets start **PPP** from the command line. Note that in all of our examples we will use *example* as the hostname of the machine running **PPP**. You start `ppp` by just typing `ppp`:

```
# ppp
```

We have now started `ppp`.

```
ppp ON example> set device /dev/cuau1
```

We set our modem device, in this case it is `cuau1`.

```
ppp ON example> set speed 115200
```

Set the connection speed, in this case we are using 115,200 kbps.

```
ppp ON example> enable dns
```

Tell `ppp` to configure our resolver and add the nameserver lines to `/etc/resolv.conf`. If `ppp` cannot determine our hostname, we can set one manually later.

```
ppp ON example> term
```

Switch to “terminal” mode so that we can manually control the modem.

```
deflink: Entering terminal mode on /dev/cuau1
type '~h' for help

at
OK
atdt123456789
```

Use `at` to initialize the modem, then use `atdt` and the number for your ISP to begin the dial in process.

```
CONNECT
```

Confirmation of the connection, if we are going to have any connection problems, unrelated to hardware, here is where we will attempt to resolve them.

```
ISP Login:myusername
```

Here you are prompted for a username, return the prompt with the username that was provided by the ISP.

```
ISP Pass:mypassword
```

This time we are prompted for a password, just reply with the password that was provided by the ISP. Just like logging into FreeBSD, the password will not echo.

```
Shell or PPP:ppp
```

Depending on your ISP this prompt may never appear. Here we are being asked if we wish to use a shell on the provider, or to start ppp. In this example, we have chosen to use ppp as we want an Internet connection.

```
Ppp ON example>
```

Notice that in this example the first p has been capitalized. This shows that we have successfully connected to the ISP.

```
PPp ON example>
```

We have successfully authenticated with our ISP and are waiting for the assigned IP address.

```
PPP ON example>
```

We have made an agreement on an IP address and successfully completed our connection.

```
PPP ON example>add default HISADDR
```

Here we add our default route, we need to do this before we can talk to the outside world as currently the only established connection is with the peer. If this fails due to existing routes you can put a bang character ! in front of the add. Alternatively, you can set this before making the actual connection and it will negotiate a new route accordingly.

If everything went good we should now have an active connection to the Internet, which could be thrown into the background using **CTRL**+**z** If you notice the PPP return to ppp then we have lost our connection. This is good to know because it shows our connection status. Capital P's show that we have a connection to the ISP and lowercase p's show that the connection has been lost for whatever reason. ppp only has these 2 states.

28.4.2.1 Debugging

If you have a direct line and cannot seem to make a connection, then turn hardware flow CTS/RTS to off with the `set ctsrts off`. This is mainly the case if you are connected to some **PPP** capable terminal servers, where **PPP** hangs when it tries to write data to your communication link, so it would be waiting for a CTS, or Clear To Send signal which may never come. If you use this option however, you should also use the `set accmap` option, which may be required to defeat hardware dependent on passing certain characters from end to end, most of the time XON/XOFF. See the ppp(8) manual page for more information on this option, and how it is used.

If you have an older modem, you may need to use the `set parity even`. Parity is set at none be default, but is used for error checking (with a large increase in traffic) on older modems and some ISPs. You may need this option for the Compuserve ISP.

PPP may not return to the command mode, which is usually a negotiation error where the ISP is waiting for your side to start negotiating. At this point, using the `~p` command will force ppp to start sending the configuration information.

If you never obtain a login prompt, then most likely you need to use PAP or CHAP authentication instead of the UNIX style in the example above. To use PAP or CHAP just add the following options to **PPP** before going into terminal mode:

```
ppp ON example> set authname myusername
```

Where *myusername* should be replaced with the username that was assigned by the ISP.

```
ppp ON example> set authkey mypassword
```

Where *mypassword* should be replaced with the password that was assigned by the ISP.

If you connect fine, but cannot seem to find any domain name, try to use ping(8) with an IP address and see if you can get any return information. If you experience 100 percent (100%) packet loss, then it is most likely that you were not assigned a default route. Double check that the option `add default HISADDR` was set during the connection. If you can connect to a remote IP address then it is possible that a resolver address has not been added to the `/etc/resolv.conf`. This file should look like:

```
domain example.com
nameserver x.x.x.x
nameserver y.y.y.y
```

Where *x.x.x.x* and *y.y.y.y* should be replaced with the IP address of your ISP's DNS servers. This information may or may not have been provided when you signed up, but a quick call to your ISP should remedy that.

You could also have syslog(3) provide a logging function for your **PPP** connection. Just add:

```
!ppp
*.*     /var/log/ppp.log
```

to `/etc/syslog.conf`. In most cases, this functionality already exists.

28.5 Using PPP over Ethernet (PPPoE)

Contributed (from http://node.to/freebsd/how-tos/how-to-freebsd-pppoe.html) by Jim Mock.

This section describes how to set up PPP over Ethernet (PPPoE).

28.5.1 Configuring the Kernel

No kernel configuration is necessary for PPPoE any longer. If the necessary netgraph support is not built into the kernel, it will be dynamically loaded by **ppp**.

28.5.2 Setting Up `ppp.conf`

Here is an example of a working `ppp.conf`:

```
default:
  set log Phase tun command # you can add more detailed logging if you wish
  set ifaddr 10.0.0.1/0 10.0.0.2/0

name_of_service_provider:
  set device PPPoE:xl1 # replace xl1 with your Ethernet device
  set authname YOURLOGINNAME
  set authkey YOURPASSWORD
  set dial
  set login
  add default HISADDR
```

28.5.3 Running ppp

As `root`, you can run:

```
# ppp -ddial name_of_service_provider
```

28.5.4 Starting ppp at Boot

Add the following to your `/etc/rc.conf` file:

```
ppp_enable="YES"
ppp_mode="ddial"
ppp_nat="YES"  # if you want to enable nat for your local network, otherwise NO
ppp_profile="name_of_service_provider"
```

28.5.5 Using a PPPoE Service Tag

Sometimes it will be necessary to use a service tag to establish your connection. Service tags are used to distinguish between different PPPoE servers attached to a given network.

You should have been given any required service tag information in the documentation provided by your ISP. If you cannot locate it there, ask your ISP's tech support personnel.

As a last resort, you could try the method suggested by the Roaring Penguin PPPoE (http://www.roaringpenguin.com/pppoe/) program which can be found in the Ports Collection. Bear in mind however, this may de-program your modem and render it useless, so think twice before doing it. Simply install the program shipped with the modem by your provider. Then, access the System menu from the program. The name of your profile should be listed there. It is usually *ISP*.

The profile name (service tag) will be used in the PPPoE configuration entry in `ppp.conf` as the provider part of the `set device` command (see the ppp(8) manual page for full details). It should look like this:

```
set device PPPoE:xl1:ISP
```

Do not forget to change *xl1* to the proper device for your Ethernet card.

Do not forget to change *ISP* to the profile you have just found above.

For additional information, see:

- Cheaper Broadband with FreeBSD on DSL (http://renaud.waldura.com/doc/freebsd/pppoe/) by Renaud Waldura.

28.5.6 PPPoE with a 3Com® HomeConnect® ADSL Modem Dual Link

This modem does not follow RFC 2516 (http://www.faqs.org/rfcs/rfc2516.html) (*A Method for transmitting PPP over Ethernet (PPPoE)*, written by L. Mamakos, K. Lidl, J. Evarts, D. Carrel, D. Simone, and R. Wheeler). Instead, different packet type codes have been used for the Ethernet frames. Please complain to 3Com (http://www.3com.com/) if you think it should comply with the PPPoE specification.

In order to make FreeBSD capable of communicating with this device, a sysctl must be set. This can be done automatically at boot time by updating `/etc/sysctl.conf`:

```
net.graph.nonstandard_pppoe=1
```

or can be done immediately with the command:

```
# sysctl net.graph.nonstandard_pppoe=1
```

Unfortunately, because this is a system-wide setting, it is not possible to talk to a normal PPPoE client or server and a 3Com HomeConnect® ADSL Modem at the same time.

28.6 Using PPP over ATM (PPPoA)

The following describes how to set up PPP over ATM (PPPoA). PPPoA is a popular choice among European DSL providers.

28.6.1 Using PPPoA with the Alcatel SpeedTouch™ USB

PPPoA support for this device is supplied as a port in FreeBSD because the firmware is distributed under Alcatel's license agreement (http://www.speedtouchdsl.com/disclaimer_lx.htm) and can not be redistributed freely with the base system of FreeBSD.

To install the software, simply use the Ports Collection. Install the `net/pppoa` port and follow the instructions provided with it.

Like many USB devices, the Alcatel SpeedTouch™ USB needs to download firmware from the host computer to operate properly. It is possible to automate this process in FreeBSD so that this transfer takes place whenever the device is plugged into a USB port. The following information can be added to the `/etc/usbd.conf` file to enable this automatic firmware transfer. This file must be edited as the `root` user.

```
device "Alcatel SpeedTouch USB"
    devname "ugen[0-9]+"
    vendor 0x06b9
    product 0x4061
```

```
    attach "/usr/local/sbin/modem_run -f /usr/local/libdata/mgmt.o"
```

To enable the USB daemon, **usbd**, put the following the line into `/etc/rc.conf`:

```
usbd_enable="YES"
```

It is also possible to set up **ppp** to dial up at startup. To do this add the following lines to `/etc/rc.conf`. Again, for this procedure you will need to be logged in as the `root` user.

```
ppp_enable="YES"
ppp_mode="ddial"
ppp_profile="adsl"
```

For this to work correctly you will need to have used the sample `ppp.conf` which is supplied with the `net/pppoa` port.

28.6.2 Using mpd

You can use **mpd** to connect to a variety of services, in particular PPTP services. You can find **mpd** in the Ports Collection, `net/mpd`. Many ADSL modems require that a PPTP tunnel is created between the modem and computer, one such modem is the Alcatel SpeedTouch Home.

First you must install the port, and then you can configure **mpd** to suit your requirements and provider settings. The port places a set of sample configuration files which are well documented in *`PREFIX`*`/etc/mpd/`. Note here that *`PREFIX`* means the directory into which your ports are installed, this defaults to `/usr/local/`. A complete guide to configure **mpd** is available in HTML format once the port has been installed. It is placed in *`PREFIX`*`/share/doc/mpd/`. Here is a sample configuration for connecting to an ADSL service with **mpd**. The configuration is spread over two files, first the `mpd.conf`:

Note: This example of the `mpd.conf` file only works with **mpd** 4.x.

```
default:
    load adsl

adsl:
    new -i ng0 adsl adsl
    set bundle authname username ❶
    set bundle password password ❷
    set bundle disable multilink

    set link no pap acfcomp protocomp
    set link disable chap
    set link accept chap
    set link keep-alive 30 10

    set ipcp no vjcomp
    set ipcp ranges 0.0.0.0/0 0.0.0.0/0

    set iface route default
    set iface disable on-demand
```

```
    set iface enable proxy-arp
    set iface idle 0

    open
```

❶ The username used to authenticate with your ISP.

❷ The password used to authenticate with your ISP.

The `mpd.links` file contains information about the link, or links, you wish to establish. An example `mpd.links` to accompany the above example is given beneath:

```
adsl:
    set link type pptp
    set pptp mode active
    set pptp enable originate outcall
    set pptp self 10.0.0.1 ❶
    set pptp peer 10.0.0.138 ❷
```

❶ The IP address of your FreeBSD computer which you will be using **mpd** from.

❷ The IP address of your ADSL modem. For the Alcatel SpeedTouch Home this address defaults to `10.0.0.138`.

It is possible to initialize the connection easily by issuing the following command as `root`:

```
# mpd -b adsl
```

You can see the status of the connection with the following command:

```
% ifconfig ng0
ng0: flags=88d1<UP,POINTOPOINT,RUNNING,NOARP,SIMPLEX,MULTICAST> mtu 1500
     inet 216.136.204.117 --> 204.152.186.171 netmask 0xffffffff
```

Using **mpd** is the recommended way to connect to an ADSL service with FreeBSD.

28.6.3 Using pptpclient

It is also possible to use FreeBSD to connect to other PPPoA services using `net/pptpclient`.

To use `net/pptpclient` to connect to a DSL service, install the port or package and edit your `/etc/ppp/ppp.conf`. You will need to be `root` to perform both of these operations. An example section of `ppp.conf` is given below. For further information on `ppp.conf` options consult the **ppp** manual page, ppp(8).

```
adsl:
 set log phase chat lcp ipcp ccp tun command
 set timeout 0
 enable dns
 set authname username ❶
 set authkey password ❷
 set ifaddr 0 0
 add default HISADDR
```

❶ The username of your account with the DSL provider.

❷ The password for your account.

Warning: Because you must put your account's password in the `ppp.conf` file in plain text form you should make sure than nobody can read the contents of this file. The following series of commands will make sure the file is only readable by the `root` account. Refer to the manual pages for chmod(1) and chown(8) for further information.

```
# chown root:wheel /etc/ppp/ppp.conf
# chmod 600 /etc/ppp/ppp.conf
```

This will open a tunnel for a PPP session to your DSL router. Ethernet DSL modems have a preconfigured LAN IP address which you connect to. In the case of the Alcatel SpeedTouch Home this address is `10.0.0.138`. Your router documentation should tell you which address your device uses. To open the tunnel and start a PPP session execute the following command:

```
# pptp address adsl
```

Tip: You may wish to add an ampersand ("&") to the end of the previous command because **pptp** will not return your prompt to you otherwise.

A `tun` virtual tunnel device will be created for interaction between the **pptp** and **ppp** processes. Once you have been returned to your prompt, or the **pptp** process has confirmed a connection you can examine the tunnel like so:

```
% ifconfig tun0
tun0: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST> mtu 1500
        inet 216.136.204.21 --> 204.152.186.171 netmask 0xffffff00
       Opened by PID 918
```

If you are unable to connect, check the configuration of your router, which is usually accessible via **telnet** or with a web browser. If you still cannot connect you should examine the output of the `pptp` command and the contents of the **ppp** log file, `/var/log/ppp.log` for clues.

28.7 Using SLIP

Originally contributed by Satoshi Asami. With input from Guy Helmer and Piero Serini.

Warning: This section applies and is valid only for FreeBSD 7.X.

28.7.1 Setting Up a SLIP Client

The following is one way to set up a FreeBSD machine for SLIP on a static host network. For dynamic hostname assignments (your address changes each time you dial up), you probably need to have a more complex setup.

First, determine which serial port your modem is connected to. Many people set up a symbolic link, such as `/dev/modem`, to point to the real device name, `/dev/cuadN`. This allows you to abstract the actual device name should you ever need to move the modem to a different port. It can become quite cumbersome when you need to fix a bunch of files in `/etc` and `.kermrc` files all over the system!

> **Note:** `/dev/cuad0` is `COM1`, `/dev/cuad1` is `COM2`, etc.

Make sure you have the following in your kernel configuration file:

```
device   sl
```

It is included in the `GENERIC` kernel, so this should not be a problem unless you have deleted it.

28.7.1.1 Things You Have to Do Only Once

1. Add your home machine, the gateway and nameservers to your `/etc/hosts` file. Ours looks like this:

   ```
   127.0.0.1               localhost loghost
   136.152.64.181          water.CS.Example.EDU water.CS water
   136.152.64.1            inr-3.CS.Example.EDU inr-3 slip-gateway
   128.32.136.9            ns1.Example.EDU ns1
   128.32.136.12           ns2.Example.EDU ns2
   ```

2. Make sure you have `files` before `dns` in the `hosts:` section of your `/etc/nsswitch.conf` file. Without these parameters funny things may happen.
3. Edit the `/etc/rc.conf` file.
   1. Set your hostname by editing the line that says:

      ```
      hostname="myname.my.domain"
      ```

      Your machine's full Internet hostname should be placed here.
   2. Designate the default router by changing the line:

      ```
      defaultrouter="NO"
      ```

      to:

      ```
      defaultrouter="slip-gateway"
      ```
4. Make a file `/etc/resolv.conf` which contains:

   ```
   domain CS.Example.EDU
   nameserver 128.32.136.9
   nameserver 128.32.136.12
   ```

   As you can see, these set up the nameserver hosts. Of course, the actual domain names and addresses depend on your environment.
5. Set the password for `root` and `toor` (and any other accounts that do not have a password).
6. Reboot your machine and make sure it comes up with the correct hostname.

28.7.1.2 Making a SLIP Connection

1. Dial up, type `slip` at the prompt, enter your machine name and password. What is required to be entered depends on your environment. If you use **Kermit**, you can try a script like this:

```
# kermit setup
set modem hayes
set line /dev/modem
set speed 115200
set parity none
set flow rts/cts
set terminal bytesize 8
set file type binary
# The next macro will dial up and login
define slip dial 643-9600, input 10 =>, if failure stop, -
output slip\x0d, input 10 Username:, if failure stop, -
output silvia\x0d, input 10 Password:, if failure stop, -
output ***\x0d, echo \x0aCONNECTED\x0a
```

   Of course, you have to change the username and password to fit yours. After doing so, you can just type `slip` from the **Kermit** prompt to connect.

   > **Note:** Leaving your password in plain text anywhere in the filesystem is generally a *bad* idea. Do it at your own risk.

2. Leave the **Kermit** there (you can suspend it by **Ctrl**-**z**) and as `root`, type:

```
# slattach -h -c -s 115200 /dev/modem
```

   If you are able to `ping` hosts on the other side of the router, you are connected! If it does not work, you might want to try `-a` instead of `-c` as an argument to `slattach`.

28.7.1.3 How to Shutdown the Connection

Do the following:

```
# kill -INT ‘cat /var/run/slattach.modem.pid‘
```

to kill `slattach`. Keep in mind you must be `root` to do the above. Then go back to `kermit` (by running `fg` if you suspended it) and exit from it (**q**).

The slattach(8) manual page says you have to use `ifconfig sl0 down` to mark the interface down, but this does not seem to make any difference. (`ifconfig sl0` reports the same thing.)

Some times, your modem might refuse to drop the carrier. In that case, simply start `kermit` and quit it again. It usually goes out on the second try.

28.7.1.4 Troubleshooting

If it does not work, feel free to ask on freebsd-net (http://lists.FreeBSD.org/mailman/listinfo/freebsd-net) mailing list. The things that people tripped over so far:

- Not using `-c` or `-a` in `slattach` (This should not be fatal, but some users have reported that this solves their problems.)
- Using `s10` instead of `sl0` (might be hard to see the difference on some fonts).
- Try `ifconfig sl0` to see your interface status. For example, you might get:

```
# ifconfig sl0
sl0: flags=10<POINTOPOINT>
        inet 136.152.64.181 --> 136.152.64.1 netmask ffffff00
```

- If you get `no route to host` messages from ping(8), there may be a problem with your routing table. You can use the `netstat -r` command to display the current routes :

```
# netstat -r
Routing tables
Destination      Gateway            Flags     Refs     Use  IfaceMTU    Rtt    Netmasks:

(root node)
(root node)

Route Tree for Protocol Family inet:
(root node) =>
default          inr-3.Example.EDU  UG          8   224515  sl0 -      -
localhost.Exampl localhost.Example. UH          5    42127  lo0 -       0.438
inr-3.Example.ED water.CS.Example.E UH          1        0  sl0 -      -
water.CS.Example localhost.Example. UGH        34 47641234  lo0 -       0.438
(root node)
```

  The preceding examples are from a relatively busy system. The numbers on your system will vary depending on network activity.

28.7.2 Setting Up a SLIP Server

This document provides suggestions for setting up SLIP Server services on a FreeBSD system, which typically means configuring your system to automatically start up connections upon login for remote SLIP clients.

28.7.2.1 Prerequisites

This section is very technical in nature, so background knowledge is required. It is assumed that you are familiar with the TCP/IP network protocol, and in particular, network and node addressing, network address masks, subnetting, routing, and routing protocols, such as RIP. Configuring SLIP services on a dial-up server requires a knowledge of these concepts, and if you are not familiar with them, please read a copy of either Craig Hunt's *TCP/IP Network Administration* published by O'Reilly & Associates, Inc. (ISBN Number 0-937175-82-X), or Douglas Comer's books on the TCP/IP protocol.

It is further assumed that you have already set up your modem(s) and configured the appropriate system files to allow logins through your modems. If you have not prepared your system for this yet, please see Section 27.4 for details on

dialup services configuration. You may also want to check the manual pages or sio(4) for information on the serial port device driver and ttys(5), gettytab(5), getty(8), & init(8) for information relevant to configuring the system to accept logins on modems, and perhaps stty(1) for information on setting serial port parameters (such as `clocal` for directly-connected serial interfaces).

28.7.2.2 Quick Overview

In its typical configuration, using FreeBSD as a SLIP server works as follows: a SLIP user dials up your FreeBSD SLIP Server system and logs in with a special SLIP login ID that uses `/usr/sbin/sliplogin` as the special user's shell. The `sliplogin` program browses the file `/etc/sliphome/slip.hosts` to find a matching line for the special user, and if it finds a match, connects the serial line to an available SLIP interface and then runs the shell script `/etc/sliphome/slip.login` to configure the SLIP interface.

28.7.2.2.1 An Example of a SLIP Server Login

For example, if a SLIP user ID were `Shelmerg`, `Shelmerg`'s entry in `/etc/master.passwd` would look something like this:

```
Shelmerg:password:1964:89::0:0:Guy Helmer - SLIP:/usr/users/Shelmerg:/usr/sbin/sliplogin
```

When `Shelmerg` logs in, `sliplogin` will search `/etc/sliphome/slip.hosts` for a line that had a matching user ID; for example, there may be a line in `/etc/sliphome/slip.hosts` that reads:

```
Shelmerg        dc-slip sl-helmer       0xfffffc00              autocomp
```

`sliplogin` will find that matching line, hook the serial line into the next available SLIP interface, and then execute `/etc/sliphome/slip.login` like this:

```
/etc/sliphome/slip.login 0 19200 Shelmerg dc-slip sl-helmer 0xfffffc00 autocomp
```

If all goes well, `/etc/sliphome/slip.login` will issue an `ifconfig` for the SLIP interface to which `sliplogin` attached itself (SLIP interface 0, in the above example, which was the first parameter in the list given to `slip.login`) to set the local IP address (`dc-slip`), remote IP address (`sl-helmer`), network mask for the SLIP interface (`0xfffffc00`), and any additional flags (`autocomp`). If something goes wrong, `sliplogin` usually logs good informational messages via the **syslogd** daemon facility, which usually logs to `/var/log/messages` (see the manual pages for syslogd(8) and syslog.conf(5) and perhaps check `/etc/syslog.conf` to see to what **syslogd** is logging and where it is logging to).

28.7.2.3 Kernel Configuration

FreeBSD's default kernel (`GENERIC`) comes with SLIP (sl(4)) support; in case of a custom kernel, you have to add the following line to your kernel configuration file:

```
device   sl
```

By default, your FreeBSD machine will not forward packets. If you want your FreeBSD SLIP Server to act as a router, you will have to edit the `/etc/rc.conf` file and change the setting of the `gateway_enable` variable to `YES`. This will make sure that setting the routing option will be persistent after a reboot.

To apply the settings immediately you can execute the following command as `root`:

```
# service routing start
```

Please refer to Chapter 9 on Configuring the FreeBSD Kernel for help in reconfiguring your kernel.

28.7.2.4 Sliplogin Configuration

As mentioned earlier, there are three files in the `/etc/sliphome` directory that are part of the configuration for `/usr/sbin/sliplogin` (see sliplogin(8) for the actual manual page for `sliplogin`): `slip.hosts`, which defines the SLIP users and their associated IP addresses; `slip.login`, which usually just configures the SLIP interface; and (optionally) `slip.logout`, which undoes `slip.login`'s effects when the serial connection is terminated.

28.7.2.4.1 `slip.hosts` Configuration

`/etc/sliphome/slip.hosts` contains lines which have at least four items separated by whitespace:

- SLIP user's login ID
- Local address (local to the SLIP server) of the SLIP link
- Remote address of the SLIP link
- Network mask

The local and remote addresses may be host names (resolved to IP addresses by `/etc/hosts` or by the domain name service, depending on your specifications in the file `/etc/nsswitch.conf`), and the network mask may be a name that can be resolved by a lookup into `/etc/networks`. On a sample system, `/etc/sliphome/slip.hosts` looks like this:

```
#
# login local-addr      remote-addr     mask            opt1    opt2
#                                               (normal,compress,noicmp)
#
Shelmerg  dc-slip       sl-helmerg      0xfffffc00      autocomp
```

At the end of the line is one or more of the options:

- `normal` — no header compression
- `compress` — compress headers
- `autocomp` — compress headers if the remote end allows it
- `noicmp` — disable ICMP packets (so any "ping" packets will be dropped instead of using up your bandwidth)

Your choice of local and remote addresses for your SLIP links depends on whether you are going to dedicate a TCP/IP subnet or if you are going to use "proxy ARP" on your SLIP server (it is not "true" proxy ARP, but that is the terminology used in this section to describe it). If you are not sure which method to select or how to assign IP addresses, please refer to the TCP/IP books referenced in the SLIP Prerequisites (Section 28.7.2.1) and/or consult your IP network manager.

If you are going to use a separate subnet for your SLIP clients, you will need to allocate the subnet number out of your assigned IP network number and assign each of your SLIP client's IP numbers out of that subnet. Then, you will probably need to configure a static route to the SLIP subnet via your SLIP server on your nearest IP router.

Otherwise, if you will use the "proxy ARP" method, you will need to assign your SLIP client's IP addresses out of your SLIP server's Ethernet subnet, and you will also need to adjust your `/etc/sliphome/slip.login` and `/etc/sliphome/slip.logout` scripts to use arp(8) to manage the "proxy ARP" entries in the SLIP server's ARP table.

28.7.2.4.2 `slip.login` Configuration

The typical `/etc/sliphome/slip.login` file looks like this:

```
#!/bin/sh -
#
#       @(#)slip.login  5.1 (Berkeley) 7/1/90

#
# generic login file for a slip line.  sliplogin invokes this with
# the parameters:
#      1        2         3        4          5         6     7-n
#   slipunit ttyspeed loginname local-addr remote-addr mask opt-args
#
/sbin/ifconfig sl$1 inet $4 $5 netmask $6
```

This `slip.login` file merely runs `ifconfig` for the appropriate SLIP interface with the local and remote addresses and network mask of the SLIP interface.

If you have decided to use the "proxy ARP" method (instead of using a separate subnet for your SLIP clients), your `/etc/sliphome/slip.login` file will need to look something like this:

```
#!/bin/sh -
#
#       @(#)slip.login  5.1 (Berkeley) 7/1/90

#
# generic login file for a slip line.  sliplogin invokes this with
# the parameters:
#      1        2         3        4          5         6     7-n
#   slipunit ttyspeed loginname local-addr remote-addr mask opt-args
#
/sbin/ifconfig sl$1 inet $4 $5 netmask $6
# Answer ARP requests for the SLIP client with our Ethernet addr
/usr/sbin/arp -s $5 00:11:22:33:44:55 pub
```

The additional line in this `slip.login`, `arp -s $5 00:11:22:33:44:55 pub`, creates an ARP entry in the SLIP server's ARP table. This ARP entry causes the SLIP server to respond with the SLIP server's Ethernet MAC address whenever another IP node on the Ethernet asks to speak to the SLIP client's IP address.

When using the example above, be sure to replace the Ethernet MAC address (`00:11:22:33:44:55`) with the MAC address of your system's Ethernet card, or your "proxy ARP" will definitely not work! You can discover your SLIP server's Ethernet MAC address by looking at the results of running `netstat -i`; the second line of the output should look something like:

```
ed0   1500  <Link>0.2.c1.28.5f.4a         191923        0   129457     0   116
```

This indicates that this particular system's Ethernet MAC address is `00:02:c1:28:5f:4a` — the periods in the Ethernet MAC address given by `netstat -i` must be changed to colons and leading zeros should be added to each single-digit hexadecimal number to convert the address into the form that arp(8) desires; see the manual page on arp(8) for complete information on usage.

> **Note:** When you create `/etc/sliphome/slip.login` and `/etc/sliphome/slip.logout`, the "execute" bit (i.e., `chmod 755 /etc/sliphome/slip.login /etc/sliphome/slip.logout`) must be set, or `sliplogin` will be unable to execute it.

28.7.2.4.3 `slip.logout` Configuration

`/etc/sliphome/slip.logout` is not strictly needed (unless you are implementing "proxy ARP"), but if you decide to create it, this is an example of a basic `slip.logout` script:

```
#!/bin/sh -
#
#       slip.logout

#
# logout file for a slip line.  sliplogin invokes this with
# the parameters:
#      1        2         3        4          5         6     7-n
#   slipunit ttyspeed loginname local-addr remote-addr mask opt-args
#
/sbin/ifconfig sl$1 down
```

If you are using "proxy ARP", you will want to have `/etc/sliphome/slip.logout` remove the ARP entry for the SLIP client:

```
#!/bin/sh -
#
#       @(#)slip.logout

#
# logout file for a slip line.  sliplogin invokes this with
# the parameters:
#      1        2         3        4          5         6     7-n
#   slipunit ttyspeed loginname local-addr remote-addr mask opt-args
#
/sbin/ifconfig sl$1 down
# Quit answering ARP requests for the SLIP client
/usr/sbin/arp -d $5
```

The `arp -d $5` removes the ARP entry that the "proxy ARP" `slip.login` added when the SLIP client logged in.

It bears repeating: make sure `/etc/sliphome/slip.logout` has the execute bit set after you create it (i.e., `chmod 755 /etc/sliphome/slip.logout`).

28.7.2.5 Routing Considerations

If you are not using the “proxy ARP” method for routing packets between your SLIP clients and the rest of your network (and perhaps the Internet), you will probably have to add static routes to your closest default router(s) to route your SLIP clients subnet via your SLIP server.

28.7.2.5.1 Static Routes

Adding static routes to your nearest default routers can be troublesome (or impossible if you do not have authority to do so...). If you have a multiple-router network in your organization, some routers, such as those made by Cisco and Proteon, may not only need to be configured with the static route to the SLIP subnet, but also need to be told which static routes to tell other routers about, so some expertise and troubleshooting/tweaking may be necessary to get static-route-based routing to work.

Chapter 29 Electronic Mail

Original work by Bill Lloyd. Rewritten by Jim Mock.

29.1 Synopsis

"Electronic Mail", better known as email, is one of the most widely used forms of communication today. This chapter provides a basic introduction to running a mail server on FreeBSD, as well as an introduction to sending and receiving email using FreeBSD. For more complete coverage of this subject, refer to the books listed in Appendix B.

After reading this chapter, you will know:

- Which software components are involved in sending and receiving electronic mail.
- Where basic **sendmail** configuration files are located in FreeBSD.
- The difference between remote and local mailboxes.
- How to block spammers from illegally using a mail server as a relay.
- How to install and configure an alternate Mail Transfer Agent, replacing **sendmail**.
- How to troubleshoot common mail server problems.
- How to set up the system to send mail only.
- How to use mail with a dialup connection.
- How to configure SMTP authentication for added security.
- How to install and use a Mail User Agent, such as **mutt**, to send and receive email.
- How to download mail from a remote POP or IMAP server.
- How to automatically apply filters and rules to incoming email.

Before reading this chapter, you should:

- Properly set up a network connection (Chapter 32).
- Properly set up the DNS information for a mail host (Chapter 30).
- Know how to install additional third-party software (Chapter 5).

29.2 Using Electronic Mail

There are five major parts involved in an email exchange: the Mail User Agent MUA>, the Mail Transfer AgentMTA, DNS, a remote or local mailbox, and the mail host.

29.2.1 The Mail User Agent

This includes command line programs such as **mutt**, **alpine**, **elm**, and `mail`, GUI programs such as **balsa** or **xfmail**, and web mail programs which can be accessed from a web browser. User programs pass the email transactions to the local "mail host", either by a MTA, or by delivering it over TCP.

29.2.2 The Mail Transfer Agent

FreeBSD ships with **Sendmail** as the default MTA, but it also supports numerous other mail server daemons, including:

- **Exim**;
- **Postfix**;
- **qmail**.

The MTA usually has two functions. It is responsible for receiving incoming mail as well as delivering outgoing mail. It is *not* responsible for the collection of mail using protocols such as POP or IMAP, nor does it allow connecting to local `mbox` or Maildir mailboxes. An additional daemon may be required for these functions.

> **Warning:** Older versions of **Sendmail** contain serious security issues which may result in an attacker gaining local or remote access to the system. Run a current version to FreeBSD to avoid these problems. Optionally, install an alternative MTA from the FreeBSD Ports Collection.

29.2.3 Email and DNS

The Domain Name System (DNS) and its daemon `named` play a large role in the delivery of email. In order to deliver mail from one site to another, the MTA will look up the remote site in DNS to determine which host will receive mail for the destination. This process also occurs when mail is sent from a remote host to the MTA.

DNS is responsible for mapping hostnames to IP addresses, as well as for storing information specific to mail delivery, known as Mail eXchanger MX records. The MX record specifies which host, or hosts, will receive mail for a particular domain. If there is no MX record for the hostname or domain, the mail will be delivered directly to the host, provided there is an `A` record pointing the hostname to the IP address.

To view the MX records for a domain, specify the type of record using host(1), as seen in the example below:

```
% host -t mx FreeBSD.org
FreeBSD.org mail is handled by 10 mx1.FreeBSD.org
```

29.2.4 Receiving Mail

Receiving mail for a domain is done by the mail host. It will collect all mail sent to the domain and store it either in the default `mbox` or the alternative Maildir format, depending on the configuration. Once mail has been stored, it may either be read locally using a MUA, or remotely accessed and collected using protocols such as POP or IMAP. In order to read mail locally, a POP or IMAP server does not need to be installed.

29.2.4.1 Accessing Remote Mailboxes Using POP and IMAP

To access mailboxes remotely, access to a POP or IMAP server is required. These protocols allow users to connect to their mailboxes from remote locations. Though both POP and IMAP allow users to remotely access mailboxes, IMAP offers many advantages, including:

- IMAP can store messages on a remote server as well as fetch them.

- IMAP supports concurrent updates.
- IMAP can be useful over low-speed links as it allows users to fetch the structure of messages without downloading them. It can also perform tasks such as searching on the server in order to minimize data transfer between clients and servers.

In order to install a POP or IMAP server, the following steps should be performed:

1. Use the Ports Collection to install an IMAP or POP server. The following POP and IMAP servers are well known:
   - `mail/qpopper`
   - `mail/teapop`
   - `mail/imap-uw`
   - `mail/courier-imap`
   - `mail/dovecot2`
2. Where required, use the startup script that came with the application to load the POP or IMAP server. Those programs will also provide a variable which can be added to `/etc/rc.conf` to automate the startup of the application's daemon whenever the system boots.

> **Warning:** It should be noted that both POP and IMAP transmit information, including username and password credentials, in clear-text. To secure the transmission of information across these protocols, consider tunneling sessions over ssh(1) (Section 15.10.8) or using SSL (Section 15.8).

29.2.4.2 Accessing Local Mailboxes

Mailboxes may be accessed locally by directly using an MUA on the server on which the mailbox resides. This can be done using a built-in application such as mail(1) or by installing a MUA from the Ports Collection..

29.2.5 The Mail Host

The mail host is a server that is responsible for delivering and receiving mail for a host, or a network.

29.3 Sendmail Configuration

Contributed by Christopher Shumway.

sendmail(8) is the default MTA which is installed with FreeBSD. **Sendmail** accepts mail from MUAs and delivers it to the appropriate mailer as defined by its configuration file. **Sendmail** can also accept network connections and deliver mail to local mailboxes or to another program.

Sendmail uses the following configuration files. This section describes these files in more detail.

Filename	Function
/etc/mail/access	**Sendmail** access database file.
/etc/mail/aliases	Mailbox aliases
/etc/mail/local-host-names	Lists of hosts **Sendmail** accepts mail for.
/etc/mail/mailer.conf	Mailer program configuration.
/etc/mail/mailertable	Mailer delivery table.
/etc/mail/sendmail.cf	**Sendmail** master configuration file.
/etc/mail/virtusertable	Virtual users and domain tables.

29.3.1 /etc/mail/access

This database defines which host(s) or IP addresses have access to the local mail server and what kind of access they have. Hosts can be listed as OK, REJECT, or RELAY, or can be passed to **Sendmail**'s error handling routine with a given mailer error. Hosts that are listed as OK, which is the default option, are allowed to send mail to this host as long as the mail's final destination is the local machine. Hosts that are listed as REJECT are rejected for all mail connections. Hosts that are listed as RELAY are allowed to send mail for any destination using this mail server.

Example 29-1. Configuring the Sendmail Access Database

```
cyberspammer.com    550 We do not accept mail from spammers
FREE.STEALTH.MAILER@            550 We do not accept mail from spammers
another.source.of.spam          REJECT
okay.cyberspammer.com           OK
128.32                          RELAY
```

This example shows five entries. Mail senders that match the left side of the table are affected by the action on the right side of the table. The first two examples give an error code to **Sendmail**'s error handling routine. The message is sent to the remote host when a mail matches the left side of the table. The third entry rejects mail from a specific host on the Internet, another.source.of.spam. The fourth entry accepts mail connections from okay.cyberspammer.com, which is more specific than the cyberspammer.com line above. More specific matches override less exact matches. The last entry allows relaying of email from hosts with an IP address that begins with 128.32. These hosts can send mail through this mail server that is destined for other mail servers.

Whenever this file is updated, run make in /etc/mail/ to update the database.

29.3.2 /etc/mail/aliases

This database contains a list of virtual mailboxes that are expanded to other user(s), files, programs, or other aliases. Here are a few examples to illustrate the file format:

Example 29-2. Mail Aliases

```
root: localuser
ftp-bugs: joe,eric,paul
bit.bucket:  /dev/null
procmail: "|/usr/local/bin/procmail"
```

The mailbox name on the left side of the colon is expanded to the target(s) on the right. The first entry expands the mailbox `root` to the mailbox `localuser`, which is then looked up again in the `aliases` database. If no match is found, the message is delivered to `localuser`. The second entry shows a mail list. Mail to the mailbox `ftp-bugs` is expanded to the three local mailboxes `joe`, `eric`, and `paul`. A remote mailbox could be specified as <`user@example.com`>. The third entry shows how to write mail to a file, in this case `/dev/null`. The last entry demonstrates how to send mail to a program, `/usr/local/bin/procmail`, through a UNIX pipe.

Whenever this file is updated, run `make` in `/etc/mail/` to update the database.

29.3.3 `/etc/mail/local-host-names`

This is a list of hostnames sendmail(8) is to accept as the local host name. Place any domains or hosts that **Sendmail** will receive mail for. For example, to configure a mail server to accept mail for the domain `example.com` and the host `mail.example.com`, add these entries to `local-host-names`:

```
example.com
mail.example.com
```

Whenever this file is updated, sendmail(8) needs to be restarted so that it will read the changes.

29.3.4 `/etc/mail/sendmail.cf`

This is the master configuration file for **Sendmail**. It controls the overall behavior of **Sendmail**, including everything from rewriting email addresses to printing rejection messages to remote mail servers. Accordingly, this configuration file is quite complex. Fortunately, this file rarely needs to be changed for standard mail servers.

The master **Sendmail** configuration file can be built from m4(1) macros that define the features and behavior of **Sendmail**. Refer to `/usr/src/contrib/sendmail/cf/README` for some of the details.

Whenever changes to this file are made, **Sendmail** needs to be restarted for the changes to take effect.

29.3.5 `/etc/mail/virtusertable`

The `virtusertable` maps mail addresses for virtual domains and mailboxes to real mailboxes. These mailboxes can be local, remote, aliases defined in `/etc/mail/aliases`, or files.

Example 29-3. Example Virtual Domain Mail Map

```
root@example.com                root
postmaster@example.com          postmaster@noc.example.net
@example.com                    joe
```

The above example contains a mapping for the domain `example.com`. This file is processed in a first match order. The first item maps <`root@example.com`> to the local mailbox `root`. The second entry maps <`postmaster@example.com`> to the mailbox `postmaster` on the host `noc.example.net`. Finally, if nothing from `example.com` has matched so far, it will match the last mapping, which matches every other mail message addressed to someone at `example.com` to the local mailbox `joe`.

29.4 Changing the Mail Transfer Agent

Written by Andrew Boothman. Information taken from emails written byGregory Neil Shapiro.

FreeBSD comes with **Sendmail** already installed as the MTA which is in charge of outgoing and incoming mail.

However, the system administrator can change the system's MTA. The reasons for doing so range from wanting to try out another MTA to needing a specific feature or package which relies on another MTA. Whatever the reason, FreeBSD makes it easy to make the change.

29.4.1 Install a New MTA

A wide choice of MTAs is available from the `mail` category of the FreeBSD Ports Collection.

Once a new MTA is installed, configure the new software and decide if it really fulfills your needs before replacing **Sendmail**.

Refer to the new chosen MTA's documentation for information on how to configure the software.

29.4.2 Disable Sendmail

> **Warning:** If **Sendmail**'s outgoing mail service is disabled, it is important that it is replaced with an alternative mail delivery system. Otherwise, system functions such as periodic(8) will be unable to deliver their results by email. Many parts of the system expect a functional MTA. If applications continue to use **Sendmail**'s binaries to try to send email they are disabled, mail could go into an inactive **Sendmail** queue, and never be delivered.

In order to completely disable **Sendmail**, including the outgoing mail service, add or edit the following lines in `/etc/rc.conf`:

```
sendmail_enable="NO"
sendmail_submit_enable="NO"
sendmail_outbound_enable="NO"
sendmail_msp_queue_enable="NO"
```

To only disable **Sendmail**'s incoming mail service, set

```
sendmail_enable="NO"
```

in `/etc/rc.conf`. More information on **Sendmail**'s startup options is available in rc.sendmail(8).

29.4.3 Running the New MTA on Boot

The new MTA can be started during boot by adding a configuration line to `/etc/rc.conf`. This example enables the Postfix MTA:

```
# echo
'postfix_enable="YES"'
>> /etc/rc.conf
```

The specified MTA will now be automatically started during boot.

29.4.4 Replacing Sendmail as the System's Default Mailer

Sendmail is so ubiquitous as standard software on UNIX systems that some software assumes it is already installed and configured. For this reason, many alternative MTAs provide their own compatible implementations of the **Sendmail** command-line interface in order to facilitate using them as "drop-in" replacements for **Sendmail**.

When using an alternative MTA, make sure that software trying to execute standard **Sendmail** binaries, such as `/usr/bin/sendmail`, actually execute the chosen mailer instead. Fortunately, FreeBSD provides a system called mailwrapper(8) for this purpose.

When **Sendmail** is operating as installed, `/etc/mail/mailer.conf` will look like this:

```
sendmail        /usr/libexec/sendmail/sendmail
send-mail       /usr/libexec/sendmail/sendmail
mailq           /usr/libexec/sendmail/sendmail
newaliases      /usr/libexec/sendmail/sendmail
hoststat        /usr/libexec/sendmail/sendmail
purgestat       /usr/libexec/sendmail/sendmail
```

When any of the commands listed on the left are run, the system actually executes the associated command shown on the right instead. This system makes it easy to change what binaries are executed when these default `Sendmail` functions are invoked.

As an example, to run `/usr/local/supermailer/bin/sendmail-compat` instead of **Sendmail**, specify the paths to the installed applications in `/etc/mail/mailer.conf`:

```
sendmail /usr/local/supermailer/bin/sendmail-compat
send-mail       /usr/local/supermailer/bin/sendmail-compat
mailq           /usr/local/supermailer/bin/mailq-compat
newaliases      /usr/local/supermailer/bin/newaliases-compat
hoststat        /usr/local/supermailer/bin/hoststat-compat
purgestat       /usr/local/supermailer/bin/purgestat-compat
```

29.4.5 Finishing

Once everything is configured, either kill the unneeded **sendmail** processes and start the processes belonging to the new software, or reboot. Rebooting provides the opportunity to ensure that the system is correctly configured to start the new MTA automatically on boot.

29.5 Troubleshooting

1. Why do I have to use the FQDN for hosts on my site?

The host may actually be in a different domain. For example, in order for a host in `foo.bar.edu` to reach a host called `mumble` in the `bar.edu` domain, refer to it by the Fully-Qualified Domain Name FQDN, `mumble.bar.edu`, instead of just `mumble`.

This is because the version of **BIND** which ships with FreeBSD no longer provides default abbreviations for non-FQDNs other than the local domain. An unqualified host such as `mumble` must either be found as `mumble.foo.bar.edu`, or it will be searched for in the root domain.

In older versions of **BIND**, the search continued across `mumble.bar.edu`, and `mumble.edu`. RFC 1535 details why this is considered bad practice or even a security hole.

As a good workaround, place the line:

```
search foo.bar.edu bar.edu
```

instead of the previous:

```
domain foo.bar.edu
```

into `/etc/resolv.conf`. However, make sure that the search order does not go beyond the "boundary between local and public administration", as RFC 1535 calls it.

2. Sendmail says `mail loops back to myself`.

This is answered in the Sendmail FAQ (http://www.sendmail.org/faq/) as follows. This FAQ is recommended reading when "tweaking" the mail setup.

```
I'm getting these error messages:

553 MX list for domain.net points back to relay.domain.net
554 <user@domain.net>... Local configuration error

How can I solve this problem?

You have asked mail to the domain (e.g., domain.net) to be
forwarded to a specific host (in this case, relay.domain.net)
by using an MX record, but the relay machine does not recognize
itself as domain.net. Add domain.net to /etc/mail/local-host-names
[known as /etc/sendmail.cw prior to version 8.10]
(if you are using FEATURE(use_cw_file)) or add "Cw domain.net"
to /etc/mail/sendmail.cf.
```

3. How can I run a mail server on a dial-up PPP host?

Connect to a FreeBSD mail gateway on the LAN. The PPP connection is non-dedicated.

One way to do this is to get a full-time Internet server to provide secondary MX services for the domain. In this example, the domain is `example.com` and the ISP has configured `example.net` to provide secondary MX services to the domain:

```
example.com.          MX        10      example.com.
                      MX        20      example.net.
```

Only one host should be specified as the final recipient. For **Sendmail**, add `Cw example.com` in `/etc/mail/sendmail.cf` on `example.com`.

When the sending MTA attempts to deliver mail, it will try to connect to the system, `example.com`, over the PPP link. This will time out if the destination is offline. The MTA will automatically deliver it to the secondary MX site at the Internet Service Provider (ISP), `example.net`. The secondary MX site will periodically try to connect to the primary MX host, `example.com`.

Use something like this as a login script:

```
#!/bin/sh
# Put me in /usr/local/bin/pppmyisp
( sleep 60 ; /usr/sbin/sendmail -q ) &
/usr/sbin/ppp -direct pppmyisp
```

When creating a separate login script for users, instead use `sendmail -qRexample.com` in the script above. This will force all mail in the queue for `example.com` to be processed immediately.

A further refinement of the situation can be seen from this example from the FreeBSD Internet service provider's mailing list (http://lists.FreeBSD.org/mailman/listinfo/freebsd-isp):

```
> we provide the secondary MX for a customer. The customer connects to
> our services several times a day automatically to get the mails to
> his primary MX (We do not call his site when a mail for his domains
> arrived). Our sendmail sends the mailqueue every 30 minutes. At the
> moment he has to stay 30 minutes online to be sure that all mail is
> gone to the primary MX.
>
> Is there a command that would initiate sendmail to send all the mails
> now? The user has not root-privileges on our machine of course.

In the “privacy flags” section of sendmail.cf, there is a
definition Opgoaway,restrictqrun

Remove restrictqrun to allow non-root users to start the queue processing.
You might also like to rearrange the MXs. We are the 1st MX for our
customers like this, and we have defined:

# If we are the best MX for a host, try directly instead of generating
# local config error.
OwTrue

That way a remote site will deliver straight to you, without trying
the customer connection.  You then send to your customer.  Only works for
“hosts”, so you need to get your customer to name their mail
machine “customer.com” as well as
“hostname.customer.com” in the DNS.  Just put an A record in
the DNS for “customer.com”.
```

4. Why do I keep getting `Relaying Denied` errors when sending mail from other hosts?

In a default FreeBSD installation, **Sendmail** is configured to only send mail from the host it is running on. For example, if a POP server is available, users will be able to check mail from remote locations but they will not be able to send outgoing emails from outside locations. Typically, a few moments after the attempt, an email will be sent from `MAILER-DAEMON` with a `5.7 Relaying Denied`.

The most straightforward solution is to add the ISP's FQDN to `/etc/mail/relay-domains`, as seen in this example:

```
# echo "your.isp.example.com" > /etc/mail/relay-domains
```

After creating or editing this file, restart **Sendmail**. This works great if the server administrator does not wish to send mail locally, would like to use a MUA on a remote machine, or would like to use another ISP for remote connections. It is also useful when there is only one or two email accounts. If there are a large number of addresses, add them one per line:

```
your.isp.example.com
other.isp.example.net
users-isp.example.org
www.example.org
```

Now any mail sent through the system by any host in this list, provided the user has an account on the system, will succeed. This allows users to send mail from the system remotely without opening the system up to relaying SPAM from the Internet.

29.6 Advanced Topics

This section covers more involved topics such as mail configuration and setting up mail for an entire domain.

29.6.1 Basic Configuration

Out of the box, one can send email to external hosts as long as `/etc/resolv.conf` is configured or the network has access to a configured DNS server. If order to have mail delivered to the MTA on the FreeBSD host, do one of the following:

- Run a DNS server for the domain.
- Get mail delivered directly to to the FQDN for the machine.

In order to have mail delivered directly to a host, it must have a permanent static IP address, not a dynamic IP address. If the system is behind a firewall, it must be configured to allow SMTP traffic. To receive mail directly at a host, one of these two must be configured:

- Make sure that the lowest-numbered MX record in DNS points to the host's static IP address.
- Make sure there is no MX entry in the DNS for the host.

Either of the above will allow mail to be received directly at the host.

Try this:

```
# hostname
example.FreeBSD.org
# host example.FreeBSD.org
example.FreeBSD.org has address 204.216.27.XX
```

In this example, mail sent directly to <`yourlogin@example.FreeBSD.org`> should work without problems, assuming **Sendmail** is running correctly on `example.FreeBSD.org`.

For this example:

```
# host example.FreeBSD.org
example.FreeBSD.org has address 204.216.27.XX
example.FreeBSD.org mail is handled (pri=10) by hub.FreeBSD.org
```

All mail sent to `example.FreeBSD.org` will be collected on `hub` under the same username instead of being sent directly to your host.

The above information is handled by the DNS server. The DNS record that carries mail routing information is the MX entry. If no MX record exists, mail will be delivered directly to the host by way of its IP address.

The MX entry for `freefall.FreeBSD.org` at one time looked like this:

```
freefall                MX      30      mail.crl.net
freefall                MX      40      agora.rdrop.com
freefall                MX      10      freefall.FreeBSD.org
freefall                MX      20      who.cdrom.com
```

`freefall` had many MX entries. The lowest MX number is the host that receives mail directly, if available. If it is not accessible for some reason, the next lower-numbered host will accept messages temporarily, and pass it along when a lower-numbered host becomes available.

Alternate MX sites should have separate Internet connections in order to be most useful. Your ISP can provide this service.

29.6.2 Mail for a Domain

When configuring a MTA for a network, any mail sent to hosts in its domain should be diverted to the MTA so that users can receive their mail on the master mail server.

To make life easiest, a user account with the same *username* should exist on both the MTA and the system with the MUA. Use adduser(8) to create the user accounts.

The MTA must be the designated mail exchanger for each workstation on the network. This is done in theDNS configuration with an MX record:

```
example.FreeBSD.org     A       204.216.27.XX           ; Workstation
                        MX      10 hub.FreeBSD.org      ; Mailhost
```

This will redirect mail for the workstation to the MTA no matter where the A record points. The mail is sent to the MX host.

This must be configured on a DNS server. If the network does not run its own DNS server, talk to the ISP or DNS provider.

The following is an example of virtual email hosting. Consider a customer with the domain `customer1.org`, where all the mail for `customer1.org` should be sent to `mail.myhost.com`. The DNS entry should look like this:

```
customer1.org           MX      10      mail.myhost.com
```

An `A`> record is *not* needed for `customer1.org` in order to only handle email for that domain. However, running `ping` against `customer1.org` will not work unless an `A` record exists for it.

Tell the MTA which domains and/or hostnames it should accept mail for. Either of the following will work for **Sendmail**:

- Add the hosts to `/etc/mail/local-host-names` when using the `FEATURE(use_cw_file)`. For versions of **Sendmail** earlier than 8.10, edit `/etc/sendmail.cw` instead.
- Add a `Cwyour.host.com` line to `/etc/sendmail.cf`. For **Sendmail** 8.10 or higher, add that line to `/etc/mail/sendmail.cf`.

29.7 Setting Up to Send Only

Contributed by Bill Moran.

There are many instances where one may only want to send mail through a relay. Some examples are:

- The computer is a desktop machine that needs to use programs such as send-pr(1), using the ISP's mail relay.
- The computer is a server that does not handle mail locally, but needs to pass off all mail to a relay for processing.

While any MTA is capable of filling this particular niche, it can be difficult to properly configure a full-featured MTA just to handle offloading mail. Programs such as **Sendmail** and **Postfix** are overkill for this use.

Additionally, a typical Internet access service agreement may forbid one from running a "mail server".

The easiest way to fulfill those needs is to install the `mail/ssmtp` port:

```
# cd /usr/ports/mail/ssmtp
# make install replace clean
```

Once installed, `mail/ssmtp` can be configured with `/usr/local/etc/ssmtp/ssmtp.conf`:

```
root=yourrealemail@example.com
mailhub=mail.example.com
rewriteDomain=example.com
hostname=_HOSTNAME_
```

Use the real email address for `root`. Enter the ISP's outgoing mail relay in place of `mail.example.com`. Some ISPs call this the "outgoing mail server" or "SMTP server").

Make sure to disable **Sendmail**, including the outgoing mail service. See Section 29.4.2 for details.

`mail/ssmtp` has some other options available. Refer to the examples in `/usr/local/etc/ssmtp` or the manual page of **ssmtp** for more information.

Setting up **ssmtp** in this manner allows any software on the computer that needs to send mail to function properly, while not violating the ISP's usage policy or allowing the computer to be hijacked for spamming.

29.8 Using Mail with a Dialup Connection

When using a static IP address, one should not need to adjust the default configuration. Set the hostname to the assigned Internet name and **Sendmail** will do the rest.

When using a dynamically assigned IP address and a dialup PPP connection to the Internet, one usually has a mailbox on the ISP's mail server. In this example, the ISP's domain is `example.net`, the user name is `user`, the hostname is `bsd.home`, and the ISP has allowed `relay.example.net` as a mail relay.

In order to retrieve mail from the ISP's mailbox, install a retrieval agent from the Ports Collection. `mail/fetchmail` is a good choice as it supports many different protocols. Usually, the ISP will provide POP. When using user PPP, email can be automatically fetched when an Internet connection is established with the following entry in `/etc/ppp/ppp.linkup`:

```
MYADDR:
!bg su user -c fetchmail
```

When using **Sendmail** to deliver mail to non-local accounts, configure **Sendmail** to process the mail queue as soon as the Internet connection is established. To do this, add this line after the above `fetchmail` entry in `/etc/ppp/ppp.linkup`:

```
  !bg su user -c "sendmail -q"
```

In this example, there is an account for `user` on `bsd.home`. In the home directory of `user` on `bsd.home`, create a `.fetchmailrc` which contains this line:

```
poll example.net protocol pop3 fetchall pass MySecret
```

This file should not be readable by anyone except `user` as it contains the password `MySecret`.

In order to send mail with the correct `from:` header, configure **Sendmail** to use <`user@example.net`> rather than <`user@bsd.home`> and to send all mail via `relay.example.net`, allowing quicker mail transmission.

The following `.mc` file should suffice:

```
VERSIONID('bsd.home.mc version 1.0')
OSTYPE(bsd4.4)dnl
FEATURE(nouucp)dnl
MAILER(local)dnl
MAILER(smtp)dnl
Cwlocalhost
Cwbsd.home
MASQUERADE_AS('example.net')dnl
FEATURE(allmasquerade)dnl
FEATURE(masquerade_envelope)dnl
FEATURE(nocanonify)dnl
FEATURE(nodns)dnl
define('SMART_HOST', 'relay.example.net')
Dmbsd.home
define('confDOMAIN_NAME','bsd.home')dnl
define('confDELIVERY_MODE','deferred')dnl
```

Refer to the previous section for details of how to convert this file into the `sendmail.cf` format. Do not forget to restart **Sendmail** after updating `sendmail.cf`.

29.9 SMTP Authentication

Written by James Gorham.

Configuring SMTP authentication on the MTA provides a number of benefits. SMTP authentication adds a layer of security to **Sendmail**, and provides mobile users who switch hosts the ability to use the same MTA without the need

to reconfigure their mail client's settings each time.

1. Install `security/cyrus-sasl2` from the Ports Collection. This port supports a number of compile-time options. For the SMTP authentication method demonstrated in this example, make sure that `LOGIN` is not disabled.
2. After installing `security/cyrus-sasl2`, edit `/usr/local/lib/sasl2/Sendmail.conf`, or create it if it does not exist, and add the following line:

   ```
   pwcheck_method: saslauthd
   ```
3. Next, install `security/cyrus-sasl2-saslauthd` and add the following line to `/etc/rc.conf`:

   ```
   saslauthd_enable="YES"
   ```

   Finally, start the saslauthd daemon:

   ```
   # service saslauthd start
   ```

   This daemon serves as a broker for **sendmail** to authenticate against the FreeBSD passwd(5) database. This saves the trouble of creating a new set of usernames and passwords for each user that needs to use SMTP authentication, and keeps the login and mail password the same.
4. Next, edit `/etc/make.conf` and add the following lines:

   ```
   SENDMAIL_CFLAGS=-I/usr/local/include/sasl -DSASL
   SENDMAIL_LDFLAGS=-L/usr/local/lib
   SENDMAIL_LDADD=-lsasl2
   ```

   These lines provide **Sendmail** the proper configuration options for linking to `cyrus-sasl2` at compile time. Make sure that `cyrus-sasl2` has been installed before recompiling **Sendmail**.
5. Recompile **Sendmail** by executing the following commands:

   ```
   # cd /usr/src/lib/libsmutil
   # make cleandir && make obj && make
   # cd /usr/src/lib/libsm
   # make cleandir && make obj && make
   # cd /usr/src/usr.sbin/sendmail
   # make cleandir && make obj && make && make install
   ```

   This compile should not have any problems if `/usr/src` has not changed extensively and the shared libraries it needs are available.
6. After **Sendmail** has been compiled and reinstalled, edit `/etc/mail/freebsd.mc` or the local `.mc` file. Many administrators choose to use the output from hostname(1) as the name of the `.mc` file for uniqueness. Add these lines:

   ```
   dnl set SASL options
   TRUST_AUTH_MECH('GSSAPI DIGEST-MD5 CRAM-MD5 LOGIN')dnl
   define('confAUTH_MECHANISMS', 'GSSAPI DIGEST-MD5 CRAM-MD5 LOGIN')dnl
   ```

   These options configure the different methods available to **Sendmail** for authenticating users. To use a method other than **pwcheck**, refer to the **Sendmail** documentation.
7. Finally, run make(1) while in `/etc/mail`. That will run the new `.mc` and create a `.cf` named either `freebsd.cf` or the name used for the local `.mc`. Then, run `make install restart`, which will copy the file to `sendmail.cf`, and properly restart **Sendmail**. For more information about this process, refer to `/etc/mail/Makefile`.

To test the configuration, use a MUA to send a test message. For further investigation, set the `LogLevel` of **Sendmail** to `13` and watch `/var/log/maillog` for any errors.

For more information, refer to SMTP authentication (http://www.sendmail.org/~ca/email/auth.html).

29.10 Mail User Agents

Contributed by Marc Silver.

A MUA is an application that is used to send and receive email. As email "evolves" and becomes more complex, MUAs are becoming increasingly powerful and provide users increased functionality and flexibility. The `mail` category of the FreeBSD Ports Collection contains numerous MUAs. These include graphical email clients such as **Evolution** or **Balsa** and console based clients such as **mutt** or **alpine**.

29.10.1 `mail`

mail(1) is the default MUA installed with FreeBSD. It is a console based MUA that offers the basic functionality required to send and receive text-based email. It provides limited attachment support and can only access local mailboxes.

Although `mail` does not natively support interaction with POP or IMAP servers, these mailboxes may be downloaded to a local `mbox` using an application such as **fetchmail**.

In order to send and receive email, run `mail`:

```
% mail
```

The contents of the user's mailbox in `/var/mail` are automatically read by `mail`. Should the mailbox be empty, the utility exits with a message indicating that no mail could be found. If mail exists, the application interface starts, and a list of messages will be displayed. Messages are automatically numbered, as can be seen in the following example:

```
Mail version 8.1 6/6/93.  Type ? for help.
"/var/mail/marcs": 3 messages 3 new
>N  1 root@localhost        Mon Mar  8 14:05  14/510   "test"
 N  2 root@localhost        Mon Mar  8 14:05  14/509   "user account"
 N  3 root@localhost        Mon Mar  8 14:05  14/509   "sample"
```

Messages can now be read by typing **t** followed by the message number. This example reads the first email:

```
& t 1
Message 1:
From root@localhost  Mon Mar  8 14:05:52 2004
X-Original-To: marcs@localhost
Delivered-To: marcs@localhost
To: marcs@localhost
Subject: test
Date: Mon,  8 Mar 2004 14:05:52 +0200 (SAST)
From: root@localhost (Charlie Root)

This is a test message, please reply if you receive it.
```

As seen in this example, the message will be displayed with full headers. To display the list of messages again, press **h**.

If the email requires a reply, press either **R** or **r** `mail` keys. **R** instructs `mail` to reply only to the sender of the email, while **r** replies to all other recipients of the message. These commands can be suffixed with the mail number of the message to reply to. After typing the response, the end of the message should be marked by a single **.** on its own line. An example can be seen below:

```
& R 1
To: root@localhost
Subject: Re: test

Thank you, I did get your email.
.
EOT
```

In order to send a new email, press **m**, followed by the recipient email address. Multiple recipients may be specified by separating each address with the **,** delimiter. The subject of the message may then be entered, followed by the message contents. The end of the message should be specified by putting a single **.** on its own line.

```
& mail root@localhost
Subject: I mastered mail

Now I can send and receive email using mail ... :)
.
EOT
```

While using `mail`, press **?** to display help at any time. Refer to mail(1) for more help on how to use `mail`.

> **Note:** mail(1) was not designed to handle attachments and thus deals with them poorly. Newer MUAs handle attachments in a more intelligent way. Users who prefer to use `mail` may find the `converters/mpack` port to be of considerable use.

29.10.2 mutt

mutt is a powerful MUA, with many features, including:

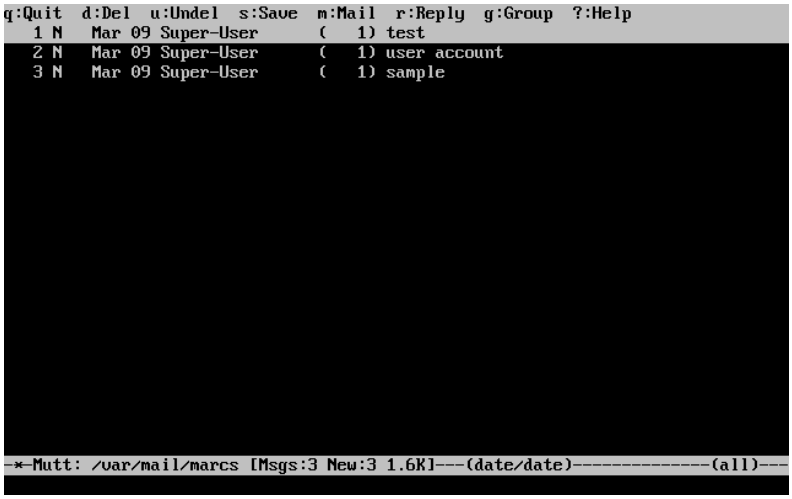
- The ability to thread messages.
- PGP support for digital signing and encryption of email.
- MIME support.
- Maildir support.
- Highly customizable.

Refer to http://www.mutt.org for more information on **mutt**.

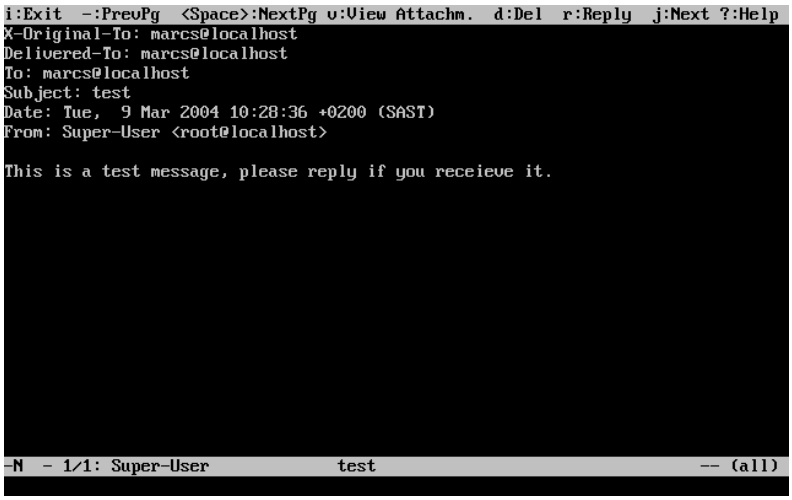
mutt may be installed using the `mail/mutt` port. After the port has been installed, **mutt** can be started by issuing the following command:

```
% mutt
```

mutt will automatically read and display the contents of the user mailbox in `/var/mail`. If no mails are found, **mutt** will wait for commands from the user. The example below shows **mutt** displaying a list of messages:



To read an email, select it using the cursor keys and press **Enter**. An example of **mutt** displaying email can be seen below:

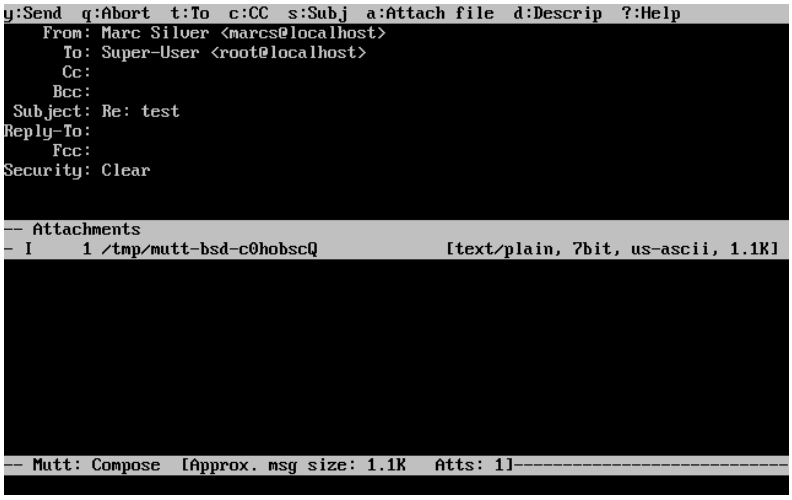


Similar to mail(1), **mutt** can be used to reply only to the sender of the message as well as to all recipients. To reply only to the sender of the email, press **r**. To send a group reply to the original sender as well as all the message recipients, press **g**.

> **Note:** By default, **mutt** uses the vi(1) editor for creating and replying to emails. Each user can customize this by creating or editing the `.muttrc` in their home directory and setting the `editor` variable or by setting the `EDITOR` environment variable. Refer to http://www.mutt.org/ for more information about configuring **mutt**.

To compose a new mail message, press **m**. After a valid subject has been given, **mutt** will start vi(1) so the email can be written. Once the contents of the email are complete, save and quit from `vi`. **mutt** will resume, displaying a

summary screen of the mail that is to be delivered. In order to send the mail, press **y**. An example of the summary screen can be seen below:



mutt contains extensive help which can be accessed from most of the menus by pressing **?**. The top line also displays the keyboard shortcuts where appropriate.

29.10.3 alpine

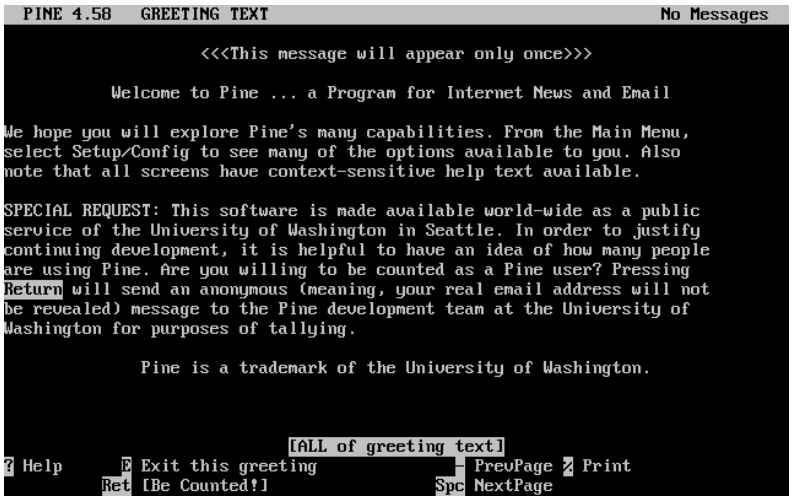
alpine is aimed at a beginner user, but also includes some advanced features.

> **Warning: alpine** has had several remote vulnerabilities discovered in the past, which allowed remote attackers to execute arbitrary code as users on the local system, by the action of sending a specially-prepared email. While *known* problems have been fixed, **alpine** code is written in an insecure style and the FreeBSD Security Officer believes there are likely to be other undiscovered vulnerabilities. Users install **alpine** at their own risk.

The current version of **alpine** may be installed using the `mail/alpine` port. Once the port has installed, **alpine** can be started by issuing the following command:

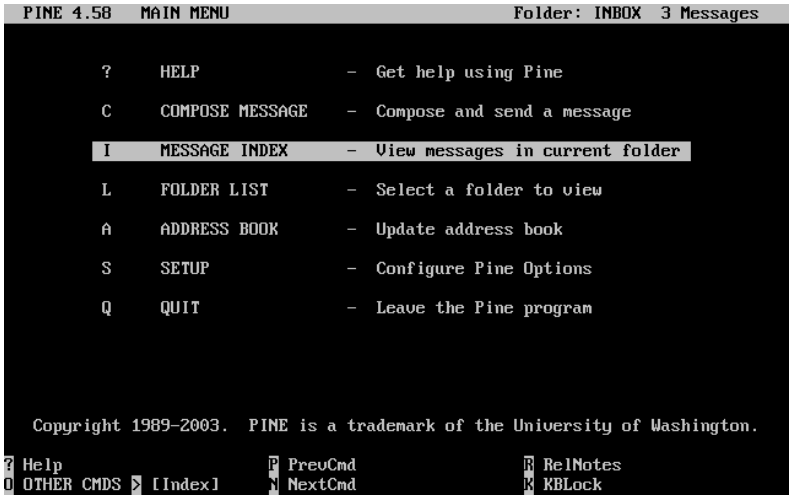
```
% alpine
```

The first time **alpine** runs, it displays a greeting page with a brief introduction, as well as a request from the **alpine** development team to send an anonymous email message allowing them to judge how many users are using their client. To send this anonymous message, press **Enter**. Alternatively, press **E** to exit the greeting without sending an anonymous message. An example of the greeting page is shown below:

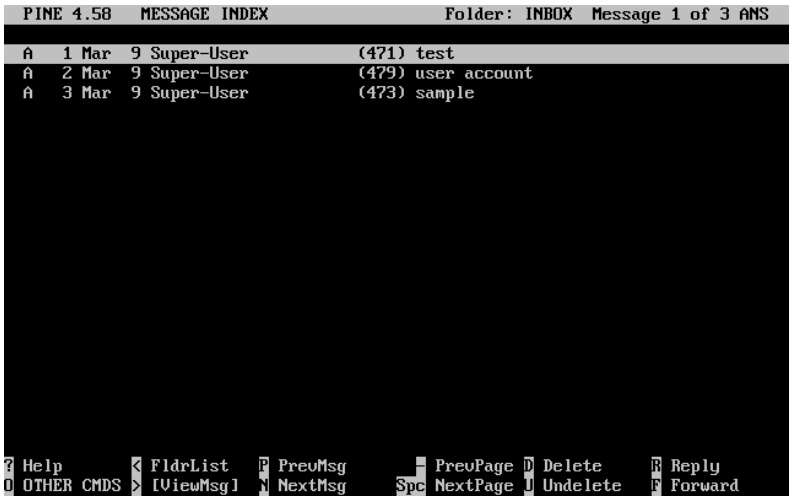


The main menu is then presented, which can be navigated using the cursor keys. This main menu provides shortcuts for the composing new mails, browsing mail directories, and administering address book entries. Below the main menu, relevant keyboard shortcuts to perform functions specific to the task at hand are shown.

The default directory opened by **alpine** is `inbox`. To view the message index, press **I**, or select the MESSAGE INDEX option shown below:

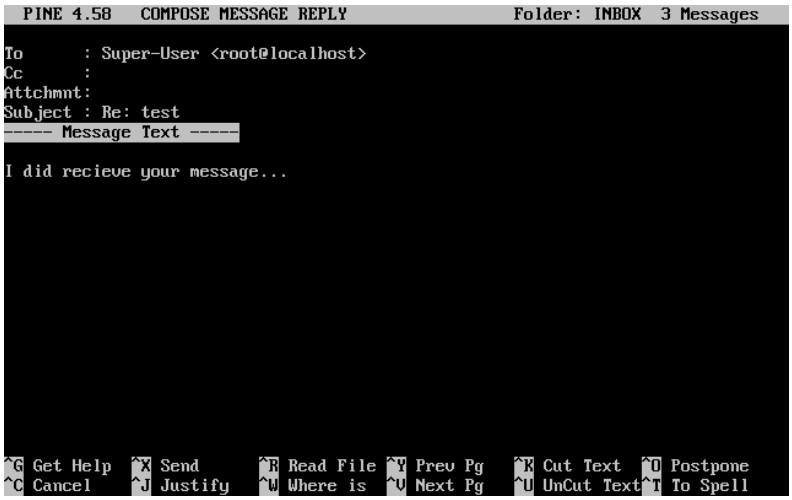


The message index shows messages in the current directory and can be navigated by using the cursor keys. Highlighted messages can be read by pressing **Enter**.



In the screenshot below, a sample message is displayed by **alpine**. Contextual keyboard shortcuts are displayed at the bottom of the screen. An example of one of a shortcut is **r**, which tells the MUA to reply to the current message being displayed.

Replying to an email in **alpine** is done using the **pico** editor, which is installed by default with **alpine**. **pico** makes it easy to navigate the message and is easier for novice users to use than vi(1) or mail(1). Once the reply is complete, the message can be sent by pressing **Ctrl**+**X**. **alpine** will ask for confirmation before sending the message.



alpine can be customized using the SETUP option from the main menu. Consult http://www.washington.edu/alpine/ for more information.

29.11 Using fetchmail

Contributed by Marc Silver.

fetchmail is a full-featured IMAP and POP client. It allows users to automatically download mail from remote IMAP and POP servers and save it into local mailboxes where it can be accessed more easily. **fetchmail** can be installed using the `mail/fetchmail` port, and offers various features, including:

- Support for the POP3, APOP, KPOP, IMAP, ETRN and ODMR protocols.
- Ability to forward mail using SMTP, which allows filtering, forwarding, and aliasing to function normally.
- May be run in daemon mode to check periodically for new messages.
- Can retrieve multiple mailboxes and forward them, based on configuration, to different local users.

This section explains some of the basic features of **fetchmail**. This utility requires a `.fetchmailrc` configuration in the user's home directory in order to run correctly. This file includes server information as well as login credentials. Due to the sensitive nature of the contents of this file, it is advisable to make it readable only by the user, with the following command:

```
% chmod 600 .fetchmailrc
```

The following `.fetchmailrc` serves as an example for downloading a single user mailbox using POP. It tells **fetchmail** to connect to `example.com` using a username of `joesoap` and a password of `XXX`. This example assumes that the user `joesoap` exists on the local system.

```
poll example.com protocol pop3 username "joesoap" password "XXX"
```

The next example connects to multiple POP and IMAP servers and redirects to different local usernames where applicable:

```
poll example.com proto pop3:
user "joesoap", with password "XXX", is "jsoap" here;
user "andrea", with password "XXXX";
poll example2.net proto imap:
user "john", with password "XXXXX", is "myth" here;
```

fetchmail can be run in daemon mode by running it with `-d`, followed by the interval (in seconds) that **fetchmail** should poll servers listed in `.fetchmailrc`. The following example configures **fetchmail** to poll every 600 seconds:

```
% fetchmail -d 600
```

More information on **fetchmail** can be found at http://fetchmail.berlios.de/.

29.12 Using procmail

Contributed by Marc Silver.

procmail is a powerful application used to filter incoming mail. It allows users to define "rules" which can be matched to incoming mails to perform specific functions or to reroute mail to alternative mailboxes or email addresses. **procmail** can be installed using the `mail/procmail` port. Once installed, it can be directly integrated into most MTAs. Consult the MTA documentation for more information. Alternatively, **procmail** can be integrated by adding the following line to a `.forward` in the home directory of the user:

```
"|exec /usr/local/bin/procmail || exit 75"
```

The following section displays some basic **procmail** rules, as well as brief descriptions of what they do. Rules must be inserted into a `.procmailrc`, which must reside in the user's home directory.

The majority of these rules can be found in procmailex(5).

To forward all mail from <`user@example.com`> to an external address of <`goodmail@example2.com`>:

```
:0
* ^From.*user@example.com
! goodmail@example2.com
```

To forward all mails shorter than 1000 bytes to an external address of <`goodmail@example2.com`>:

```
:0
* < 1000
! goodmail@example2.com
```

To send all mail sent to <`alternate@example.com`> to a mailbox called `alternate`:

```
:0
* ^TOalternate@example.com
alternate
```

To send all mail with a subject of "Spam" to `/dev/null`:

```
:0
^Subject:.*Spam
/dev/null
```

A useful recipe that parses incoming `FreeBSD.org` mailing lists and places each list in its own mailbox:

```
:0
* ^Sender:.owner-freebsd-\/[^@]+@FreeBSD.ORG
{
        LISTNAME=${MATCH}
        :0
        * LISTNAME??^\/[^@]+
        FreeBSD-${MATCH}
}
```

Chapter 30 Network Servers

Reorganized by Murray Stokely.

30.1 Synopsis

This chapter will cover some of the more frequently used network services on UNIX systems. We will cover how to install, configure, test, and maintain many different types of network services. Example configuration files are included throughout this chapter for you to benefit from.

After reading this chapter, you will know:

- How to manage the **inetd** daemon.
- How to set up a network file system.
- How to set up a network information server for sharing user accounts.
- How to set FreeBSD up to act as an LDAP server or client
- How to set FreeBSD up to act as an LDAP server or client
- How to set up automatic network settings using DHCP.
- How to set up a domain name server.
- How to set up the **Apache** HTTP Server.
- How to set up a File Transfer Protocol (FTP) Server.
- How to set up a file and print server for Windows clients using **Samba**.
- How to synchronize the time and date, and set up a time server, with the NTP protocol.
- How to configure the standard logging daemon, `syslogd`, to accept logs from remote hosts.

Before reading this chapter, you should:

- Understand the basics of the `/etc/rc` scripts.
- Be familiar with basic network terminology.
- Know how to install additional third-party software (Chapter 5).

30.2 The inetd “Super-Server”

Contributed by Chern Lee. Updated by The FreeBSD Documentation Project.

30.2.1 Overview

The inetd(8) daemon is sometimes referred to as the “Internet Super-Server” because it manages connections for many services. When a connection is received by **inetd**, it determines which program the connection is destined for, spawns the particular process and delegates the socket to it (the program is invoked with the service socket as its

standard input, output and error descriptors). Running **inetd** for servers that are not heavily used can reduce the overall system load, when compared to running each daemon individually in stand-alone mode.

Primarily, **inetd** is used to spawn other daemons, but several trivial protocols are handled directly, such as **chargen**, **auth**, and **daytime**.

This section will cover the basics in configuring **inetd** through its command-line options and its configuration file, `/etc/inetd.conf`.

30.2.2 Settings

inetd is initialized through the rc(8) system. The `inetd_enable` option is set to `NO` by default, but may be turned on by **sysinstall** during installation, depending on the configuration chosen by the user. Placing:

```
inetd_enable="YES"
```

or

```
inetd_enable="NO"
```

into `/etc/rc.conf` will enable or disable **inetd** starting at boot time. The command:

```
# service inetd rcvar
```

can be run to display the current effective setting.

Additionally, different command-line options can be passed to **inetd** via the `inetd_flags` option.

30.2.3 Command-Line Options

Like most server daemons, **inetd** has a number of options that it can be passed in order to modify its behaviour. See the inetd(8) manual page for the full list of options.

Options can be passed to **inetd** using the `inetd_flags` option in `/etc/rc.conf`. By default, `inetd_flags` is set to `-wW -C 60`, which turns on TCP wrapping for **inetd**'s services, and prevents any single IP address from requesting any service more than 60 times in any given minute.

Although we mention rate-limiting options below, novice users may be pleased to note that these parameters usually do not need to be modified. These options may be useful if an excessive amount of connections are being established. A full list of options can be found in the inetd(8) manual.

-c maximum

> Specify the default maximum number of simultaneous invocations of each service; the default is unlimited. May be overridden on a per-service basis with the `max-child` parameter.

-C rate

> Specify the default maximum number of times a service can be invoked from a single IP address in one minute; the default is unlimited. May be overridden on a per-service basis with the `max-connections-per-ip-per-minute` parameter.

-R rate

Specify the maximum number of times a service can be invoked in one minute; the default is 256. A rate of 0 allows an unlimited number of invocations.

-s maximum

Specify the maximum number of times a service can be invoked from a single IP address at any one time; the default is unlimited. May be overridden on a per-service basis with the `max-child-per-ip` parameter.

30.2.4 `inetd.conf`

Configuration of **inetd** is done via the file `/etc/inetd.conf`.

When a modification is made to `/etc/inetd.conf`, **inetd** can be forced to re-read its configuration file by running the command:

Example 30-1. Reloading the inetd Configuration File

```
# service inetd reload
```

Each line of the configuration file specifies an individual daemon. Comments in the file are preceded by a "#". The format of each entry in `/etc/inetd.conf` is as follows:

```
service-name
socket-type
protocol
{wait|nowait}[/max-child[/max-connections-per-ip-per-minute[/max-child-per-ip]]]
user[:group][/login-class]
server-program
server-program-arguments
```

An example entry for the ftpd(8) daemon using IPv4 might read:

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

service-name

This is the service name of the particular daemon. It must correspond to a service listed in `/etc/services`. This determines which port **inetd** must listen to. If a new service is being created, it must be placed in `/etc/services` first.

socket-type

Either `stream`, `dgram`, `raw`, or `seqpacket`. `stream` must be used for connection-based, TCP daemons, while `dgram` is used for daemons utilizing the UDP transport protocol.

protocol

One of the following:

Protocol	Explanation
tcp, tcp4	TCP IPv4

Protocol	Explanation
udp, udp4	UDP IPv4
tcp6	TCP IPv6
udp6	UDP IPv6
tcp46	Both TCP IPv4 and v6
udp46	Both UDP IPv4 and v6

{wait|nowait}[/max-child[/max-connections-per-ip-per-minute[/max-child-per-ip]]]

`wait|nowait` indicates whether the daemon invoked from **inetd** is able to handle its own socket or not. `dgram` socket types must use the `wait` option, while stream socket daemons, which are usually multi-threaded, should use `nowait`. `wait` usually hands off multiple sockets to a single daemon, while `nowait` spawns a child daemon for each new socket.

The maximum number of child daemons **inetd** may spawn can be set using the `max-child` option. If a limit of ten instances of a particular daemon is needed, a `/10` would be placed after `nowait`. Specifying `/0` allows an unlimited number of children

In addition to `max-child`, two other options which limit the maximum connections from a single place to a particular daemon can be enabled. `max-connections-per-ip-per-minute` limits the number of connections from any particular IP address per minutes, e.g., a value of ten would limit any particular IP address connecting to a particular service to ten attempts per minute. `max-child-per-ip` limits the number of children that can be started on behalf on any single IP address at any moment. These options are useful to prevent intentional or unintentional excessive resource consumption and Denial of Service (DoS) attacks to a machine.

In this field, either of `wait` or `nowait` is mandatory. `max-child`, `max-connections-per-ip-per-minute` and `max-child-per-ip` are optional.

A stream-type multi-threaded daemon without any `max-child`, `max-connections-per-ip-per-minute` or `max-child-per-ip` limits would simply be: `nowait`.

The same daemon with a maximum limit of ten daemons would read: `nowait/10`.

The same setup with a limit of twenty connections per IP address per minute and a maximum total limit of ten child daemons would read: `nowait/10/20`.

These options are utilized by the default settings of the fingerd(8) daemon, as seen here:

```
finger stream  tcp     nowait/3/10 nobody /usr/libexec/fingerd fingerd -s
```

Finally, an example of this field with a maximum of 100 children in total, with a maximum of 5 for any one IP address would read: `nowait/100/0/5`.

user

This is the username that the particular daemon should run as. Most commonly, daemons run as the `root` user. For security purposes, it is common to find some servers running as the `daemon` user, or the least privileged `nobody` user.

server-program

The full path of the daemon to be executed when a connection is received. If the daemon is a service provided by **inetd** internally, then `internal` should be used.

server-program-arguments

This works in conjunction with `server-program` by specifying the arguments, starting with `argv[0]`, passed to the daemon on invocation. If `mydaemon -d` is the command line, `mydaemon -d` would be the value of `server-program-arguments`. Again, if the daemon is an internal service, use `internal` here.

30.2.5 Security

Depending on the choices made at install time, many of **inetd**'s services may be enabled by default. If there is no apparent need for a particular daemon, consider disabling it. Place a "#" in front of the daemon in question in `/etc/inetd.conf`, and then reload the inetd configuration. Some daemons, such as **fingerd**, may not be desired at all because they provide information that may be useful to an attacker.

Some daemons are not security-conscious and have long, or non-existent, timeouts for connection attempts. This allows an attacker to slowly send connections to a particular daemon, thus saturating available resources. It may be a good idea to place `max-connections-per-ip-per-minute`, `max-child` or `max-child-per-ip` limitations on certain daemons if there are too many connections.

By default, TCP wrapping is turned on. Consult the hosts_access(5) manual page for more information on placing TCP restrictions on various **inetd** invoked daemons.

30.2.6 Miscellaneous

daytime, **time**, **echo**, **discard**, **chargen**, and **auth** are all internally provided services of **inetd**.

The **auth** service provides identity network services, and is configurable to a certain degree, whilst the others are simply on or off.

Consult the inetd(8) manual page for more in-depth information.

30.3 Network File System (NFS)

Reorganized and enhanced by Tom Rhodes. Written by Bill Swingle.

Among the many different file systems that FreeBSD supports is the Network File System, also known as NFS. NFS allows a system to share directories and files with others over a network. By using NFS, users and programs can access files on remote systems almost as if they were local files.

Some of the most notable benefits that NFS can provide are:

- Local workstations use less disk space because commonly used data can be stored on a single machine and still remain accessible to others over the network.
- There is no need for users to have separate home directories on every network machine. Home directories could be set up on the NFS server and made available throughout the network.
- Storage devices such as floppy disks, CDROM drives, and Zip® drives can be used by other machines on the network. This may reduce the number of removable media drives throughout the network.

30.3.1 How NFS Works

NFS consists of at least two main parts: a server and one or more clients. The client remotely accesses the data that is stored on the server machine. In order for this to function properly a few processes have to be configured and running.

The server has to be running the following daemons:

Daemon	Description
nfsd	The NFS daemon which services requests from the NFS clients.
mountd	The NFS mount daemon which carries out the requests that nfsd(8) passes on to it.
rpcbind	This daemon allows NFS clients to discover which port the NFS server is using.

The client can also run a daemon, known as **nfsiod**. The **nfsiod** daemon services the requests from the NFS server. This is optional, and improves performance, but is not required for normal and correct operation. See the nfsiod(8) manual page for more information.

30.3.2 Configuring NFS

NFS configuration is a relatively straightforward process. The processes that need to be running can all start at boot time with a few modifications to `/etc/rc.conf`.

On the NFS server, make sure that the following options are configured in the `/etc/rc.conf` file:

```
rpcbind_enable="YES"
nfs_server_enable="YES"
mountd_flags="-r"
```

mountd runs automatically whenever the NFS server is enabled.

On the client, make sure this option is present in `/etc/rc.conf`:

```
nfs_client_enable="YES"
```

The `/etc/exports` file specifies which file systems NFS should export (sometimes referred to as “share”). Each line in `/etc/exports` specifies a file system to be exported and which machines have access to that file system. Along with what machines have access to that file system, access options may also be specified. There are many such options that can be used in this file but only a few will be mentioned here. Other options are discussed in the exports(5) manual page.

Here are a few example `/etc/exports` entries:

The following examples give an idea of how to export file systems, although the settings may be different depending on the environment and network configuration. For instance, to export the `/cdrom` directory to three example machines that have the same domain name as the server (hence the lack of a domain name for each) or have entries in the `/etc/hosts` file. The `-ro` flag makes the exported file system read-only. With this flag, the remote system will not be able to write any changes to the exported file system.

```
/cdrom -ro host1 host2 host3
```

The following line exports `/home` to three hosts by IP address. This is a useful setup on a private network without a DNS server configured. Optionally the `/etc/hosts` file could be configured for internal hostnames; please review

hosts(5) for more information. The `-alldirs` flag allows the subdirectories to be mount points. In other words, it will not mount the subdirectories but permit the client to mount only the directories that are required or needed.

```
/home  -alldirs  10.0.0.2 10.0.0.3 10.0.0.4
```

The following line exports `/a` so that two clients from different domains may access the file system. The `-maproot=root` flag allows the `root` user on the remote system to write data on the exported file system as `root`. If the `-maproot=root` flag is not specified, then even if a user has `root` access on the remote system, he will not be able to modify files on the exported file system.

```
/a  -maproot=root  host.example.com box.example.org
```

In order for a client to access an exported file system, the client must have permission to do so. Make sure the client is listed in `/etc/exports`.

In `/etc/exports`, each line represents the export information for one file system to one host. A remote host can only be specified once per file system, and may only have one default entry. For example, assume that `/usr` is a single file system. The following `/etc/exports` would be invalid:

```
# Invalid when /usr is one file system
/usr/src   client
/usr/ports client
```

One file system, `/usr`, has two lines specifying exports to the same host, `client`. The correct format for this situation is:

```
/usr/src /usr/ports  client
```

The properties of one file system exported to a given host must all occur on one line. Lines without a client specified are treated as a single host. This limits how file systems may be exported; however, for most environments, this is not an issue.

The following is an example of a valid export list, where `/usr` and `/exports` are local file systems:

```
# Export src and ports to client01 and client02, but only
# client01 has root privileges on it
/usr/src /usr/ports -maproot=root    client01
/usr/src /usr/ports               client02
# The client machines have root and can mount anywhere
# on /exports. Anyone in the world can mount /exports/obj read-only
/exports -alldirs -maproot=root      client01 client02
/exports/obj -ro
```

The **mountd** daemon must be forced to recheck the `/etc/exports` file whenever it has been modified, so the changes can take effect. This can be accomplished either by sending a HUP signal to the running daemon:

```
# kill -HUP ‘cat /var/run/mountd.pid‘
```

or by invoking the `mountd` rc(8) script with the appropriate parameter:

```
# service mountd onereload
```

Please refer to Section 12.7 for more information about using rc scripts.

Alternatively, a reboot will make FreeBSD set everything up properly. A reboot is not necessary though. Executing the following commands as `root` should start everything up.

On the NFS server:

```
# rpcbind
# nfsd -u -t -n 4
# mountd -r
```

On the NFS client:

```
# nfsiod -n 4
```

Now everything should be ready to actually mount a remote file system. In these examples the server's name will be `server` and the client's name will be `client`. For testing or to temporarily mount a remote file system execute a command like this as `root` on the client:

```
# mount server:/home /mnt
```

This will mount the `/home` directory on the server at `/mnt` on the client. If everything is set up correctly, the server's files should be visible and available in the `/mnt` directory.

To permanently mount a remote file system each time the computer boots, add the file system to the `/etc/fstab` file. Here is an example:

```
server:/home    /mnt    nfs     rw      0       0
```

The fstab(5) manual page lists all the available options.

30.3.3 Locking

Some applications (e.g., **mutt**) require file locking to operate correctly. In the case of NFS, **rpc.lockd** can be used for file locking. To enable it, add the following to the `/etc/rc.conf` file on both client and server (it is assumed that the NFS client and server are configured already):

```
rpc_lockd_enable="YES"
rpc_statd_enable="YES"
```

Start the application by using:

```
# service lockd start
# service statd start
```

If real locking between the NFS clients and NFS server is not required, it is possible to let the NFS client do locking locally by passing `-L` to mount_nfs(8). Refer to the mount_nfs(8) manual page for further details.

30.3.4 Practical Uses

NFS has many practical uses. Some of the more common ones are listed below:

- Set several machines to share a CDROM or other media among them. This is cheaper and often a more convenient method to install software on multiple machines.
- On large networks, it might be more convenient to configure a central NFS server in which to store all the user home directories. These home directories can then be exported to the network so that users would always have the same home directory, regardless of which workstation they log in to.
- Several machines could have a common `/usr/ports/distfiles` directory. This allows for quick access to the source files without downloading them on each machine.

30.3.5 Automatic Mounts with amd

Contributed by Wylie Stilwell. Rewritten by Chern Lee.

amd(8) (the automatic mounter daemon) automatically mounts a remote file system whenever a file or directory within that file system is accessed. Filesystems that are inactive for a period of time will also be automatically unmounted by **amd**. Using **amd** provides a simple alternative to permanent mounts, as permanent mounts are usually listed in `/etc/fstab`.

amd operates by attaching itself as an NFS server to the `/host` and `/net` directories. When a file is accessed within one of these directories, **amd** looks up the corresponding remote mount and automatically mounts it. `/net` is used to mount an exported file system from an IP address, while `/host` is used to mount an export from a remote hostname.

An access to a file within `/host/foobar/usr` would tell **amd** to attempt to mount the `/usr` export on the host `foobar`.

Example 30-2. Mounting an Export with amd

The `showmount` command shows the available mounts on a remote host. For example, to view the mounts of a host named `foobar`:

```
% showmount -e foobar
Exports list on foobar:
/usr                                10.10.10.0
/a                                  10.10.10.0
% cd /host/foobar/usr
```

As seen in the example, the `showmount` shows `/usr` as an export. When changing directories to `/host/foobar/usr`, **amd** attempts to resolve the hostname `foobar` and automatically mount the desired export.

amd can be started by the startup scripts by placing the following lines in `/etc/rc.conf`:

```
amd_enable="YES"
```

Additionally, custom flags can be passed to **amd** from the `amd_flags` option. By default, `amd_flags` is set to:

```
amd_flags="-a /.amd_mnt -l syslog /host /etc/amd.map /net /etc/amd.map"
```

The `/etc/amd.map` file defines the default options that exports are mounted with. The `/etc/amd.conf` file defines some of the more advanced features of **amd**.

Consult the amd(8) and amd.conf(5) manual pages for more information.

30.3.6 Problems Integrating with Other Systems

Contributed by John Lind.

Certain Ethernet adapters for ISA PC systems have limitations which can lead to serious network problems, particularly with NFS. This difficulty is not specific to FreeBSD, but FreeBSD systems are affected by it.

The problem nearly always occurs when (FreeBSD) PC systems are networked with high-performance workstations, such as those made by Silicon Graphics, Inc., and Sun Microsystems, Inc. The NFS mount will work fine, and some operations may succeed, but suddenly the server will seem to become unresponsive to the client, even though requests to and from other systems continue to be processed. This happens to the client system, whether the client is the FreeBSD system or the workstation. On many systems, there is no way to shut down the client gracefully once this problem has manifested itself. The only solution is often to reset the client, because the NFS situation cannot be resolved.

Though the "correct" solution is to get a higher performance and capacity Ethernet adapter for the FreeBSD system, there is a simple workaround that will allow satisfactory operation. If the FreeBSD system is the *server*, include the option `-w=1024` on the mount from the client. If the FreeBSD system is the *client*, then mount the NFS file system with the option `-r=1024`. These options may be specified using the fourth field of the `fstab` entry on the client for automatic mounts, or by using the `-o` parameter of the mount(8) command for manual mounts.

It should be noted that there is a different problem, sometimes mistaken for this one, when the NFS servers and clients are on different networks. If that is the case, make *certain* that the routers are routing the necessary UDP information.

In the following examples, `fastws` is the host (interface) name of a high-performance workstation, and `freebox` is the host (interface) name of a FreeBSD system with a lower-performance Ethernet adapter. Also, `/sharedfs` will be the exported NFS file system (see exports(5)), and `/project` will be the mount point on the client for the exported file system. In all cases, note that additional options, such as `hard` or `soft` and `bg` may be desirable in the application.

Examples for the FreeBSD system (`freebox`) as the client in `/etc/fstab` on `freebox`:

```
fastws:/sharedfs /project nfs rw,-r=1024 0 0
```

As a manual mount command on `freebox`:

```
# mount -t nfs -o -r=1024 fastws:/sharedfs /project
```

Examples for the FreeBSD system as the server in `/etc/fstab` on `fastws`:

```
freebox:/sharedfs /project nfs rw,-w=1024 0 0
```

As a manual mount command on `fastws`:

```
# mount -t nfs -o -w=1024 freebox:/sharedfs /project
```

Nearly any 16-bit Ethernet adapter will allow operation without the above restrictions on the read or write size.

For anyone who cares, here is what happens when the failure occurs, which also explains why it is unrecoverable. NFS typically works with a "block" size of 8 K (though it may do fragments of smaller sizes). Since the maximum Ethernet packet is around 1500 bytes, the NFS "block" gets split into multiple Ethernet packets, even though it is still a single unit to the upper-level code, and must be received, assembled, and *acknowledged* as a unit. The high-performance workstations can pump out the packets which comprise the NFS unit one right after the other, just as close together as the standard allows. On the smaller, lower capacity cards, the later packets overrun the earlier

packets of the same unit before they can be transferred to the host and the unit as a whole cannot be reconstructed or acknowledged. As a result, the workstation will time out and try again, but it will try again with the entire 8 K unit, and the process will be repeated, ad infinitum.

By keeping the unit size below the Ethernet packet size limitation, we ensure that any complete Ethernet packet received can be acknowledged individually, avoiding the deadlock situation.

Overruns may still occur when a high-performance workstations is slamming data out to a PC system, but with the better cards, such overruns are not guaranteed on NFS "units". When an overrun occurs, the units affected will be retransmitted, and there will be a fair chance that they will be received, assembled, and acknowledged.

30.4 Network Information System (NIS/YP)

Written by Bill Swingle. Enhanced by Eric Ogren and Udo Erdelhoff.

30.4.1 What Is It?

NIS, which stands for Network Information Services, was developed by Sun Microsystems to centralize administration of UNIX (originally SunOS) systems. It has now essentially become an industry standard; all major UNIX like systems (Solaris, HP-UX, AIX®, Linux, NetBSD, OpenBSD, FreeBSD, etc) support NIS.

NIS was formerly known as Yellow Pages, but because of trademark issues, Sun changed the name. The old term (and yp) is still often seen and used.

It is a RPC-based client/server system that allows a group of machines within an NIS domain to share a common set of configuration files. This permits a system administrator to set up NIS client systems with only minimal configuration data and add, remove or modify configuration data from a single location.

It is similar to the Windows NT® domain system; although the internal implementation of the two are not at all similar, the basic functionality can be compared.

30.4.2 NISTerms and Processes

There are several terms and important user processes that will be explained while attempting to implement NIS on FreeBSD, regardless if the system is a NIS server or a NIS client:

Term	Description
NIS domainname	An NIS master server and all of its clients (including its slave servers) have a NIS domainname. Similar to an Windows NT domain name, the NIS domainname does not have anything to do with DNS.
rpcbind	Must be running in order to enable RPC (Remote Procedure Call, a network protocol used by NIS). If **rpcbind** is not running, it will be impossible to run an NIS server, or to act as an NIS client.
ypbind	"Binds" an NIS client to its NIS server. It will take the NIS domainname from the system, and using RPC, connect to the server. **ypbind** is the core of client-server communication in an NIS environment; if **ypbind** dies on a client machine, it will not be able to access the NIS server.

Term	Description
ypserv	Should only be running on NIS servers; this is the NIS server process itself. If ypserv(8) dies, then the server will no longer be able to respond to NIS requests (hopefully, there is a slave server to take over for it). There are some implementations of NIS (but not the FreeBSD one), that do not try to reconnect to another server if the server it used before dies. Often, the only thing that helps in this case is to restart the server process (or even the whole server) or the **ypbind** process on the client.
rpc.yppasswdd	Another process that should only be running on NIS master servers; this is a daemon that will allow NIS clients to change their NIS passwords. If this daemon is not running, users will have to login to the NIS master server and change their passwords there.

30.4.3 How Does It Work?

There are three types of hosts in an NIS environment: master servers, slave servers, and clients. Servers act as a central repository for host configuration information. Master servers hold the authoritative copy of this information, while slave servers mirror this information for redundancy. Clients rely on the servers to provide this information to them.

Information in many files can be shared in this manner. The `master.passwd`, `group`, and `hosts` files are commonly shared via NIS. Whenever a process on a client needs information that would normally be found in these files locally, it makes a query to the NIS server that it is bound to instead.

30.4.3.1 Machine Types

- A *NIS master server*. This server, analogous to a Windows NT primary domain controller, maintains the files used by all of the NIS clients. The `passwd`, `group`, and other various files used by the NIS clients live on the master server.

  Note: It is possible for one machine to be an NIS master server for more than one NIS domain. However, this will not be covered in this introduction, which assumes a relatively small-scale NIS environment.

- *NIS slave servers*. Similar to the Windows NT backup domain controllers, NIS slave servers maintain copies of the NIS master's data files. NIS slave servers provide the redundancy, which is needed in important environments. They also help to balance the load of the master server: NIS Clients always attach to the NIS server whose response they get first, and this includes slave-server-replies.

- *NIS clients*. NIS clients, like most Windows NT workstations, authenticate against the NIS server (or the Windows NT domain controller in the Windows NT workstations case) to log on.

30.4.4 Using NIS/YP

This section will deal with setting up a sample NIS environment.

30.4.4.1 Planning

Let us assume that an administrator of a small university lab, which consists of 15 FreeBSD machines, currently has no centralized point of administration. Each machine has its own `/etc/passwd` and `/etc/master.passwd`. These files are kept in sync with each other only through manual intervention; currently, a user is added to the lab, the process must be ran on all 15 machines. The lab would clearly benefit from the addition of two NIS servers.

Therefore, the configuration of the lab now looks something like:

Machine name	IP address	Machine role
`ellington`	`10.0.0.2`	NIS master
`coltrane`	`10.0.0.3`	NIS slave
`basie`	`10.0.0.4`	Faculty workstation
`bird`	`10.0.0.5`	Client machine
`cli[1-11]`	`10.0.0.[6-17]`	Other client machines

If this is the first time a NIS scheme is being developed, it should be thoroughly planned ahead of time. Regardless of network size, several decisions need to be made as part of the planning process.

30.4.4.1.1 Choosing a NIS Domain Name

This might not be the normal “domainname” for the network. It is more accurately called the “NIS domainname”. When a client broadcasts its requests for info, it includes the name of the NIS domain that it is part of. This is how multiple servers on one network can tell which server should answer which request. Think of the NIS domainname as the name for a group of hosts that are related in some way.

Some organizations choose to use their Internet domainname for their NIS domainname. This is not recommended as it can cause confusion when trying to debug network problems. The NIS domainname should be unique within the network and it is helpful if it describes the group of machines it represents. For example, the Art department at Acme Inc. might be in the “acme-art” NIS domain. For this example, assume the chosen name will be `test-domain`.

However, some operating systems (notably SunOS) use their NIS domain name as their Internet domain name. If one or more machines on the network have this restriction, it *must* be used as the Internet domain name for the NIS domain name.

30.4.4.1.2 Physical Server Requirements

There are several things to keep in mind when choosing a machine to use as a NIS server. One of the unfortunate things about NIS is the level of dependency the clients have on the server. If a client cannot contact the server for its NIS domain, very often the machine becomes unusable. The lack of user and group information causes most systems to temporarily freeze up. With this in mind be sure to choose a machine that will not be prone to being rebooted frequently, or one that might be used for development. The NIS server should ideally be a stand alone machine whose sole purpose in life is to be an NIS server. If the network is not very heavily used, it is acceptable to put the NIS server on a machine running other services, however; if the NIS server becomes unavailable, it will adversely affect *all* NIS clients.

30.4.4.2 NIS Servers

The canonical copies of all NIS information are stored on a single machine called the NIS master server. The databases used to store the information are called NIS maps. In FreeBSD, these maps are stored in `/var/yp/[domainname]` where `[domainname]` is the name of the NIS domain being served. A single NIS server can support several domains at once, therefore it is possible to have several such directories, one for each supported domain. Each domain will have its own independent set of maps.

NIS master and slave servers handle all NIS requests with the `ypserv` daemon. `ypserv` is responsible for receiving incoming requests from NIS clients, translating the requested domain and map name to a path to the corresponding database file and transmitting data from the database back to the client.

30.4.4.2.1 Setting Up a NIS Master Server

Setting up a master NIS server can be relatively straight forward, depending on environmental needs. FreeBSD comes with support for NIS out-of-the-box. It only needs to be enabled by adding the following lines to `/etc/rc.conf`:

1. `nisdomainname="test-domain"`
   This line will set the NIS domainname to `test-domain` upon network setup (e.g., after reboot).
2. `nis_server_enable="YES"`
   This will tell FreeBSD to start up the NIS server processes when the networking is next brought up.
3. `nis_yppasswdd_enable="YES"`
   This will enable the `rpc.yppasswdd` daemon which, as mentioned above, will allow users to change their NIS password from a client machine.

> **Note:** Depending on the NIS setup, additional entries may be required. See the section about NIS servers that are also NIS clients, below, for details.

After setting up the above entries, run the command `/etc/netstart` as superuser. It will set up everything, using the values defined in `/etc/rc.conf`. As a last step, before initializing the NIS maps, start the **ypserv** daemon manually:

```
# service ypserv start
```

30.4.4.2.2 Initializing the NIS Maps

The *NIS maps* are database files, that are kept in the `/var/yp` directory. They are generated from configuration files in the `/etc` directory of the NIS master, with one exception: `/etc/master.passwd`. This is for a good reason, never propagate passwords for `root` and other administrative accounts to all the servers in the NIS domain. Therefore, before the the NIS maps are initialized, configure the primary password files:

```
# cp /etc/master.passwd /var/yp/master.passwd
```

```
# cd /var/yp
# vi master.passwd
```

It is advisable to remove all entries regarding system accounts (`bin`, `tty`, `kmem`, `games`, etc), as well as any accounts that do not need to be propagated to the NIS clients (for example `root` and any other UID 0 (superuser) accounts).

> **Note:** Ensure the `/var/yp/master.passwd` is neither group or world readable (mode 600)! Use the `chmod` command, as appropriate.

When this task has been completed, it is time to initialize the NIS maps. FreeBSD includes a script named `ypinit` to do this (see its manual page for more information). Note that this script is available on most UNIX Operating Systems, but not on all. On Digital UNIX/Compaq Tru64 UNIX it is called `ypsetup`. Because we are generating maps for an NIS master, we are going to pass the `-m` option to `ypinit`. To generate the NIS maps run:

```
ellington# ypinit -m test-domain
Server Type: MASTER Domain: test-domain
Creating an YP server will require that you answer a few questions.
Questions will all be asked at the beginning of the procedure.
Do you want this procedure to quit on non-fatal errors? [y/n: n] n
Ok, please remember to go back and redo manually whatever fails.
If you don't, something might not work.
At this point, we have to construct a list of this domains YP servers.
rod.darktech.org is already known as master server.
Please continue to add any slave servers, one per line. When you are
done with the list, type a <control D>.
master server   :  ellington
next host to add:  coltrane
next host to add:  ^D
The current list of NIS servers looks like this:
ellington
coltrane
Is this correct?  [y/n: y] y

[..output from map generation..]

NIS Map update completed.
ellington has been setup as an YP master server without any errors.
```

At this point, `ypinit` should have created `/var/yp/Makefile` from `/var/yp/Makefile.dist`. When created, this file assumes that the operating environment is a single server NIS system with only FreeBSD machines. Since `test-domain` has a slave server as well, edit `/var/yp/Makefile` as well:

```
ellington# vi /var/yp/Makefile
```

You should comment out the line that says

```
NOPUSH = "True"
```

(if it is not commented out already).

30.4.4.2.3 Setting up a NIS Slave Server

Setting up an NIS slave server is even more simple than setting up the master. Log on to the slave server and edit the file `/etc/rc.conf` as you did before. The only difference is that we now must use the `-s` option when running `ypinit`. The `-s` option requires the name of the NIS master be passed to it as well, so our command line looks like:

```
coltrane# ypinit -s ellington test-domain

Server Type: SLAVE Domain: test-domain Master: ellington

Creating an YP server will require that you answer a few questions.
Questions will all be asked at the beginning of the procedure.

Do you want this procedure to quit on non-fatal errors? [y/n: n]  n

Ok, please remember to go back and redo manually whatever fails.
If you don't, something might not work.
There will be no further questions. The remainder of the procedure
should take a few minutes, to copy the databases from ellington.
Transferring netgroup...
ypxfr: Exiting: Map successfully transferred
Transferring netgroup.byuser...
ypxfr: Exiting: Map successfully transferred
Transferring netgroup.byhost...
ypxfr: Exiting: Map successfully transferred
Transferring master.passwd.byuid...
ypxfr: Exiting: Map successfully transferred
Transferring passwd.byuid...
ypxfr: Exiting: Map successfully transferred
Transferring passwd.byname...
ypxfr: Exiting: Map successfully transferred
Transferring group.bygid...
ypxfr: Exiting: Map successfully transferred
Transferring group.byname...
ypxfr: Exiting: Map successfully transferred
Transferring services.byname...
ypxfr: Exiting: Map successfully transferred
Transferring rpc.bynumber...
ypxfr: Exiting: Map successfully transferred
Transferring rpc.byname...
ypxfr: Exiting: Map successfully transferred
Transferring protocols.byname...
ypxfr: Exiting: Map successfully transferred
Transferring master.passwd.byname...
ypxfr: Exiting: Map successfully transferred
Transferring networks.byname...
ypxfr: Exiting: Map successfully transferred
Transferring networks.byaddr...
ypxfr: Exiting: Map successfully transferred
Transferring netid.byname...
ypxfr: Exiting: Map successfully transferred
Transferring hosts.byaddr...
ypxfr: Exiting: Map successfully transferred
```

```
Transferring protocols.bynumber...
ypxfr: Exiting: Map successfully transferred
Transferring ypservers...
ypxfr: Exiting: Map successfully transferred
Transferring hosts.byname...
ypxfr: Exiting: Map successfully transferred

coltrane has been setup as an YP slave server without any errors.
Don't forget to update map ypservers on ellington.
```

There should be a directory called `/var/yp/test-domain`. Copies of the NIS master server's maps should be in this directory. These files must always be up to date. The following `/etc/crontab` entries on the slave servers should do the job:

```
20      *       *       *       *       root   /usr/libexec/ypxfr passwd.byname
21      *       *       *       *       root   /usr/libexec/ypxfr passwd.byuid
```

These two lines force the slave to sync its maps with the maps on the master server. These entries are not mandatory because the master server automatically attempts to push any map changes to its slaves; however, due to the importance of correct password information on other clients depending on the slave server, it is recommended to specifically force the password map updates frequently. This is especially important on busy networks where map updates might not always complete.

Now, run the command `/etc/netstart` on the slave server as well, which again starts the NIS server.

30.4.4.3 NIS Clients

An NIS client establishes what is called a binding to a particular NIS server using the `ypbind` daemon. The `ypbind` command checks the system's default domain (as set by the `domainname` command), and begins broadcasting RPC requests on the local network. These requests specify the name of the domain for which `ypbind` is attempting to establish a binding. If a server that has been configured to serve the requested domain receives one of the broadcasts, it will respond to `ypbind`, which will record the server's address. If there are several servers available (a master and several slaves, for example), `ypbind` will use the address of the first one to respond. From that point on, the client system will direct all of its NIS requests to that server. `ypbind` will occasionally "ping" the server to make sure it is still up and running. If it fails to receive a reply to one of its pings within a reasonable amount of time, `ypbind` will mark the domain as unbound and begin broadcasting again in the hopes of locating another server.

30.4.4.3.1 Setting Up a NIS Client

Setting up a FreeBSD machine to be a NIS client is fairly straightforward.

1. Edit `/etc/rc.conf` and add the following lines in order to set the NIS domainname and start `ypbind` during network startup:

   ```
   nisdomainname="test-domain"
   nis_client_enable="YES"
   ```

2. To import all possible password entries from the NIS server, remove all user accounts from the `/etc/master.passwd` file and use `vipw` to add the following line to the end of the file:

   ```
   +:::::::::
   ```

Note: This line will afford anyone with a valid account in the NIS server's password maps an account. There are many ways to configure the NIS client by changing this line. See the netgroups section below for more information. For more detailed reading see O'Reilly's book on `Managing NFS and NIS`.

Note: Keep in mind that at least one local account (i.e. not imported via NIS) must exist in `/etc/master.passwd` and this account should also be a member of the group `wheel`. If there is something wrong with NIS, this account can be used to log in remotely, become `root`, and fix things.

3. To import all possible group entries from the NIS server, add this line to `/etc/group`:

   ```
   +:*::
   ```

To start the NIS client immediately, execute the following commands as the superuser:

```
# /etc/netstart
# service ypbind start
```

After completing these steps, the command, `ypcat passwd`, should show the server's passwd map.

30.4.5 NIS Security

In general, any remote user may issue an RPC to ypserv(8) and retrieve the contents of the NIS maps, provided the remote user knows the domainname. To prevent such unauthorized transactions, ypserv(8) supports a feature called "securenets" which can be used to restrict access to a given set of hosts. At startup, ypserv(8) will attempt to load the securenets information from a file called `/var/yp/securenets`.

Note: This path varies depending on the path specified with the `-p` option. This file contains entries that consist of a network specification and a network mask separated by white space. Lines starting with "#" are considered to be comments. A sample securenets file might look like this:

```
# allow connections from local host -- mandatory
127.0.0.1     255.255.255.255
# allow connections from any host
# on the 192.168.128.0 network
192.168.128.0 255.255.255.0
# allow connections from any host
# between 10.0.0.0 to 10.0.15.255
# this includes the machines in the testlab
10.0.0.0      255.255.240.0
```

If ypserv(8) receives a request from an address that matches one of these rules, it will process the request normally. If the address fails to match a rule, the request will be ignored and a warning message will be logged. If the `/var/yp/securenets` file does not exist, `ypserv` will allow connections from any host.

The ypserv program also has support for Wietse Venema's **TCP Wrapper** package. This allows the administrator to use the **TCP Wrapper** configuration files for access control instead of /var/yp/securenets.

> **Note:** While both of these access control mechanisms provide some security, they, like the privileged port test, are vulnerable to "IP spoofing" attacks. All NIS-related traffic should be blocked at the firewall.
>
> Servers using /var/yp/securenets may fail to serve legitimate NIS clients with archaic TCP/IP implementations. Some of these implementations set all host bits to zero when doing broadcasts and/or fail to observe the subnet mask when calculating the broadcast address. While some of these problems can be fixed by changing the client configuration, other problems may force the retirement of the client systems in question or the abandonment of /var/yp/securenets.
>
> Using /var/yp/securenets on a server with such an archaic implementation of TCP/IP is a really bad idea and will lead to loss of NIS functionality for large parts of the network.
>
> The use of **TCP Wrapper** increases the latency of the NIS server. The additional delay may be long enough to cause timeouts in client programs, especially in busy networks or with slow NIS servers. If one or more of the client systems suffers from these symptoms, convert the client systems in question into NIS slave servers and force them to bind to themselves.

30.4.6 Barring Some Users from Logging On

In our lab, there is a machine basie that is supposed to be a faculty only workstation. We do not want to take this machine out of the NIS domain, yet the passwd file on the master NIS server contains accounts for both faculty and students. What can we do?

There is a way to bar specific users from logging on to a machine, even if they are present in the NIS database. To do this, add *-username* with the correct number of colons like other entries to the end of the /etc/master.passwd file on the client machine, where *username* is the username of the user to bar from logging in. The line with the blocked user must be before the + line for allowing NIS users. This should preferably be done using vipw, since vipw will sanity check the changes to /etc/master.passwd, as well as automatically rebuild the password database after editing. For example, to bar user bill from logging on to basie:

```
basie# vipw
[add -bill::::::::: to the end, exit]
vipw: rebuilding the database...
vipw: done

basie# cat /etc/master.passwd

root:[password]:0:0::0:0:The super-user:/root:/bin/csh
toor:[password]:0:0::0:0:The other super-user:/root:/bin/sh
daemon:*:1:1::0:0:Owner of many system processes:/root:/sbin/nologin
operator:*:2:5::0:0:System &:/:/sbin/nologin
bin:*:3:7::0:0:Binaries Commands and Source,,:/:/sbin/nologin
tty:*:4:65533::0:0:Tty Sandbox:/:/sbin/nologin
kmem:*:5:65533::0:0:KMem Sandbox:/:/sbin/nologin
games:*:7:13::0:0:Games pseudo-user:/usr/games:/sbin/nologin
news:*:8:8::0:0:News Subsystem:/:/sbin/nologin
man:*:9:9::0:0:Mister Man Pages:/usr/share/man:/sbin/nologin
bind:*:53:53::0:0:Bind Sandbox:/:/sbin/nologin
```

```
uucp:*:66:66::0:0:UUCP pseudo-user:/var/spool/uucppublic:/usr/libexec/uucp/uucico
xten:*:67:67::0:0:X-10 daemon:/usr/local/xten:/sbin/nologin
pop:*:68:6::0:0:Post Office Owner:/nonexistent:/sbin/nologin
nobody:*:65534:65534::0:0:Unprivileged user:/nonexistent:/sbin/nologin
-bill:::::::::
+:::::::::

basie#
```

30.4.7 Using Netgroups

Contributed by Udo Erdelhoff.

The method shown in the previous section works reasonably well for special rules in an environment with small numbers of users and/or machines. On larger networks, administrators *will* likely forget to bar some users from logging onto sensitive machines, or may even have to modify each machine separately, thus losing the main benefit of NIS: *centralized* administration.

The NIS developers' solution for this problem is called *netgroups*. Their purpose and semantics can be compared to the normal groups used by UNIX file systems. The main differences are the lack of a numeric ID and the ability to define a netgroup by including both user accounts and other netgroups.

Netgroups were developed to handle large, complex networks with hundreds of users and machines. On one hand, this is a Good Thing in such a situation. On the other hand, this complexity makes it almost impossible to explain netgroups with really simple examples. The example used in the remainder of this section demonstrates this problem.

Let us assume that the successful introduction of NIS in the laboratory caught a superiors' interest. The next task is to extend the NIS domain to cover some of the other machines on campus. The two tables contain the names of the new users and new machines as well as brief descriptions of them.

User Name(s)	Description
`alpha`, `beta`	Normal employees of the IT department
`charlie`, `delta`	The new apprentices of the IT department
`echo`, `foxtrott`, `golf`, ...	Ordinary employees
`able`, `baker`, ...	The current interns

Machine Name(s)	Description
`war`, `death`, `famine`, `pollution`	The most important servers deployed. Only the IT employees are allowed to log onto these machines.
`pride`, `greed`, `envy`, `wrath`, `lust`, `sloth`	Less important servers. All members of the IT department are allowed to login onto these machines.
`one`, `two`, `three`, `four`, ...	Ordinary workstations. Only the *real* employees are allowed to use these machines.
`trashcan`	A very old machine without any critical data. Even the intern is allowed to use this box.

An attempt to implement these restrictions by separately blocking each user, would require the addition of the `-user` line to each system's `passwd`. One line for each user who is not allowed to login onto that system. Forgetting just

one entry could cause significant trouble. It may be feasible to do this correctly during the initial setup; however, eventually someone will forget to add these lines for new users.

Handling this situation with netgroups offers several advantages. Each user need not be handled separately; they would be assigned to one or more netgroups and logins would be allowed or forbidden for all members of the netgroup. While adding a new machine, login restrictions must be defined for all netgroups. If a new user is added, they must be added to one or more netgroups. Those changes are independent of each other: no more “for each combination of user and machine do...” If the NIS setup is planned carefully, only one central configuration file needs modification to grant or deny access to machines.

The first step is the initialization of the NIS map netgroup. FreeBSD’s ypinit(8) does not create this map by default, but its NIS implementation will support it after creation. To create an empty map, simply type

```
ellington# vi /var/yp/netgroup
```

and begin adding content. For our example, we need at least four netgroups: IT employees, IT apprentices, normal employees and interns.

```
IT_EMP  (,alpha,test-domain)    (,beta,test-domain)
IT_APP  (,charlie,test-domain)  (,delta,test-domain)
USERS   (,echo,test-domain)     (,foxtrott,test-domain) \
        (,golf,test-domain)
INTERNS (,able,test-domain)     (,baker,test-domain)
```

`IT_EMP`, `IT_APP` etc. are the names of the netgroups. Each bracketed group adds one or more user accounts to it. The three fields inside a group are:

1. The name of the host(s) where the following items are valid. If a hostname is not specified, the entry is valid on all hosts. If a hostname is specified, it will need to be micro-managed within this configuration.
2. The name of the account that belongs to this netgroup.
3. The NIS domain for the account. Accounts may be imported from other NIS domains into a netgroup.

Each of these fields may contain wildcards. See netgroup(5) for details.

> **Note:** Netgroup names longer than 8 characters should not be used, especially with machines running other operating systems within the NIS domain. The names are case sensitive; using capital letters for netgroup names is an easy way to distinguish between user, machine and netgroup names.
>
> Some NIS clients (other than FreeBSD) cannot handle netgroups with a large number of entries. For example, some older versions of SunOS start to cause trouble if a netgroup contains more than 15 *entries*. This limit may be circumvented by creating several sub-netgroups with 15 users or fewer and a real netgroup consisting of the sub-netgroups:
>
> ```
> BIGGRP1  (,joe1,domain)  (,joe2,domain)  (,joe3,domain) [...]
> BIGGRP2  (,joe16,domain)  (,joe17,domain) [...]
> BIGGRP3  (,joe31,domain)  (,joe32,domain)
> BIGGROUP  BIGGRP1 BIGGRP2 BIGGRP3
> ```
>
> Repeat this process if more than 225 users will exist within a single netgroup.

Activating and distributing the new NIS map is easy:

```
ellington# cd /var/yp
ellington# make
```

This will generate the three NIS maps `netgroup`, `netgroup.byhost` and `netgroup.byuser`. Use ypcat(1) to check if the new NIS maps are available:

```
ellington% ypcat -k netgroup
ellington% ypcat -k netgroup.byhost
ellington% ypcat -k netgroup.byuser
```

The output of the first command should resemble the contents of `/var/yp/netgroup`. The second command will not produce output without specified host-specific netgroups. The third command may be used to get the list of netgroups for a user.

The client setup is quite simple. To configure the server `war`, use vipw(8) to replace the line

```
+::::::::::
```

with

```
+@IT_EMP:::::::::
```

Now, only the data for the users defined in the netgroup `IT_EMP` is imported into `war`'s password database and only these users are allowed to login.

Unfortunately, this limitation also applies to the ~ function of the shell and all routines converting between user names and numerical user IDs. In other words, `cd ~`*`user`* will not work, `ls -l` will show the numerical ID instead of the username and `find . -user joe -print` will fail with `No such user`. To fix this, import all user entries *without allowing them to login into the servers*.

This can be achieved by adding another line to `/etc/master.passwd`. This line should contain:

`+::::::::::/sbin/nologin`, meaning "Import all entries but replace the shell with `/sbin/nologin` in the imported entries". It is possible to replace any field in the `passwd` entry by placing a default value in `/etc/master.passwd`.

> **Warning:** Make sure that the line `+::::::::::/sbin/nologin` is placed after `+@IT_EMP:::::::::`. Otherwise, all user accounts imported from NIS will have `/sbin/nologin` as their login shell.

After this change, the NIS map will only need modification when a new employee joins the IT department. A similar approach for the less important servers may be used by replacing the old `+::::::::::` in their local version of `/etc/master.passwd` with something like this:

```
+@IT_EMP:::::::::
+@IT_APP:::::::::
+::::::::::/sbin/nologin
```

The corresponding lines for the normal workstations could be:

```
+@IT_EMP:::::::::
+@USERS:::::::::
+::::::::::/sbin/nologin
```

And everything would be fine until there is a policy change a few weeks later: The IT department starts hiring interns. The IT interns are allowed to use the normal workstations and the less important servers; and the IT apprentices are allowed to login onto the main servers. Add a new netgroup `IT_INTERN`, then add the new IT interns to this netgroup and start to change the configuration on each and every machine. As the old saying goes: "Errors in centralized planning lead to global mess".

NIS' ability to create netgroups from other netgroups can be used to prevent situations like these. One possibility is the creation of role-based netgroups. For example, one might create a netgroup called `BIGSRV` to define the login restrictions for the important servers, another netgroup called `SMALLSRV` for the less important servers and a third netgroup called `USERBOX` for the normal workstations. Each of these netgroups contains the netgroups that are allowed to login onto these machines. The new entries for the NIS map netgroup should look like this:

```
BIGSRV    IT_EMP  IT_APP
SMALLSRV  IT_EMP  IT_APP  ITINTERN
USERBOX   IT_EMP  ITINTERN USERS
```

This method of defining login restrictions works reasonably well when it is possible to define groups of machines with identical restrictions. Unfortunately, this is the exception and not the rule. Most of the time, the ability to define login restrictions on a per-machine basis is required.

Machine-specific netgroup definitions are the other possibility to deal with the policy change outlined above. In this scenario, the `/etc/master.passwd` of each box contains two lines starting with "+". The first of them adds a netgroup with the accounts allowed to login onto this machine, the second one adds all other accounts with `/sbin/nologin` as shell. It is a good idea to use the "ALL-CAPS" version of the machine name as the name of the netgroup. In other words, the lines should look like this:

```
+@BOXNAME:::::::::
+::::::::::/sbin/nologin
```

Once this task is completed on all the machines, there is no longer a need to modify the local versions of `/etc/master.passwd` ever again. All further changes can be handled by modifying the NIS map. Here is an example of a possible netgroup map for this scenario with some additional goodies:

```
# Define groups of users first
IT_EMP    (,alpha,test-domain)    (,beta,test-domain)
IT_APP    (,charlie,test-domain)  (,delta,test-domain)
DEPT1     (,echo,test-domain)     (,foxtrott,test-domain)
DEPT2     (,golf,test-domain)     (,hotel,test-domain)
DEPT3     (,india,test-domain)    (,juliet,test-domain)
ITINTERN  (,kilo,test-domain)     (,lima,test-domain)
D_INTERNS (,able,test-domain)     (,baker,test-domain)
#
# Now, define some groups based on roles
USERS     DEPT1   DEPT2     DEPT3
BIGSRV    IT_EMP  IT_APP
SMALLSRV  IT_EMP  IT_APP    ITINTERN
USERBOX   IT_EMP  ITINTERN  USERS
#
# And a groups for a special tasks
# Allow echo and golf to access our anti-virus-machine
SECURITY  IT_EMP  (,echo,test-domain)  (,golf,test-domain)
#
# machine-based netgroups
```

```
# Our main servers
WAR       BIGSRV
FAMINE    BIGSRV
# User india needs access to this server
POLLUTION  BIGSRV  (,india,test-domain)
#
# This one is really important and needs more access restrictions
DEATH     IT_EMP
#
# The anti-virus-machine mentioned above
ONE       SECURITY
#
# Restrict a machine to a single user
TWO       (,hotel,test-domain)
# [...more groups to follow]
```

If some kind of database is used to manage the user accounts, it may be possible to create the first part of the map using the database's reporting tools. This way, new users will automatically have access to the boxes.

One last word of caution: It may not always be advisable to use machine-based netgroups. When deploying a couple of dozen or even hundreds of identical machines for student labs, role-based netgroups instead of machine-based netgroups may be used to keep the size of the NIS map within reasonable limits.

30.4.8 Important Things to Remember

There are still a couple of things administrators need to do differently now that machines are in an NIS environment.

- Every time a new user is added to the lab, they must be added to the master NIS server and the NIS maps will need rebuilt. If this step is omitted, the new user will not be able to login anywhere except on the NIS master. For example, if we needed to add a new user `jsmith` to the lab, we would:

  ```
  # pw useradd jsmith
  # cd /var/yp
  # make test-domain
  ```

  The user may also be added using `adduser jsmith` instead of `pw useradd jsmith`.

- *Keep the administration accounts out of the NIS maps*. This is undesirable as it will create a security risk. These users and passwords should not be propagated to all machines. Especially if these machines will have users whom should not have access to those accounts.

- *Keep the NIS master and slave secure, and minimize their downtime*. If somebody either hacks or simply turns off these machines, they have effectively rendered many people without the ability to login to the lab.

  This is the chief weakness of any centralized administration system. If the NIS servers are not protected, there will be a lot of angry users and unhappy management!

30.4.9 NIS v1 Compatibility

FreeBSD's **ypserv** has some support for serving NIS v1 clients. FreeBSD's NIS implementation only uses the NIS v2 protocol; however, other implementations include support for the v1 protocol for backwards compatibility with older systems. The **ypbind** daemons supplied with these systems will attempt to establish a binding to an NIS v1

server even though they may never actually need it (and they may persist in broadcasting in search of one even after they receive a response from a v2 server). Note that while support for normal client calls is provided, this version of **ypserv** does not handle v1 map transfer requests. Additionally, it cannot be used as a master or slave in conjunction with older NIS servers that only support the v1 protocol. Fortunately, there probably are not any such servers still in use today.

30.4.10 NIS Servers That Are Also NIS Clients

Care must be taken when running **ypserv** in a multi-server domain where the server machines are also NIS clients. It is generally a good idea to force the servers to bind to themselves rather than allowing them to broadcast bind requests and possibly become bound to each other. Strange failure modes can result if one server goes down and others are dependent upon it. Eventually all the clients will time out and attempt to bind to other servers, but the delay involved can be considerable and the failure mode is still present since the servers might bind to each other all over again.

A host may be forced to bind to a particular server by running `ypbind` with the `-S` flag. Add the following lines to `/etc/rc.conf` to enable this feature during every system boot:

```
nis_client_enable="YES" # run client stuff as well
nis_client_flags="-S NIS domain,server"
```

See ypbind(8) for further information.

30.4.11 Password Formats

One of the most common issues that people run into when trying to implement NIS is password format compatibility. If the NIS server is using DES encrypted passwords, it will only support clients that are also using DES. For example, if any Solaris NIS clients exist on the network, there is a highly likelihood DES must be used for encrypted passwords.

To check which format the servers and clients are using, look at `/etc/login.conf`. If the host is configured to use DES encrypted passwords, then the `default` class will contain an entry like this:

```
default:\
       :passwd_format=des:\
       :copyright=/etc/COPYRIGHT:\
       [Further entries elided]
```

Other possible values for the `passwd_format` capability include `blf` and `md5` (for Blowfish and MD5 encrypted passwords, respectively).

If any changes were made to `/etc/login.conf`, the login capability database must be rebuilt by running the following command as `root`:

```
# cap_mkdb /etc/login.conf
```

> **Note:** The format of passwords already in `/etc/master.passwd` will not be updated until a user changes his password for the first time *after* the login capability database is rebuilt.

Next, in order to ensure that passwords are encrypted with the chosen format, check that the `crypt_default` in `/etc/auth.conf` gives precedence to the chosen password format. To do this, place the chosen format first in the list. For example, when using DES encrypted passwords, the entry would be:

```
crypt_default   =       des blf md5
```

Having followed the above steps on each of the FreeBSD based NIS servers and clients, verify that they all agree on which password format is used within the network. If users have trouble authenticating on an NIS client, this is a pretty good place to start looking for possible problems. Remember: to deploy an NIS server for a heterogeneous network, they will probably have to use DES on all systems because it is the lowest common standard.

30.5 FreeBSD and LDAP

Written by Tom Rhodes.

LDAP, the Lightweight Directory Access Protocol, is an application layer protocol used to access, modify, and authenticate (bind) using a distributed directory information service. Think of it as a phone or record book which stores several levels of hierarchical, homogeneous information. It is often used in networks where users often need access to several levels of internal information utilizing a single account. For example, email authentication, pulling employee contact information, and internal website authentication might all make use of a single user in the LDAP server's record base.

This section will not provide a history or the implementation details of the protocol. These sections were authored to get an LDAP server and/or client configured both quickly and securely; however, any information base requires planning and this is no exception.

Planning should include what type of information will be stored, what that information will be used for, whom should have access to said information, and how to secure this information from prying eyes.

30.5.1 LDAP Terminology and Structure

Before continuing, several parts of LDAP must be explained to prevent confusion. And confusion with this configuration is relatively simple. To begin, all directory entries consist of a group of *attributes*. Each of these attribute sets contain a name, a unique identifier known as a DN or distinguished name normally built from several other attributes such as the RDN. The RDN or relative distinguished name, is a more common name for the attribute. Like directories have absolute and relative paths, consider a DN as an absolute path and the RDN as the relative path.

As an example, an entry might look like the following:

```
% ldapsearch -xb "uid=trhodes,ou=users,o=example.com"

# extended LDIF
#
# LDAPv3
# base <uid=trhodes,ou=users,o=example.com> with scope subtree
# filter: (objectclass=*)
# requesting: ALL
#

# trhodes, users, example.com
```

```
dn: uid=trhodes,ou=users,o=example.com
mail: trhodes@example.com
cn: Tom Rhodes
uid: trhodes
telephoneNumber: (xxx) xxx-xxxx

# search result
search: 2
result: 0 Success

# numResponses: 2
# numEntries: 1
```

In this example, it is very obvious what the various attributes are; however, the cn attribute should be noticed. This is the RDN discussed previously. In addition, there is a unique user id provided here. It is common practice to have specific uid or uuids for entries to ease in any future migration.

30.5.2 Configuring an LDAP Server

To configure FreeBSD to act as an LDAP server, the OpenLDAP port needs installed. This may be accomplished using the `pkg_add` command or by installing the `net/openldap24-server` port. Building the port is recommended as the administrator may select a great deal of options at this time and disable some options. In most cases, the defaults will be fine; however, this is the time to enable SQL support if needed.

A few directories will be required from this point on, at minimal, a data directory and a directory to store the certificates in. Create them both with the following commands:

```
# mkdir /var/db/openldap-data
```

```
# mkdir /usr/local/etc/openldap/private
```

Copy over the database configuration file:

```
# cp /usr/local/etc/openldap/DB_CONFIG.example /var/db/openldap-data/DB_CONFIG
```

The next phase is to configure the SSL certificates. While creating certificates is discussed in the OpenSSL section in this book, a certificate authority is needed so a different method will be used. It is recommended that this section be reviewed prior to configuring to ensure correct information is entered during the certificate creation process below.

The following commands must be executed in the `/usr/local/etc/openldap/private` directory. This is important as the file permissions will need to be restrictive and users should not have access to these files directly. To create the certificates, issues the following commands.

```
# openssl req -days 365 -nodes -new -x509 -keyout ca.key -out ../ca.crt
```

The entries for these may be completely generic *except* for the *Common Name* entry. This entry must have something different than the system hostname. If the entry is the hostname, it would be like the hostname is attempting to verify hostname. In cases with a self signed certificate like this example, just prefix the hostname with CA for certificate authority.

The next task is to create a certificate signing request and a private key. To do this, issue the following commands:

```
# openssl req -days 365 -nodes -new -keyout server.key -out server.csr
```

During the certificate generation process, be sure to correctly set the common name attribute. After this has been completed, the key will need signed:

```
# openssl x509 -req -days 365 -in server.csr -out ../server.crt -CA ../ca.crt -CAkey ca.key -CAcreateserial
```

The final part of the certificate generation process is to generate and sign the client certificates:

```
# openssl req -days 365 -nodes -new -keyout client.key -out client.csr

# openssl x509 -req -days 3650 -in client.csr -out ../client.crt -CA ../ca.crt -CAkey ca.key
```

Remember, again, to respect the common name attribute. This is a common cause for confusion during the first attempt to configure LDAP. In addition, ensure that a total of eight (8) new files have been generated through the proceeding commands. If so, the next step is to edit `/usr/local/etc/openldap/slapd.conf` and add the following options:

```
TLSCipherSuite HIGH:MEDIUM:+SSLv3
TLSCertificateFile /usr/local/etc/openldap/server.crt
TLSCertificateKeyFile /usr/local/etc/openldap/private/server.key
TLSCACertificateFile /usr/local/etc/openldap/ca.crt
```

In addition, edit `/usr/local/etc/openldap/ldap.conf` and add the following lines:

```
TLS_CACERT /usr/local/etc/openldap/ca.crt
TLS_CIPHER_SUITE HIGH:MEDIUM:+SSLv3
```

While editing these this file, set the `BASE` to the desired values, and uncomment all three of the `URI`, `SIZELIMIT` and `TIMELIMIT` options. In addition, set the `URI` to contain `ldap://` and `ldaps://`.

The resulting file should look similar to the following shown here:

```
BASE    dc=example,dc=com
URI     ldap:// ldaps://

SIZELIMIT       12
TIMELIMIT       15
#DEREF          never

TLS_CACERT /usr/local/etc/openldap/ca.crt
TLS_CIPHER_SUITE HIGH:MEDIUM:+SSLv3
```

A password for the server will need to be created as the default is extremely poor as is normal in this industry. To do this, issue the following command, sending the output to `slapd.conf`:

```
# slappasswd -h "{SHA}" >> /usr/local/etc/openldap/slapd.conf
```

There will be a prompt for entering the password and, if the process does not fail, a password hash will be added to the end of `slapd.conf`. The `slappasswd` understands several hashing formats, refer to the manual page for more information.

Edit `/usr/local/etc/openldap/slapd.conf` and add the following lines:

```
password-hash {sha}
allow bind_v2
```

In addition, the `suffix` in this file must be updated to match the `BASE` from the previous configuration. The `rootdn` option should also be set. A good recommendation is something like `cn=Manager`. Before saving this file, place the `rootpw` option in front of the password output from the `slappasswd` and delete the old `rootpw` option above. The end result should look similar to this:

```
TLSCipherSuite HIGH:MEDIUM:+SSLv3
TLSCertificateFile /usr/local/etc/openldap/server.crt
TLSCertificateKeyFile /usr/local/etc/openldap/private/server.key
TLSCACertificateFile /usr/local/etc/openldap/ca.crt
rootpw  {SHA}W6ph5Mm5Pz8GgiULbPgzG37mj9g=
```

Finally, enable the **OpenLDAP** service in `rc.conf`. At this time, setting up a URI and providing the group and user to run as may be useful. Edit `/etc/rc.conf` and add the following lines:

```
slapd_enable="YES"
slapd_flags="-4 -h ldaps:///"
```

At this point the server should be ready to be brought up and tested. To perform this task, issue the following command:

```
# service slapd start
```

If everything was configured correctly, a search of the directory should show a successful connection with a single response as in this example:

```
# ldapsearch -Z

# extended LDIF
#
# LDAPv3
# base <dc=example,dc=com> (default) with scope subtree
# filter: (objectclass=*)
# requesting: ALL
#

# search result
search: 3
result: 32 No such object

# numResponses: 1
```

Considering the service should now be responding, as it is above, the directory may be populated using the `ldapadd` command. In this example, there is a file containing a list of users to be added to this particular directory. First, create a file to be imported with the following dataset:

```
dn: dc=example,dc=com
objectclass: dcObject
objectclass: organization
o: Example
dc: Example

dn: cn=Manager,dc=example,dc=com
objectclass: organizationalRole
```

```
cn: Manager
```

> **Note:** To debug any of the following, stop the `slapd` service using the `service` command and start it using with debugging options. To accomplish this, issue the following command:
>
> ```
> # /usr/local/libexec/slapd -d -1
> ```

To import this datafile, issue the following command, assuming the file is `import.ldif`:

```
# ldapadd -Z -D "cn=Manager,dc=example,dc=com" -W -f import.ldif
```

There will be a request for the password specified earlier, and the output should look like this:

```
Enter LDAP Password:
adding new entry "dc=example,dc=com"

adding new entry "cn=Manager,dc=example,dc=com"
```

Verify the data was added by issuing a search on the server using `ldapsearch`. In this case the output should look like this:

```
% ldapsearch -Z

# extended LDIF
#
# LDAPv3
# base <dc=example,dc=com> (default) with scope subtree
# filter: (objectclass=*)
# requesting: ALL
#

# example.com
dn: dc=example,dc=com
objectClass: dcObject
objectClass: organization
o: Example
dc: Example

# Manager, example.com
dn: cn=Manager,dc=example,dc=com
objectClass: organizationalRole
cn: Manager

# search result
search: 3
result: 0 Success

# numResponses: 3
# numEntries: 2
```

It is of course advisable to read about the structure of LDAP directories and the various manual pages mentioned in this section. At this point, the server should be configured and functioning properly.

30.6 Automatic Network Configuration (DHCP)

Written by Greg Sutter.

30.6.1 What Is DHCP?

DHCP, the Dynamic Host Configuration Protocol, describes the means by which a system can connect to a network and obtain the necessary information for communication upon that network. FreeBSD uses the OpenBSD `dhclient` taken from OpenBSD 3.7. All information here regarding `dhclient` is for use with either of the ISC or OpenBSD DHCP clients. The DHCP server is the one included in the ISC distribution.

30.6.2 What This Section Covers

This section describes both the client-side components of the ISC and OpenBSD DHCP client and server-side components of the ISC DHCP system. The client-side program, `dhclient`, comes integrated within FreeBSD, and the server-side portion is available from the `net/isc-dhcp42-server` port. The dhclient(8), dhcp-options(5), and dhclient.conf(5) manual pages, in addition to the references below, are useful resources.

30.6.3 How It Works

When `dhclient`, the DHCP client, is executed on the client machine, it begins broadcasting requests for configuration information. By default, these requests are on UDP port 68. The server replies on UDP 67, giving the client an IP address and other relevant network information such as netmask, router, and DNS servers. All of this information comes in the form of a DHCP “lease” and is only valid for a certain time (configured by the DHCP server maintainer). In this manner, stale IP addresses for clients no longer connected to the network can be automatically reclaimed.

DHCP clients can obtain a great deal of information from the server. An exhaustive list may be found in dhcp-options(5).

30.6.4 FreeBSD Integration

FreeBSD fully integrates the OpenBSD DHCP client, `dhclient`. DHCP client support is provided within both the installer and the base system, obviating the need for detailed knowledge of network configurations on any network that runs a DHCP server.

DHCP is supported by **sysinstall**. When configuring a network interface within **sysinstall**, the second question asked is: “Do you want to try DHCP configuration of the interface?”. Answering affirmatively will execute `dhclient`, and if successful, will fill in the network configuration information automatically.

There are two things required to have the system use DHCP upon startup:

- Make sure that the `bpf` device is compiled into the kernel. To do this, add `device bpf` to the kernel configuration file, and rebuild the kernel. For more information about building kernels, see Chapter 9.

  The `bpf` device is already part of the `GENERIC` kernel that is supplied with FreeBSD, thus there is no need to build a custom kernel for DHCP. In the case of a custom kernel configuration file, this device must be present for DHCP to function properly.

  > **Note:** For those who are particularly security conscious, take note that `bpf` is also the device that allows packet sniffers to work correctly (although they still have to be run as `root`). `bpf` *is* required to use DHCP; however, the security sensitive types should probably not add `bpf` to the kernel in the expectation that at some point in the future the system will be using DHCP.

- By default, DHCP configuration on FreeBSD runs in the background, or *asynchronously*. Other startup scripts continue to run while DHCP completes, speeding up system startup.

  Background DHCP works well when the DHCP server responds quickly to requests and the DHCP configuration process goes quickly. However, DHCP may take a long time to complete on some systems. If network services attempt to run before DHCP has completed, they will fail. Using DHCP in *synchronous* mode prevents the problem, pausing startup until DHCP configuration has completed.

  To connect to a DHCP server in the background while other startup continues (asynchronous mode), use the "`DHCP`" value in `/etc/rc.conf`:

  ```
  ifconfig_fxp0="DHCP"
  ```

  To pause startup while DHCP completes, use synchronous mode with the "`SYNCDHCP`" value:

  ```
  ifconfig_fxp0="SYNCDHCP"
  ```

  > **Note:** Replace the *`fxp0`* shown in these examples with the name of the interface to be dynamically configured, as described in Section 12.8.

  When using a different file system location for `dhclient`, or if additional flags must be passed to `dhclient`, include (editing as necessary):

  ```
  dhclient_program="/sbin/dhclient"
  dhclient_flags=""
  ```

The DHCP server, **dhcpd**, is included as part of the `net/isc-dhcp42-server` port in the ports collection. This port contains the ISC DHCP server and documentation.

30.6.5 Files

- `/etc/dhclient.conf`

  `dhclient` requires a configuration file, `/etc/dhclient.conf`. Typically the file contains only comments, the defaults being reasonably sane. This configuration file is described by the dhclient.conf(5) manual page.

- `/sbin/dhclient`

`dhclient` is statically linked and resides in `/sbin`. The dhclient(8) manual page gives more information about `dhclient`.

- `/sbin/dhclient-script`

  `dhclient-script` is the FreeBSD-specific DHCP client configuration script. It is described in dhclient-script(8), but should not need any user modification to function properly.

- `/var/db/dhclient.leases.`*`interface`*

  The DHCP client keeps a database of valid leases in this file, which is written as a log. dhclient.leases(5) gives a slightly longer description.

30.6.6 Further Reading

The DHCP protocol is fully described in RFC 2131 (http://www.freesoft.org/CIE/RFC/2131/). An informational resource has also been set up at http://www.dhcp.org/.

30.6.7 Installing and Configuring a DHCP Server

30.6.7.1 What This Section Covers

This section provides information on how to configure a FreeBSD system to act as a DHCP server using the ISC (Internet Systems Consortium) implementation of the DHCP server.

The server is not provided as part of FreeBSD, and so the `net/isc-dhcp42-server` port must be installed to provide this service. See Chapter 5 for more information on using the Ports Collection.

30.6.7.2 DHCP Server Installation

In order to configure the FreeBSD system as a DHCP server, first ensure that the bpf(4) device is compiled into the kernel. To do this, add `device bpf` to the kernel configuration file, and rebuild the kernel. For more information about building kernels, see Chapter 9.

The `bpf` device is already part of the `GENERIC` kernel that is supplied with FreeBSD, so there is no need to create a custom kernel in order to get DHCP working.

> **Note:** Those who are particularly security conscious should note that `bpf` is also the device that allows packet sniffers to function correctly (although such programs still need privileged access). The `bpf` device *is* required to use DHCP, but if the sensitivity of the system's security is high, this device should not be included in the kernel purely because the use of DHCP may, at some point in the future, be desired.

The next thing that is needed is to edit the sample `dhcpd.conf` which was installed by the `net/isc-dhcp42-server` port. By default, this will be `/usr/local/etc/dhcpd.conf.sample`, and you should copy this to `/usr/local/etc/dhcpd.conf` before proceeding to make changes.

30.6.7.3 Configuring the DHCP Server

`dhcpd.conf` is comprised of declarations regarding subnets and hosts, and is perhaps most easily explained using an example :

```
option domain-name "example.com";❶
option domain-name-servers 192.168.4.100;❷
option subnet-mask 255.255.255.0;❸

default-lease-time 3600;❹
max-lease-time 86400;❺
ddns-update-style none;❻

subnet 192.168.4.0 netmask 255.255.255.0 {
  range 192.168.4.129 192.168.4.254;❼
  option routers 192.168.4.1;❽
}

host mailhost {
  hardware ethernet 02:03:04:05:06:07;❾
  fixed-address mailhost.example.com;(10)
}
```

❶ This option specifies the domain that will be provided to clients as the default search domain. See resolv.conf(5) for more information on what this means.

❷ This option specifies a comma separated list of DNS servers that the client should use.

❸ The netmask that will be provided to clients.

❹ A client may request a specific length of time that a lease will be valid. Otherwise the server will assign a lease with this expiry value (in seconds).

❺ This is the maximum length of time that the server will lease for. Should a client request a longer lease, a lease will be issued, although it will only be valid for `max-lease-time` seconds.

❻ This option specifies whether the DHCP server should attempt to update DNS when a lease is accepted or released. In the ISC implementation, this option is *required*.

❼ This denotes which IP addresses should be used in the pool reserved for allocating to clients. IP addresses between, and including, the ones stated are handed out to clients.

❽ Declares the default gateway that will be provided to clients.

❾ The hardware MAC address of a host (so that the DHCP server can recognize a host when it makes a request).

(10) Specifies that the host should always be given the same IP address. Note that using a hostname is correct here, since the DHCP server will resolve the hostname itself before returning the lease information.

Once the configuration of `dhcpd.conf` has been completed, enable the DHCP server in `/etc/rc.conf`, i.e., by adding:

```
dhcpd_enable="YES"
dhcpd_ifaces="dc0"
```

Replace the `dc0` interface name with the interface (or interfaces, separated by whitespace) that the DHCP server should listen on for DHCP client requests.

Proceed to start the server by issuing the following command:

```
# service isc-dhcpd start
```

Any future changes to the configuration of the server will require the sending of a `SIGTERM` signal to **dhcpd** rather than a `SIGHUP`. It is definitely more simple to use service(8) to completely restart the service.

30.6.7.4 Files

- `/usr/local/sbin/dhcpd`

  dhcpd is statically linked and resides in `/usr/local/sbin`. The dhcpd(8) manual page installed with the port gives more information about **dhcpd**.

- `/usr/local/etc/dhcpd.conf`

  dhcpd requires a configuration file, `/usr/local/etc/dhcpd.conf` before it will start providing service to clients. This file needs to contain all the information that should be provided to clients that are being serviced, along with information regarding the operation of the server. This configuration file is described by the dhcpd.conf(5) manual page installed by the port.

- `/var/db/dhcpd.leases`

  The DHCP server keeps a database of leases it has issued in this file, which is written as a log. The manual page dhcpd.leases(5), installed by the port gives a slightly longer description.

- `/usr/local/sbin/dhcrelay`

  dhcrelay is used in advanced environments where one DHCP server forwards a request from a client to another DHCP server on a separate network. If this functionality is required, then install the `net/isc-dhcp42-relay` port. The dhcrelay(8) manual page provided with the port contains more detail.

30.7 Domain Name System (DNS)

Contributed by Chern Lee, Tom Rhodes, and Daniel Gerzo.

30.7.1 Overview

FreeBSD utilizes, by default, a version of BIND (Berkeley Internet Name Domain), which is the most common implementation of the DNS protocol. DNS is the protocol through which names are mapped to IP addresses, and vice versa. For example, a query for `www.FreeBSD.org` will receive a reply with the IP address of The FreeBSD Project's web server, whereas, a query for `ftp.FreeBSD.org` will return the IP address of the corresponding FTP machine. Likewise, the opposite can happen. A query for an IP address can resolve its hostname. It is not necessary to run a name server to perform DNS lookups on a system.

FreeBSD currently comes with BIND9 DNS server software by default. Our installation provides enhanced security features, a new file system layout and automated chroot(8) configuration.

DNS is coordinated across the Internet through a somewhat complex system of authoritative root, Top Level Domain (TLD), and other smaller-scale name servers which host and cache individual domain information.

Currently, BIND is maintained by the Internet Systems Consortium https://www.isc.org/.

30.7.2 Terminology

To understand this document, some terms related to DNS must be understood.

Term	Definition
Forward DNS	Mapping of hostnames to IP addresses.
Origin	Refers to the domain covered in a particular zone file.
named, BIND	Common names for the BIND name server package within FreeBSD.
Resolver	A system process through which a machine queries a name server for zone information.
Reverse DNS	Mapping of IP addresses to hostnames.
Root zone	The beginning of the Internet zone hierarchy. All zones fall under the root zone, similar to how all files in a file system fall under the root directory.
Zone	An individual domain, subdomain, or portion of the DNS administered by the same authority.

Examples of zones:

- `.` is how the root zone is usually referred to in documentation.
- `org.` is a Top Level Domain (TLD) under the root zone.
- `example.org.` is a zone under the `org.` TLD.
- `1.168.192.in-addr.arpa` is a zone referencing all IP addresses which fall under the `192.168.1.*` IP address space.

As one can see, the more specific part of a hostname appears to its left. For example, `example.org.` is more specific than `org.`, as `org.` is more specific than the root zone. The layout of each part of a hostname is much like a file system: the `/dev` directory falls within the root, and so on.

30.7.3 Reasons to Run a Name Server

Name servers generally come in two forms: authoritative name servers, and caching (also known as resolving) name servers.

An authoritative name server is needed when:

- One wants to serve DNS information to the world, replying authoritatively to queries.
- A domain, such as `example.org`, is registered and IP addresses need to be assigned to hostnames under it.
- An IP address block requires reverse DNS entries (IP to hostname).
- A backup or second name server, called a slave, will reply to queries.

A caching name server is needed when:

- A local DNS server may cache and respond more quickly than querying an outside name server.

When one queries for `www.FreeBSD.org`, the resolver usually queries the uplink ISP's name server, and retrieves the reply. With a local, caching DNS server, the query only has to be made once to the outside world by the caching DNS server. Additional queries will not have to go outside the local network, since the information is cached locally.

30.7.4 How It Works

In FreeBSD, the BIND daemon is called **named**.

File	Description
named(8)	The BIND daemon.
rndc(8)	Name server control utility.
`/etc/namedb`	Directory where BIND zone information resides.
`/etc/namedb/named.conf`	Configuration file of the daemon.

Depending on how a given zone is configured on the server, the files related to that zone can be found in the `master`, `slave`, or `dynamic` subdirectories of the `/etc/namedb` directory. These files contain the DNS information that will be given out by the name server in response to queries.

30.7.5 Starting BIND

Since BIND is installed by default, configuring it is relatively simple.

The default **named** configuration is that of a basic resolving name server, running in a chroot(8) environment, and restricted to listening on the local IPv4 loopback address (127.0.0.1). To start the server one time with this configuration, use the following command:

```
# service named onestart
```

To ensure the **named** daemon is started at boot each time, put the following line into the `/etc/rc.conf`:

```
named_enable="YES"
```

There are obviously many configuration options for `/etc/namedb/named.conf` that are beyond the scope of this document. There are other startup options for **named** on FreeBSD, take a look at the `named_*` flags in `/etc/defaults/rc.conf` and consult the rc.conf(5) manual page. The Section 12.7 section is also a good read.

30.7.6 Configuration Files

Configuration files for **named** currently reside in `/etc/namedb` directory and will need modification before use unless all that is needed is a simple resolver. This is where most of the configuration will be performed.

30.7.6.1 `/etc/namedb/named.conf`

```
// $FreeBSD$
//
// Refer to the named.conf(5) and named(8) man pages, and the documentation
// in /usr/share/doc/bind9 for more details.
```

```
//
// If you are going to set up an authoritative server, make sure you
// understand the hairy details of how DNS works.  Even with
// simple mistakes, you can break connectivity for affected parties,
// or cause huge amounts of useless Internet traffic.

options {
       // All file and path names are relative to the chroot directory,
       // if any, and should be fully qualified.
       directory       "/etc/namedb/working";
       pid-file        "/var/run/named/pid";
       dump-file       "/var/dump/named_dump.db";
       statistics-file "/var/stats/named.stats";

// If named is being used only as a local resolver, this is a safe default.
// For named to be accessible to the network, comment this option, specify
// the proper IP address, or delete this option.
       listen-on       { 127.0.0.1; };

// If you have IPv6 enabled on this system, uncomment this option for
// use as a local resolver.  To give access to the network, specify
// an IPv6 address, or the keyword "any".
//     listen-on-v6    { ::1; };

// These zones are already covered by the empty zones listed below.
// If you remove the related empty zones below, comment these lines out.
       disable-empty-zone "255.255.255.255.IN-ADDR.ARPA";
       disable-empty-zone "0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.IP6.AR
       disable-empty-zone "1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.IP6.AR

// If you've got a DNS server around at your upstream provider, enter
// its IP address here, and enable the line below.  This will make you
// benefit from its cache, thus reduce overall DNS traffic in the Internet.
/*
       forwarders {
               127.0.0.1;
       };
*/

// If the 'forwarders' clause is not empty the default is to 'forward first'
// which will fall back to sending a query from your local server if the name
// servers in 'forwarders' do not have the answer.  Alternatively you can
// force your name server to never initiate queries of its own by enabling the
// following line:
//     forward only;

// If you wish to have forwarding configured automatically based on
// the entries in /etc/resolv.conf, uncomment the following line and
// set named_auto_forward=yes in /etc/rc.conf.  You can also enable
// named_auto_forward_only (the effect of which is described above).
//     include "/etc/namedb/auto_forward.conf";
```

Just as the comment says, to benefit from an uplink's cache, `forwarders` can be enabled here. Under normal circumstances, a name server will recursively query the Internet looking at certain name servers until it finds the answer it is looking for. Having this enabled will have it query the uplink's name server (or name server provided) first, taking advantage of its cache. If the uplink name server in question is a heavily trafficked, fast name server, enabling this may be worthwhile.

Warning: `127.0.0.1` will *not* work here. Change this IP address to a name server at the uplink.

```
        /*
          Modern versions of BIND use a random UDP port for each outgoing
          query by default in order to dramatically reduce the possibility
          of cache poisoning.  All users are strongly encouraged to utilize
          this feature, and to configure their firewalls to accommodate it.

          AS A LAST RESORT in order to get around a restrictive firewall
          policy you can try enabling the option below.  Use of this option
          will significantly reduce your ability to withstand cache poisoning
          attacks, and should be avoided if at all possible.

          Replace NNNNN in the example with a number between 49160 and 65530.
      */
      // query-source address * port NNNNN;
};

// If you enable a local name server, don't forget to enter 127.0.0.1
// first in your /etc/resolv.conf so this server will be queried.
// Also, make sure to enable it in /etc/rc.conf.

// The traditional root hints mechanism. Use this, OR the slave zones below.
zone "." { type hint; file "/etc/namedb/named.root"; };

/*     Slaving the following zones from the root name servers has some
       significant advantages:
       1. Faster local resolution for your users
       2. No spurious traffic will be sent from your network to the roots
       3. Greater resilience to any potential root server failure/DDoS

       On the other hand, this method requires more monitoring than the
       hints file to be sure that an unexpected failure mode has not
       incapacitated your server.  Name servers that are serving a lot
       of clients will benefit more from this approach than individual
       hosts.  Use with caution.

       To use this mechanism, uncomment the entries below, and comment
       the hint zone above.

       As documented at http://dns.icann.org/services/axfr/ these zones:
       "." (the root), ARPA, IN-ADDR.ARPA, IP6.ARPA, and ROOT-SERVERS.NET
       are available for AXFR from these servers on IPv4 and IPv6:
       xfr.lax.dns.icann.org, xfr.cjr.dns.icann.org
*/
```

```
/*
zone "." {
       type slave;
       file "/etc/namedb/slave/root.slave";
       masters {
               192.5.5.241;    // F.ROOT-SERVERS.NET.
       };
       notify no;
};
zone "arpa" {
       type slave;
       file "/etc/namedb/slave/arpa.slave";
       masters {
               192.5.5.241;    // F.ROOT-SERVERS.NET.
       };
       notify no;
};
*/

/*     Serving the following zones locally will prevent any queries
       for these zones leaving your network and going to the root
       name servers.  This has two significant advantages:
       1. Faster local resolution for your users
       2. No spurious traffic will be sent from your network to the roots
*/
// RFCs 1912 and 5735 (and BCP 32 for localhost)
zone "localhost"       { type master; file "/etc/namedb/master/localhost-forward.db"; };
zone "127.in-addr.arpa"        { type master; file "/etc/namedb/master/localhost-reverse.db"; };
zone "255.in-addr.arpa"        { type master; file "/etc/namedb/master/empty.db"; };

// RFC 1912-style zone for IPv6 localhost address
zone "0.ip6.arpa"      { type master; file "/etc/namedb/master/localhost-reverse.db"; };

// "This" Network (RFCs 1912 and 5735)
zone "0.in-addr.arpa"  { type master; file "/etc/namedb/master/empty.db"; };

// Private Use Networks (RFCs 1918 and 5735)
zone "10.in-addr.arpa"    { type master; file "/etc/namedb/master/empty.db"; };
zone "16.172.in-addr.arpa" { type master; file "/etc/namedb/master/empty.db"; };
zone "17.172.in-addr.arpa" { type master; file "/etc/namedb/master/empty.db"; };
zone "18.172.in-addr.arpa" { type master; file "/etc/namedb/master/empty.db"; };
zone "19.172.in-addr.arpa" { type master; file "/etc/namedb/master/empty.db"; };
zone "20.172.in-addr.arpa" { type master; file "/etc/namedb/master/empty.db"; };
zone "21.172.in-addr.arpa" { type master; file "/etc/namedb/master/empty.db"; };
zone "22.172.in-addr.arpa" { type master; file "/etc/namedb/master/empty.db"; };
zone "23.172.in-addr.arpa" { type master; file "/etc/namedb/master/empty.db"; };
zone "24.172.in-addr.arpa" { type master; file "/etc/namedb/master/empty.db"; };
zone "25.172.in-addr.arpa" { type master; file "/etc/namedb/master/empty.db"; };
zone "26.172.in-addr.arpa" { type master; file "/etc/namedb/master/empty.db"; };
zone "27.172.in-addr.arpa" { type master; file "/etc/namedb/master/empty.db"; };
zone "28.172.in-addr.arpa" { type master; file "/etc/namedb/master/empty.db"; };
zone "29.172.in-addr.arpa" { type master; file "/etc/namedb/master/empty.db"; };
zone "30.172.in-addr.arpa" { type master; file "/etc/namedb/master/empty.db"; };
```

```
zone "31.172.in-addr.arpa" { type master; file "/etc/namedb/master/empty.db"; };
zone "168.192.in-addr.arpa" { type master; file "/etc/namedb/master/empty.db"; };

// Link-local/APIPA (RFCs 3927 and 5735)
zone "254.169.in-addr.arpa" { type master; file "/etc/namedb/master/empty.db"; };

// IETF protocol assignments (RFCs 5735 and 5736)
zone "0.0.192.in-addr.arpa" { type master; file "/etc/namedb/master/empty.db"; };

// TEST-NET-[1-3] for Documentation (RFCs 5735 and 5737)
zone "2.0.192.in-addr.arpa" { type master; file "/etc/namedb/master/empty.db"; };
zone "100.51.198.in-addr.arpa" { type master; file "/etc/namedb/master/empty.db"; };
zone "113.0.203.in-addr.arpa" { type master; file "/etc/namedb/master/empty.db"; };

// IPv6 Range for Documentation (RFC 3849)
zone "8.b.d.0.1.0.0.2.ip6.arpa" { type master; file "/etc/namedb/master/empty.db"; };

// Domain Names for Documentation and Testing (BCP 32)
zone "test" { type master; file "/etc/namedb/master/empty.db"; };
zone "example" { type master; file "/etc/namedb/master/empty.db"; };
zone "invalid" { type master; file "/etc/namedb/master/empty.db"; };
zone "example.com" { type master; file "/etc/namedb/master/empty.db"; };
zone "example.net" { type master; file "/etc/namedb/master/empty.db"; };
zone "example.org" { type master; file "/etc/namedb/master/empty.db"; };

// Router Benchmark Testing (RFCs 2544 and 5735)
zone "18.198.in-addr.arpa" { type master; file "/etc/namedb/master/empty.db"; };
zone "19.198.in-addr.arpa" { type master; file "/etc/namedb/master/empty.db"; };

// IANA Reserved - Old Class E Space (RFC 5735)
zone "240.in-addr.arpa"       { type master; file "/etc/namedb/master/empty.db"; };
zone "241.in-addr.arpa"       { type master; file "/etc/namedb/master/empty.db"; };
zone "242.in-addr.arpa"       { type master; file "/etc/namedb/master/empty.db"; };
zone "243.in-addr.arpa"       { type master; file "/etc/namedb/master/empty.db"; };
zone "244.in-addr.arpa"       { type master; file "/etc/namedb/master/empty.db"; };
zone "245.in-addr.arpa"       { type master; file "/etc/namedb/master/empty.db"; };
zone "246.in-addr.arpa"       { type master; file "/etc/namedb/master/empty.db"; };
zone "247.in-addr.arpa"       { type master; file "/etc/namedb/master/empty.db"; };
zone "248.in-addr.arpa"       { type master; file "/etc/namedb/master/empty.db"; };
zone "249.in-addr.arpa"       { type master; file "/etc/namedb/master/empty.db"; };
zone "250.in-addr.arpa"       { type master; file "/etc/namedb/master/empty.db"; };
zone "251.in-addr.arpa"       { type master; file "/etc/namedb/master/empty.db"; };
zone "252.in-addr.arpa"       { type master; file "/etc/namedb/master/empty.db"; };
zone "253.in-addr.arpa"       { type master; file "/etc/namedb/master/empty.db"; };
zone "254.in-addr.arpa"       { type master; file "/etc/namedb/master/empty.db"; };

// IPv6 Unassigned Addresses (RFC 4291)
zone "1.ip6.arpa"     { type master; file "/etc/namedb/master/empty.db"; };
zone "3.ip6.arpa"     { type master; file "/etc/namedb/master/empty.db"; };
zone "4.ip6.arpa"     { type master; file "/etc/namedb/master/empty.db"; };
zone "5.ip6.arpa"     { type master; file "/etc/namedb/master/empty.db"; };
zone "6.ip6.arpa"     { type master; file "/etc/namedb/master/empty.db"; };
zone "7.ip6.arpa"     { type master; file "/etc/namedb/master/empty.db"; };
```

```
zone "8.ip6.arpa"      { type master; file "/etc/namedb/master/empty.db"; };
zone "9.ip6.arpa"      { type master; file "/etc/namedb/master/empty.db"; };
zone "a.ip6.arpa"      { type master; file "/etc/namedb/master/empty.db"; };
zone "b.ip6.arpa"      { type master; file "/etc/namedb/master/empty.db"; };
zone "c.ip6.arpa"      { type master; file "/etc/namedb/master/empty.db"; };
zone "d.ip6.arpa"      { type master; file "/etc/namedb/master/empty.db"; };
zone "e.ip6.arpa"      { type master; file "/etc/namedb/master/empty.db"; };
zone "0.f.ip6.arpa"    { type master; file "/etc/namedb/master/empty.db"; };
zone "1.f.ip6.arpa"    { type master; file "/etc/namedb/master/empty.db"; };
zone "2.f.ip6.arpa"    { type master; file "/etc/namedb/master/empty.db"; };
zone "3.f.ip6.arpa"    { type master; file "/etc/namedb/master/empty.db"; };
zone "4.f.ip6.arpa"    { type master; file "/etc/namedb/master/empty.db"; };
zone "5.f.ip6.arpa"    { type master; file "/etc/namedb/master/empty.db"; };
zone "6.f.ip6.arpa"    { type master; file "/etc/namedb/master/empty.db"; };
zone "7.f.ip6.arpa"    { type master; file "/etc/namedb/master/empty.db"; };
zone "8.f.ip6.arpa"    { type master; file "/etc/namedb/master/empty.db"; };
zone "9.f.ip6.arpa"    { type master; file "/etc/namedb/master/empty.db"; };
zone "a.f.ip6.arpa"    { type master; file "/etc/namedb/master/empty.db"; };
zone "b.f.ip6.arpa"    { type master; file "/etc/namedb/master/empty.db"; };
zone "0.e.f.ip6.arpa"  { type master; file "/etc/namedb/master/empty.db"; };
zone "1.e.f.ip6.arpa"  { type master; file "/etc/namedb/master/empty.db"; };
zone "2.e.f.ip6.arpa"  { type master; file "/etc/namedb/master/empty.db"; };
zone "3.e.f.ip6.arpa"  { type master; file "/etc/namedb/master/empty.db"; };
zone "4.e.f.ip6.arpa"  { type master; file "/etc/namedb/master/empty.db"; };
zone "5.e.f.ip6.arpa"  { type master; file "/etc/namedb/master/empty.db"; };
zone "6.e.f.ip6.arpa"  { type master; file "/etc/namedb/master/empty.db"; };
zone "7.e.f.ip6.arpa"  { type master; file "/etc/namedb/master/empty.db"; };

// IPv6 ULA (RFC 4193)
zone "c.f.ip6.arpa"    { type master; file "/etc/namedb/master/empty.db"; };
zone "d.f.ip6.arpa"    { type master; file "/etc/namedb/master/empty.db"; };

// IPv6 Link Local (RFC 4291)
zone "8.e.f.ip6.arpa"  { type master; file "/etc/namedb/master/empty.db"; };
zone "9.e.f.ip6.arpa"  { type master; file "/etc/namedb/master/empty.db"; };
zone "a.e.f.ip6.arpa"  { type master; file "/etc/namedb/master/empty.db"; };
zone "b.e.f.ip6.arpa"  { type master; file "/etc/namedb/master/empty.db"; };

// IPv6 Deprecated Site-Local Addresses (RFC 3879)
zone "c.e.f.ip6.arpa"  { type master; file "/etc/namedb/master/empty.db"; };
zone "d.e.f.ip6.arpa"  { type master; file "/etc/namedb/master/empty.db"; };
zone "e.e.f.ip6.arpa"  { type master; file "/etc/namedb/master/empty.db"; };
zone "f.e.f.ip6.arpa"  { type master; file "/etc/namedb/master/empty.db"; };

// IP6.INT is Deprecated (RFC 4159)
zone "ip6.int"         { type master; file "/etc/namedb/master/empty.db"; };

// NB: Do not use the IP addresses below, they are faked, and only
// serve demonstration/documentation purposes!
//
// Example slave zone config entries.  It can be convenient to become
// a slave at least for the zone your own domain is in.  Ask
// your network administrator for the IP address of the responsible
```

```
// master name server.
//
// Do not forget to include the reverse lookup zone!
// This is named after the first bytes of the IP address, in reverse
// order, with ".IN-ADDR.ARPA" appended, or ".IP6.ARPA" for IPv6.
//
// Before starting to set up a master zone, make sure you fully
// understand how DNS and BIND work.  There are sometimes
// non-obvious pitfalls.  Setting up a slave zone is usually simpler.
//
// NB: Don't blindly enable the examples below. :-)  Use actual names
// and addresses instead.

/* An example dynamic zone
key "exampleorgkey" {
       algorithm hmac-md5;
       secret "sf87HJqjkqh8ac87a02lla==";
};
zone "example.org" {
       type master;
       allow-update {
               key "exampleorgkey";
       };
       file "/etc/namedb/dynamic/example.org";
};
*/

/* Example of a slave reverse zone
zone "1.168.192.in-addr.arpa" {
       type slave;
       file "/etc/namedb/slave/1.168.192.in-addr.arpa";
       masters {
               192.168.1.1;
       };
};
*/
```

In `named.conf`, these are examples of slave entries for a forward and reverse zone.

For each new zone served, a new zone entry must be added to `named.conf`.

For example, the simplest zone entry for `example.org` can look like:

```
zone "example.org" {
       type master;
       file "master/example.org";
};
```

The zone is a master, as indicated by the `type` statement, holding its zone information in `/etc/namedb/master/example.org` indicated by the `file` statement.

```
zone "example.org" {
       type slave;
       file "slave/example.org";
```

```
};
```

In the slave case, the zone information is transferred from the master name server for the particular zone, and saved in the file specified. If and when the master server dies or is unreachable, the slave name server will have the transferred zone information and will be able to serve it.

30.7.6.2 Zone Files

An example master zone file for `example.org` (existing within `/etc/namedb/master/example.org`) is as follows:

```
$TTL 3600        ; 1 hour default TTL
example.org.    IN      SOA      ns1.example.org. admin.example.org. (
                                 2006051501      ; Serial
                                 10800           ; Refresh
                                 3600            ; Retry
                                 604800          ; Expire
                                 300             ; Negative Response TTL
                         )

; DNS Servers
                IN      NS       ns1.example.org.
                IN      NS       ns2.example.org.

; MX Records
                IN      MX 10    mx.example.org.
                IN      MX 20    mail.example.org.

                IN      A        192.168.1.1

; Machine Names
localhost       IN      A        127.0.0.1
ns1             IN      A        192.168.1.2
ns2             IN      A        192.168.1.3
mx              IN      A        192.168.1.4
mail            IN      A        192.168.1.5

; Aliases
www             IN      CNAME    example.org.
```

Note that every hostname ending in a "." is an exact hostname, whereas everything without a trailing "." is relative to the origin. For example, `ns1` is translated into `ns1.`*`example.org.`*

The format of a zone file follows:

```
recordname      IN recordtype   value
```

The most commonly used DNS records:

SOA

start of zone authority

NS

an authoritative name server

A

a host address

CNAME

the canonical name for an alias

MX

mail exchanger

PTR

a domain name pointer (used in reverse DNS)

```
example.org. IN SOA ns1.example.org. admin.example.org. (
                        2006051501      ; Serial
                        10800           ; Refresh after 3 hours
                        3600            ; Retry after 1 hour
                        604800          ; Expire after 1 week
                        300 )           ; Negative Response TTL
```

`example.org.`

the domain name, also the origin for this zone file.

`ns1.example.org.`

the primary/authoritative name server for this zone.

`admin.example.org.`

the responsible person for this zone, email address with "@" replaced. (<`admin@example.org`> becomes `admin.example.org`)

`2006051501`

the serial number of the file. This must be incremented each time the zone file is modified. Nowadays, many admins prefer a `yyyymmddrr` format for the serial number. `2006051501` would mean last modified 05/15/2006, the latter `01` being the first time the zone file has been modified this day. The serial number is important as it alerts slave name servers for a zone when it is updated.

```
    IN NS           ns1.example.org.
```

This is an NS entry. Every name server that is going to reply authoritatively for the zone must have one of these entries.

```
localhost       IN      A       127.0.0.1
ns1             IN      A       192.168.1.2
ns2             IN      A       192.168.1.3
mx              IN      A       192.168.1.4
```

```
mail            IN      A       192.168.1.5
```

The A record indicates machine names. As seen above, `ns1.example.org` would resolve to `192.168.1.2`.

```
                IN      A       192.168.1.1
```

This line assigns IP address `192.168.1.1` to the current origin, in this case `example.org`.

```
www             IN CNAME        @
```

The canonical name record is usually used for giving aliases to a machine. In the example, `www` is aliased to the "master" machine whose name happens to be the same as the domain name `example.org` (`192.168.1.1`). CNAMEs can never be used together with another kind of record for the same hostname.

```
               IN MX   10      mail.example.org.
```

The MX record indicates which mail servers are responsible for handling incoming mail for the zone. `mail.example.org` is the hostname of a mail server, and 10 is the priority of that mail server.

One can have several mail servers, with priorities of 10, 20 and so on. A mail server attempting to deliver to `example.org` would first try the highest priority MX (the record with the lowest priority number), then the second highest, etc, until the mail can be properly delivered.

For in-addr.arpa zone files (reverse DNS), the same format is used, except with PTR entries instead of A or CNAME.

```
$TTL 3600

1.168.192.in-addr.arpa. IN SOA ns1.example.org. admin.example.org. (
                        2006051501      ; Serial
                        10800           ; Refresh
                        3600            ; Retry
                        604800          ; Expire
                        300 )           ; Negative Response TTL

        IN      NS      ns1.example.org.
        IN      NS      ns2.example.org.

1       IN      PTR     example.org.
2       IN      PTR     ns1.example.org.
3       IN      PTR     ns2.example.org.
4       IN      PTR     mx.example.org.
5       IN      PTR     mail.example.org.
```

This file gives the proper IP address to hostname mappings for the above fictitious domain.

It is worth noting that all names on the right side of a PTR record need to be fully qualified (i.e., end in a ".").

30.7.7 Caching Name Server

A caching name server is a name server whose primary role is to resolve recursive queries. It simply asks queries of its own, and remembers the answers for later use.

30.7.8 DNSSEC

Domain Name System Security Extensions, or DNSSEC for short, is a suite of specifications to protect resolving name servers from forged DNS data, such as spoofed DNS records. By using digital signatures, a resolver can verify the integrity of the record. Note that DNSSEC only provides integrity via digitally signing the Resource Records (RRs). It provides neither confidentiality nor protection against false end-user assumptions. This means that it cannot protect against people going to `example.net` instead of `example.com`. The only thing DNSSEC does is authenticate that the data has not been compromised in transit. The security of DNS is an important step in securing the Internet in general. For more in-depth details of how DNSSEC works, the relevant RFCs are a good place to start. See the list in Section 30.7.10.

The following sections will demonstrate how to enable DNSSEC for an authoritative DNS server and a recursive (or caching) DNS server running BIND 9. While all versions of BIND 9 support DNSSEC, it is necessary to have at least version 9.6.2 in order to be able to use the signed root zone when validating DNS queries. This is because earlier versions lack the required algorithms to enable validation using the root zone key. It is strongly recommended to use the latest version of BIND 9.7 or later to take advantage of automatic key updating for the root key, as well as other features to automatically keep zones signed and signatures up to date. Where configurations differ between 9.6.2 and 9.7 and later, differences will be pointed out.

30.7.8.1 Recursive DNS Server Configuration

Enabling DNSSEC validation of queries performed by a recursive DNS server requires a few changes to `named.conf`. Before making these changes the root zone key, or trust anchor, must be acquired. Currently the root zone key is not available in a file format BIND understands, so it has to be manually converted into the proper format. The key itself can be obtained by querying the root zone for it using **dig**. By running

```
% dig +multi +noall +answer DNSKEY . > root.dnskey
```

the key will end up in `root.dnskey`. The contents should look something like this:

```
. 93910 IN DNSKEY 257 3 8 (
       AwEAAagAIKlVZrpC6Ia7gEzahOR+9W29euxhJhVVLOyQ
       bSEW0O8gcCjFFVQUTf6v58fLjwBd0YI0EzrAcQqBGCzh
       /RStIoO8g0NfnfL2MTJRkxoXbfDaUeVPQuYEhg37NZWA
       JQ9VnMVDxP/VHL496M/QZxkjf5/Efucp2gaDX6RS6CXp
       oY68LsvPVjR0ZSwzz1apAzvN9dlzEheX7ICJBBtuA6G3
       LQpzW5hOA2hzCTMjJPJ8LbqF6dsV6DoBQzgul0sGIcGO
       Yl7OyQdXfZ57relSQageu+ipAdTTJ25AsRTAoub8ONGc
       LmqrAmRLKBP1dfwhYB4N7knNnulqQxA+Uk1ihz0=
       ) ; key id = 19036
. 93910 IN DNSKEY 256 3 8 (
       AwEAAcaGQEA+OJmOzfzVfoYN249JId7gx+OZMbxy69Hf
       UyuGBbRN0+HuTOpBxxBCkNOL+EJB9qJxt+0FEY6ZUVjE
       g58sRr4ZQ6Iu6b1xTBKgc193zUARk4mmQ/PPGxn7Cn5V
       EGJ/1h6dNaiXuRHwR+7oWh7DnzkIJChcTqlFrXDW3tjt
) ; key id = 34525
```

Do not be alarmed if the obtained keys differ from this example. They might have changed since these instructions were last updated. This output actually contains two keys. The first key in the listing, with the value 257 after the DNSKEY record type, is the one needed. This value indicates that this is a Secure Entry Point (SEP), commonly known as a Key Signing Key (KSK). The second key, with value 256, is a subordinate key, commonly called a Zone Signing Key (ZSK). More on the different key types later in Section 30.7.8.2.

Now the key must be verified and formatted so that BIND can use it. To verify the key, generate a DS RR set. Create a file containing these RRs with

```
% dnssec-dsfromkey -f root-dnskey . > root.ds
```

These records use SHA-1 and SHA-256 respectively, and should look similar to the following example, where the longer is using SHA-256.

```
.  IN DS 19036 8 1
       B256BD09DC8DD59F0E0F0D8541B8328DD986DF6E
. IN DS 19036 8 2 49AAC11D7B6F6446702E54A1607371607A1A41855200FD2CE1CDDE32F24E8FB5
```

The SHA-256 RR can now be compared to the digest in https://data.iana.org/root-anchors/root-anchors.xml. To be absolutely sure that the key has not been tampered with the data in the XML file can be verified using the PGP signature in https://data.iana.org/root-anchors/root-anchors.asc.

Next, the key must be formatted properly. This differs a little between BIND versions 9.6.2 and 9.7 and later. In version 9.7 support was added to automatically track changes to the key and update it as necessary. This is done using `managed-keys` as seen in the example below. When using the older version, the key is added using a `trusted-keys` statement and updates must be done manually. For BIND 9.6.2 the format should look like:

```
trusted-keys {
       "." 257 3 8
       "AwEAAagAIKlVZrpC6Ia7gEzahOR+9W29euxhJhVVLOyQbSEW0O8gcCjF
       FVQUTf6v58fLjwBd0YI0EzrAcQqBGCzh/RStIoO8g0NfnfL2MTJRkxoX
       bfDaUeVPQuYEhg37NZWAJQ9VnMVDxP/VHL496M/QZxkjf5/Efucp2gaD
       X6RS6CXpoY68LsvPVjR0ZSwzz1apAzvN9dlzEheX7ICJBBtuA6G3LQpz
       W5hOA2hzCTMjJPJ8LbqF6dsV6DoBQzgul0sGIcGOYl7OyQdXfZ57relS
       Qageu+ipAdTTJ25AsRTAoub8ONGcLmqrAmRLKBP1dfwhYB4N7knNnulq
       QxA+Uk1ihz0=";
};
```

For 9.7 the format will instead be:

```
managed-keys {
       "." initial-key 257 3 8
       "AwEAAagAIKlVZrpC6Ia7gEzahOR+9W29euxhJhVVLOyQbSEW0O8gcCjF
       FVQUTf6v58fLjwBd0YI0EzrAcQqBGCzh/RStIoO8g0NfnfL2MTJRkxoX
       bfDaUeVPQuYEhg37NZWAJQ9VnMVDxP/VHL496M/QZxkjf5/Efucp2gaD
       X6RS6CXpoY68LsvPVjR0ZSwzz1apAzvN9dlzEheX7ICJBBtuA6G3LQpz
       W5hOA2hzCTMjJPJ8LbqF6dsV6DoBQzgul0sGIcGOYl7OyQdXfZ57relS
       Qageu+ipAdTTJ25AsRTAoub8ONGcLmqrAmRLKBP1dfwhYB4N7knNnulq
       QxA+Uk1ihz0=";
};
```

The root key can now be added to `named.conf` either directly or by including a file containing the key. After these steps, configure BIND to do DNSSEC validation on queries by editing `named.conf` and adding the following to the `options` directive:

```
dnssec-enable yes;
dnssec-validation yes;
```

To verify that it is actually working use **dig** to make a query for a signed zone using the resolver just configured. A successful reply will contain the AD flag to indicate the data was authenticated. Running a query such as

```
% dig @resolver +dnssec se ds
```

should return the DS RR for the .se zone. In the flags: section the AD flag should be set, as seen in:

```
...
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 3, AUTHORITY: 0, ADDITIONAL: 1
...
```

The resolver is now capable of authenticating DNS queries.

30.7.8.2 Authoritative DNS Server Configuration

In order to get an authoritative name server to serve a DNSSEC signed zone a little more work is required. A zone is signed using cryptographic keys which must be generated. It is possible to use only one key for this. The preferred method however is to have a strong well-protected Key Signing Key (KSK) that is not rotated very often and a Zone Signing Key (ZSK) that is rotated more frequently. Information on recommended operational practices can be found in RFC 4641: DNSSEC Operational Practices (http://tools.ietf.org/rfc/rfc4641.txt). Practices regarding the root zone can be found in DNSSEC Practice Statement for the Root Zone KSK operator (http://www.root-dnssec.org/wp-content/uploads/2010/06/icann-dps-00.txt) and DNSSEC Practice Statement for the Root Zone ZSK operator (http://www.root-dnssec.org/wp-content/uploads/2010/06/vrsn-dps-00.txt). The KSK is used to build a chain of authority to the data in need of validation and as such is also called a Secure Entry Point (SEP) key. A message digest of this key, called a Delegation Signer (DS) record, must be published in the parent zone to establish the trust chain. How this is accomplished depends on the parent zone owner. The ZSK is used to sign the zone, and only needs to be published there.

To enable DNSSEC for the example.com zone depicted in previous examples, the first step is to use **dnssec-keygen** to generate the KSK and ZSK key pair. This key pair can utilize different cryptographic algorithms. It is recommended to use RSA/SHA256 for the keys and 2048 bits key length should be enough. To generate the KSK for example.com, run

```
% dnssec-keygen -f KSK -a RSASHA256 -b 2048 -n ZONE example.com
```

and to generate the ZSK, run

```
% dnssec-keygen -a RSASHA256 -b 2048 -n ZONE example.com
```

dnssec-keygen outputs two files, the public and the private keys in files named similar to Kexample.com.+005+nnnnn.key (public) and Kexample.com.+005+nnnnn.private (private). The nnnnn part of the file name is a five digit key ID. Keep track of which key ID belongs to which key. This is especially important when having more than one key in a zone. It is also possible to rename the keys. For each KSK file do:

```
% mv Kexample.com.+005+nnnnn.key Kexample.com.+005+nnnnn.KSK.key
% mv Kexample.com.+005+nnnnn.private Kexample.com.+005+nnnnn.KSK.private
```

For the ZSK files, substitute KSK for ZSK as necessary. The files can now be included in the zone file, using the $include statement. It should look something like this:

```
$include Kexample.com.+005+nnnnn.KSK.key ; KSK
$include Kexample.com.+005+nnnnn.ZSK.key    ; ZSK
```

Finally, sign the zone and tell BIND to use the signed zone file. To sign a zone **dnssec-signzone** is used. The command to sign the zone `example.com`, located in `example.com.db` would look similar to

```
% dnssec-signzone -o
 example.com -k Kexample.com.+005+nnnnn.KSK example.com.db
 Kexample.com.+005+nnnnn.ZSK.key
```

The key supplied to the `-k` argument is the KSK and the other key file is the ZSK that should be used in the signing. It is possible to supply more than one KSK and ZSK, which will result in the zone being signed with all supplied keys. This can be needed to supply zone data signed using more than one algorithm. The output of **dnssec-signzone** is a zone file with all RRs signed. This output will end up in a file with the extension `.signed`, such as `example.com.db.signed`. The DS records will also be written to a separate file `dsset-example.com`. To use this signed zone just modify the zone directive in `named.conf` to use `example.com.db.signed`. By default, the signatures are only valid 30 days, meaning that the zone needs to be resigned in about 15 days to be sure that resolvers are not caching records with stale signatures. It is possible to make a script and a cron job to do this. See relevant manuals for details.

Be sure to keep private keys confidential, as with all cryptographic keys. When changing a key it is best to include the new key into the zone, while still signing with the old one, and then move over to using the new key to sign. After these steps are done the old key can be removed from the zone. Failure to do this might render the DNS data unavailable for a time, until the new key has propagated through the DNS hierarchy. For more information on key rollovers and other DNSSEC operational issues, see RFC 4641: DNSSEC Operational practices (http://www.ietf.org/rfc/rfc4641.txt).

30.7.8.3 Automation Using BIND 9.7 or Later

Beginning with BIND version 9.7 a new feature called *Smart Signing* was introduced. This feature aims to make the key management and signing process simpler by automating parts of the task. By putting the keys into a directory called a *key repository*, and using the new option `auto-dnssec`, it is possible to create a dynamic zone which will be resigned as needed. To update this zone use **nsupdate** with the new option `-l`. **rndc** has also grown the ability to sign zones with keys in the key repository, using the option `sign`. To tell BIND to use this automatic signing and zone updating for `example.com`, add the following to `named.conf`:

```
zone example.com {
       type master;
       key-directory "/etc/named/keys";
       update-policy local;
       auto-dnssec maintain;
       file "/etc/named/dynamic/example.com.zone";
};
```

After making these changes, generate keys for the zone as explained in Section 30.7.8.2, put those keys in the key repository given as the argument to the `key-directory` in the zone configuration and the zone will be signed automatically. Updates to a zone configured this way must be done using **nsupdate**, which will take care of re-signing the zone with the new data added. For further details, see Section 30.7.10 and the BIND documentation.

30.7.9 Security

Although BIND is the most common implementation of DNS, there is always the issue of security. Possible and exploitable security holes are sometimes found.

While FreeBSD automatically drops **named** into a chroot(8) environment; there are several other security mechanisms in place which could help to lure off possible DNS service attacks.

It is always good idea to read CERT (http://www.cert.org/)'s security advisories and to subscribe to the FreeBSD security notifications mailing list (http://lists.FreeBSD.org/mailman/listinfo/freebsd-security-notifications) to stay up to date with the current Internet and FreeBSD security issues.

> **Tip:** If a problem arises, keeping sources up to date and having a fresh build of **named** may help.

30.7.10 Further Reading

BIND/**named** manual pages: rndc(8) named(8) named.conf(5) nsupdate(1) dnssec-signzone(8) dnssec-keygen(8)

- Official ISC BIND Page (https://www.isc.org/software/bind)
- Official ISC BIND Forum (https://www.isc.org/software/guild)
- O'Reilly DNS and BIND 5th Edition (http://www.oreilly.com/catalog/dns5/)
- Root DNSSEC (http://www.root-dnssec.org/documentation/)
- DNSSEC Trust Anchor Publication for the Root Zone (http://data.iana.org/root-anchors/draft-icann-dnssec-trust-anchor.html)
- RFC1034 - Domain Names - Concepts and Facilities (http://tools.ietf.org/html/rfc1034)
- RFC1035 - Domain Names - Implementation and Specification (http://tools.ietf.org/html/rfc1035)
- RFC4033 - DNS Security Introduction and Requirements (http://tools.ietf.org/html/rfc4033)
- RFC4034 - Resource Records for the DNS Security Extensions (http://tools.ietf.org/html/rfc4034)
- RFC4035 - Protocol Modifications for the DNS Security Extensions (http://tools.ietf.org/html/rfc4035)
- RFC4641 - DNSSEC Operational Practices (http://tools.ietf.org/html/rfc4641)
- RFC 5011 - Automated Updates of DNS Security (DNSSEC Trust Anchors (http://tools.ietf.org/html/rfc5011)

30.8 Apache HTTP Server

Contributed by Murray Stokely.

30.8.1 Overview

FreeBSD is used to run some of the busiest web sites in the world. The majority of web servers on the Internet are using the **Apache HTTP Server**. **Apache** software packages should be included on the FreeBSD installation media. If **Apache** was not installed while installing FreeBSD, then it can be installed from the `www/apache22` port.

Once **Apache** has been installed successfully, it must be configured.

> **Note:** This section covers version 2.2.X of the **Apache HTTP Server** as that is the most widely used version for FreeBSD. For more detailed information beyond the scope of this document about **Apache** 2.X, please see http://httpd.apache.org/.

30.8.2 Configuration

The main **Apache HTTP Server** configuration file is installed as `/usr/local/etc/apache22/httpd.conf` on FreeBSD. This file is a typical UNIX text configuration file with comment lines beginning with the # character. A comprehensive description of all possible configuration options is outside the scope of this book, so only the most frequently modified directives will be described here.

`ServerRoot "/usr/local"`

> This specifies the default directory hierarchy for the **Apache** installation. Binaries are stored in the `bin` and `sbin` subdirectories of the server root, and configuration files are stored in `etc/apache`.

`ServerAdmin you@your.address`

> The address to which problems with the server should be emailed. This address also appears on some server-generated pages, such as error documents.

`ServerName www.example.com`

> `ServerName` allows an administrator to set a host name which is sent back to clients for the server. This is useful if the host is different than the one that it is configured with (i.e., use `www` instead of the host's real name).

`DocumentRoot "/usr/local/www/apache22/data"`

> `DocumentRoot`: The directory where documents will be served from. By default, all requests are taken from this directory, but symbolic links and aliases may be used to point to other locations.

It is always a good idea to make backup copies of the **Apache** configuration file before making changes. When the configuration of **Apache**, is complete, save the file and verify the configuration using apachectl(8). To do this, issue `apachectl configtest` which should return `Syntax OK`.

30.8.3 Running Apache

The `www/apache22` port installs an rc(8) script to aid in starting, stopping, and restarting **Apache**, which can be found in `/usr/local/etc/rc.d/`.

To launch **Apache** at system startup, add the following line to `/etc/rc.conf`:

```
apache22_enable="YES"
```

If **Apache** should be started with non-default options, the following line may be added to `/etc/rc.conf`:

```
apache22_flags=""
```

The **Apache** configuration can be tested for errors after making subsequent configuration changes while `httpd` is running. This can be done by the rc(8) script directly, or by the service(8) utility by issuing one of the following commands:

```
# service apache22 configtest
```

> **Note:** It is important to note that the `configtest` is not an rc(8) standard, and should not be expected to work for all rc(8) startup scripts.

If **Apache** does not report configuration errors, the **Apache** `httpd` can be started with service(8):

```
# service apache22 start
```

The `httpd` service can be tested by entering `http://localhost` in a web browser, replacing *`localhost`* with the fully-qualified domain name of the machine running `httpd`, if it is not the local machine. The default web page that is displayed is `/usr/local/www/apache22/data/index.html`.

30.8.4 Virtual Hosting

Apache supports two different types of Virtual Hosting. The first method is Name-based Virtual Hosting. Name-based virtual hosting uses the clients HTTP/1.1 headers to figure out the hostname. This allows many different domains to share the same IP address.

To setup **Apache** to use Name-based Virtual Hosting add an entry like the following to `httpd.conf`:

```
NameVirtualHost *
```

If the webserver was named `www.domain.tld` and a virtual domain for `www.someotherdomain.tld` then add the following entries to `httpd.conf`:

```
<VirtualHost *>
ServerName www.domain.tld
DocumentRoot /www/domain.tld
</VirtualHost>

<VirtualHost *>
ServerName www.someotherdomain.tld
DocumentRoot /www/someotherdomain.tld
</VirtualHost>
```

Replace the addresses with the addresses needed and the path to the documents with what are being used.

For more information about setting up virtual hosts, please consult the official **Apache** documentation at: http://httpd.apache.org/docs/vhosts/.

30.8.5 Apache Modules

There are many different **Apache** modules available to add functionality to the basic server. The FreeBSD Ports Collection provides an easy way to install **Apache** together with some of the more popular add-on modules.

30.8.5.1 mod_ssl

The **mod_ssl** module uses the OpenSSL library to provide strong cryptography via the Secure Sockets Layer (SSL v2/v3) and Transport Layer Security (TLS v1) protocols. This module provides everything necessary to request a signed certificate from a trusted certificate signing authority to run a secure web server on FreeBSD.

The **mod_ssl** module is built by default, but can be enabled by specifying `-DWITH_SSL` at compile time.

30.8.5.2 Language Bindings

There are Apache modules for most major scripting languages. These modules typically make it possible to write **Apache** modules entirely in a scripting language. They are also often used as a persistent interpreter embedded into the server that avoids the overhead of starting an external interpreter and the startup-time penalty for dynamic websites, as described in the next section.

30.8.6 Dynamic Websites

In the last decade, more businesses have turned to the Internet in order to enhance their revenue and increase exposure. This has also increased the need for interactive web content. While some companies, such as Microsoft, have introduced solutions into their proprietary products, the open source community answered the call. Modern options for dynamic web content include Django, Ruby on Rails, **mod_perl2**, and **mod_php**.

30.8.6.1 Django

Django is a BSD licensed framework designed to allow developers to write high performance, elegant web applications quickly. It provides an object-relational mapper so that data types are developed as Python objects, and a rich dynamic database-access API is provided for those objects without the developer ever having to write SQL. It also provides an extensible template system so that the logic of the application is separated from the HTML presentation.

Django depends on **mod_python**, **Apache**, and an SQL database engine. The FreeBSD Port will install all of these pre-requisites with the appropriate flags.

Example 30-3. Installing Django with Apache2, mod_python3, and PostgreSQL

```
# cd /usr/ports/www/py-django; make all install clean -DWITH_MOD_PYTHON3 -DWITH_POSTGRESQL
```

Once Django and these pre-requisites are installed, the application will need a Django project directory along with the Apache configuration to use the embedded Python interpreter. This will be the interpreter to call the application for specific URLs on the site.

Example 30-4. Apache Configuration for Django/mod_python

A line must be added to the apache `httpd.conf` file to configure Apache to pass requests for certain URLs to the web application:

```
<Location "/">
    SetHandler python-program
    PythonPath "['/dir/to/the/django/packages/'] + sys.path"
    PythonHandler django.core.handlers.modpython
```

```
    SetEnv DJANGO_SETTINGS_MODULE mysite.settings
    PythonAutoReload On
    PythonDebug On
</Location>
```

30.8.6.2 Ruby on Rails

Ruby on Rails is another open source web framework that provides a full development stack and is optimized to make web developers more productive and capable of writing powerful applications quickly. It can be installed easily from the ports system.

```
# cd /usr/ports/www/rubygem-rails; make all install clean
```

30.8.6.3 mod_perl2

The **Apache**/Perl integration project brings together the full power of the Perl programming language and the **Apache HTTP Server**. With the **mod_perl2** module it is possible to write **Apache** modules entirely in Perl. In addition, the persistent interpreter embedded in the server avoids the overhead of starting an external interpreter and the penalty of Perl start-up time.

mod_perl2 is available in the `www/mod_perl2` port.

30.8.6.4 mod_php

Written by Tom Rhodes.

PHP, also known as "PHP: Hypertext Preprocessor" is a general-purpose scripting language that is especially suited for Web development. Capable of being embedded into HTML its syntax draws upon C, Java, and Perl with the intention of allowing web developers to write dynamically generated webpages quickly.

To gain support for PHP5 for the **Apache** web server, begin by installing the `lang/php5` port.

If the `lang/php5` port is being installed for the first time, available `OPTIONS` will be displayed automatically. If a menu is not displayed, i.e., because the `lang/php5` port has been installed some time in the past, it is always possible to bring the options dialog up again by running:

```
# make config
```

in the port directory.

In the options dialog, check the `APACHE` option to build **mod_php5** as a loadable module for the **Apache** web server.

> **Note:** A lot of sites are still using PHP4 for various reasons (i.e., compatibility issues or already deployed web applications). If the **mod_php4** is needed instead of **mod_php5**, then please use the `lang/php4` port. The `lang/php4` port supports many of the configuration and build-time options of the `lang/php5` port.

This will install and configure the modules required to support dynamic PHP applications. Check to ensure the following sections have been added to `/usr/local/etc/apache22/httpd.conf`:

```
LoadModule php5_module        libexec/apache/libphp5.so
```

```
AddModule mod_php5.c
    <IfModule mod_php5.c>
        DirectoryIndex index.php index.html
    </IfModule>
    <IfModule mod_php5.c>
        AddType application/x-httpd-php .php
        AddType application/x-httpd-php-source .phps
    </IfModule>
```

Once completed, a simple call to the `apachectl` command for a graceful restart is needed to load the PHP module:

```
# apachectl graceful
```

For future upgrades of PHP, the `make config` command will not be required; the selected `OPTIONS` are saved automatically by the FreeBSD Ports framework.

The PHP support in FreeBSD is extremely modular so the base install is very limited. It is very easy to add support using the `lang/php5-extensions` port. This port provides a menu driven interface to PHP extension installation. Alternatively, individual extensions can be installed using the appropriate port.

For instance, to add support for the **MySQL** database server to PHP5, simply install the port `databases/php5-mysql`.

After installing an extension, the **Apache** server must be reloaded to pick up the new configuration changes:

```
# apachectl graceful
```

30.9 File Transfer Protocol (FTP)

Contributed by Murray Stokely.

30.9.1 Overview

The File Transfer Protocol (FTP) provides users with a simple way to transfer files to and from an FTP server. FreeBSD includes FTP server software, **ftpd**, in the base system. This makes setting up and administering an FTP server on FreeBSD very straightforward.

30.9.2 Configuration

The most important configuration step is deciding which accounts will be allowed access to the FTP server. A normal FreeBSD system has a number of system accounts used for various daemons, but unknown users should not be allowed to log in with these accounts. The `/etc/ftpusers` file is a list of users disallowed any FTP access. By default, it includes the aforementioned system accounts, but it is possible to add specific users here that should not be allowed access to FTP.

In some cases it may be desirable to restrict the access of some users without preventing them completely from using FTP. This can be accomplished with the `/etc/ftpchroot` file. This file lists users and groups subject to FTP access restrictions. The ftpchroot(5) manual page has all of the details so it will not be described in detail here.

To enable anonymous FTP access to the server, create a user named `ftp` on the FreeBSD system. Users will then be able to log on to the FTP server with a username of `ftp` or `anonymous` and with any password (by convention an email address for the user should be used as the password). The FTP server will call chroot(2) when an anonymous user logs in, to restrict access to only the home directory of the `ftp` user.

There are two text files that specify welcome messages to be displayed to FTP clients. The contents of the file `/etc/ftpwelcome` will be displayed to users before they reach the login prompt. After a successful login, the contents of the file `/etc/ftpmotd` will be displayed. Note that the path to this file is relative to the login environment, so the file `~ftp/etc/ftpmotd` would be displayed for anonymous users.

Once the FTP server has been configured properly, it must be enabled in `/etc/inetd.conf`. All that is required here is to remove the comment symbol "#" from in front of the existing **ftpd** line :

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

As explained in Example 30-1, the **inetd** configuration must be reloaded after this configuration file is changed. Please refer to Section 30.2.2 for details on enabling **inetd** on the system.

Alternatively, **ftpd** can also be started as a stand-alone server. In this case, it is sufficient to set the appropriate variable in `/etc/rc.conf`:

```
ftpd_enable="YES"
```

After setting the above variable, the stand-alone server will be started at the next reboot, or it can be started manually by executing the following command as `root`:

```
# service ftpd start
```

You can now log on to the FTP server by typing:

```
% ftp localhost
```

30.9.3 Maintaining

The **ftpd** daemon uses syslog(3) to log messages. By default, the system log daemon will put messages related to FTP in the `/var/log/xferlog` file. The location of the FTP log can be modified by changing the following line in `/etc/syslog.conf`:

```
ftp.info      /var/log/xferlog
```

Be aware of the potential problems involved with running an anonymous FTP server. In particular, think twice about allowing anonymous users to upload files. It may turn out that the FTP site becomes a forum for the trade of unlicensed commercial software or worse. If anonymous FTP uploads are required, then verify the permissions so that these files can not be read by other anonymous users until they have been reviewed by an administrator.

30.10 File and Print Services for Microsoft® Windows Clients (Samba)

Contributed by Murray Stokely.

30.10.1 Overview

Samba is a popular open source software package that provides file and print services for Microsoft Windows clients. Such clients can connect to and use FreeBSD filespace as if it was a local disk drive, or FreeBSD printers as if they were local printers.

Samba software packages should be included on the FreeBSD installation media. If they were not installed when first installing FreeBSD, then they may be installed from the `net/samba34` port or package.

30.10.2 Configuration

A default **Samba** configuration file is installed as `/usr/local/share/examples/samba34/smb.conf.default`. This file must be copied to `/usr/local/etc/smb.conf` and customized before **Samba** can be used.

The `smb.conf` file contains runtime configuration information for **Samba**, such as definitions of the printers and "file system shares" that will be shared with Windows clients. The **Samba** package includes a web based tool called **swat** which provides a simple way of configuring the `smb.conf` file.

30.10.2.1 Using the Samba Web Administration Tool (SWAT)

The Samba Web Administration Tool (SWAT) runs as a daemon from **inetd**. Therefore, **inetd** must be enabled as shown in Section 30.2, and the following line in `/etc/inetd.conf` should be uncommented before **swat** can be used to configure **Samba**:

```
swat    stream  tcp     nowait/400      root    /usr/local/sbin/swat    swat
```

As explained in Example 30-1, the **inetd** configuration must be reloaded after this configuration file is changed.

Once **swat** has been enabled in `inetd.conf`, a web browser may be used to connect to http://localhost:901. At first login, the system `root` account must be used.

Once successfully logging on to the main **Samba** configuration page, the system documentation will be available, or configuration may begin by clicking on the Globals tab. The Globals section corresponds to the variables that are set in the `[global]` section of `/usr/local/etc/smb.conf`.

30.10.2.2 Global Settings

Whether **swat** is being used or `/usr/local/etc/smb.conf` is being edited directly, the first directives encountered when configuring **Samba** are:

`workgroup`

NT Domain-Name or Workgroup-Name for the computers that will be accessing this server.

`netbios name`

This sets the NetBIOS name by which a **Samba** server is known. By default it is the same as the first component of the host's DNS name.

`server string`

This sets the string that will be displayed with the `net view` command and some other networking tools that seek to display descriptive text about the server.

30.10.2.3 Security Settings

Two of the most important settings in `/usr/local/etc/smb.conf` are the security model chosen, and the backend password format for client users. The following directives control these options:

`security`

The two most common options here are `security = share` and `security = user`. If the clients use usernames that are the same as their usernames on the FreeBSD machine then user level security should be used. This is the default security policy and it requires clients to first log on before they can access shared resources.

In share level security, clients do not need to log onto the server with a valid username and password before attempting to connect to a shared resource. This was the default security model for older versions of **Samba**.

`passdb backend`

Samba has several different backend authentication models. Clients may be authenticated with LDAP, NIS+, an SQL database, or a modified password file. The default authentication method is `smbpasswd`, and that is all that will be covered here.

Assuming that the default `smbpasswd` backend is used, the `/usr/local/etc/samba/smbpasswd` file must be created to allow **Samba** to authenticate clients. To provide the UNIX user accounts access from Windows clients, use the following command:

```
# smbpasswd -a username
```

Note: The recommended backend is now `tdbsam`, and the following command should be used to add user accounts:

```
# pdbedit -a -u username
```

Please see the Official Samba HOWTO (http://www.samba.org/samba/docs/man/Samba-HOWTO-Collection/) for additional information about configuration options. With the basics outlined here, the minimal required start running **Samba** will be explained. Other documentation should be consulted in addition to the information here.

30.10.3 Starting Samba

The `net/samba34` port adds a new startup script, which can be used to control **Samba**. To enable this script, so that it can be used for example to start, stop or restart **Samba**, add the following line to the `/etc/rc.conf` file:

```
samba_enable="YES"
```

Or, for fine grain control:

```
nmbd_enable="YES"
```

```
smbd_enable="YES"
```

> **Note:** This will also configure **Samba** to automatically start at system boot time.

It is possible then to start **Samba** at any time by typing:

```
# service samba start
Starting SAMBA: removing stale tdbs :
Starting nmbd.
Starting smbd.
```

Please refer to Section 12.7 for more information about using rc scripts.

Samba actually consists of three separate daemons. Notice that both the **nmbd** and **smbd** daemons are started by the `samba` script. If winbind, name resolution services were enabled in `smb.conf`, the **winbindd** daemon will be started as well.

Samba may be stopped at any time by typing:

```
# service samba stop
```

Samba is a complex software suite with functionality that allows broad integration with Microsoft Windows networks. For more information about functionality beyond the basic installation described here, please see http://www.samba.org.

30.11 Clock Synchronization with NTP

Contributed by Tom Hukins.

30.11.1 Overview

Over time, a computer's clock is prone to drift. The Network Time Protocol (NTP) is one way to ensure the clock stays accurate.

Many Internet services rely on, or greatly benefit from, computers' clocks being accurate. For example, a web server may receive requests to send a file if it has been modified since a certain time. In a local area network environment, it is essential that computers sharing files from the same file server have synchronized clocks so that file timestamps

stay consistent. Services such as cron(8) also rely on an accurate system clock to run commands at the specified times.

FreeBSD ships with the ntpd(8) NTP server which can be used to query other NTP servers to set the clock on the machine or provide time services to others.

30.11.2 Choosing Appropriate NTP Servers

In order to synchronize the clock, one or more NTP servers must be defined. The network administrator or ISP may have set up an NTP server for this purpose—check their documentation to see if this is the case. There is an online list of publicly accessible NTP servers (http://support.ntp.org/bin/view/Servers/WebHome) which may be referenced to find an NTP server nearest to the system. Take care to review the policy for any chosen servers, and ask for permission if required.

Choosing several unconnected NTP servers is a good idea in case one of the servers being used becomes unreachable or its clock is unreliable. ntpd(8) uses the responses it receives from other servers intelligently—it will favor unreliable servers less than reliable ones.

30.11.3 Configuring The Machine

30.11.3.1 Basic Configuration

To synchronize the clock only when the machine boots up, use ntpdate(8). This may be appropriate for some desktop machines which are frequently rebooted and only require infrequent synchronization, but most machines should run ntpd(8).

Using ntpdate(8) at boot time is also a good idea for machines that run ntpd(8). The ntpd(8) program changes the clock gradually, whereas ntpdate(8) sets the clock, no matter how great the difference between a machine's current clock setting and the correct time.

To enable ntpdate(8) at boot time, add `ntpdate_enable="YES"` to `/etc/rc.conf`. Also specify all synchronization servers and any flags to be passed to ntpdate(8) in `ntpdate_flags`.

30.11.3.2 General Configuration

NTP is configured by the `/etc/ntp.conf` file in the format described in ntp.conf(5). Here is a simple example:

```
server ntplocal.example.com prefer
server timeserver.example.org
server ntp2a.example.net

driftfile /var/db/ntp.drift
```

The `server` option specifies which servers are to be used, with one server listed on each line. If a server is specified with the `prefer` argument, as with `ntplocal.example.com`, that server is preferred over other servers. A response from a preferred server will be discarded if it differs significantly from other servers' responses, otherwise it will be used without any consideration to other responses. The `prefer` argument is normally used for NTP servers that are known to be highly accurate, such as those with special time monitoring hardware.

The `driftfile` option specifies which file is used to store the system clock's frequency offset. The ntpd(8) program uses this to automatically compensate for the clock's natural drift, allowing it to maintain a reasonably correct setting even if it is cut off from all external time sources for a period of time.

The `driftfile` option specifies which file is used to store information about previous responses from the NTP servers being used. This file contains internal information for NTP. It should not be modified by any other process.

30.11.3.3 Controlling Access to Your Server

By default, the NTP server will be accessible to all hosts on the Internet. The `restrict` option in `/etc/ntp.conf` controls which machines can access the server.

To deny all machines from accessing the NTP server, add the following line to `/etc/ntp.conf`:

```
restrict default ignore
```

> **Note:** This will also prevent access from the server to any servers listed in the local configuration. If there is a need to synchronise the NTP server with an external NTP server, allow only that specific server. See the ntp.conf(5) manual for more information.

To allow machines within the network to synchronize their clocks with the server, but ensure they are not allowed to configure the server or used as peers to synchronize against, add

```
restrict 192.168.1.0 mask 255.255.255.0 nomodify notrap
```

instead, where `192.168.1.0` is an IP address on the network and `255.255.255.0` is the network's netmask.

The `/etc/ntp.conf` file can contain multiple `restrict` options. For more details, see the `Access Control Support` subsection of ntp.conf(5).

30.11.4 Running the NTP Server

To ensure the NTP server is started at boot time, add the line `ntpd_enable="YES"` to `/etc/rc.conf`. To pass additional flags to ntpd(8), edit the `ntpd_flags` parameter in `/etc/rc.conf`.

To start the server without rebooting the machine, run `ntpd` being sure to specify any additional parameters from `ntpd_flags` in `/etc/rc.conf`. For example:

```
# ntpd -p /var/run/ntpd.pid
```

30.11.5 Using ntpd with a Temporary Internet Connection

The ntpd(8) program does not need a permanent connection to the Internet to function properly. However, if there is a temporary connection that is configured to dial out on demand, it is a good idea to prevent NTP traffic from triggering a dial out or keeping the connection alive. PPP users can use the `filter` directives in `/etc/ppp/ppp.conf`. For example:

```
 set filter dial 0 deny udp src eq 123
```

```
# Prevent NTP traffic from initiating dial out
set filter dial 1 permit 0 0
set filter alive 0 deny udp src eq 123
# Prevent incoming NTP traffic from keeping the connection open
set filter alive 1 deny udp dst eq 123
# Prevent outgoing NTP traffic from keeping the connection open
set filter alive 2 permit 0/0 0/0
```

For more details see the `PACKET FILTERING` section in ppp(8) and the examples in `/usr/share/examples/ppp/`.

> **Note:** Some Internet access providers block low-numbered ports, preventing NTP from functioning since replies never reach the machine.

30.11.6 Further Information

Documentation for the NTP server can be found in `/usr/share/doc/ntp/` in HTML format.

30.12 Remote Host Logging with `syslogd`

Contributed by Tom Rhodes.

Interacting with system logs is a crucial aspect of both security and system administration. Monitoring the log files of multiple hosts can get very unwieldy when these hosts are distributed across medium or large networks, or when they are parts of various different types of networks. In these cases, configuring remote logging may make the whole process a lot more comfortable.

Centralized logging to a specific logging host can reduce some of the administrative burden of log file administration. Log file aggregation, merging and rotation may be configured in one location, using the native tools of FreeBSD, such as syslogd(8) and newsyslog(8). In the following example configuration, host `A`, named `logserv.example.com`, will collect logging information for the local network. Host `B`, named `logclient.example.com` will pass logging information to the server system. In live configurations, both hosts require proper forward and reverse DNS or entries in `/etc/hosts`. Otherwise, data will be rejected by the server.

30.12.1 Log Server Configuration

Log servers are machines configured to accept logging information from remote hosts. In most cases this is to ease configuration, in other cases it may just be a better administration move. Regardless of reason, there are a few requirements before continuing.

A properly configured logging server has met the following minimal requirements:

- The firewall ruleset allows for UDP to be passed on port 514 on both the client and server;
- `syslogd` has been configured to accept remote messages from client machines;

- The `syslogd` server and all client machines must have valid entries for both forward and reverse DNS, or be properly configured in `/etc/hosts`.

To configure the log server, the client must be listed in `/etc/syslog.conf`, and the logging facility must be specified:

```
+logclient.example.com
*.*     /var/log/logclient.log
```

> **Note:** More information on various supported and available *facilities* may be found in the syslog.conf(5) manual page.

Once added, all `facility` messages will be logged to the file specified previously, `/var/log/logclient.log`.

The server machine must also have the following listing placed inside `/etc/rc.conf`:

```
syslogd_enable="YES"
syslogd_flags="-a logclient.example.com -v -v"
```

The first option will enable the `syslogd` daemon on boot up, and the second option allows data from the specified client to be accepted on this server. The latter part, using `-v -v`, will increase the verbosity of logged messages. This is extremely useful for tweaking facilities as administrators are able to see what type of messages are being logged under which facility.

Multiple `-a` options may be specified to allow logging from multiple clients. IP addresses and whole netblocks may also be specified, see the syslog(3) manual page for a full list of possible options.

Finally, the log file should be created. The method used does not matter, but touch(1) works great for situations such as this:

```
# touch
        /var/log/logclient.log
```

At this point, the `syslogd` daemon should be restarted and verified:

```
# service syslogd restart
# pgrep syslog
```

If a PID is returned, the server has been restarted successfully, and client configuration may begin. If the server has not restarted, consult the `/var/log/messages` log for any output.

30.12.2 Log Client Configuration

A logging client is a machine which sends log information to a logging server in addition to keeping local copies.

Similar to log servers, clients must also meet a few minimum requirements:

- syslogd(8) must be configured to send messages of specific types to a log server, which must accept them;
- The firewall must allow UDP packets through on port 514;
- Both forward and reverse DNS must be configured or have proper entries in the `/etc/hosts`.

Client configuration is a bit more relaxed when compared to that of the servers. The client machine must have the following listing placed inside `/etc/rc.conf`:

```
syslogd_enable="YES"
syslogd_flags="-s -v -v"
```

As before, these entries will enable the `syslogd` daemon on boot up, and increases the verbosity of logged messages. The `-s` option prevents logs from being accepted by this client from other hosts.

Facilities describe the system part for which a message is generated. For an example, ftp and ipfw are both facilities. When log messages are generated for those two services, they will normally include those two utilities in any log messages. Facilities are accompanied with a priority or level, which is used to mark how important a log message is. The most common will be the `warning` and `info`. Please refer to the syslog(3) manual page for a full list of available facilities and priorities.

The logging server must be defined in the client's `/etc/syslog.conf`. In this instance, the `@` symbol is used to send logging data to a remote server and would look similar to the following entry:

```
*.*             @logserv.example.com
```

Once added, `syslogd` must be restarted for the changes to take effect:

```
# service syslogd restart
```

To test that log messages are being sent across the network, use logger(1) on the client to send a message to `syslogd`:

```
# logger
        "Test message from logclient"
```

This message should now exist both in `/var/log/messages` on the client, and `/var/log/logclient.log` on the log server.

30.12.3 Debugging Log Servers

In certain cases, debugging may be required if messages are not being received on the log server. There are several reasons this may occur; however, the most common two are network connection issues and DNS issues. To test these cases, ensure both hosts are able to reach one another using the hostname specified in `/etc/rc.conf`. If this appears to be working properly, an alternation to the `syslogd_flags` option in `/etc/rc.conf` will be required.

In the following example, `/var/log/logclient.log` is empty, and the `/var/log/messages` files indicate no reason for the failure. To increase debugging output, change the `syslogd_flags` option to look like the following example, and issue a restart:

```
syslogd_flags="-d -a logclien.example.com -v -v"
```

```
# service syslogd restart
```

Debugging data similar to the following will flash on the screen immediately after the restart:

```
logmsg: pri 56, flags 4, from logserv.example.com, msg syslogd: restart
syslogd: restarted
logmsg: pri 6, flags 4, from logserv.example.com, msg syslogd: kernel boot file is /boot/kernel/k
```

```
Logging to FILE /var/log/messages
syslogd: kernel boot file is /boot/kernel/kernel
cvthname(192.168.1.10)
validate: dgram from IP 192.168.1.10, port 514, name logclient.example.com;
rejected in rule 0 due to name mismatch.
```

It appears obvious the messages are being rejected due to a name mismatch. After reviewing the configuration bit by bit, it appears a typo in the following `/etc/rc.conf` line has an issue:

```
syslogd_flags="-d -a logclien.example.com -v -v"
```

The line should contain `logclient`, not `logclien`. After the proper alterations are made, a restart is issued with expected results:

```
# service syslogd restart
logmsg: pri 56, flags 4, from logserv.example.com, msg syslogd: restart
syslogd: restarted
logmsg: pri 6, flags 4, from logserv.example.com, msg syslogd: kernel boot file is /boot/kernel/k
syslogd: kernel boot file is /boot/kernel/kernel
logmsg: pri 166, flags 17, from logserv.example.com,
msg Dec 10 20:55:02 <syslog.err> logserv.example.com syslogd: exiting on signal 2
cvthname(192.168.1.10)
validate: dgram from IP 192.168.1.10, port 514, name logclient.example.com;
accepted in rule 0.
logmsg: pri 15, flags 0, from logclient.example.com, msg Dec 11 02:01:28 trhodes: Test message 2
Logging to FILE /var/log/logclient.log
Logging to FILE /var/log/messages
```

At this point, the messages are being properly received and placed in the correct file.

30.12.4 Security Considerations

As with any network service, security requirements should be considered before implementing this configuration. At times, log files may contain sensitive data about services enabled on the local host, user accounts, and configuration data. Network data sent from the client to the server will not be encrypted nor password protected. If a need for encryption exists, it might be possible to use `security/stunnel`, which will transmit data over an encrypted tunnel.

Local security is also an issue. Log files are not encrypted during use or after log rotation. Local users may access these files to gain additional insight on system configuration. In those cases, setting proper permissions on these files will be critical. The newsyslog(8) utility supports setting permissions on newly created and rotated log files. Setting log files to mode `600` should prevent any unwanted snooping by local users.

Chapter 31 Firewalls

Contributed by Joseph J. Barbish. Converted to SGML and updated by Brad Davis.

31.1 Introduction

Firewalls make it possible to filter the incoming and outgoing traffic that flows through a system. A firewall can use one or more sets of "rules" to inspect network packets as they come in or go out of network connections and either allows the traffic through or blocks it. The rules of a firewall can inspect one or more characteristics of the packets such as the protocol type, source or destination host address, and source or destination port.

Firewalls can enhance the security of a host or a network. They can be used to do one or more of the following:

- Protect and insulate the applications, services, and machines of an internal network from unwanted traffic from the public Internet.
- Limit or disable access from hosts of the internal network to services of the public Internet.
- Support network address translation (NAT), which allows an internal network to use private IP addresses and share a single connection to the public Internet using either a single IP address or a shared pool of automatically assigned public addresses.

After reading this chapter, you will know:

- How to define packet filtering rules.
- The differences between the firewalls built into FreeBSD.
- How to use and configure the **PF** firewall.
- How to use and configure the **IPFILTER** firewall.
- How to use and configure the **IPFW** firewall.

Before reading this chapter, you should:

- Understand basic FreeBSD and Internet concepts.

31.2 Firewall Concepts

A firewall ruleset can be either "exclusive" or "inclusive". An exclusive firewall allows all traffic through except for the traffic matching the ruleset. An inclusive firewall does the reverse as it only allows traffic matching the rules through and blocks everything else.

An inclusive firewall offers better control of the outgoing traffic, making it a better choice for systems that offer services to the public Internet. It also controls the type of traffic originating from the public Internet that can gain access to a private network. All traffic that does not match the rules is blocked and logged. Inclusive firewalls are generally safer than exclusive firewalls because they significantly reduce the risk of allowing unwanted traffic.

> **Note:** Unless noted otherwise, all configuration and example rulesets in this chapter create inclusive firewall rulesets.

Security can be tightened further using a “stateful firewall”. This type of firewall keeps track of open connections and only allows traffic which either matches an existing connection or opens a new, allowed connection. The disadvantage of a stateful firewall is that it can be vulnerable to Denial of Service (DoS) attacks if a lot of new connections are opened very fast. Most firewalls use a combination of stateful and non-stateful behavior.

31.3 Firewall Packages

FreeBSD has three firewalls built into the base system: *IPFILTER*, also known as IPF, *IPFIREWALL*, also known as IPFW, and PF). FreeBSD also provides two traffic shapers for controlling bandwidth usage: altq(4) and dummynet(4). Dummynet has traditionally been closely tied with IPFW, and ALTQ with PF. Each firewall uses rules to control the access of packets to and from a FreeBSD system, although they go about it in different ways and each has a different rule syntax.

FreeBSD provides multiple firewalls in order to meet the different requirements and preferences for a wide variety of users. Each user should evaluate which firewall best meets their needs.

Since all firewalls are based on inspecting the values of selected packet control fields, the creator of the firewall ruleset must have an understanding of how TCP/IP works, what the different values in the packet control fields are, and how these values are used in a normal session conversation. For a good introduction, refer to Daryl’s TCP/IP Primer (http://www.ipprimer.com/overview.cfm).

31.4 PF and ALTQ

Revised and updated by John Ferrell.

Since FreeBSD 5.3, a ported version of OpenBSD’s PF firewall has been included as an integrated part of the base system. PF is a complete, full-featured firewall that has optional support for ALTQ (Alternate Queuing), which provides Quality of Service (QoS).

Since the OpenBSD Project maintains the definitive reference for PF in thePF FAQ (http://www.openbsd.org/faq/pf/), this section of the Handbook focuses on PF as it pertains to FreeBSD, while providing some general usage information.

More information about porting PF to FreeBSD can be found at http://pf4freebsd.love2party.net/.

31.4.1 Using the PF Loadable Kernel Modules

In order to use PF, the PF kernel module must be first loaded. Add the following line to `/etc/rc.conf`:

```
pf_enable="YES"
```

Then, run the startup script to load the module:

```
# service pf start
```

The PF module will not load if it cannot find the ruleset configuration file. The default location is `/etc/pf.conf`. If the PF ruleset is located somewhere else, add a line to `/etc/rc.conf` which specifies the full path to the file:

```
pf_rules="/path/to/pf.conf"
```

The sample `pf.conf` can be found in `/usr/share/examples/pf/`.

The PF module can also be loaded manually from the command line:

```
# kldload pf.ko
```

Logging support for PF is provided by `pflog.ko` which can be loaded by adding the following line to `/etc/rc.conf`:

```
pflog_enable="YES"
```

Then, run the startup script to load the module:

```
# service pflog start
```

31.4.2 PF Kernel Options

While it is not necessary to compile PF support into the FreeBSD kernel, some of PF's advanced features are not included in the loadable module, namely pfsync(4), which is a pseudo-device that exposes certain changes to the state table used by PF. It can be paired with carp(4) to create failover firewalls using PF. More information on CARP can be found in of the Handbook.

The following PF kernel options can be found in `/usr/src/sys/conf/NOTES`:

```
device pf
device pflog
device pfsync
```

`device pf` enables PF support.

`device pflog` enables the optional pflog(4) pseudo network device which can be used to log traffic to a bpf(4) descriptor. The pflogd(8) daemon can then be used to store the logging information to disk.

`device pfsync` enables the optional pfsync(4) pseudo-network device that is used to monitor "state changes".

31.4.3 Available `rc.conf` Options

The following rc.conf(5) statements can be used to configure PF and pflog(4) at boot:

```
pf_enable="YES"                 # Enable PF (load module if required)
pf_rules="/etc/pf.conf"         # rules definition file for pf
pf_flags=""                     # additional flags for pfctl startup
pflog_enable="YES"              # start pflogd(8)
pflog_logfile="/var/log/pflog"  # where pflogd should store the logfile
pflog_flags=""                  # additional flags for pflogd startup
```

If there is a LAN behind the firewall and packets need to be forwarded for the computers on the LAN, or NAT is required, add the following option:

```
gateway_enable="YES"            # Enable as LAN gateway
```

31.4.4 Creating Filtering Rules

By default, PF reads its configuration rules from `/etc/pf.conf` and modifies, drops, or passes packets according to the rules or definitions specified in this file. The FreeBSD installation includes several sample files located in `/usr/share/examples/pf/`. Refer to the PF FAQ (http://www.openbsd.org/faq/pf/) for complete coverage of PF rulesets.

> **Warning:** When reading the PF FAQ (http://www.openbsd.org/faq/pf/), keep in mind that different versions of FreeBSD contain different versions of PF. Currently, FreeBSD 8.*X* is using the same version of PF as OpenBSD 4.1. FreeBSD 9.*X* and later is using the same version of PF as OpenBSD 4.5.

The FreeBSD packet filter mailing list (http://lists.FreeBSD.org/mailman/listinfo/freebsd-pf) is a good place to ask questions about configuring and running the PF firewall. Do not forget to check the mailing list archives before asking questions.

To control PF, use pfctl(8). Below are some useful options to this command. Review pfctl(8) for a description of all available options:

Command	Purpose
`pfctl -e`	Enable PF.
`pfctl -d`	Disable PF.
`pfctl -F all -f /etc/pf.conf`	Flush all NAT, filter, state, and table rules and reload `/etc/pf.conf`.
`pfctl -s [ rules | nat state ]`	Report on the filter rules, NAT rules, or state table.
`pfctl -vnf /etc/pf.conf`	Check `/etc/pf.conf` for errors, but do not load ruleset.

31.4.5 Enabling ALTQ

ALTQ is only available by compiling its support into the FreeBSD kernel. ALTQ is not supported by all network card drivers. Refer to altq(4) for a list of drivers that are supported by the release of FreeBSD.

The following kernel options will enable ALTQ and add additional functionality:

```
options         ALTQ
options         ALTQ_CBQ        # Class Based Queuing (CBQ)
options         ALTQ_RED        # Random Early Detection (RED)
options         ALTQ_RIO        # RED In/Out
options         ALTQ_HFSC       # Hierarchical Packet Scheduler (HFSC)
options         ALTQ_PRIQ       # Priority Queuing (PRIQ)
options         ALTQ_NOPCC      # Required for SMP build
```

`options ALTQ` enables the ALTQ framework.

`options ALTQ_CBQ` enables *Class Based Queuing* (CBQ). CBQ can be used to divide a connection's bandwidth into different classes or queues to prioritize traffic based on filter rules.

`options ALTQ_RED` enables *Random Early Detection* (RED). RED is used to avoid network congestion by measuring the length of the queue and comparing it to the minimum and maximum thresholds for the queue. If the

queue is over the maximum, all new packets will be dropped. RED drops packets from different connections randomly.

`options ALTQ_RIO` enables *Random Early Detection In and Out*.

`options ALTQ_HFSC` enables the *Hierarchical Fair Service Curve Packet Scheduler* HFSC. For more information, refer to http://www-2.cs.cmu.edu/~hzhang/HFSC/main.html.

`options ALTQ_PRIQ` enables *Priority Queuing* (PRIQ). PRIQ will always pass traffic that is in a higher queue first.

`options ALTQ_NOPCC` enables SMP support for ALTQ. This option is required on SMP systems.

31.4.6 PF Rule Sets and Tools

Contributed by Peter N. M. Hansteen.

This section demonstrates some useful PF features and PF related tools in a series of examples. A more thorough tutorial is available at http://home.nuug.no/~peter/pf/.

> **Tip:** `security/sudo` is useful for running commands like `pfctl` that require elevated privileges. It can be installed from the Ports Collection.

31.4.6.1 The Simplest Rule Set Ever

The simplest possible setup is for a single machine which will not run any services, and which will talk to one network which may be the Internet. A minimal `/etc/pf.conf` looks like this:

```
block in all
pass out all keep state
```

Here we deny any incoming traffic, allow traffic we make ourselves to pass, and retain state information on our connections. Keeping state information allows return traffic for all connections we have initiated to pass back to us. This rule set is used on machines that can be trusted. The rule set can be loaded with

```
# pfctl -e ; pfctl -f /etc/pf.conf
```

31.4.6.2 Tighter and More Elegant

For a slightly more structured and complete setup, we start by denying everything and then allowing only those things we know that we need [1]. This gives us the opportunity to introduce two of the features which make PF such a wonderful tool: *lists* and *macros*.

We will make some changes to `/etc/pf.conf`, starting with

```
block all
```

Then we back up a little. Macros need to be defined before use, so at the very top of the file, we add:

```
tcp_services = "{ ssh, smtp, domain, www, pop3, auth, pop3s }"
udp_services = "{ domain }"
```

Now we have demonstrated several things at once - what macros look like, that macros may be lists, and that PF understands rules using port names equally well as it does port numbers. The names are the ones listed in `/etc/services`. This gives us something to put in our rules, which we edit slightly to look like this:

```
block all
pass out proto tcp to any port $tcp_services keep state
pass proto udp to any port $udp_services keep state
```

At this point some of us will point out that UDP is stateless, but PF actually manages to maintain state information despite this. Keeping state for a UDP connection means that for example when you ask a name server about a domain name, you will be able to receive its answer.

Since we have made changes to our `pf.conf`, we load the new rules:

```
# pfctl -f /etc/pf.conf
```

and the new rules are applied. If there are no syntax errors, `pfctl` will not output any messages during the rule load. The `-v` flag will produce more verbose `pfctl` output.

If there have been extensive changes to the rule set, the rules can be tested before attempting to load them. The command to do this is

```
# pfctl -nf /etc/pf.conf
```

`-n` causes the rules to be interpreted only, but does not load them. This provides an opportunity to correct any errors. Under any circumstances, the last valid rule set loaded will be in force until PF is disabled or a new rule set is loaded.

> **Use `pfctl -v` to Show the Parsed Rule Set:** Adding the `-v` to a `pfctl` ruleset load (even a dry run with `-n`) will display the fully parsed rules exactly the way they will be loaded. This is extremely useful when debugging rules.

31.4.6.3 A Simple Gateway with NAT

To most users, a single machine setup will be of limited interest, and at this point we move on to more realistic or at least more common setups, concentrating on a machine which is running PF and also acts as a gateway for at least one other machine.

31.4.6.3.1 Gateways and the Pitfalls of `in`, `out` and `on`

In the single machine setup, life is relatively simple. Traffic created on it should either pass out to the rest of the world or not, and the administrator decides what to let in from elsewhere.

On a gateway, the perspective changes from “me versus the network out there” to “I am the one who decides what to pass to or from all the networks I am connected to”. The machine has at least two network interfaces, each connected to a separate net.

It is very reasonable to think that for traffic to pass from the network connected to `xl1` to hosts on the network connected to `xl0`, a rule like this is needed:

```
pass in on xl1 from xl1:network to xl0:network port $ports keep state
```

This rule keeps track of states as well.

However, one of the most common and most complained-about mistakes in firewall configuration is not realizing that the “to” keyword does not in itself guarantee passage all the way there. The rule we just wrote only lets the traffic pass in to the gateway on the internal interface. To let the packets get a bit further, a matching rule is needed which says

```
pass out on xl0 from xl1:network to xl0:network port $ports keep state
```

These rules will work, but they will not necessarily achieve the desired effect.

Rules this specific are rarely needed. For the basic gateway configurations we will be dealing with here, a better rule says

```
pass from xl1:network to any port $ports keep state
```

This provides local net access to the Internet and leaves the detective work to the *antispoof* and *scrub* code. They are both pretty good these days, and we will get back to them later. For now we just accept the fact that for simple setups, interface-bound rules with in/out rules tend to add more clutter than they are worth to rule sets.

For a busy network admin, a readable rule set is a safer rule set.

For the remainder of this section, with some exceptions, we will keep the rules as simple as possible for readability.

31.4.6.3.2 What is the Local Network, Anyway?

Above, we introduced the `interface:network` notation. That is a nice piece of shorthand, but the rule set can be made even more readable and maintainable by taking the macro use a tiny bit further.

For example, a `$localnet` macro could be defined as the network directly attached to your internal interface (`$xl1:network` in the examples above).

Alternatively, the definition of `$localnet` could be changed to an *IP address/netmask* notation to denote a network, such as `192.168.100.1/24` for a subnet of private addresses.

If required, `$localnet` could even be defined as a list of networks. Whatever the specific needs, a sensible `$localnet` definition and a typical pass rule of the type

```
pass from $localnet to any port $ports keep state
```

could end up saving you a few headaches. We will stick to that convention from here on.

31.4.6.3.3 Setting Up

We assume that the machine has acquired another network card or at any rate there is a network connection from the local network, via PPP or other means. We will not consider the specific interface configurations.

For the discussion and examples below, only the interface names will differ between a PPP setup and an Ethernet one, and we will do our best to get rid of the actual interface names as quickly as possible.

First, we need to turn on gatewaying in order to let the machine forward the network traffic it receives on one interface to other networks via a separate interface. Initially we will do this on the command line with sysctl(8), for traditional *IP version four*.

```
# sysctl net.inet.ip.forwarding=1
```

If we need to forward *IP version six* traffic, the command is

```
# sysctl net.inet6.ip6.forwarding=1
```

In order for this to continue working after the computer has been restarted at some time in the future, enter these settings into `/etc/rc.conf`:

```
gateway_enable="YES"            #for ipv4
ipv6_gateway_enable="YES"      #for ipv6
```

Use `ifconfig -a`, or `ifconfig` *`interface_name`* to find out if both of the interfaces to be used are up and running.

If all traffic initiated by machines on the inside is to be allowed, `/etc/pf.conf` could look roughly like this [2]:

```
ext_if = "xl0"  # macro for external interface - use tun0 for PPPoE
int_if = "xl1" # macro for internal interface
localnet = $int_if:network
# ext_if IP address could be dynamic, hence ($ext_if)
nat on $ext_if from $localnet to any -> ($ext_if)
block all
pass from { lo0, $localnet } to any keep state
```

Note the use of macros to assign logical names to the network interfaces. Here 3Com cards are used, but this is the last time during this tutorial we will find this of any interest whatsoever. In truly simple setups like this one, we may not gain very much by using macros like these, but once the rule sets grow somewhat larger, you will learn to appreciate the readability this provides.

Also note the `nat` rule. This is where we handle the network address translation from the non-routable address inside the local net to the sole official address we assume has been assigned.

The parentheses surrounding the last part of the nat rule `($ext_if)` are there to compensate for the possibility that the IP address of the external interface may be dynamically assigned. This detail will ensure that network traffic runs without serious interruptions even if the external IP address changes.

On the other hand, this rule set probably allows more traffic to pass out of the network than actually desired. One reasonable setup could contain the macro

```
client_out = "{ ftp-data, ftp, ssh, domain, pop3, auth, nntp, http, \
    https, cvspserver, 2628, 5999, 8000, 8080 }"
```

and the main pass rule

```
pass inet proto tcp from $localnet to any port $client_out \
    flags S/SA keep state
```

This may be a somewhat peculiar selection of ports, but it is based on a real life example. Individual needs probably differ at least in some specifics, but this should cover at least some of the more useful services.

In addition, we have a few other pass rules. We will be returning to some of the more interesting ones rather soon. One pass rule which is useful to those of us who want the ability to administer our machines from elsewhere is

```
pass in inet proto tcp to port ssh
```

or for that matter

```
pass in inet proto tcp to $ext_if port ssh
```

whichever is preferred. Lastly we need to make the name service work for our clients:

```
udp_services = "{ domain, ntp }"
```

This is supplemented with a rule which passes the traffic we want through our firewall:

```
pass quick inet proto { tcp, udp } to any port $udp_services keep state
```

Note the `quick` keyword in this rule. We have started writing rule sets which consist of several rules, and it is time to take a look at the relationships between the rules in a rule set. The rules are evaluated from top to bottom, in the sequence they are written in the configuration file. For each packet or connection evaluated by PF, *the last matching rule* in the rule set is the one which is applied. The `quick` keyword offers an escape from the ordinary sequence. When a packet matches a quick rule, the packet is treated according to the present rule. The rule processing stops without considering any further rules which might have matched the packet. This is very useful when a few isolated exceptions to the general rules are needed.

This rule also takes care of NTP, which is used for time synchronization. One thing common to both protocols is that they may under certain circumstances communicate alternately over TCP and UDP.

31.4.6.4 That Sad Old FTP Thing

The short list of real life TCP ports above contained, among other things, FTP. FTP is a sad old thing and a problem child, emphatically so for anyone trying to combine FTP and firewalls. FTP is an old and weird protocol, with a lot to not like. The most common points against it are

- Passwords are transferred in the clear
- The protocol demands the use of at least two TCP connections (control and data) on separate ports
- When a session is established, data is communicated via ports selected at random

All of these points make for challenges security-wise, even before considering any potential weaknesses in client or server software which may lead to security issues. These things have tended to happen.

Under any circumstances, other more modern and more secure options for file transfer exist, such as sftp(1) or scp(1), which feature both authentication and data transfer via encrypted connections. Competent IT professionals should have a preference for some other form of file transfer than FTP.

Regardless of our professionalism and preferences, we are all too aware that at times we will need to handle things we would prefer not to. In the case of FTP through firewalls, the main part of our handling consists of redirecting the traffic to a small program which is written specifically for this purpose.

*31.4.6.4.1 FTP Via Redirect: **ftp-proxy***

Enabling FTP transfers through your gateway is amazingly simple, thanks to the FTP proxy program (called ftp-proxy(8)) included in the base system on FreeBSD and other systems which offer PF.

The FTP protocol being what it is, the proxy needs to dynamically insert rules in your rule set. ftp-proxy(8) interacts with your configuration via a set of anchors where the proxy inserts and deletes the rules it constructs to handle your FTP traffic.

To enable ftp-proxy(8), add this line to /etc/rc.conf:

```
ftpproxy_flags=""
```

Starting the proxy manually by running /usr/sbin/ftp-proxy allows testing of the PF configuration changes we are about to make.

For a basic configuration, only three elements need to be added to /etc/pf.conf. First, the anchors:

```
nat-anchor "ftp-proxy/*"
rdr-anchor "ftp-proxy/*"
```

The proxy will insert the rules it generates for the FTP sessions here. A pass rule is needed to let FTP traffic in to the proxy.

Now for the actual redirection. Redirection rules and NAT rules fall into the same rule class. These rules may be referenced directly by other rules, and filtering rules may depend on these rules. Logically, rdr and nat rules need to be defined before the filtering rules.

We insert our rdr rule immediately after the nat rule in /etc/pf.conf

```
rdr pass on $int_if proto tcp from any to any port ftp -> 127.0.0.1 port 8021
```

In addition, the redirected traffic must be allowed to pass. We achieve this with

```
pass out proto tcp from $proxy to any port ftp
```

where $proxy expands to the address the proxy daemon is bound to.

Save pf.conf, then load the new rules with

```
# pfctl -f /etc/pf.conf
```

At this point, users will probably begin noticing that FTP works before they have been told.

This example covers a basic setup where the clients in the local net need to contact FTP servers elsewhere. The basic configuration here should work well with most combinations of FTP clients and servers. As shown in the man page, the proxy's behavior can be changed in various ways by adding options to the ftpproxy_flags= line. Some clients or servers may have specific quirks that must be compensated for in the configuration, or there may be a need to integrate the proxy in specific ways such as assigning FTP traffic to a specific queue. For these and other finer points of ftp-proxy(8) configuration, start by studying the man page.

For ways to run an FTP server protected by PF and ftp-proxy(8), look into running a separate ftp-proxy in reverse mode (using -R), on a separate port with its own redirecting pass rule.

31.4.6.5 Easing Troubleshooting

Making network troubleshooting friendly is a potentially large subject. At most times, the debugging or troubleshooting friendliness of a TCP/IP network depends on treatment of the Internet protocol which was designed specifically with debugging in mind, the *Internet Control Message Protocol*, or ICMP as it is usually abbreviated.

ICMP is the protocol for sending and receiving *control messages* between hosts and gateways, mainly to provide feedback to a sender about any unusual or difficult conditions enroute to the target host.

There is a lot of ICMP traffic which usually just happens in the background while users are surfing the web, reading mail or transferring files. Routers use ICMP to negotiate packet sizes and other transmission parameters in a process often referred to as *path MTU discovery*.

Some admins refer to ICMP as either “just evil”, or, if their understanding runs a little deeper, “a necessary evil”. The reason for this attitude is purely historical. The reason can be found a few years back when it was discovered that several operating systems contained code in their networking stack which could make a machine running one of the affected systems crash and fall over, or in some cases just do really strange things, with a sufficiently large ICMP request.

One of the companies which was hit hard was Microsoft, and you can find rather a lot of material on the “ping of death” bug by using your favorite search engine. This all happened in the second half of the 1990s, and all modern operating systems, at least the ones we can read, have thoroughly sanitized their network code since then. At least that is what we are led to believe.

One of the early workarounds was to simply block either all ICMP traffic or at least ICMP ECHO, which is what ping uses. Now these rule sets have been around for roughly fifteen years, and the people who put them there are still scared.

31.4.6.5.1 Then, Do We Let it All Through?

The obvious question then becomes, if ICMP is such a good and useful thing, should we not let it all through, all the time? The answer is “It depends”.

Letting diagnostic traffic pass unconditionally of course makes debugging easier, but also makes it relatively easy for others to extract information about your network. That means that a rule like

```
pass inet proto icmp from any to any
```

might not be optimal if the internal workings of the local network should be cloaked in a bit of mystery. In all fairness it should also be said that some ICMP traffic might be found quite harmlessly riding piggyback on `keep state` rules.

31.4.6.5.2 The Easy Way Out: the Buck Stops Here

The easiest solution could very well be to let all ICMP traffic from the local net through and stop probes from elsewhere at the gateway:

```
pass inet proto icmp from $localnet to any keep state
pass inet proto icmp from any to $ext_if keep state
```

Stopping probes at the gateway might be an attractive option anyway, but let us have a look at a few other options which will show some of PF’s flexibility.

31.4.6.5.3 Letting `ping` Through

The rule set we have developed so far has one clear disadvantage: common troubleshooting commands such as ping(8) and traceroute(8) will not work. That may not matter too much to end users, and since it was `ping` which scared people into filtering or blocking ICMP traffic in the first place, there are apparently some people who feel we are better off without it. If you are in my perceived target audience, you will be rather fond of having those

troubleshooting tools avalable. With a couple of small additions to the rule set, they will be. ping(8) uses ICMP, and in order to keep our rule set tidy, we start by defining another macro:

```
icmp_types = "echoreq"
```

and a rule which uses the definition,

```
pass inet proto icmp all icmp-type $icmp_types keep state
```

More or other types of ICMP packets may need to go through, and `icmp_types` can be expanded to a list of those packet types that are allowed.

31.4.6.5.4 Helping traceroute(8)

traceroute(8) is another command which is quite useful when users claim that the Internet is not working. By default, Unix `traceroute` uses UDP connections according to a set formula based on destination. The rule below works with `traceroute` on all unixes I've had access to, including GNU/Linux:

```
# allow out the default range for traceroute(8):
# "base+nhops*nqueries-1" (33434+64*3-1)
pass out on $ext_if inet proto udp from any to any port 33433 >< 33626 keep state
```

Experience so far indicates that `traceroute` implementations on other operating systems work roughly the same. Except, of course, on Microsoft Windows. On that platform, `TRACERT.EXE` uses ICMP ECHO for this purpose. So to let Windows traceroutes through, only the first rule is needed. Unix `traceroute` can be instructed to use other protocols as well, and will behave remarkably like its Microsoft counterpart if `-I` is used. Check the traceroute(8) man page (or its source code, for that matter) for all the details.

Under any circumstances, this solution was lifted from an openbsd-misc post. I've found that list, and the searchable list archives (accessible among other places from http://marc.theaimsgroup.com/), to be a very valuable resource whenever you need OpenBSD or PF related information.

31.4.6.5.5 Path MTU Discovery

Internet protocols are designed to be device independent, and one consequence of device independence is that the optimal packet size for a given connection cannot always be predicted reliably. The main constraint on packet size is called the *Maximum Transmission Unit*, or MTU, which sets the upper limit on the packet size for an interface. ifconfig(8) shows the MTU for the network interfaces.

Modern TCP/IP implementations expect to be able to determine the right packet size for a connection through a process which, simply put, involves sending packets of varying sizes with the "Do not fragment" flag set, expecting an ICMP return packet indicating "type 3, code 4" when the upper limit has been reached. Now do not dive for the RFCs right away. Type 3 means "destination unreachable", while code 4 is short for "fragmentation needed, but the do-not-fragment flag is set". So if connections to networks which may have other MTUs than the local network seem sub-optimal, and there is no need to be that specific, the list of ICMP types can be changed slightly to let the "destination unreachable" packets through, too:

```
icmp_types = "{ echoreq, unreach }"
```

As we can see, this means we do not need to change the pass rule itself:

```
pass inet proto icmp all icmp-type $icmp_types keep state
```

PF allows filtering on all variations of ICMP types and codes. For those who want to delve into what to pass (or not) of ICMP traffic, the list of possible types and codes are documented in the icmp(4) and icmp6(4) man pages. The background information is available in the RFCs [3].

31.4.6.6 Tables Make Life Easier

By this time it may appear that this gets awfully static and rigid. There will after all be some kinds of data which are relevant to filtering and redirection at a given time, but do not deserve to be put into a configuration file! Quite right, and PF offers mechanisms for handling these situations as well. Tables are one such feature, mainly useful as lists which can be manipulated without needing to reload the entire rule set, and where fast lookups are desirable. Table names are always enclosed in `< >`, like this:

```
table <clients> { 192.168.2.0/24, !192.168.2.5 }
```

Here, the network `192.168.2.0/24` is part of the table, except the address `192.168.2.5`, which is excluded using the `!` operator (logical NOT). It is also possible to load tables from files where each item is on a separate line, such as the file `/etc/clients`.

```
192.168.2.0/24
!192.168.2.5
```

which in turn is used to initialize the table in `/etc/pf.conf`:

```
table <clients> persist file /etc/clients
```

Then, for example, one of our earlier rules can be changed to read

```
pass inet proto tcp from <clients> to any port $client_out flags S/SA keep state
```

to manage outgoing traffic from client computers. With this in hand, the table's contents can be manipulated live, such as

```
# pfctl -t clients -T add 192.168.1/16
```

Note that this changes the in-memory copy of the table only, meaning that the change will not survive a power failure or other reboot unless there are arrangements to store the changes.

One might opt to maintain the on-disk copy of the table using a cron(8) job which dumps the table content to disk at regular intervals, using a command such as `pfctl -t clients -T show >/etc/clients`. Alternatively, `/etc/clients` could be edited, replacing the in-memory table contents with the file data:

```
# pfctl -t clients -T replace -f /etc/clients
```

For operations performed frequently, administrators will sooner or later end up writing shell scripts for tasks such as inserting or removing items or replacing table contents. The only real limitations lie in individual needs and creativity.

31.4.6.7 Overload Tables

Those who run a Secure Shell login service which is accessible from the Internet have probably seen something like this in the authentication logs:

```
Sep 26 03:12:34 skapet sshd[25771]: Failed password for root from 200.72.41.31 port 40992 ssh2
Sep 26 03:12:34 skapet sshd[5279]: Failed password for root from 200.72.41.31 port 40992 ssh2
Sep 26 03:12:35 skapet sshd[5279]: Received disconnect from 200.72.41.31: 11: Bye Bye
Sep 26 03:12:44 skapet sshd[29635]: Invalid user admin from 200.72.41.31
Sep 26 03:12:44 skapet sshd[24703]: input_userauth_request: invalid user admin
Sep 26 03:12:44 skapet sshd[24703]: Failed password for invalid user admin from 200.72.41.31 port
```

And so on. This is what a brute force attack looks like. Essentially somebody, or more likely, a cracked computer somewhere, is trying by brute force to find a combination of user name and password which will let them into your system.

The simplest response would be to write a `pf.conf` rule which blocks all access. This leads to another class of problems, including what to do in order to let people with legitimate business on the system access it anyway. Some might consider moving the service to another port, but then again, the ones flooding on port 22 would probably be able to scan their way to port 22222 for a repeat performance.

Since OpenBSD 3.7, and soon after in FreeBSD version 6.0, PF has offered a slightly more elegant solution. Pass rules can be written so they maintain certain limits on what connecting hosts can do. For good measure, violators can be banished to a table of addresses which are denied some or all access. If desired, it's even possible to drop all existing connections from machines which overreach the limits. Here is how it is done:

First, set up the table. In the tables section, add

```
table <bruteforce> persist
```

Then somewhere fairly early in the rule set, add a rule to block the bruteforcers:

```
block quick from <bruteforce>
```

And finally, the pass rule.

```
pass inet proto tcp from any to $localnet port $tcp_services \
    flags S/SA keep state \
    (max-src-conn 100, max-src-conn-rate 15/5, \
    overload <bruteforce> flush global)
```

The first part here is identical to the main rule we constructed earlier. The part in parentheses is the new stuff which will ease network load even further.

`max-src-conn` is the number of simultaneous connections allowed from one host. In this example, it is set at 100. Other setups may want a slightly higher or lower value.

`max-src-conn-rate` is the rate of new connections allowed from any single host, here 15 connections per 5 seconds. Again, the administrator is the one to judge what suits their setup.

`overload <bruteforce>` means that any host which exceeds these limits gets its address added to the table `bruteforce`. Our rule set blocks all traffic from addresses in the bruteforce table.

Finally, `flush global` says that when a host reaches the limit, that host's connections will be terminated (flushed). The global part says that for good measure, this applies to connections which match other pass rules too.

The effect is dramatic. From here on, bruteforcers more often than not will end up with `"Fatal: timeout before authentication"` messages, getting nowhere.

> **Note:** These rules will *not* block slow bruteforcers, sometimes referred to as the Hail Mary Cloud (http://home.nuug.no/~peter/hailmary2013/).

Once again, please keep in mind that this example rule is intended mainly as an illustration. It is not unlikely that a particular network's needs are better served by rather different rules or combinations of rules.

If, for example, a generous number of connections in general are wanted, but the desire is to be a little more tight fisted when it comes to **ssh**, supplement the rule above with something like the one below, early on in the rule set:

```
pass quick proto { tcp, udp } from any to any port ssh \
    flags S/SA keep state \
    (max-src-conn 15, max-src-conn-rate 5/3, \
    overload <bruteforce> flush global)
```

It should be possible to find the set of parameters which is just right for individual situations by reading the relevant man pages and the PF User Guide (http://www.openbsd.org/faq/pf/), and perhaps a bit of experimentation.

> **It May Not be Necessary to Block All Overloaders:** It is probably worth noting at this point that the *overload* mechanism is a general technique which does not have to apply exclusively to the *ssh* service, and it is not always optimal to block all traffic from offenders entirely.
>
> For example, an overload rule could be used to protect a mail service or a web service, and the overload table could be used in a rule to assign offenders to a queue with a minimal bandwidth allocation or, in the web case, to redirect to a specific web page.

*31.4.6.7.1 Expiring Table Entries with **pfctl***

At this point, we have tables which will be filled by our `overload` rules, and since we could reasonably expect our gateways to have months of uptime, the tables will grow incrementally, taking up more memory as time goes by.

Sometimes an IP address that was blocked last week due to a brute force attack was in fact a dynamically assigned one, which is now assigned to a different ISP customer who has a legitimate reason to try communicating with hosts in the local network.

Situations like these were what caused Henning Brauer to add to **pfctl** the ability to expire table entries not referenced in a specified number of seconds (in OpenBSD 4.1). For example, the command

```
# pfctl -t bruteforce -T expire 86400
```

will remove `<bruteforce>` table entries which have not been referenced for 86400 seconds.

*31.4.6.7.2 The **expiretable** Tool*

Before **pfctl** acquired the ability to expire table entries, Henrik Gustafsson had written **expiretable**, which removes table entries which have not been accessed for a specified period of time.

One useful example is to use the **expiretable** program as a way of removing outdated `<bruteforce>` table entries.

For example, let **expiretable** remove `<bruteforce>` table entries older than 24 hours by adding an entry containing the following to `/etc/rc.local`:

```
/usr/local/sbin/expiretable -v -d -t 24h bruteforce
```

expiretable is in the Ports Collection on FreeBSD as `security/expiretable`.

31.4.6.8 Other PF Tools

Over time, a number of tools have been developed which interact with PF in various ways.

*31.4.6.8.1 The **pftop** Traffic Viewer*

Can Erkin Acar's **pftop** makes it possible to keep an eye on what passes into and out of the network. **pftop** is available through the ports system as `sysutils/pftop`. The name is a strong hint at what it does - **pftop** shows a running snapshot of traffic in a format which is strongly inspired by top(1).

*31.4.6.8.2 The **spamd** Spam Deferral Daemon*

Not to be confused with the **spamd** daemon which comes bundled with **spamassassin**, the PF companion **spamd** was designed to run on a PF gateway to form part of the outer defense against spam. **spamd** hooks into the PF configuration via a set of redirections.

The main point underlying the **spamd** design is the fact that spammers send a large number of messages, and the probability that you are the first person receiving a particular message is incredibly small. In addition, spam is mainly sent via a few spammer friendly networks and a large number of hijacked machines. Both the individual messages and the machines will be reported to blacklists fairly quickly, and this is the kind of data **spamd** can use to our advantage with *blacklists*.

What **spamd** does to SMTP connections from addresses in the blacklist is to present its banner and immediately switch to a mode where it answers SMTP traffic one byte at the time. This technique, which is intended to waste as much time as possible on the sending end while costing the receiver pretty much nothing, is called *tarpitting*. The specific implementation with one byte SMTP replies is often referred to as *stuttering*.

*31.4.6.8.2.1 A Basic Blacklisting **spamd***

Here is the basic procedure for setting up **spamd** with automatically updated blacklists:

1. Install the `mail/spamd/` port. In particular, be sure to read the package message and act upon what it says. Specifically, to use **spamd**'s greylisting features, a file descriptor file system (see fdescfs(5) (http://www.freebsd.org/cgi/man.cgi?query=fdescfs&sektion=5)) must be mounted at `/dev/fd/`. Do this by adding the following line to `/etc/fstab`:

   ```
   fdescfs /dev/fd fdescfs rw 0 0
   ```

   Make sure the `fdescfs` code is in the kernel, either compiled in or by loading the module with kldload(8).

2. Next, edit the rule set to include

   ```
   table <spamd> persist
   table <spamd-white> persist
   ```

```
rdr pass on $ext_if inet proto tcp from <spamd> to \
    { $ext_if, $localnet } port smtp -> 127.0.0.1 port 8025
rdr pass on $ext_if inet proto tcp from !<spamd-white> to \
    { $ext_if, $localnet } port smtp -> 127.0.0.1 port 8025
```

The two tables <spamd> and <spamd-white> are essential. SMTP traffic from the addresses in the first table plus the ones which are not in the other table are redirected to a daemon listening at port 8025.

3. The next step is to set up **spamd**'s own configuration in `/usr/local/etc/spamd.conf` supplemented by `rc.conf` parameters.

   The supplied sample file offers quite a bit of explanation, and the man page offers additional information, but we will recap the essentials here.

   One of the first lines without a # comment sign at the start contains the block which defines the `all` list, which specifies the lists actually used:

   ```
   all:\
       :traplist:whitelist:
   ```

   Here, all the desired black lists are added, separated by colons (`:`). To use whitelists to subtract addresses from the blacklist, add the name of the whitelist immediately after the name of each blacklist, i.e., `:blacklist:whitelist:`.

   Next up is a blacklist definition:

   ```
   traplist:\
       :black:\
       :msg="SPAM. Your address %A has sent spam within the last 24 hours":\
       :method=http:\
       :file=www.openbsd.org/spamd/traplist.gz
   ```

   Following the name, the first data field specifies the list type, in this case `black`. The `msg` field contains the message to display to blacklisted senders during the SMTP dialogue. The `method` field specifies how spamd-setup fetches the list data, here `http`. The other options are fetching via `ftp`, from a `file` in a mounted file system or via `exec` of an external program. Finally the `file` field specifies the name of the file spamd expects to receive.

   The definition of a whitelist follows much the same pattern:

   ```
   whitelist:\
       :white:\
       :method=file:\
       :file=/var/mail/whitelist.txt
   ```

   but omits the message parameters since a message is not needed.

   > **Choose Data Sources with Care:** Using all the blacklists in the sample `spamd.conf` will end up blacklisting large blocks of the Internet, including several Asian nations. Administrators need to edit the file to end up with an optimal configuration. The administrator is the judge of which data sources to use, and using lists other than the ones suggested in the sample file is possible.

   Put the lines for spamd and any startup parameters desired in `/etc/rc.conf`, for example:

   ```
   spamd_flags="-v" # for normal use: "" and see spamd-setup(8)
   ```

When done with editing the setup, reload the rule set, start **spamd** with the options desired using the `/usr/local/etc/rc.d/obspamd` script, and complete the configuration using `spamd-setup`. Finally, create a cron(8) job which calls `spamd-setup` to update the tables at reasonable intervals.

On a typical gateway in front of a mail server, hosts will start getting trapped within a few seconds to several minutes.

*31.4.6.8.2.2 Adding Greylisting to the **spamd** Setup*

spamd also supports *greylisting*, which works by rejecting messages from unknown hosts temporarily with *45n* codes, letting messages from hosts which try again within a reasonable time through. Traffic from well behaved hosts, that is, senders which are set up to behave within the limits set up in the relevant RFCs [4], will be let through.

Greylisting as a technique was presented in a 2003 paper by Evan Harris [5], and a number of implementations followed over the next few months. OpenBSD's **spamd** acquired its ability to greylist in OpenBSD 3.5, which was released in May 2004.

The most amazing thing about greylisting, apart from its simplicity, is that it still works. Spammers and malware writers have been very slow to adapt.

The basic procedure for adding greylisting to your setup follows below.

1. If not done already, make sure the file descriptor file system (see fdescfs(5)) is mounted at `/dev/fd/`. Do this by adding the following line to `/etc/fstab`:

   ```
   fdescfs /dev/fd fdescfs rw 0 0
   ```

   and make sure the fdescfs(5) code is in the kernel, either compiled in or by loading the module with kldload(8).

2. To run **spamd** in greylisting mode, `/etc/rc.conf` must be changed slightly by adding

   ```
   spamd_grey="YES"  # use spamd greylisting if YES
   ```

   Several greylisting related parameters can be fine-tuned with `spamd`'s command line parameters and the corresponding `/etc/rc.conf` settings. Check the **spamd** man page to see what the parameters mean.

3. To complete the greylisting setup, restart **spamd** using the `/usr/local/etc/rc.d/obspamd` script.

Behind the scenes, rarely mentioned and barely documented are two of **spamd**'s helpers, the **spamdb** database tool and the **spamlogd** whitelist updater, which both perform essential functions for the greylisting feature. Of the two **spamlogd** works quietly in the background, while **spamdb** has been developed to offer some interesting features.

> **Restart spamd to Enable Greylisting:** After following all steps in the tutorial exactly up to this point, **spamlogd** has been started automatically already. However, if the initial **spamd** configuration did not include greylisting, **spamlogd** may not have been started, and there may be strange symptoms, such as greylists and whitelists not getting updated properly.
>
> Under normal circumstances, it should not be necessary to start **spamlogd** by hand. Restarting **spamd** after enabling greylisting ensures **spamlogd** is loaded and available too.

spamdb is the administrator's main interface to managing the black, grey and white lists via the contents of the `/var/db/spamdb` database.

31.4.6.8.3 Network Hygiene: Blocking, Scrubbing and so On

Our gateway does not feel quite complete without a few more items in the configuration which will make it behave a bit more sanely towards hosts on the wide net and our local network.

31.4.6.8.3.1 `block-policy`

`block-policy` is an option which can be set in the `options` part of the ruleset, which precedes the redirection and filtering rules. This option determines which feedback, if any, PF will give to hosts which try to create connections which are subsequently blocked. The option has two possible values, `drop`, which drops blocked packets with no feedback, and `return`, which returns with status codes such as `Connection refused` or similar.

The correct strategy for block policies has been the subject of rather a lot of discussion. We choose to play nicely and instruct our firewall to issue returns:

```
set block-policy return
```

31.4.6.8.3.2 `scrub`

In PF versions up to OpenBSD 4.5 inclusive, `scrub` is a keyword which enables network packet normalization, causing fragmented packets to be assembled and removing ambiguity. Enabling `scrub` provides a measure of protection against certain kinds of attacks based on incorrect handling of packet fragments. A number of supplementing options are available, but we choose the simplest form which is suitable for most configurations.

```
scrub in all
```

Some services, such as NFS, require some specific fragment handling options. This is extensively documented in the PF user guide and man pages provide all the information you could need.

One fairly common example is this,

```
scrub in all fragment reassemble no-df max-mss 1440
```

meaning, we reassemble fragments, clear the “do not fragment” bit and set the maximum segment size to 1440 bytes. Other variations are possible, and you should be able to cater to various specific needs by consulting the man pages and some experimentation.

31.4.6.8.3.3 `antispoof`

`antispoof` is a common special case of filtering and blocking. This mechanism protects against activity from spoofed or forged IP addresses, mainly by blocking packets appearing on interfaces and in directions which are logically not possible.

We specify that we want to weed out spoofed traffic coming in from the rest of the world and any spoofed packets which, however unlikely, were to originate in our own network:

```
antispoof for $ext_if
antispoof for $int_if
```

31.4.6.8.3.4 Handling Non-Routable Addresses from Elsewhere

Even with a properly configured gateway to handle network address translation for your own network, you may find yourself in the unenviable position of having to compensate for other people's misconfigurations.

One depressingly common class of misconfigurations is the kind which lets traffic with non-routable addresses out to the Internet. Traffic from non-routeable addresses have also played a part in several DOS attack techniques, so it may be worth considering explicitly blocking traffic from non-routeable addresses from entering your network.

One possible solution is the one outlined below, which for good measure also blocks any attempt to initiate contact to non-routable addresses through the gateway's external interface:

```
martians = "{ 127.0.0.0/8, 192.168.0.0/16, 172.16.0.0/12, \
              10.0.0.0/8, 169.254.0.0/16, 192.0.2.0/24, \
              0.0.0.0/8, 240.0.0.0/4 }"

block drop in quick on $ext_if from $martians to any
block drop out quick on $ext_if from any to $martians
```

Here, the `martians` macro denotes the RFC 1918 addresses and a few other ranges which are mandated by various RFCs not to be in circulation on the open Internet. Traffic to and from such addresses is quietly dropped on the gateway's external interface.

The specific details of how to implement this kind of protection will vary, among other things according to your specific network configuration. Your network design could for example dictate that you include or exclude other address ranges than these.

This completes our simple NATing firewall for a small local network. A more thorough tutorial is available at http://home.nuug.no/~peter/pf/, where you will also find slides from related presentations.

31.5 The IPFILTER (IPF) Firewall

IPFILTER is a cross-platform, open source firewall which has been ported to FreeBSD, NetBSD, OpenBSD, SunOS, HP/UX, and Solaris operating systems.

IPFILTER is based on a kernel-side firewall and NAT mechanism that can be controlled and monitored by userland interface programs. The firewall rules can be set or deleted using ipf(8). The NAT rules can be set or deleted using ipnat(8). Run-time statistics for the kernel parts of IPFILTER can be printed using ipfstat(8). To log IPFILTER actions to the system log files, use ipmon(8).

IPF was originally written using a rule processing logic of "the last matching rule wins" and only used stateless rules. Over time, IPF has been enhanced to include a "quick" option and a stateful "keep state" option which modernized the rules processing logic. IPF's official documentation covers only the legacy rule coding parameters and rule file processing logic and the modernized functions are only included as additional options.

The instructions contained in this section are based on using rules that contain "quick" and "keep state" as these provide the basic framework for configuring an inclusive firewall ruleset.

For a detailed explanation of the legacy rules processing method, refer to http://www.munk.me.uk/ipf/ipf-howto.html and http://coombs.anu.edu.au/~avalon/ip-filter.html.

The IPF FAQ is at http://www.phildev.net/ipf/index.html.

A searchable archive of the IPFilter mailing list is available at http://marc.theaimsgroup.com/?l=ipfilter.

31.5.1 Enabling IPF

IPF is included in the basic FreeBSD install as a kernel loadable module. The system will dynamically load this module at boot time when `ipfilter_enable="YES"` is added to `rc.conf`. The module enables logging and `default pass all`. To change the default to `block all`, add a `block all` rule at the end of the ruleset.

31.5.2 Kernel Options

For users who prefer to statically compile IPF support into a custom kernel, the following IPF option statements, listed in `/usr/src/sys/conf/NOTES`, are available:

```
options IPFILTER
options IPFILTER_LOG
options IPFILTER_DEFAULT_BLOCK
```

`options IPFILTER` enables support for the “IPFILTER” firewall.

`options IPFILTER_LOG` enables IPF logging using the `ipl` packet logging pseudo—device for every rule that has the `log` keyword.

`options IPFILTER_DEFAULT_BLOCK` changes the default behavior so that any packet not matching a firewall `pass` rule gets blocked.

These settings will take effect only after installing a kernel that has been built with the above options set.

31.5.3 Available `rc.conf` Options

To activate IPF at boot time, the following statements need to be added to `/etc/rc.conf`:

```
ipfilter_enable="YES"             # Start ipf firewall
ipfilter_rules="/etc/ipf.rules"   # loads rules definition text file
ipmon_enable="YES"                # Start IP monitor log
ipmon_flags="-Ds"                 # D = start as daemon
                                  # s = log to syslog
                                  # v = log tcp window, ack, seq
                                  # n = map IP & port to names
```

If there is a LAN behind the firewall that uses the reserved private IP address ranges, the following lines have to be added to enable NAT functionality:

```
gateway_enable="YES"              # Enable as LAN gateway
ipnat_enable="YES"                # Start ipnat function
ipnat_rules="/etc/ipnat.rules"    # rules definition file for ipnat
```

31.5.4 IPF

To load the ruleset file, use ipf(8). Custom rules are normally placed in a file, and the following command can be used to replace the currently running firewall rules:

```
# ipf -Fa -f /etc/ipf.rules
```

`-Fa` flushes all the internal rules tables.

`-f` specifies the file containing the rules to load.

This provides the ability to make changes to a custom rules file, run the above IPF command, and thus update the running firewall with a fresh copy of the rules without having to reboot the system. This method is convenient for testing new rules as the procedure can be executed as many times as needed.

Refer to ipf(8) for details on the other flags available with this command.

ipf(8) expects the rules file to be a standard text file. It will not accept a rules file written as a script with symbolic substitution.

There is a way to build IPF rules that utilize the power of script symbolic substitution. For more information, see Section 31.5.9.

31.5.5 IPFSTAT

The default behavior of ipfstat(8) is to retrieve and display the totals of the accumulated statistics gathered by applying the rules against packets going in and out of the firewall since it was last started, or since the last time the accumulators were reset to zero using `ipf -Z`.

Refer to ipfstat(8) for details.

The default ipfstat(8) output will look something like this:

```
input packets: blocked 99286 passed 1255609 nomatch 14686 counted 0
 output packets: blocked 4200 passed 1284345 nomatch 14687 counted 0
 input packets logged: blocked 99286 passed 0
 output packets logged: blocked 0 passed 0
 packets logged: input 0 output 0
 log failures: input 3898 output 0
 fragment state(in): kept 0 lost 0
 fragment state(out): kept 0 lost 0
 packet state(in): kept 169364 lost 0
 packet state(out): kept 431395 lost 0
 ICMP replies: 0 TCP RSTs sent: 0
 Result cache hits(in): 1215208 (out): 1098963
 IN Pullups succeeded: 2 failed: 0
 OUT Pullups succeeded: 0 failed: 0
 Fastroute successes: 0 failures: 0
 TCP cksum fails(in): 0 (out): 0
 Packet log flags set: (0)
```

When supplied with either `-i` for inbound or `-o` for outbound, the command will retrieve and display the appropriate list of filter rules currently installed and in use by the kernel.

`ipfstat -in` displays the inbound internal rules table with rule numbers.

`ipfstat -on` displays the outbound internal rules table with rule numbers.

The output will look something like this:

```
@1 pass out on xl0 from any to any
@2 block out on dc0 from any to any
@3 pass out quick on dc0 proto tcp/udp from any to any keep state
```

`ipfstat -ih` displays the inbound internal rules table, prefixing each rule with a count of how many times the rule was matched.

`ipfstat -oh` displays the outbound internal rules table, prefixing each rule with a count of how many times the rule was matched.

The output will look something like this:

```
2451423 pass out on xl0 from any to any
354727 block out on dc0 from any to any
430918 pass out quick on dc0 proto tcp/udp from any to any keep state
```

One of the most important options of `ipfstat` is `-t` which displays the state table in a way similar to how top(1) shows the FreeBSD running process table. When a firewall is under attack, this function provides the ability to identify and see the attacking packets. The optional sub-flags give the ability to select the destination or source IP, port, or protocol to be monitored in real time. Refer to ipfstat(8) for details.

31.5.6 IPMON

In order for `ipmon` to work properly, the kernel option `IPFILTER_LOG` must be turned on. This command has two different modes. Native mode is the default mode when the command is used without `-D`.

Daemon mode provides a continuous system log file so that logging of past events may be reviewed. FreeBSD has a built in facility to automatically rotate system logs. This is why outputting the log information to syslogd(8) is better than the default of outputting to a regular file. The default `rc.conf` `ipmon_flags` statement uses `-Ds`:

```
ipmon_flags="-Ds" # D = start as daemon
                  # s = log to syslog
                  # v = log tcp window, ack, seq
                  # n = map IP & port to names
```

Logging provides the ability to review, after the fact, information such as which packets were dropped, what addresses they came from and where they were going. These can all provide a significant edge in tracking down attackers.

Even with the logging facility enabled, IPF will not generate any rule logging by default. The firewall administrator decides which rules in the ruleset should be logged and adds the log keyword to those rules. Normally, only deny rules are logged.

It is customary to include a “default deny everything” rule with the log keyword included as the last rule in the ruleset. This makes it possible to see all the packets that did not match any of the rules in the ruleset.

31.5.7 IPMON Logging

syslogd(8) uses its own method for segregation of log data. It uses groupings called “facility” and “level”. By default, IPMON in `-Ds` mode uses `local0` as the “facility” name. The following levels can be used to further segregate the logged data:

```
LOG_INFO - packets logged using the "log" keyword as the action rather than pass or block.
LOG_NOTICE - packets logged which are also passed
LOG_WARNING - packets logged which are also blocked
LOG_ERR - packets which have been logged and which can be considered short
```

In order to setup IPFILTER to log all data to `/var/log/ipfilter.log`, first create the empty file:

```
# touch /var/log/ipfilter.log
```

syslogd(8) is controlled by definition statements in `/etc/syslog.conf`. This file offers considerable flexibility in how **syslog** will deal with system messages issued by software applications like IPF.

To write all logged messages to the specified file, add the following statement to `/etc/syslog.conf`:

```
local0.* /var/log/ipfilter.log
```

To activate the changes and instruct syslogd(8) to read the modified `/etc/syslog.conf`, run `service syslogd reload`.

Do not forget to change `/etc/newsyslog.conf` to rotate the new log file.

31.5.8 The Format of Logged Messages

Messages generated by `ipmon` consist of data fields separated by white space. Fields common to all messages are:

1. The date of packet receipt.
2. The time of packet receipt. This is in the form HH:MM:SS.F, for hours, minutes, seconds, and fractions of a second.
3. The name of the interface that processed the packet.
4. The group and rule number of the rule in the format `@0:17`.

These can be viewed with `ipfstat -in`.

1. The action: `p` for passed, `b` for blocked, `S` for a short packet, `n` did not match any rules, and `L` for a log rule. The order of precedence in showing flags is: `S`, `p`, `b`, `n`, `L`. A capital `P` or `B` means that the packet has been logged due to a global logging setting, not a particular rule.
2. The addresses written as three fields: the source address and port separated by a comma, the -> symbol, and the destination address and port. For example: `209.53.17.22,80 -> 198.73.220.17,1722`.
3. `PR` followed by the protocol name or number: for example, `PR tcp`.
4. `len` followed by the header length and total length of the packet: for example, `len 20 40`.

If the packet is a TCP packet, there will be an additional field starting with a hyphen followed by letters corresponding to any flags that were set. Refer to ipf(5) for a list of letters and their flags.

If the packet is an ICMP packet, there will be two fields at the end: the first always being “ICMP” and the next being the ICMP message and sub-message type, separated by a slash. For example: ICMP 3/3 for a port unreachable message.

31.5.9 Building the Rule Script with Symbolic Substitution

Some experienced IPF users create a file containing the rules and code them in a manner compatible with running them as a script with symbolic substitution. The major benefit of doing this is that only the value associated with the symbolic name needs to be changed, and when the script is run all the rules containing the symbolic name will have the value substituted in the rules. Being a script, symbolic substitution can be used to code frequently used values and substitute them in multiple rules. This can be seen in the following example.

The script syntax used here is compatible with the sh(1), csh(1), and tcsh(1) shells.

Symbolic substitution fields are prefixed with a $.

Symbolic fields do not have the $ prefix.

The value to populate the symbolic field must be enclosed between double quotes (").

Start the rule file with something like this:

```
############# Start of IPF rules script ########################

oif="dc0"            # name of the outbound interface
odns="192.0.2.11"    # ISP’s DNS server IP address
myip="192.0.2.7"     # my static IP address from ISP
ks="keep state"
fks="flags S keep state"

# You can choose between building /etc/ipf.rules file
# from this script or running this script "as is".
#
# Uncomment only one line and comment out another.
#
# 1) This can be used for building /etc/ipf.rules:
#cat > /etc/ipf.rules << EOF
#
# 2) This can be used to run script "as is":
/sbin/ipf -Fa -f - << EOF

# Allow out access to my ISP’s Domain name server.
pass out quick on $oif proto tcp from any to $odns port = 53 $fks
pass out quick on $oif proto udp from any to $odns port = 53 $ks

# Allow out non-secure standard www function
pass out quick on $oif proto tcp from $myip to any port = 80 $fks

# Allow out secure www function https over TLS SSL
pass out quick on $oif proto tcp from $myip to any port = 443 $fks
EOF
################## End of IPF rules script ######################
```

The rules are not important in this example as it instead focuses on how the symbolic substitution fields are populated. If this example was in a file named `/etc/ipf.rules.script`, these rules could be reloaded by running:

```
# sh /etc/ipf.rules.script
```

There is one problem with using a rules file with embedded symbolics: IPF does not understand symbolic substitution, and cannot read such scripts directly.

This script can be used in one of two ways:

- Uncomment the line that begins with `cat`, and comment out the line that begins with `/sbin/ipf`. Place `ipfilter_enable="YES"` into `/etc/rc.conf`, and run the script once after each modification to create or update `/etc/ipf.rules`.
- Disable IPFILTER in the system startup scripts by adding `ipfilter_enable="NO"`to `/etc/rc.conf`.

  Then, add a script like the following to `/usr/local/etc/rc.d/`. The script should have an obvious name like `ipf.loadrules.sh`, where the `.sh` extension is mandatory.

  ```
  #!/bin/sh
  sh /etc/ipf.rules.script
  ```

  The permissions on this script file must be read, write, execute for owner `root`:

  ```
  # chmod 700 /usr/local/etc/rc.d/ipf.loadrules.sh
  ```

Now, when the system boots, the IPF rules will be loaded.

31.5.10 IPF Rulesets

A ruleset contains a group of IPF rules which pass or block packets based on the values contained in the packet. The bi-directional exchange of packets between hosts comprises a session conversation. The firewall ruleset processes both the packets arriving from the public Internet, as well as the packets produced by the system as a response to them. Each TCP/IP service is predefined by its protocol and listening port. Packets destined for a specific service originate from the source address using an unprivileged port and target the specific service port on the destination address. All the above parameters can be used as selection criteria to create rules which will pass or block services.

> **Warning:** When working with the firewall rules, be *very careful*. Some configurations *can lock the administrator out* of the server. To be on the safe side, consider performing the initial firewall configuration from the local console rather than doing it remotely over **ssh**.

31.5.11 Rule Syntax

The rule syntax presented here has been simplified to only address the modern stateful rule context and “first matching rule wins” logic. For the complete legacy rule syntax, refer to ipf(8).

A # character is used to mark the start of a comment and may appear at the end of a rule line or on its own line. Blank lines are ignored.

Rules contain keywords which must be written in a specific order from left to right on the line. Keywords are identified in bold type. Some keywords have sub-options which may be keywords themselves and also include more sub-options. Each of the headings in the below syntax has a bold section header which expands on the content.

ACTION IN-OUT OPTIONS SELECTION STATEFUL PROTO SRC_ADDR,DST_ADDR OBJECT PORT_NUM TCP_FLAG STATEFUL

ACTION = block | pass

IN-OUT = in | out

OPTIONS = log | quick | on interface-name

SELECTION = proto value | source/destination IP | port = number | flags flag-value

PROTO = tcp/udp | udp | tcp | icmp

SRC_ADD,DST_ADDR = all | from object to object

OBJECT = IP address | any

PORT_NUM = port number

TCP_FLAG = S

STATEFUL = keep state

31.5.11.1 ACTION

The action keyword indicates what to do with the packet if it matches the rest of the filter rule. Each rule *must* have an action. The following actions are recognized:

`block` indicates that the packet should be dropped if the selection parameters match the packet.

`pass` indicates that the packet should exit the firewall if the selection parameters match the packet.

31.5.11.2 IN-OUT

A mandatory requirement is that each filter rule explicitly state which side of the I/O it is to be used on. The next keyword must be either `in` or `out` and one or the other has to be included or the rule will not pass syntax checks.

`in` means this rule is being applied against an inbound packet which has just been received on the interface facing the public Internet.

`out` means this rule is being applied against an outbound packet destined for the interface facing the public Internet.

31.5.11.3 OPTIONS

> **Note:** These options must be used in the order shown here.

`log` indicates that the packet header will be written to the ipl(4) packet log pseudo-device if the selection parameters match the packet.

`quick` indicates that if the selection parameters match the packet, this rule will be the last rule checked, and no further processing of any following rules will occur for this packet.

on indicates the interface name to be incorporated into the selection parameters. Interface names are as displayed by ifconfig(8). Using this option, the rule will only match if the packet is going through that interface in the specified direction.

When a packet is logged, the headers of the packet are written to the ipl(4) packet logging pseudo-device. Immediately following the log keyword, the following qualifiers may be used in this order:

body indicates that the first 128 bytes of the packet contents will be logged after the headers.

first. If the log keyword is being used in conjunction with a keep state option, this option is recommended so that only the triggering packet is logged and not every packet which matches the stateful connection.

31.5.11.4 SELECTION

The keywords described in this section are used to describe attributes of the packet to be checked when determining whether or not rules match. There is a keyword subject, and it has sub-option keywords, one of which has to be selected. The following general-purpose attributes are provided for matching, and must be used in this order:

31.5.11.5 PROTO

proto is the subject keyword which must include one of its corresponding keyword sub-option values. The sub-option indicates a specific protocol to be matched against.

tcp/udp | udp | tcp | icmp or any protocol names found in /etc/protocols are recognized and may be used. The special protocol keyword tcp/udp may be used to match either a TCP or a UDP packet, and has been added as a convenience to save duplication of otherwise identical rules.

31.5.11.6 SRC_ADDR/DST_ADDR

The all keyword is equivalent to “from any to any” with no other match parameters.

from | to src to dst: the from and to keywords are used to match against IP addresses. Rules must specify *both* the source and destination parameters. any is a special keyword that matches any IP address. Examples include: from any to any, from 0.0.0.0/0 to any, from any to 0.0.0.0/0, from 0.0.0.0 to any, and from any to 0.0.0.0.

There is no way to match ranges of IP addresses which do not express themselves easily using the dotted numeric form / mask-length notation. The net-mgmt/ipcalc port may be used to ease the calculation. Additional information is available at the utility’s web page: http://jodies.de/ipcalc.

31.5.11.7 PORT

If a port match is included, for either or both of source and destination, it is only applied to TCP and UDP packets. When composing port comparisons, either the service name from /etc/services or an integer port number may be used. When the port appears as part of the from object, it matches the source port number. When it appears as part of the to object, it matches the destination port number. An example usage is from any to any port = 80

Single port comparisons may be done in a number of ways, using a number of different comparison operators. Instead of the = shown in the example above, the following operators may be used: !=, <, >, <=, >=, eq, ne, lt, gt, le, and ge.

To specify port ranges, place the two port numbers between <> or ><

31.5.11.8 TCP_FLAG

Flags are only effective for TCP filtering. The letters represent one of the possible flags that can be matched against the TCP packet header.

The modernized rules processing logic uses the `flags S` parameter to identify the TCP session start request.

31.5.11.9 STATEFUL

`keep state` indicates that on a pass rule, any packets that match the rules selection parameters should activate the stateful filtering facility.

31.5.12 Stateful Filtering

Stateful filtering treats traffic as a bi-directional exchange of packets comprising a session. When activated, `keep-state` dynamically generates internal rules for each anticipated packet being exchanged during the session. It has sufficient matching capabilities to determine if a packet is valid for a session. Any packets that do not properly fit the session template are automatically rejected.

IPF stateful filtering will also allow ICMP packets related to an existing TCP or UDP session. So, if an ICMP type 3 code 4 packet is a response in a session started by a keep state rule, it will automatically be allowed. Any packet that IPF can be certain is part of an active session, even if it is a different protocol, will be allowed.

Packets destined to go out through the interface connected to the public Internet are first checked against the dynamic state table. If the packet matches the next expected packet comprising an active session conversation, it exits the firewall and the state of the session conversation flow is updated in the dynamic state table. Packets that do not belong to an already active session, are checked against the outbound ruleset.

Packets coming in from the interface connected to the public Internet are first checked against the dynamic state table. If the packet matches the next expected packet comprising an active session, it exits the firewall and the state of the session conversation flow is updated in the dynamic state table. Packets that do not belong to an already active session, are checked against the inbound ruleset.

When the session completes, it is removed from the dynamic state table.

Stateful filtering allows one to focus on blocking/passing new sessions. If the new session is passed, all its subsequent packets are allowed automatically and any impostor packets are automatically rejected. If a new session is blocked, none of its subsequent packets are allowed. Stateful filtering provides advanced matching abilities capable of defending against the flood of different attack methods employed by attackers.

31.5.13 Inclusive Ruleset Example

The following ruleset is an example of an inclusive type of firewall which only allows services matching `pass` rules and blocks all others by default. Network firewalls intended to protect other machines should have at least two interfaces, and are generally configured to trust the LAN and to not trust the public Internet. Alternatively, a host based firewall might be configured to protect only the system it is running on, and is appropriate for servers on an untrusted network or a desktop system not protected by firewall on the network.

FreeBSD uses interface `lo0` and IP address `127.0.0.1` for internal communication within the operating system. The firewall rules must contain rules to allow free movement of these internally used packets.

The interface which faces the public Internet is the one specified in the rules that authorize and control access of the outbound and inbound connections.

In cases where one or more NICs are cabled to private network segments, those interfaces may require rules to allow packets originating from those LAN interfaces transit to each other or to the Internet.

The rules should be organized into three major sections: the trusted interfaces, then the public interface outbound, and lastly, the public untrusted interface inbound.

The rules in each of the public interface sections should have the most frequently matched rules placed before less commonly matched rules, with the last rule in the section blocking and logging all packets on that interface and direction.

The outbound section in the following ruleset only contains `pass` rules which uniquely identify the services that are authorized for public Internet access. All the rules use `quick`, `on`, `proto`, `port`, and `keep state`. The `proto tcp` rules include `flag` to identify the session start request as the triggering packet to activate the stateful facility.

The inbound section blocks undesirable packets first, for two different reasons. The first is that malicious packets may be partial matches for legitimate traffic. These packets have to be discarded rather than allowed, based on their partial matches against the `allow` rules. The second reason is that known and uninteresting rejects may be blocked silently, rather than being logged by the last rule in the section.

The ruleset should ensure that there is no response returned for any undesirable traffic. Invalid packets should be silently dropped so that the attacker has no knowledge if the packets reached the system. Rules that include a `log first` option, will only log the event the first time they are triggered. This option is included in the sample `nmap OS fingerprint` rule. The `security/nmap` utility is commonly used by attackers who attempt to identify the operating system of the server.

Any time there are logged messages on a rule with the `log first` option, `ipfstat -hio` should be executed to evaluate how many times the rule has been matched. A large number of matches usually indicates that the system is being flooded or is under attack.

To lookup unknown port numbers, refer to `/etc/services`. Alternatively, visit http://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers and do a port number lookup to find the purpose of a particular port number.

Check out this link for port numbers used by Trojans http://www.sans.org/security-resources/idfaq/oddports.php.

The following ruleset creates an `inclusive` firewall ruleset which can be easily customized by commenting out `pass` rules for services that should not be authorized.

To avoid logging unwanted messages, add a `block` rule in the inbound section.

Change the `dc0` interface name in every rule to the interface name that connects the system to the public Internet.

The following statements were added to `/etc/ipf.rules`:

```
################################################################
# No restrictions on Inside LAN Interface for private network
# Not needed unless you have LAN
################################################################

#pass out quick on xl0 all
#pass in quick on xl0 all
```

```
#####################################################################
# No restrictions on Loopback Interface
#####################################################################
pass in quick on lo0 all
pass out quick on lo0 all

#####################################################################
# Interface facing Public Internet (Outbound Section)
# Match session start requests originating from behind the
# firewall on the private network
# or from this gateway server destined for the public Internet.
#####################################################################

# Allow out access to my ISP's Domain name server.
# xxx must be the IP address of your ISP's DNS.
# Dup these lines if your ISP has more than one DNS server
# Get the IP addresses from /etc/resolv.conf file
pass out quick on dc0 proto tcp from any to xxx port = 53 flags S keep state
pass out quick on dc0 proto udp from any to xxx port = 53 keep state

# Allow out access to my ISP's DHCP server for cable or DSL networks.
# This rule is not needed for 'user ppp' type connection to the
# public Internet, so you can delete this whole group.
# Use the following rule and check log for IP address.
# Then put IP address in commented out rule & delete first rule
pass out log quick on dc0 proto udp from any to any port = 67 keep state
#pass out quick on dc0 proto udp from any to z.z.z.z port = 67 keep state


# Allow out non-secure standard www function
pass out quick on dc0 proto tcp from any to any port = 80 flags S keep state

# Allow out secure www function https over TLS SSL
pass out quick on dc0 proto tcp from any to any port = 443 flags S keep state

# Allow out send & get email function
pass out quick on dc0 proto tcp from any to any port = 110 flags S keep state
pass out quick on dc0 proto tcp from any to any port = 25 flags S keep state

# Allow out Time
pass out quick on dc0 proto tcp from any to any port = 37 flags S keep state

# Allow out nntp news
pass out quick on dc0 proto tcp from any to any port = 119 flags S keep state

# Allow out gateway & LAN users' non-secure FTP ( both passive & active modes)
# This function uses the IPNAT built in FTP proxy function coded in
# the nat rules file to make this single rule function correctly.
# If you want to use the pkg_add command to install application packages
# on your gateway system you need this rule.
pass out quick on dc0 proto tcp from any to any port = 21 flags S keep state
```

```
# Allow out ssh/sftp/scp (telnet/rlogin/FTP replacements)
# This function is using SSH (secure shell)
pass out quick on dc0 proto tcp from any to any port = 22 flags S keep state

# Allow out insecure Telnet
pass out quick on dc0 proto tcp from any to any port = 23 flags S keep state

# Allow out FreeBSD CVSup
pass out quick on dc0 proto tcp from any to any port = 5999 flags S keep state

# Allow out ping to public Internet
pass out quick on dc0 proto icmp from any to any icmp-type 8 keep state

# Allow out whois from LAN to public Internet
pass out quick on dc0 proto tcp from any to any port = 43 flags S keep state

# Block and log only the first occurrence of everything
# else that's trying to get out.
# This rule implements the default block
block out log first quick on dc0 all

#################################################################
# Interface facing Public Internet (Inbound Section)
# Match packets originating from the public Internet
# destined for this gateway server or the private network.
#################################################################

# Block all inbound traffic from non-routable or reserved address spaces
block in quick on dc0 from 192.168.0.0/16 to any    #RFC 1918 private IP
block in quick on dc0 from 172.16.0.0/12 to any     #RFC 1918 private IP
block in quick on dc0 from 10.0.0.0/8 to any        #RFC 1918 private IP
block in quick on dc0 from 127.0.0.0/8 to any       #loopback
block in quick on dc0 from 0.0.0.0/8 to any         #loopback
block in quick on dc0 from 169.254.0.0/16 to any    #DHCP auto-config
block in quick on dc0 from 192.0.2.0/24 to any      #reserved for docs
block in quick on dc0 from 204.152.64.0/23 to any   #Sun cluster interconnect
block in quick on dc0 from 224.0.0.0/3 to any       #Class D & E multicast

##### Block a bunch of different nasty things. ############
# That I do not want to see in the log

# Block frags
block in quick on dc0 all with frags

# Block short tcp packets
block in quick on dc0 proto tcp all with short

# block source routed packets
block in quick on dc0 all with opt lsrr
block in quick on dc0 all with opt ssrr

# Block nmap OS fingerprint attempts
# Log first occurrence of these so I can get their IP address
```

```
block in log first quick on dc0 proto tcp from any to any flags FUP

# Block anything with special options
block in quick on dc0 all with ipopts

# Block public pings
block in quick on dc0 proto icmp all icmp-type 8

# Block ident
block in quick on dc0 proto tcp from any to any port = 113

# Block all Netbios service. 137=name, 138=datagram, 139=session
# Netbios is MS/Windows sharing services.
# Block MS/Windows hosts2 name server requests 81
block in log first quick on dc0 proto tcp/udp from any to any port = 137
block in log first quick on dc0 proto tcp/udp from any to any port = 138
block in log first quick on dc0 proto tcp/udp from any to any port = 139
block in log first quick on dc0 proto tcp/udp from any to any port = 81

# Allow traffic in from ISP's DHCP server. This rule must contain
# the IP address of your ISP's DHCP server as it is the only
# authorized source to send this packet type. Only necessary for
# cable or DSL configurations. This rule is not needed for
# 'user ppp' type connection to the public Internet.
# This is the same IP address you captured and
# used in the outbound section.
pass in quick on dc0 proto udp from z.z.z.z to any port = 68 keep state

# Allow in standard www function because I have apache server
pass in quick on dc0 proto tcp from any to any port = 80 flags S keep state

# Allow in non-secure Telnet session from public Internet
# labeled non-secure because ID/PW passed over public Internet as clear text.
# Delete this sample group if you do not have telnet server enabled.
#pass in quick on dc0 proto tcp from any to any port = 23 flags S keep state

# Allow in secure FTP, Telnet, and SCP from public Internet
# This function is using SSH (secure shell)
pass in quick on dc0 proto tcp from any to any port = 22 flags S keep state

# Block and log only first occurrence of all remaining traffic
# coming into the firewall. The logging of only the first
# occurrence avoids filling up disk with Denial of Service logs.
# This rule implements the default block.
block in log first quick on dc0 all
################### End of rules file ######################################
```

31.5.14 NAT

NAT stands for *Network Address Translation*. In Linux, NAT is called "IP Masquerading". The IPF NAT function enables the private LAN behind the firewall to share a single ISP-assigned IP address, even if that address is dynamically assigned. NAT allows each computer in the LAN to have Internet access, without having to pay the ISP for multiple Internet accounts or IP addresses.

NAT will automatically translate the private LAN IP address for each system on the LAN to the single public IP address as packets exit the firewall bound for the public Internet. It also performs the reverse translation for returning packets.

According to RFC 1918, the following IP address ranges are reserved for private networks which will never be routed directly to the public Internet, and therefore are available for use with NAT:

- `10.0.0.0/8`.
- `172.16.0.0/12`.
- `192.168.0.0/16`.

31.5.15 IPNAT

NAT rules are loaded using `ipnat`. Typically, the NAT rules are stored in `/etc/ipnat.rules`. See ipnat(8) for details.

When the file containing the NAT rules is edited after NAT has been started, run `ipnat` with `-CF` to delete the internal in use NAT rules and flush the contents of the translation table of all active entries.

To reload the NAT rules, issue a command like this:

```
# ipnat -CF -f
   /etc/ipnat.rules
```

To display some NAT statistics, use this command:

```
# ipnat -s
```

To list the NAT table's current mappings, use this command:

```
# ipnat -l
```

To turn verbose mode on and display information relating to rule processing and active rules/table entries:

```
# ipnat -v
```

31.5.16 IPNAT Rules

NAT rules are flexible and can accomplish many different things to fit the needs of commercial and home users.

The rule syntax presented here has been simplified to what is most commonly used in a non-commercial environment. For a complete rule syntax description, refer to ipnat(5).

The syntax for a NAT rule looks like this:

```
map IF LAN_IP_RANGE -> PUBLIC_ADDRESS
```

The keyword `map` starts the rule.

Replace *`IF`* with the external interface.

The *`LAN_IP_RANGE`* is used by the internal clients use for IP Addressing. Usually, this is something like `192.168.1.0/24`.

The *`PUBLIC_ADDRESS`* can either be the static external IP address or the special keyword `0/32` which uses the IP address assigned to *`IF`*.

31.5.17 How NAT Works

In IPF, when a packet arrives at the firewall from the LAN with a public destination, it passes through the outbound filter rules. NAT gets its turn at the packet and applies its rules top down, where the first matching rule wins. NAT tests each of its rules against the packet's interface name and source IP address. When a packet's interface name matches a NAT rule, the packet's source IP address in the private LAN is checked to see if it falls within the IP address range specified to the left of the arrow symbol on the NAT rule. On a match, the packet has its source IP address rewritten with the public IP address obtained by the `0/32` keyword. NAT posts an entry in its internal NAT table so when the packet returns from the public Internet it can be mapped back to its original private IP address and then passed to the filter rules for processing.

31.5.18 Enabling IPNAT

To enable IPNAT, add these statements to `/etc/rc.conf`.

To enable the machine to route traffic between interfaces:

```
gateway_enable="YES"
```

To start IPNAT automatically each time:

```
ipnat_enable="YES"
```

To specify where to load the IPNAT rules from:

```
ipnat_rules="/etc/ipnat.rules"
```

31.5.19 NAT for a Large LAN

For networks that have large numbers of systems on the LAN or networks with more than a single LAN, the process of funneling all those private IP addresses into a single public IP address becomes a resource problem that may cause problems with the same port numbers being used many times across many connections, causing collisions. There are two ways to relieve this resource problem.

31.5.19.1 Assigning Ports to Use

A normal NAT rule would look like:

```
map dc0 192.168.1.0/24 -> 0/32
```

In the above rule, the packet's source port is unchanged as the packet passes through IPNAT. By adding the `portmap` keyword, IPNAT can be directed to only use source ports in the specified range. For example, the following rule will tell IPNAT to modify the source port to be within the range shown:

```
map dc0 192.168.1.0/24 -> 0/32 portmap tcp/udp 20000:60000
```

Additionally, the `auto` keyword tells IPNAT to determine which ports are available for use:

```
map dc0 192.168.1.0/24 -> 0/32 portmap tcp/udp auto
```

31.5.19.2 Using a Pool of Public Addresses

In very large LANs there comes a point where there are just too many LAN addresses to fit into a single public address. If a block of public IP addresses is available, these addresses can be used as a "pool", and IPNAT may pick one of the public IP addresses as packet addresses are mapped on their way out.

For example, instead of mapping all packets through a single public IP address:

```
map dc0 192.168.1.0/24 -> 204.134.75.1
```

A range of public IP addresses can be specified either with a netmask:

```
map dc0 192.168.1.0/24 -> 204.134.75.0/255.255.255.0
```

or using CIDR notation:

```
map dc0 192.168.1.0/24 -> 204.134.75.0/24
```

31.5.20 Port Redirection

A common practice is to have a web server, email server, database server, and DNS server each segregated to a different system on the LAN. In this case, the traffic from these servers still has to undergo NAT, but there has to be some way to direct the inbound traffic to the correct server. For example, a web server operating on LAN address `10.0.10.25` and using a single public IP address of `20.20.20.5`, would use this rule:

```
rdr dc0 20.20.20.5/32 port 80 -> 10.0.10.25 port 80
```

or:

```
rdr dc0 0.0.0.0/0 port 80 -> 10.0.10.25 port 80
```

For a LAN DNS server on a private address of `10.0.10.33` that needs to receive public DNS requests:

```
rdr dc0 20.20.20.5/32 port 53 -> 10.0.10.33 port 53 udp
```

31.5.21 FTP and NAT

FTP has two modes: active mode and passive mode. The difference is in how the data channel is acquired. Passive mode is more secure as the data channel is acquired by the ordinal ftp session requester. For a good explanation of FTP and the different modes, see http://www.slacksite.com/other/ftp.html.

31.5.21.1 IPNAT Rules

IPNAT has a built in FTP proxy option which can be specified on the NAT map rule. It can monitor all outbound packet traffic for FTP active or passive start session requests and dynamically create temporary filter rules containing the port number being used by the data channel. This eliminates the security risk FTP normally exposes the firewall to as it no longer needs to open large ranges of high order ports for FTP connections.

This rule will handle all the traffic for the internal LAN:

```
map dc0 10.0.10.0/29 -> 0/32 proxy port 21 ftp/tcp
```

This rule handles the FTP traffic from the gateway:

```
map dc0 0.0.0.0/0 -> 0/32 proxy port 21 ftp/tcp
```

This rule handles all non-FTP traffic from the internal LAN:

```
map dc0 10.0.10.0/29 -> 0/32
```

The FTP `map` rules go before the NAT rule so that when a packet matches an FTP rule, the FTP proxy creates temporary filter rules to let the FTP session packets pass and undergo NAT. All LAN packets that are not FTP will not match the FTP rules but will undergo NAT if they match the third rule.

31.5.21.2 IPNAT FTP Filter Rules

Only one filter rule is needed for FTP if the NAT FTP proxy is used.

Without the FTP proxy, the following three rules will be needed:

```
# Allow out LAN PC client FTP to public Internet
# Active and passive modes
pass out quick on rl0 proto tcp from any to any port = 21 flags S keep state

# Allow out passive mode data channel high order port numbers
pass out quick on rl0 proto tcp from any to any port > 1024 flags S keep state

# Active mode let data channel in from FTP server
pass in quick on rl0 proto tcp from any to any port = 20 flags S keep state
```

31.6 IPFW

IPFW is a stateful firewall written for FreeBSD which also provides a traffic shaper, packet scheduler, and in-kernel NAT.

FreeBSD provides a sample ruleset in `/etc/rc.firewall`. The sample ruleset define several firewall types for common scenarios to assist novice users in generating an appropriate ruleset. ipfw(8) provides a powerful syntax which advanced users can use to craft customized rulesets that meet the security requirements of a given environment.

IPFW is composed of several components: the kernel firewall filter rule processor and its integrated packet accounting facility, the logging facility, the `divert` rule which triggers NAT, the dummynet traffic shaper facilities, the `fwd rule` forward facility, the bridge facility, and the ipstealth facility. IPFW supports both IPv4 and IPv6.

31.6.1 Enabling IPFW

IPFW is included in the basic FreeBSD install as a run time loadable module. The system will dynamically load the kernel module when `rc.conf` contains the statement `firewall_enable="YES"`. After rebooting the system, the following white highlighted message is displayed on the screen as part of the boot process:

```
ipfw2 initialized, divert disabled, rule-based forwarding disabled, default to deny, logging disa
```

The loadable module includes logging ability. To enable logging and set the verbose logging limit, add these statements to `/etc/sysctl.conf` before rebooting:

```
net.inet.ip.fw.verbose=1
net.inet.ip.fw.verbose_limit=5
```

31.6.2 Kernel Options

For those users who wish to statically compile kernel IPFW support, the following options are available for the custom kernel configuration file:

```
options     IPFIREWALL
```

This option enables IPFW as part of the kernel.

```
options     IPFIREWALL_VERBOSE
```

This option enables logging of packets that pass through IPFW and have the `log` keyword specified in the ruleset.

```
options     IPFIREWALL_VERBOSE_LIMIT=5
```

This option limits the number of packets logged through syslogd(8), on a per-entry basis. This option may be used in hostile environments, when firewall activity logging is desired. This will close a possible denial of service attack via syslog flooding.

```
options     IPFIREWALL_DEFAULT_TO_ACCEPT
```

This option allows everything to pass through the firewall by default, which is a good idea when the firewall is being set up for the first time.

```
options     IPDIVERT
```

This option enables the use of NAT functionality.

Note: The firewall will block all incoming and outgoing packets if either the `IPFIREWALL_DEFAULT_TO_ACCEPT` kernel option or a rule to explicitly allow these connections is missing.

31.6.3 `/etc/rc.conf` Options

Enables the firewall:

```
firewall_enable="YES"
```

To select one of the default firewall types provided by FreeBSD, select one by reading `/etc/rc.firewall` and specify it in the following:

```
firewall_type="open"
```

Available values for this setting are:

- `open`: passes all traffic.
- `client`: protects only this machine.
- `simple`: protects the whole network.
- `closed`: entirely disables IP traffic except for the loopback interface.
- `UNKNOWN`: disables the loading of firewall rules.
- *`filename`*: absolute path of the file containing the firewall rules.

Two methods are available for loading custom **ipfw** rules. One is to set the `firewall_type` variable to the absolute path of the file which contains the firewall rules.

The other method is to set the `firewall_script` variable to the absolute path of an executable script that includes `ipfw` commands. A ruleset script that blocks all incoming and outgoing traffic would look like this:

```
#!/bin/sh

ipfw -q flush

ipfw add deny in
ipfw add deny out
```

Note: If `firewall_type` is set to either `client` or `simple`, modify the default rules found in `/etc/rc.firewall` to fit the configuration of the system. The examples used in this section assume that the `firewall_script` is set to `/etc/ipfw.rules`.

Enable logging:

```
firewall_logging="YES"
```

Warning: `firewall_logging` sets the `net.inet.ip.fw.verbose` sysctl variable to the value of `1`. There is no `rc.conf` variable to set log limitations, but the desired value can be set using `sysctl` or by adding the following variable and desired value to `/etc/sysctl.conf`:

```
net.inet.ip.fw.verbose_limit=5
```

If the machine is acting as a gateway providing NAT using natd(8), refer to Section 32.9 for information regarding the required `/etc/rc.conf` options.

31.6.4 The IPFW Command

`ipfw` can be used to make manual, single rule additions or deletions to the active firewall while it is running. The problem with using this method is that all the changes are lost when the system reboots. It is recommended to instead write all the rules in a file and to use that file to load the rules at boot time and to replace the currently running firewall rules whenever that file changes.

`ipfw` is a useful way to display the running firewall rules to the console screen. The IPFW accounting facility dynamically creates a counter for each rule that counts each packet that matches the rule. During the process of testing a rule, listing the rule with its counter is one way to determine if the rule is functioning as expected.

To list all the running rules in sequence:

```
# ipfw list
```

To list all the running rules with a time stamp of when the last time the rule was matched:

```
# ipfw -t list
```

The next example lists accounting information and the packet count for matched rules along with the rules themselves. The first column is the rule number, followed by the number of matched packets and bytes, followed by the rule itself.

```
# ipfw -a list
```

To list dynamic rules in addition to static rules:

```
# ipfw -d list
```

To also show the expired dynamic rules:

```
# ipfw -d -e list
```

To zero the counters:

```
# ipfw zero
```

To zero the counters for just the rule with number *`NUM`*:

```
# ipfw zero NUM
```

31.6.5 IPFW Rulesets

When a packet enters the IPFW firewall, it is compared against the first rule in the ruleset and progresses one rule at a time, moving from top to bottom of the set in ascending rule number sequence order. When the packet matches the selection parameters of a rule, the rule's action field value is executed and the search of the ruleset terminates for that packet. This is referred to as "first match wins". If the packet does not match any of the rules, it gets caught by the mandatory IPFW default rule, number 65535, which denies all packets and silently discards them. However, if the packet matches a rule that contains the `count`, `skipto`, or `tee` keywords, the search continues. Refer to ipfw(8) for details on how these keywords affect rule processing.

The examples in this section create an inclusive type firewall ruleset containing the stateful `keep state`, `limit`, `in`, `out` and `via` options. For a complete rule syntax description, refer to ipfw(8).

> **Warning:** Be careful when working with firewall rules, as it is easy to lock out even the administrator.

31.6.5.1 Rule Syntax

This section describes the keywords which comprise an IPFW rule. Keywords must be written in the following order. # is used to mark the start of a comment and may appear at the end of a rule line or on its own line. Blank lines are ignored.

`CMD RULE_NUMBER ACTION LOGGING SELECTION STATEFUL`

31.6.5.1.1 CMD

Each new rule has to be prefixed with *`add`* to add the rule to the internal table.

31.6.5.1.2 RULE_NUMBER

Each rule is associated with a rule_number in the range of `1` to `65535`.

31.6.5.1.3 ACTION

A rule can be associated with one of the following actions. The specified action will be executed when the packet matches the selection criterion of the rule.

`allow | accept | pass | permit`

These keywords are equivalent as they allow packets that match the rule to exit the firewall rule processing. The search terminates at this rule.

`check-state`

Checks the packet against the dynamic rules table. If a match is found, execute the action associated with the rule which generated this dynamic rule, otherwise move to the next rule. A `check-state` rule does not have selection criterion. If no `check-state` rule is present in the ruleset, the dynamic rules table is checked at the first `keep-state` or `limit` rule.

`deny | drop`

Both words mean the same thing, which is to discard packets that match this rule. The search terminates.

31.6.5.1.4 Logging

When a packet matches a rule with the `log` keyword, a message will be logged to syslogd(8) with a facility name of `SECURITY`. Logging only occurs if the number of packets logged for that particular rule does not exceed the `logamount` parameter. If no `logamount` is specified, the limit is taken from the `sysctl` value of `net.inet.ip.fw.verbose_limit`. In both cases, a value of zero removes the logging limit. Once the limit is reached, logging can be re-enabled by clearing the logging counter or the packet counter for that rule, using `ipfw reset log`.

> **Note:** Logging is done after all other packet matching conditions have been met, and before performing the final action on the packet. The administrator decides which rules to enable logging on.

31.6.5.1.5 Selection

The keywords described in this section are used to describe attributes of the packet to be checked when determining whether rules match the packet or not. The following general-purpose attributes are provided for matching, and must be used in this order:

`udp | tcp | icmp`

Any other protocol names found in `/etc/protocols` can be used. The value specified is the protocol to be matched against. This is a mandatory keyword.

`from src to dst`

The `from` and `to` keywords are used to match against IP addresses. Rules must specify *both* source and destination parameters. `any` is a special keyword that matches any IP address. `me` is a special keyword that matches any IP address configured on an interface in the FreeBSD system to represent the PC the firewall is running on. Example usage includes `from me to any`, `from any to me`, `from 0.0.0.0/0 to any`, `from any to 0.0.0.0/0`, `from 0.0.0.0 to any`. `from any to 0.0.0.0`, and `from me to 0.0.0.0`. IP addresses are specified in dotted IP address format followed by the mask in CIDR notation, or as a single host in dotted IP address format. This keyword is a mandatory requirement. The `net-mgmt/ipcalc` port may be used to assist the mask calculation.

`port number`

For protocols which support port numbers, such as TCP and UDP, it is mandatory to include the port number of the service that will be matched. Service names from `/etc/services` may be used instead of numeric port values.

`in | out`

Matches incoming or outgoing packets. It is mandatory that one or the other is included as part of the rule matching criterion.

`via IF`

Matches packets going through the interface specified by device name. The `via` keyword causes the interface to always be checked as part of the match process.

`setup`

This mandatory keyword identifies the session start request for TCP packets.

`keep-state`

This is a mandatory keyword. Upon a match, the firewall will create a dynamic rule, whose default behavior is to match bidirectional traffic between source and destination IP/port using the same protocol.

limit {src-addr | src-port | dst-addr | dst-port}

The firewall will only allow *N* connections with the same set of parameters as specified in the rule. One or more of source and destination addresses and ports can be specified. `limit` and `keep-state` can not be used on the same rule as they provide the same stateful function.

31.6.5.2 Stateful Rule Option

The `check-state` option is used to identify where in the IPFW ruleset the packet is to be tested against the dynamic rules facility. On a match, the packet exits the firewall to continue on its way and a new rule is dynamically created for the next anticipated packet being exchanged during this session. On a no match, the packet advances to the next rule in the ruleset for testing.

The dynamic rules facility is vulnerable to resource depletion from a SYN-flood attack which would open a huge number of dynamic rules. To counter this type of attack with IPFW, use `limit`. This keyword limits the number of simultaneous sessions by checking that rule's source or destinations fields and using the packet's IP address in a search of the open dynamic rules, counting the number of times this rule and IP address combination occurred. If this count is greater than the value specified by `limit`, the packet is discarded.

31.6.5.3 Logging Firewall Messages

Even with the logging facility enabled, IPFW will not generate any rule logging on its own. The firewall administrator decides which rules in the ruleset will be logged, and adds the `log` keyword to those rules. Normally only deny rules are logged. It is customary to duplicate the "ipfw default deny everything" rule with the `log` keyword included as the last rule in the ruleset. This way, it is possible to see all the packets that did not match any of the rules in the ruleset.

Logging is a two edged sword. If one is not careful, an over abundance of log data or a DoS attack can fill the disk with log files. Log messages are not only written to **syslogd**, but also are displayed on the root console screen and soon become annoying.

The `IPFIREWALL_VERBOSE_LIMIT=5` kernel option limits the number of consecutive messages sent to syslogd(8), concerning the packet matching of a given rule. When this option is enabled in the kernel, the number of consecutive messages concerning a particular rule is capped at the number specified. There is nothing to be gained from 200 identical log messages. With this option set to five, five consecutive messages concerning a particular rule would be logged to **syslogd** and the remainder identical consecutive messages would be counted and posted to **syslogd** with a phrase like the following:

```
last message repeated 45 times
```

All logged packets messages are written by default to `/var/log/security`, which is defined in `/etc/syslog.conf`.

31.6.5.4 Building a Rule Script

Most experienced IPFW users create a file containing the rules and code them in a manner compatible with running them as a script. The major benefit of doing this is the firewall rules can be refreshed in mass without the need of

rebooting the system to activate them. This method is convenient in testing new rules as the procedure can be executed as many times as needed. Being a script, symbolic substitution can be used for frequently used values to be substituted into multiple rules.

This example script is compatible with the syntax used by the sh(1), csh(1), and tcsh(1) shells. Symbolic substitution fields are prefixed with a dollar sign ($). Symbolic fields do not have the $ prefix. The value to populate the symbolic field must be enclosed in double quotes ("").

Start the rules file like this:

```
############### start of example ipfw rules script #############
#
ipfw -q -f flush       # Delete all rules
# Set defaults
oif="tun0"             # out interface
odns="192.0.2.11"      # ISP's DNS server IP address
cmd="ipfw -q add "     # build rule prefix
ks="keep-state"        # just too lazy to key this each time
$cmd 00500 check-state
$cmd 00502 deny all from any to any frag
$cmd 00501 deny tcp from any to any established
$cmd 00600 allow tcp from any to any 80 out via $oif setup $ks
$cmd 00610 allow tcp from any to $odns 53 out via $oif setup $ks
$cmd 00611 allow udp from any to $odns 53 out via $oif $ks
################### End of example ipfw rules script ############
```

The rules are not important as the focus of this example is how the symbolic substitution fields are populated.

If the above example was in `/etc/ipfw.rules`, the rules could be reloaded by the following command:

```
# sh /etc/ipfw.rules
```

`/etc/ipfw.rules` can be located anywhere and the file can have any name.

The same thing could be accomplished by running these commands by hand:

```
# ipfw -q -f flush
# ipfw -q add check-state
# ipfw -q add deny all from any to any frag
# ipfw -q add deny tcp from any to any established
# ipfw -q add allow tcp from any to any 80 out via tun0 setup keep-state
# ipfw -q add allow tcp from any to 192.0.2.11 53 out via tun0 setup keep-state
# ipfw -q add 00611 allow udp from any to 192.0.2.11 53 out via tun0 keep-state
```

31.6.5.5 An Example Stateful Ruleset

The following sample ruleset is a complete inclusive type ruleset. Comment out any `pass` rules for services that are not required. To avoid logging undesired messages, add a `deny` rule in the inbound section. Change the `dc0` in every rule to the device name of the interface that connects the system to the Internet.

There is a noticeable pattern in the usage of these rules.

- All statements that are a request to start a session to the Internet use `keep-state`.

- All the authorized services that originate from the Internet use `limit` to prevent flooding.
- All rules use `in` or `out` to clarify direction.
- All rules use `via` *`interface-name`* to specify the interface the packet is traveling over.

The following rules go into `/etc/ipfw.rules`:

```
################ Start of IPFW rules file ###############################
# Flush out the list before we begin.
ipfw -q -f flush

# Set rules command prefix
cmd="ipfw -q add"
pif="dc0"     # public interface name of NIC
              # facing the public Internet

######################################################################
# No restrictions on Inside LAN Interface for private network
# Not needed unless you have LAN.
# Change xl0 to your LAN NIC interface name
######################################################################
#$cmd 00005 allow all from any to any via xl0

######################################################################
# No restrictions on Loopback Interface
######################################################################
$cmd 00010 allow all from any to any via lo0

######################################################################
# Allow the packet through if it has previous been added to the
# the "dynamic" rules table by a allow keep-state statement.
######################################################################
$cmd 00015 check-state

######################################################################
# Interface facing Public Internet (Outbound Section)
# Interrogate session start requests originating from behind the
# firewall on the private network or from this gateway server
# destined for the public Internet.
######################################################################

# Allow out access to my ISP's Domain name server.
# x.x.x.x must be the IP address of your ISP.s DNS
# Dup these lines if your ISP has more than one DNS server
# Get the IP addresses from /etc/resolv.conf file
$cmd 00110 allow tcp from any to x.x.x.x 53 out via $pif setup keep-state
$cmd 00111 allow udp from any to x.x.x.x 53 out via $pif keep-state

# Allow out access to my ISP's DHCP server for cable/DSL configurations.
# This rule is not needed for .user ppp. connection to the public Internet.
# so you can delete this whole group.
# Use the following rule and check log for IP address.
# Then put IP address in commented out rule & delete first rule
$cmd 00120 allow log udp from any to any 67 out via $pif keep-state
```

```
#$cmd 00120 allow udp from any to x.x.x.x 67 out via $pif keep-state

# Allow out non-secure standard www function
$cmd 00200 allow tcp from any to any 80 out via $pif setup keep-state

# Allow out secure www function https over TLS SSL
$cmd 00220 allow tcp from any to any 443 out via $pif setup keep-state

# Allow out send & get email function
$cmd 00230 allow tcp from any to any 25 out via $pif setup keep-state
$cmd 00231 allow tcp from any to any 110 out via $pif setup keep-state

# Allow out FBSD (make install & CVSUP) functions
# Basically give user root "GOD" privileges.
$cmd 00240 allow tcp from me to any out via $pif setup keep-state uid root

# Allow out ping
$cmd 00250 allow icmp from any to any out via $pif keep-state

# Allow out Time
$cmd 00260 allow tcp from any to any 37 out via $pif setup keep-state

# Allow out nntp news (i.e., news groups)
$cmd 00270 allow tcp from any to any 119 out via $pif setup keep-state

# Allow out secure FTP, Telnet, and SCP
# This function is using SSH (secure shell)
$cmd 00280 allow tcp from any to any 22 out via $pif setup keep-state

# Allow out whois
$cmd 00290 allow tcp from any to any 43 out via $pif setup keep-state

# deny and log everything else that.s trying to get out.
# This rule enforces the block all by default logic.
$cmd 00299 deny log all from any to any out via $pif

######################################################################
# Interface facing Public Internet (Inbound Section)
# Check packets originating from the public Internet
# destined for this gateway server or the private network.
######################################################################

# Deny all inbound traffic from non-routable reserved address spaces
$cmd 00300 deny all from 192.168.0.0/16 to any in via $pif  #RFC 1918 private IP
$cmd 00301 deny all from 172.16.0.0/12 to any in via $pif     #RFC 1918 private IP
$cmd 00302 deny all from 10.0.0.0/8 to any in via $pif          #RFC 1918 private IP
$cmd 00303 deny all from 127.0.0.0/8 to any in via $pif        #loopback
$cmd 00304 deny all from 0.0.0.0/8 to any in via $pif            #loopback
$cmd 00305 deny all from 169.254.0.0/16 to any in via $pif   #DHCP auto-config
$cmd 00306 deny all from 192.0.2.0/24 to any in via $pif       #reserved for docs
$cmd 00307 deny all from 204.152.64.0/23 to any in via $pif  #Sun cluster interconnect
$cmd 00308 deny all from 224.0.0.0/3 to any in via $pif         #Class D & E multicast
```

```
# Deny public pings
$cmd 00310 deny icmp from any to any in via $pif

# Deny ident
$cmd 00315 deny tcp from any to any 113 in via $pif

# Deny all Netbios service. 137=name, 138=datagram, 139=session
# Netbios is MS/Windows sharing services.
# Block MS/Windows hosts2 name server requests 81
$cmd 00320 deny tcp from any to any 137 in via $pif
$cmd 00321 deny tcp from any to any 138 in via $pif
$cmd 00322 deny tcp from any to any 139 in via $pif
$cmd 00323 deny tcp from any to any 81 in via $pif

# Deny any late arriving packets
$cmd 00330 deny all from any to any frag in via $pif

# Deny ACK packets that did not match the dynamic rule table
$cmd 00332 deny tcp from any to any established in via $pif

# Allow traffic in from ISP's DHCP server. This rule must contain
# the IP address of your ISP.s DHCP server as it.s the only
# authorized source to send this packet type.
# Only necessary for cable or DSL configurations.
# This rule is not needed for .user ppp. type connection to
# the public Internet. This is the same IP address you captured
# and used in the outbound section.
#$cmd 00360 allow udp from any to x.x.x.x 67 in via $pif keep-state

# Allow in standard www function because I have apache server
$cmd 00400 allow tcp from any to me 80 in via $pif setup limit src-addr 2

# Allow in secure FTP, Telnet, and SCP from public Internet
$cmd 00410 allow tcp from any to me 22 in via $pif setup limit src-addr 2

# Allow in non-secure Telnet session from public Internet
# labeled non-secure because ID & PW are passed over public
# Internet as clear text.
# Delete this sample group if you do not have telnet server enabled.
$cmd 00420 allow tcp from any to me 23 in via $pif setup limit src-addr 2

# Reject & Log all incoming connections from the outside
$cmd 00499 deny log all from any to any in via $pif

# Everything else is denied by default
# deny and log all packets that fell through to see what they are
$cmd 00999 deny log all from any to any
################ End of IPFW rules file ################################
```

31.6.5.6 An Example NAT and Stateful Ruleset

There are some additional configuration statements that need to be enabled to activate the NAT function of IPFW. For a customized kernel, the kernel configuration file needs `option IPDIVERT` added to the other `IPFIREWALL` options.

In addition to the normal IPFW options in `/etc/rc.conf`, the following are needed:

```
natd_enable="YES"                   # Enable NATD function
natd_interface="rl0"                # interface name of public Internet NIC
natd_flags="-dynamic -m"            # -m = preserve port numbers if possible
```

Utilizing stateful rules with a `divert natd` rule complicates the ruleset logic. The positioning of the `check-state`, and `divert natd` rules in the ruleset is critical and a new action type is used, called `skipto`. When using `skipto`, it is mandatory that each rule is numbered, so that the `skipto` rule knows which rule to jump to.

The following is an uncommented example of a ruleset which explains the sequence of the packet flow.

The processing flow starts with the first rule from the top of the ruleset and progresses one rule at a time until the end is reached or the packet matches and the packet is released out of the firewall. Take note of the location of rule numbers 100 101, 450, 500, and 510. These rules control the translation of the outbound and inbound packets so that their entries in the dynamic keep-state table always register the private LAN IP address. All the allow and deny rules specify the direction of the packet and the interface. All start outbound session requests will `skipto rule 500` to undergo NAT.

Consider a web browser which initializes a new HTTP session over port 80. When the first outbound packet enters the firewall, it does not match rule 100 because it is headed out rather than in. It passes rule 101 because this is the first packet, and it has not been posted to the dynamic keep-state table yet. The packet finally matches rule 125 as it is outbound through the NIC facing the Internet and has a source IP address as a private LAN IP address. On matching this rule, two actions take place. `keep-state` adds this rule to the dynamic keep-state rules table and the specified action is executed and posted as part of the info in the dynamic table. In this case, the action is `skipto rule 500`. Rule 500 NATs the packet IP address and sends it out to the Internet. This packet makes its way to the destination web server, where a response packet is generated and sent back. This new packet enters the top of the ruleset. It matches rule 100 and has it destination IP address mapped back to the corresponding LAN IP address. It then is processed by the `check-state` rule, is found in the table as an existing session, and is released to the LAN. It goes to the LAN system that sent it and a new packet is sent requesting another segment of the data from the remote server. This time it matches the `check-state` rule, its outbound entry is found, and the associated action, `skipto 500`, is executed. The packet jumps to rule 500, gets NATed, and is released to the Internet.

On the inbound side, everything coming in that is part of an existing session is automatically handled by the `check-state` rule and the properly placed `divert natd` rules. The ruleset only has to deny bad packets and allow only authorized services. Consider a web server running on the firewall where web requests from the Internet should have access to the local web site. An inbound start request packet will match rule 100 and its IP address will be mapped to the LAN IP address of the firewall. The packet is then matched against all the nasty things that need to be checked and finally matches rule 425 where two actions occur. The packet rule is posted to the dynamic keep-state table but this time, any new session requests originating from that source IP address are limited to 2. This defends against DoS attacks against the service running on the specified port number. The action is `allow`, so the packet is released to the LAN. The packet generated as a response is recognized by the `check-state` as belonging to an existing session. It is then sent to rule 500 for NATing and released to the outbound interface.

Example Ruleset #1:

```
#!/bin/sh
cmd="ipfw -q add"
```

```
skip="skipto 500"
pif=rl0
ks="keep-state"
good_tcpo="22,25,37,43,53,80,443,110,119"

ipfw -q -f flush

$cmd 002 allow all from any to any via xl0  # exclude LAN traffic
$cmd 003 allow all from any to any via lo0  # exclude loopback traffic

$cmd 100 divert natd ip from any to any in via $pif
$cmd 101 check-state

# Authorized outbound packets
$cmd 120 $skip udp from any to xx.168.240.2 53 out via $pif $ks
$cmd 121 $skip udp from any to xx.168.240.5 53 out via $pif $ks
$cmd 125 $skip tcp from any to any $good_tcpo out via $pif setup $ks
$cmd 130 $skip icmp from any to any out via $pif $ks
$cmd 135 $skip udp from any to any 123 out via $pif $ks


# Deny all inbound traffic from non-routable reserved address spaces
$cmd 300 deny all from 192.168.0.0/16  to any in via $pif  #RFC 1918 private IP
$cmd 301 deny all from 172.16.0.0/12   to any in via $pif  #RFC 1918 private IP
$cmd 302 deny all from 10.0.0.0/8      to any in via $pif  #RFC 1918 private IP
$cmd 303 deny all from 127.0.0.0/8     to any in via $pif  #loopback
$cmd 304 deny all from 0.0.0.0/8       to any in via $pif  #loopback
$cmd 305 deny all from 169.254.0.0/16  to any in via $pif  #DHCP auto-config
$cmd 306 deny all from 192.0.2.0/24    to any in via $pif  #reserved for docs
$cmd 307 deny all from 204.152.64.0/23 to any in via $pif  #Sun cluster
$cmd 308 deny all from 224.0.0.0/3     to any in via $pif  #Class D & E multicast

# Authorized inbound packets
$cmd 400 allow udp from xx.70.207.54 to any 68 in $ks
$cmd 420 allow tcp from any to me 80 in via $pif setup limit src-addr 1


$cmd 450 deny log ip from any to any

# This is skipto location for outbound stateful rules
$cmd 500 divert natd ip from any to any out via $pif
$cmd 510 allow ip from any to any

######################### end of rules  ##################
```

The next example is functionally equivalent, but uses descriptive comments to help the inexperienced IPFW rule writer to better understand what the rules are doing.

Example Ruleset #2:

```
#!/bin/sh
################ Start of IPFW rules file ###############################
# Flush out the list before we begin.
ipfw -q -f flush
```

```
# Set rules command prefix
cmd="ipfw -q add"
skip="skipto 800"
pif="rl0"     # public interface name of NIC
              # facing the public Internet

########################################################################
# No restrictions on Inside LAN Interface for private network
# Change xl0 to your LAN NIC interface name
########################################################################
$cmd 005 allow all from any to any via xl0

########################################################################
# No restrictions on Loopback Interface
########################################################################
$cmd 010 allow all from any to any via lo0

########################################################################
# check if packet is inbound and nat address if it is
########################################################################
$cmd 014 divert natd ip from any to any in via $pif

########################################################################
# Allow the packet through if it has previous been added to the
# the "dynamic" rules table by a allow keep-state statement.
########################################################################
$cmd 015 check-state

########################################################################
# Interface facing Public Internet (Outbound Section)
# Check session start requests originating from behind the
# firewall on the private network or from this gateway server
# destined for the public Internet.
########################################################################

# Allow out access to my ISP's Domain name server.
# x.x.x.x must be the IP address of your ISP's DNS
# Dup these lines if your ISP has more than one DNS server
# Get the IP addresses from /etc/resolv.conf file
$cmd 020 $skip tcp from any to x.x.x.x 53 out via $pif setup keep-state


# Allow out access to my ISP's DHCP server for cable/DSL configurations.
$cmd 030 $skip udp from any to x.x.x.x 67 out via $pif keep-state

# Allow out non-secure standard www function
$cmd 040 $skip tcp from any to any 80 out via $pif setup keep-state

# Allow out secure www function https over TLS SSL
$cmd 050 $skip tcp from any to any 443 out via $pif setup keep-state

# Allow out send & get email function
```

```
$cmd 060 $skip tcp from any to any 25 out via $pif setup keep-state
$cmd 061 $skip tcp from any to any 110 out via $pif setup keep-state

# Allow out FreeBSD (make install & CVSUP) functions
# Basically give user root "GOD" privileges.
$cmd 070 $skip tcp from me to any out via $pif setup keep-state uid root

# Allow out ping
$cmd 080 $skip icmp from any to any out via $pif keep-state

# Allow out Time
$cmd 090 $skip tcp from any to any 37 out via $pif setup keep-state

# Allow out nntp news (i.e., news groups)
$cmd 100 $skip tcp from any to any 119 out via $pif setup keep-state

# Allow out secure FTP, Telnet, and SCP
# This function is using SSH (secure shell)
$cmd 110 $skip tcp from any to any 22 out via $pif setup keep-state

# Allow out whois
$cmd 120 $skip tcp from any to any 43 out via $pif setup keep-state

# Allow ntp time server
$cmd 130 $skip udp from any to any 123 out via $pif keep-state

################################################################
# Interface facing Public Internet (Inbound Section)
# Check packets originating from the public Internet
# destined for this gateway server or the private network.
################################################################

# Deny all inbound traffic from non-routable reserved address spaces
$cmd 300 deny all from 192.168.0.0/16  to any in via $pif  #RFC 1918 private IP
$cmd 301 deny all from 172.16.0.0/12   to any in via $pif  #RFC 1918 private IP
$cmd 302 deny all from 10.0.0.0/8      to any in via $pif  #RFC 1918 private IP
$cmd 303 deny all from 127.0.0.0/8     to any in via $pif  #loopback
$cmd 304 deny all from 0.0.0.0/8       to any in via $pif  #loopback
$cmd 305 deny all from 169.254.0.0/16  to any in via $pif  #DHCP auto-config
$cmd 306 deny all from 192.0.2.0/24    to any in via $pif  #reserved for docs
$cmd 307 deny all from 204.152.64.0/23 to any in via $pif  #Sun cluster
$cmd 308 deny all from 224.0.0.0/3     to any in via $pif  #Class D & E multicast

# Deny ident
$cmd 315 deny tcp from any to any 113 in via $pif

# Deny all Netbios service. 137=name, 138=datagram, 139=session
# Netbios is MS/Windows sharing services.
# Block MS/Windows hosts2 name server requests 81
$cmd 320 deny tcp from any to any 137 in via $pif
$cmd 321 deny tcp from any to any 138 in via $pif
$cmd 322 deny tcp from any to any 139 in via $pif
$cmd 323 deny tcp from any to any 81  in via $pif
```

```
# Deny any late arriving packets
$cmd 330 deny all from any to any frag in via $pif

# Deny ACK packets that did not match the dynamic rule table
$cmd 332 deny tcp from any to any established in via $pif

# Allow traffic in from ISP's DHCP server. This rule must contain
# the IP address of your ISP's DHCP server as it is the only
# authorized source to send this packet type.
# Only necessary for cable or DSL configurations.
# This rule is not needed for 'user ppp' type connection to
# the public Internet. This is the same IP address you captured
# and used in the outbound section.
$cmd 360 allow udp from x.x.x.x to any 68 in via $pif keep-state

# Allow in standard www function because I have Apache server
$cmd 370 allow tcp from any to me 80 in via $pif setup limit src-addr 2

# Allow in secure FTP, Telnet, and SCP from public Internet
$cmd 380 allow tcp from any to me 22 in via $pif setup limit src-addr 2

# Allow in non-secure Telnet session from public Internet
# labeled non-secure because ID & PW are passed over public
# Internet as clear text.
# Delete this sample group if you do not have telnet server enabled.
$cmd 390 allow tcp from any to me 23 in via $pif setup limit src-addr 2

# Reject & Log all unauthorized incoming connections from the public Internet
$cmd 400 deny log all from any to any in via $pif

# Reject & Log all unauthorized out going connections to the public Internet
$cmd 450 deny log all from any to any out via $pif

# This is skipto location for outbound stateful rules
$cmd 800 divert natd ip from any to any out via $pif
$cmd 801 allow ip from any to any

# Everything else is denied by default
# deny and log all packets that fell through to see what they are
$cmd 999 deny log all from any to any
################ End of IPFW rules file ################################
```

Notes

1. Why write the rule set to default deny? The short answer is, it gives better control at the expense of some thinking. The point of packet filtering is to take control, not to run catch-up with what the bad guys do. Marcus Ranum has written a very entertaining and informative article about this, The Six Dumbest Ideas in Computer

Security (http://www.ranum.com/security/computer_security/editorials/dumb/index.html), and it is well written too.

2. For dialup users, the external interface is the `tun0` pseudo-device. Broadband users such as ADSL subscribers tend to have an Ethernet interface to play with, however for a significant subset of ADSL users, specifically those using PPP over Ethernet (PPPoE), the correct external interface will be the `tun0` pseudo-device, not the physical Ethernet interface.
3. The main internet RFCs describing ICMP and some related techhiques are RFC792, RFC950, RFC1191, RFC1256, RFC2521, rfc2765, while necessary updates for ICMP for IPv6 are found in RFC1885, RFC2463, RFC2466. These documents are available in a number of places on the net, such as the ietf.org (http://www.ietf.org) and faqs.org (http://www.faqs.org) web sites.
4. The relevant RFCs are mainly RFC1123 and RFC2821.
5. The original Harris paper and a number of other useful articles and resources can be found at the greylisting.org (http://www.greylisting.org/) web site.

Chapter 32 Advanced Networking

32.1 Synopsis

This chapter covers a number of advanced networking topics.

After reading this chapter, you will know:

- The basics of gateways and routes.
- How to set up IEEE® 802.11 and Bluetooth devices.
- How to make FreeBSD act as a bridge.
- How to set up network booting on a diskless machine.
- How to set up network PXE booting with an NFS root file system.
- How to set up network address translation.
- How to set up IPv6 on a FreeBSD machine.
- How to configure ATM.
- How to enable and utilize the features of the Common Address Redundancy Protocol (CARP) in FreeBSD.

Before reading this chapter, you should:

- Understand the basics of the `/etc/rc` scripts.
- Be familiar with basic network terminology.
- Know how to configure and install a new FreeBSD kernel (Chapter 9).
- Know how to install additional third-party software (Chapter 5).

32.2 Gateways and Routes

Contributed by Coranth Gryphon.

For one machine to be able to find another over a network, there must be a mechanism in place to describe how to get from one to the other. This is called *routing*. A “route” is a defined pair of addresses: a “destination” and a “gateway”. The pair indicates that when trying to get to this *destination*, communicate through this *gateway*. There are three types of destinations: individual hosts, subnets, and “default”. The “default route” is used if none of the other routes apply. There are also three types of gateways: individual hosts, interfaces (also called “links”), and Ethernet hardware (MAC) addresses.

32.2.1 An Example

This example netstat(1) output illustrates several aspects of routing:

```
% netstat -r
Routing tables
```

```
Destination      Gateway            Flags     Refs     Use     Netif Expire

default          outside-gw         UGSc       37      418      ppp0
localhost        localhost          UH          0      181       lo0
test0            0:e0:b5:36:cf:4f   UHLW        5    63288       ed0     77
10.20.30.255     link#1             UHLW        1     2421
example.com      link#1             UC          0        0
host1            0:e0:a8:37:8:1e    UHLW        3     4601       lo0
host2            0:e0:a8:37:8:1e    UHLW        0        5       lo0 =>
host2.example.com link#1             UC          0        0
224              link#1             UC          0        0
```

The first two lines specify the default route, described in more detail in Section 32.2.2, and the `localhost` route.

The interface (`Netif` column) that this routing table specifies to use for `localhost` is `lo0`, also known as the loopback device. This says to keep all traffic for this destination internal, rather than sending it out over the network.

The addresses beginning with `0:e0:` are Ethernet hardware addresses, also known as MAC addresses. FreeBSD will automatically identify any hosts, `test0` in the example, on the local Ethernet and add a route for that host over the Ethernet interface, `ed0`. This type of route has a timeout, seen in the `Expire` column, which is used if the host does not respond in a specific amount of time. When this happens, the route to this host will be automatically deleted. These hosts are identified using the Routing Information Protocol (RIP), which calculates routes to local hosts based upon a shortest path determination.

FreeBSD will add subnet routes for the local subnet. `10.20.30.255` is the broadcast address for the subnet `10.20.30` and `example.com` is the domain name associated with that subnet. The designation `link#1` refers to the first Ethernet card in the machine.

Local network hosts and local subnets have their routes automatically configured by a daemon called routed(8). If it is not running, only routes which are statically defined by the administrator will exist.

The `host1` line refers to the host by its Ethernet address. Since it is the sending host, FreeBSD knows to use the loopback interface (`lo0`) rather than the Ethernet interface.

The two `host2` lines represent aliases which were created using ifconfig(8). The `=>` symbol after the `lo0` interface says that an alias has been set in addition to the loopback address. Such routes only show up on the host that supports the alias; all other hosts on the local network will have a `link#1` line for such routes.

The final line (destination subnet `224`) deals with multicasting.

Finally, various attributes of each route can be seen in the `Flags` column. Below is a short table of some of these flags and their meanings:

U	Up: The route is active.
H	Host: The route destination is a single host.
G	Gateway: Send anything for this destination on to this remote system, which will figure out from there where to send it.
S	Static: This route was configured manually, not automatically generated by the system.
C	Clone: Generates a new route based upon this route for machines to connect to. This type of route is normally used for local networks.
W	WasCloned: Indicated a route that was auto-configured based upon a local area network (Clone) route.
L	Link: Route involves references to Ethernet hardware.

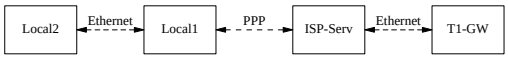
32.2.2 Default Routes

When the local system needs to make a connection to a remote host, it checks the routing table to determine if a known path exists. If the remote host falls into a subnet that it knows how to reach, the system checks to see if it can connect using that interface.

If all known paths fail, the system has one last option: the "default" route. This route is a special type of gateway route (usually the only one present in the system), and is always marked with a `C` in the flags field. For hosts on a local area network, this gateway is set to the system which has a direct connection to the Internet.

The default route for a machine which itself is functioning as the gateway to the outside world, will be the gateway machine at the Internet Service Provider (ISP).

This example is a common configuration for a default route:



The hosts `Local1` and `Local2` are on the local network. `Local1` is connected to an ISP using a PPP connection. This PPP server is connected through a local area network to another gateway computer through an external interface to the ISP.

The default routes for each machine will be:

Host	Default Gateway	Interface
Local2	Local1	Ethernet
Local1	T1-GW	PPP

A common question is "Why is `T1-GW` configured as the default gateway for `Local1`, rather than the ISP server it is connected to?".

Since the PPP interface is using an address on the ISP's local network for the local side of the connection, routes for any other machines on the ISP's local network will be automatically generated. The system already knows how to reach the `T1-GW` machine, so there is no need for the intermediate step of sending traffic to the ISP's server.

It is common to use the address `X.X.X.1` as the gateway address for the local network. So, if the local class C address space is `10.20.30` and the ISP is using `10.9.9`, the default routes would be:

Host	Default Route
Local2 (10.20.30.2)	Local1 (10.20.30.1)
Local1 (10.20.30.1, 10.9.9.30)	T1-GW (10.9.9.1)

The default route can be easily defined in `/etc/rc.conf`. In this example, on `Local2`, add the following line to `/etc/rc.conf`:

```
defaultrouter="10.20.30.1"
```

It is also possible to add the route directly using route(8):

```
# route add default 10.20.30.1
```

For more information on manual manipulation of network routing tables, refer to route(8).

32.2.3 Dual Homed Hosts

A a dual-homed system is a host which resides on two different networks.

The dual-homed machine might have two Ethernet cards, each having an address on a separate subnet. Alternately, the machine can have one Ethernet card and uses ifconfig(8) aliasing. The former is used if two physically separate Ethernet networks are in use and the latter if there is one physical network segment, but two logically separate subnets.

Either way, routing tables are set up so that each subnet knows that this machine is the defined gateway (inbound route) to the other subnet. This configuration, with the machine acting as a router between the two subnets, is often used to implement packet filtering or firewall security in either or both directions.

For this machine to forward packets between the two interfaces, FreeBSD must be configured as a router, as demonstrated in the next section.

32.2.4 Building a Router

A network router is a system that forwards packets from one interface to another. Internet standards and good engineering practice prevent the FreeBSD Project from enabling this by default in FreeBSD. This feature can be enabled by changing the following variable to `YES` in rc.conf(5):

```
gateway_enable="YES"          # Set to YES if this host will be a gateway
```

This option will set the sysctl(8) variable `net.inet.ip.forwarding` to `1`. To stop routing, reset this to `0`.

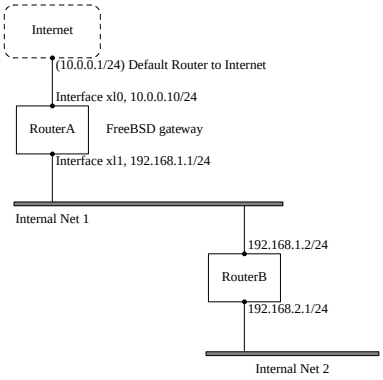
The new router will need routes to know where to send the traffic. If the network is simple enough, static routes can be used. FreeBSD comes with the standard BSD routing daemon routed(8), which speaks RIP versions 1 and 2, and IRDP. Support for BGPv4, OSPFv2, and other sophisticated routing protocols is available with the `net/zebra` package or port.

32.2.5 Setting Up Static Routes

Contributed by Al Hoang.

32.2.5.1 Manual Configuration

Consider the following network:



In this scenario, RouterA is a FreeBSD machine that is acting as a router to the rest of the Internet. It has a default route set to 10.0.0.1 which allows it to connect with the outside world. RouterB is already configured properly as it uses 192.168.1.1 as the gateway.

The routing table on RouterA looks something like this:

```
% netstat -nr
Routing tables

Internet:
Destination        Gateway            Flags    Refs      Use  Netif  Expire
default            10.0.0.1           UGS         0    49378    xl0
127.0.0.1          127.0.0.1          UH          0        6    lo0
10.0.0.0/24        link#1             UC          0        0    xl0
192.168.1.0/24     link#2             UC          0        0    xl1
```

With the current routing table, RouterA cannot reach Internal Net 2 as it does not have a route for 192.168.2.0/24. The following command adds the Internal Net 2 network to RouterA's routing table using 192.168.1.2 as the next hop:

```
# route add -net 192.168.2.0/24 192.168.1.2
```

Now RouterA can reach any hosts on the 192.168.2.0/24 network.

32.2.5.2 Persistent Configuration

The above example configures a static route on a running system. However, the routing information will not persist if the FreeBSD system reboots. Persistent static routes can be entered in /etc/rc.conf:

```
# Add Internal Net 2 as a static route
static_routes="internalnet2"
route_internalnet2="-net 192.168.2.0/24 192.168.1.2"
```

The static_routes configuration variable is a list of strings separated by a space, where each string references a route name. This example only has one string in static_routes, *internalnet2*. The variable route_*internalnet2* contains all of the configuration parameters to route(8). This example is equivalent to the command:

```
# route add -net 192.168.2.0/24 192.168.1.2
```

Using more than one string in static_routes creates multiple static routes. The following shows an example of adding static routes for the 192.168.0.0/24 and 192.168.1.0/24 networks:

```
static_routes="net1 net2"
route_net1="-net 192.168.0.0/24 192.168.0.1"
route_net2="-net 192.168.1.0/24 192.168.1.1"
```

32.2.6 Routing Propagation

When an address space is assigned to a network, the service provider configures their routing tables so that all traffic for the network will be sent to the link for the site. But how do external sites know to send their packets to the network's ISP?

There is a system that keeps track of all assigned address spaces and defines their point of connection to the Internet backbone, or the main trunk lines that carry Internet traffic across the country and around the world. Each backbone machine has a copy of a master set of tables, which direct traffic for a particular network to a specific backbone carrier, and from there down the chain of service providers until it reaches your network.

It is the task of the service provider to advertise to the backbone sites that they are the point of connection, and thus the path inward, for a site. This is known as route propagation.

32.2.7 Troubleshooting

Sometimes, there is a problem with routing propagation and some sites are unable to connect. Perhaps the most useful command for trying to figure out where routing is breaking down is traceroute(8). It is useful when ping(8) fails.

When using traceroute(8), include the name of the remote host to connect to. The output will show the gateway hosts along the path of the attempt, eventually either reaching the target host, or terminating because of a lack of connection.

For more information, refer to traceroute(8).

32.2.8 Multicast Routing

FreeBSD natively supports both multicast applications and multicast routing. Multicast applications do not require any special configuration of FreeBSD; as applications will generally run out of the box. Multicast routing requires that support be compiled into a custom kernel:

```
options MROUTING
```

The multicast routing daemon, mrouted(8), must be configured to set up tunnels and DVMRP via `/etc/mrouted.conf`. More details on multicast configuration may be found in mrouted(8).

> **Note:** The mrouted(8) multicast routing daemon implements the DVMRP multicast routing protocol, which has largely been replaced by pim(4) in many multicast installations. mrouted(8) and the related map-mbone(8) and mrinfo(8) utilities are available in the FreeBSD Ports Collection as `net/mrouted`.

32.3 Wireless Networking

Loader, Marc Fonvieille, and Murray Stokely.

32.3.1 Wireless Networking Basics

Most wireless networks are based on the IEEE 802.11 standards. A basic wireless network consists of multiple stations communicating with radios that broadcast in either the 2.4GHz or 5GHz band, though this varies according to the locale and is also changing to enable communication in the 2.3GHz and 4.9GHz ranges.

802.11 networks are organized in two ways. In *infrastructure mode*, one station acts as a master with all the other stations associating to it, the network is known as a BSS, and the master station is termed an access point (AP). In a BSS, all communication passes through the AP; even when one station wants to communicate with another wireless station, messages must go through the AP. In the second form of network, there is no master and stations communicate directly. This form of network is termed an IBSS and is commonly known as an *ad-hoc network*.

802.11 networks were first deployed in the 2.4GHz band using protocols defined by the IEEE 802.11 and 802.11b standard. These specifications include the operating frequencies and the MAC layer characteristics, including framing and transmission rates, as communication can occur at various rates. Later, the 802.11a standard defined operation in the 5GHz band, including different signaling mechanisms and higher transmission rates. Still later, the 802.11g standard defined the use of 802.11a signaling and transmission mechanisms in the 2.4GHz band in such a way as to be backwards compatible with 802.11b networks.

Separate from the underlying transmission techniques, 802.11 networks have a variety of security mechanisms. The original 802.11 specifications defined a simple security protocol called WEP. This protocol uses a fixed pre-shared key and the RC4 cryptographic cipher to encode data transmitted on a network. Stations must all agree on the fixed key in order to communicate. This scheme was shown to be easily broken and is now rarely used except to discourage transient users from joining networks. Current security practice is given by the IEEE 802.11i specification that defines new cryptographic ciphers and an additional protocol to authenticate stations to an access point and exchange keys for data communication. Cryptographic keys are periodically refreshed and there are mechanisms for detecting and countering intrusion attempts. Another security protocol specification commonly used in wireless networks is termed WPA, which was a precursor to 802.11i. WPA specifies a subset of the requirements found in 802.11i and is designed for implementation on legacy hardware. Specifically, WPA requires only the TKIP cipher that is derived from the original WEP cipher. 802.11i permits use of TKIP but also requires support for a stronger cipher, AES-CCM, for encrypting data. The AES cipher was not required in WPA because it was deemed too computationally costly to be implemented on legacy hardware.

The other standard to be aware of is 802.11e. It defines protocols for deploying multimedia applications, such as streaming video and voice over IP (VoIP), in an 802.11 network. Like 802.11i, 802.11e also has a precursor specification termed WME (later renamed WMM) that has been defined by an industry group as a subset of 802.11e that can be deployed now to enable multimedia applications while waiting for the final ratification of 802.11e. The most important thing to know about 802.11e and WME/WMM is that it enables prioritized traffic over a wireless network through Quality of Service (QoS) protocols and enhanced media access protocols. Proper implementation of these protocols enables high speed bursting of data and prioritized traffic flow.

FreeBSD supports networks that operate using 802.11a, 802.11b, and 802.11g. The WPA and 802.11i security protocols are likewise supported (in conjunction with any of 11a, 11b, and 11g) and QoS and traffic prioritization required by the WME/WMM protocols are supported for a limited set of wireless devices.

32.3.2 Basic Setup

32.3.2.1 Kernel Configuration

To use wireless networking, a wireless networking card is needed and the kernel needs to be configured with the appropriate wireless networking support. The kernel is separated into multiple modules so that only the required support needs to be configured.

The most commonly used wireless devices are those that use parts made by Atheros. These devices are supported by ath(4) and require the following line to be added to `/boot/loader.conf`:

```
if_ath_load="YES"
```

The Atheros driver is split up into three separate pieces: the driver (ath(4)), the hardware support layer that handles chip-specific functions (ath_hal(4)), and an algorithm for selecting the rate for transmitting frames. When this support is loaded as kernel modules, any dependencies are automatically handled. To load support for a different type of wireless device, specify the module for that device. This example is for devices based on the Intersil Prism parts (wi(4)) driver:

```
if_wi_load="YES"
```

> **Note:** The examples in this section use an ath(4) device and the device name in the examples must be changed according to the configuration. A list of available wireless drivers and supported adapters can be found in the FreeBSD Hardware Notes, available on the Release Information (http://www.FreeBSD.org/releases/index.html) page of the FreeBSD website. If a native FreeBSD driver for the wireless device does not exist, it may be possible to use the Windows driver with the help of the NDIS driver wrapper.

In addition, the modules that implement cryptographic support for the security protocols to use must be loaded. These are intended to be dynamically loaded on demand by the wlan(4) module, but for now they must be manually configured. The following modules are available: wlan_wep(4), wlan_ccmp(4), and wlan_tkip(4). The wlan_ccmp(4) and wlan_tkip(4) drivers are only needed when using the WPA or 802.11i security protocols. If the network does not use encryption, wlan_wep(4) support is not needed. To load these modules at boot time, add the following lines to `/boot/loader.conf`:

```
wlan_wep_load="YES"
wlan_ccmp_load="YES"
wlan_tkip_load="YES"
```

Once this information has been added to `/boot/loader.conf`, reboot the FreeBSD box. Alternately, load the modules by hand using kldload(8).

> **Note:** For users who do not want to use modules, it is possible to compile these drivers into the kernel by adding the following lines to a custom kernel configuration file:
>
> ```
> device wlan             # 802.11 support
> device wlan_wep         # 802.11 WEP support
> device wlan_ccmp        # 802.11 CCMP support
> device wlan_tkip        # 802.11 TKIP support
> device wlan_amrr        # AMRR transmit rate control algorithm
> device ath              # Atheros pci/cardbus NIC's
> device ath_hal          # pci/cardbus chip support
> ```

```
options AH_SUPPORT_AR5416 # enable AR5416 tx/rx descriptors
device ath_rate_sample   # SampleRate tx rate control for ath
```

With this information in the kernel configuration file, recompile the kernel and reboot the FreeBSD machine.

Information about the wireless device should appear in the boot messages, like this:

```
ath0: <Atheros 5212> mem 0x88000000-0x8800ffff irq 11 at device 0.0 on cardbus1
ath0: [ITHREAD]
ath0: AR2413 mac 7.9 RF2413 phy 4.5
```

32.3.3 Infrastructure Mode

Infrastructure (BSS) mode is the mode that is typically used. In this mode, a number of wireless access points are connected to a wired network. Each wireless network has its own name, called the SSID. Wireless clients connect to the wireless access points.

32.3.3.1 FreeBSD Clients

32.3.3.1.1 How to Find Access Points

To scan for available networks, use ifconfig(8). This request may take a few moments to complete as it requires the system to switch to each available wireless frequency and probe for available access points. Only the superuser can initiate a scan:

```
# ifconfig wlan0 create wlandev ath0
# ifconfig wlan0 up scan
SSID/MESH ID    BSSID              CHAN RATE   S:N     INT CAPS
dlinkap         00:13:46:49:41:76   11   54M -90:96   100 EPS  WPA WME
freebsdap       00:11:95:c3:0d:ac    1   54M -83:96   100 EPS  WPA
```

Note: The interface must be `up` before it can scan. Subsequent scan requests do not require the interface to be marked as up again.

The output of a scan request lists each BSS/IBSS network found. Besides listing the name of the network, the `SSID`, the output also shows the `BSSID`, which is the MAC address of the access point. The `CAPS` field identifies the type of each network and the capabilities of the stations operating there:

Table 32-1. Station Capability Codes

Capability Code	Meaning
`E`	Extended Service Set (ESS). Indicates that the station is part of an infrastructure network rather than an IBSS/ad-hoc network.

Capability Code	Meaning
I	IBSS/ad-hoc network. Indicates that the station is part of an ad-hoc network rather than an ESS network.
P	Privacy. Encryption is required for all data frames exchanged within the BSS using cryptographic means such as WEP, TKIP or AES-CCMP.
S	Short Preamble. Indicates that the network is using short preambles, defined in 802.11b High Rate/DSSS PHY, and utilizes a 56 bit sync field rather than the 128 bit field used in long preamble mode.
s	Short slot time. Indicates that the 802.11g network is using a short slot time because there are no legacy (802.11b) stations present.

One can also display the current list of known networks with:

```
# ifconfig wlan0 list scan
```

This information may be updated automatically by the adapter or manually with a `scan` request. Old data is automatically removed from the cache, so over time this list may shrink unless more scans are done.

32.3.3.1.2 Basic Settings

This section provides a simple example of how to make the wireless network adapter work in FreeBSD without encryption. Once familiar with these concepts, it is strongly recommend to use WPA to set up the wireless network.

There are three basic steps to configure a wireless network: select an access point, authenticate the station, and configure an IP address. The following sections discuss each step.

32.3.3.1.2.1 Selecting an Access Point

Most of the time, it is sufficient to let the system choose an access point using the builtin heuristics. This is the default behaviour when an interface is marked as up or it is listed in `/etc/rc.conf`:

```
wlans_ath0="wlan0"
ifconfig_wlan0="DHCP"
```

If there are multiple access points, a specific one can be selected by its SSID:

```
wlans_ath0="wlan0"
ifconfig_wlan0="ssid your_ssid_here DHCP"
```

In an environment where there are multiple access points with the same SSID, which is often done to simplify roaming, it may be necessary to associate to one specific device. In this case, the BSSID of the access point can be specified, with or without the SSID:

```
wlans_ath0="wlan0"
ifconfig_wlan0="ssid your_ssid_here bssid xx:xx:xx:xx:xx:xx DHCP"
```

There are other ways to constrain the choice of an access point, such as limiting the set of frequencies the system will scan on. This may be useful for a multi-band wireless card as scanning all the possible channels can be time-consuming. To limit operation to a specific band, use the `mode` parameter:

```
wlans_ath0="wlan0"
ifconfig_wlan0="mode 11g ssid your_ssid_here DHCP"
```

This example will force the card to operate in 802.11g, which is defined only for 2.4GHz frequencies so any 5GHz channels will not be considered. This can also be achieved with the `channel` parameter, which locks operation to one specific frequency, and the `chanlist` parameter, to specify a list of channels for scanning. More information about these parameters can be found in ifconfig(8).

32.3.3.1.2.2 Authentication

Once an access point is selected, the station needs to authenticate before it can pass data. Authentication can happen in several ways. The most common scheme, open authentication, allows any station to join the network and communicate. This is the authentication to use for test purposes the first time a wireless network is setup. Other schemes require cryptographic handshakes to be completed before data traffic can flow, either using pre-shared keys or secrets, or more complex schemes that involve backend services such as RADIUS. Open authentication is the default setting. The next most common setup is WPA-PSK, also known as WPA Personal, which is described in Section 32.3.3.1.3.1.

> **Note:** If using an Apple AirPort® Extreme base station for an access point, shared-key authentication together with a WEP key needs to be configured. This can be configured in `/etc/rc.conf` or by using wpa_supplicant(8). For a single AirPort base station, access can be configured with:
>
> ```
> wlans_ath0="wlan0"
> ifconfig_wlan0="authmode shared wepmode on weptxkey 1 wepkey 01234567 DHCP"
> ```
>
> In general, shared key authentication should be avoided because it uses the WEP key material in a highly-constrained manner, making it even easier to crack the key. If WEP must be used for compatibility with legacy devices, it is better to use WEP with `open` authentication. More information regarding WEP can be found in Section 32.3.3.1.4.

32.3.3.1.2.3 Getting an IP Address with DHCP

Once an access point is selected and the authentication parameters are set, an IP address must be obtained in order to communicate. Most of the time, the IP address is obtained via DHCP. To achieve that, edit `/etc/rc.conf` and add `DHCP` to the configuration for the device:

```
wlans_ath0="wlan0"
ifconfig_wlan0="DHCP"
```

The wireless interface is now ready to bring up:

```
# service netif start
```

Once the interface is running, use ifconfig(8) to see the status of the interface `ath0`:

```
# ifconfig wlan0
wlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        ether 00:11:95:d5:43:62
        inet 192.168.1.100 netmask 0xffffff00 broadcast 192.168.1.255
        media: IEEE 802.11 Wireless Ethernet OFDM/54Mbps mode 11g
        status: associated
        ssid dlinkap channel 11 (2462 Mhz 11g) bssid 00:13:46:49:41:76
        country US ecm authmode OPEN privacy OFF txpower 21.5 bmiss 7
        scanvalid 60 bgscan bgscanintvl 300 bgscanidle 250 roam:rssi 7
        roam:rate 5 protmode CTS wme burst
```

The `status: associated` line means that it is connected to the wireless network. The `bssid 00:13:46:49:41:76` is the MAC address of the access point and `authmode OPEN` indicates that the communication is not encrypted.

32.3.3.1.2.4 Static IP Address

In an IP address cannot be obtained from a DHCP server, set a fixed IP address. Replace the `DHCP` keyword shown above with the address information. Be sure to retain any other parameters for selecting the access point:

```
wlans_ath0="wlan0"
ifconfig_wlan0="inet 192.168.1.100 netmask 255.255.255.0 ssid your_ssid_here"
```

32.3.3.1.3 WPA

Wi-Fi Protected Access (WPA) is a security protocol used together with 802.11 networks to address the lack of proper authentication and the weakness of WEP. WPA leverages the 802.1X authentication protocol and uses one of several ciphers instead of WEP for data integrity. The only cipher required by WPA is the Temporary Key Integrity Protocol (TKIP). TKIP is a cipher that extends the basic RC4 cipher used by WEP by adding integrity checking, tamper detection, and measures for responding to detected intrusions. TKIP is designed to work on legacy hardware with only software modification. It represents a compromise that improves security but is still not entirely immune to attack. WPA also specifies the AES-CCMP cipher as an alternative to TKIP, and that is preferred when possible. For this specification, the term WPA2 or RSN is commonly used.

WPA defines authentication and encryption protocols. Authentication is most commonly done using one of two techniques: by 802.1X and a backend authentication service such as RADIUS, or by a minimal handshake between the station and the access point using a pre-shared secret. The former is commonly termed WPA Enterprise and the latter is known as WPA Personal. Since most people will not set up a RADIUS backend server for their wireless network, WPA-PSK is by far the most commonly encountered configuration for WPA.

The control of the wireless connection and the key negotiation or authentication with a server is done using wpa_supplicant(8). This program requires a configuration file, `/etc/wpa_supplicant.conf`, to run. More information regarding this file can be found in wpa_supplicant.conf(5).

32.3.3.1.3.1 WPA-PSK

WPA-PSK, also known as WPA Personal, is based on a pre-shared key (PSK) which is generated from a given password and used as the master key in the wireless network. This means every wireless user will share the same key. WPA-PSK is intended for small networks where the use of an authentication server is not possible or desired.

Warning: Always use strong passwords that are sufficiently long and made from a rich alphabet so that they will not be easily guessed or attacked.

The first step is the configuration of `/etc/wpa_supplicant.conf` with the SSID and the pre-shared key of the network:

```
network={
  ssid="freebsdap"
  psk="freebsdmall"
}
```

Then, in `/etc/rc.conf`, indicate that the wireless device configuration will be done with WPA and the IP address will be obtained with DHCP:

```
wlans_ath0="wlan0"
ifconfig_wlan0="WPA DHCP"
```

Then, bring up the interface:

```
# service netif start
Starting wpa_supplicant.
DHCPDISCOVER on wlan0 to 255.255.255.255 port 67 interval 5
DHCPDISCOVER on wlan0 to 255.255.255.255 port 67 interval 6
DHCPOFFER from 192.168.0.1
DHCPREQUEST on wlan0 to 255.255.255.255 port 67
DHCPACK from 192.168.0.1
bound to 192.168.0.254 -- renewal in 300 seconds.
wlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
      ether 00:11:95:d5:43:62
      inet 192.168.0.254 netmask 0xffffff00 broadcast 192.168.0.255
      media: IEEE 802.11 Wireless Ethernet OFDM/36Mbps mode 11g
      status: associated
      ssid freebsdap channel 1 (2412 Mhz 11g) bssid 00:11:95:c3:0d:ac
      country US ecm authmode WPA2/802.11i privacy ON deftxkey UNDEF
      AES-CCM 3:128-bit txpower 21.5 bmiss 7 scanvalid 450 bgscan
      bgscanintvl 300 bgscanidle 250 roam:rssi 7 roam:rate 5 protmode CTS
      wme burst roaming MANUAL
```

Or, try to configure the interface manually using the information in `/etc/wpa_supplicant.conf`:

```
# wpa_supplicant -i wlan0 -c /etc/wpa_supplicant.conf
Trying to associate with 00:11:95:c3:0d:ac (SSID='freebsdap' freq=2412 MHz)
Associated with 00:11:95:c3:0d:ac
WPA: Key negotiation completed with 00:11:95:c3:0d:ac [PTK=CCMP GTK=CCMP]
CTRL-EVENT-CONNECTED - Connection to 00:11:95:c3:0d:ac completed (auth) [id=0 id_str=]
```

The next operation is to launch dhclient(8) to get the IP address from the DHCP server:

```
# dhclient wlan0
DHCPREQUEST on wlan0 to 255.255.255.255 port 67
DHCPACK from 192.168.0.1
bound to 192.168.0.254 -- renewal in 300 seconds.
```

```
# ifconfig wlan0
wlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
      ether 00:11:95:d5:43:62
      inet 192.168.0.254 netmask 0xffffff00 broadcast 192.168.0.255
      media: IEEE 802.11 Wireless Ethernet OFDM/36Mbps mode 11g
      status: associated
      ssid freebsdap channel 1 (2412 Mhz 11g) bssid 00:11:95:c3:0d:ac
      country US ecm authmode WPA2/802.11i privacy ON deftxkey UNDEF
      AES-CCM 3:128-bit txpower 21.5 bmiss 7 scanvalid 450 bgscan
      bgscanintvl 300 bgscanidle 250 roam:rssi 7 roam:rate 5 protmode CTS
      wme burst roaming MANUAL
```

Note: If `/etc/rc.conf` has an `ifconfig_wlan0="DHCP"` entry, dhclient(8) will be launched automatically after wpa_supplicant(8) associates with the access point.

If DHCP is not possible or desired, set a static IP address after wpa_supplicant(8) has authenticated the station:

```
# ifconfig wlan0 inet 192.168.0.100 netmask 255.255.255.0
# ifconfig wlan0
wlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
      ether 00:11:95:d5:43:62
      inet 192.168.0.100 netmask 0xffffff00 broadcast 192.168.0.255
      media: IEEE 802.11 Wireless Ethernet OFDM/36Mbps mode 11g
      status: associated
      ssid freebsdap channel 1 (2412 Mhz 11g) bssid 00:11:95:c3:0d:ac
      country US ecm authmode WPA2/802.11i privacy ON deftxkey UNDEF
      AES-CCM 3:128-bit txpower 21.5 bmiss 7 scanvalid 450 bgscan
      bgscanintvl 300 bgscanidle 250 roam:rssi 7 roam:rate 5 protmode CTS
      wme burst roaming MANUAL
```

When DHCP is not used, the default gateway and the nameserver also have to be manually set:

```
# route add default your_default_router
# echo "nameserver your_DNS_server" >> /etc/resolv.conf
```

32.3.3.1.3.2 WPA with EAP-TLS

The second way to use WPA is with an 802.1X backend authentication server. In this case, WPA is called WPA Enterprise to differentiate it from the less secure WPA Personal. Authentication in WPA Enterprise is based on the Extensible Authentication Protocol (EAP).

EAP does not come with an encryption method. Instead, EAP is embedded inside an encrypted tunnel. There are many EAP authentication methods, but EAP-TLS, EAP-TTLS, and EAP-PEAP are the most common.

EAP with Transport Layer Security (EAP-TLS) is a well-supported wireless authentication protocol since it was the first EAP method to be certified by the Wi-Fi alliance (http://www.wi-fi.org/). EAP-TLS requires three certificates to run: the certificate of the Certificate Authority (CA) installed on all machines, the server certificate for the authentication server, and one client certificate for each wireless client. In this EAP method, both the authentication server and wireless client authenticate each other by presenting their respective certificates, and then verify that these certificates were signed by the organization's CA.

As previously, the configuration is done via `/etc/wpa_supplicant.conf`:

```
network={
  ssid="freebsdap" ❶
  proto=RSN  ❷
  key_mgmt=WPA-EAP ❸
  eap=TLS ❹
  identity="loader" ❺
  ca_cert="/etc/certs/cacert.pem" ❻
  client_cert="/etc/certs/clientcert.pem" ❼
  private_key="/etc/certs/clientkey.pem" ❽
  private_key_passwd="freebsdmallclient" ❾
}
```

❶ This field indicates the network name (SSID).

❷ This example uses the RSN IEEE 802.11i protocol, also known as WPA2.

❸ The `key_mgmt` line refers to the key management protocol to use. In this example, it is WPA using EAP authentication.

❹ This field indicates the EAP method for the connection.

❺ The `identity` field contains the identity string for EAP.

❻ The `ca_cert` field indicates the pathname of the CA certificate file. This file is needed to verify the server certificate.

❼ The `client_cert` line gives the pathname to the client certificate file. This certificate is unique to each wireless client of the network.

❽ The `private_key` field is the pathname to the client certificate private key file.

❾ The `private_key_passwd` field contains the passphrase for the private key.

Then, add the following lines to `/etc/rc.conf`:

```
wlans_ath0="wlan0"
ifconfig_wlan0="WPA DHCP"
```

The next step is to bring up the interface:

```
# service netif start
Starting wpa_supplicant.
DHCPREQUEST on wlan0 to 255.255.255.255 port 67 interval 7
DHCPREQUEST on wlan0 to 255.255.255.255 port 67 interval 15
DHCPACK from 192.168.0.20
bound to 192.168.0.254 -- renewal in 300 seconds.
wlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
      ether 00:11:95:d5:43:62
      inet 192.168.0.254 netmask 0xffffff00 broadcast 192.168.0.255
      media: IEEE 802.11 Wireless Ethernet DS/11Mbps mode 11g
      status: associated
      ssid freebsdap channel 1 (2412 Mhz 11g) bssid 00:11:95:c3:0d:ac
      country US ecm authmode WPA2/802.11i privacy ON deftxkey UNDEF
      AES-CCM 3:128-bit txpower 21.5 bmiss 7 scanvalid 450 bgscan
```

```
        bgscanintvl 300 bgscanidle 250 roam:rssi 7 roam:rate 5 protmode CTS
        wme burst roaming MANUAL
```

It is also possible to bring up the interface manually using wpa_supplicant(8) and ifconfig(8).

32.3.3.1.3.3 WPA with EAP-TTLS

With EAP-TTLS, both the authentication server and the client need a certificate. With EAP-TTLS, a client certificate is optional. This method is similar to a web server which creates a secure SSL tunnel even if visitors do not have client-side certificates. EAP-TTLS uses an encrypted TLS tunnel for safe transport of the authentication data.

The required configuration can be added to `/etc/wpa_supplicant.conf`:

```
network={
  ssid="freebsdap"
  proto=RSN
  key_mgmt=WPA-EAP
  eap=TTLS ❶
  identity="test" ❷
  password="test" ❸
  ca_cert="/etc/certs/cacert.pem" ❹
  phase2="auth=MD5" ❺
}
```

❶ This field specifies the EAP method for the connection.

❷ The `identity` field contains the identity string for EAP authentication inside the encrypted TLS tunnel.

❸ The `password` field contains the passphrase for the EAP authentication.

❹ The `ca_cert` field indicates the pathname of the CA certificate file. This file is needed to verify the server certificate.

❺ This field specifies the authentication method used in the encrypted TLS tunnel. In this example, EAP with MD5-Challenge is used. The “inner authentication” phase is often called “phase2”.

Next, add the following lines to `/etc/rc.conf`:

```
wlans_ath0="wlan0"
ifconfig_wlan0="WPA DHCP"
```

The next step is to bring up the interface:

```
# service netif start
Starting wpa_supplicant.
DHCPREQUEST on wlan0 to 255.255.255.255 port 67 interval 7
DHCPREQUEST on wlan0 to 255.255.255.255 port 67 interval 15
DHCPREQUEST on wlan0 to 255.255.255.255 port 67 interval 21
DHCPACK from 192.168.0.20
bound to 192.168.0.254 -- renewal in 300 seconds.
wlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
      ether 00:11:95:d5:43:62
      inet 192.168.0.254 netmask 0xffffff00 broadcast 192.168.0.255
      media: IEEE 802.11 Wireless Ethernet DS/11Mbps mode 11g
```

```
status: associated
ssid freebsdap channel 1 (2412 Mhz 11g) bssid 00:11:95:c3:0d:ac
country US ecm authmode WPA2/802.11i privacy ON deftxkey UNDEF
AES-CCM 3:128-bit txpower 21.5 bmiss 7 scanvalid 450 bgscan
bgscanintvl 300 bgscanidle 250 roam:rssi 7 roam:rate 5 protmode CTS
wme burst roaming MANUAL
```

32.3.3.1.3.4 WPA with EAP-PEAP

> **Note:** PEAPv0/EAP-MSCHAPv2 is the most common PEAP method. In this chapter, the term PEAP is used to refer to that method.

Protected EAP (PEAP) is designed as an alternative to EAP-TTLS and is the most used EAP standard after EAP-TLS. In a network with mixed operating systems, PEAP should be the most supported standard after EAP-TLS.

PEAP is similar to EAP-TTLS as it uses a server-side certificate to authenticate clients by creating an encrypted TLS tunnel between the client and the authentication server, which protects the ensuing exchange of authentication information. PEAP authentication differs from EAP-TTLS as it broadcasts the username in the clear and only the password is sent in the encrypted TLS tunnel. EAP-TTLS will use the TLS tunnel for both the username and password.

Add the following lines to `/etc/wpa_supplicant.conf` to configure the EAP-PEAP related settings:

```
network={
  ssid="freebsdap"
  proto=RSN
  key_mgmt=WPA-EAP
  eap=PEAP ❶
  identity="test" ❷
  password="test" ❸
  ca_cert="/etc/certs/cacert.pem" ❹
  phase1="peaplabel=0" ❺
  phase2="auth=MSCHAPV2" ❻
}
```

❶ This field specifies the EAP method for the connection.

❷ The `identity` field contains the identity string for EAP authentication inside the encrypted TLS tunnel.

❸ The `password` field contains the passphrase for the EAP authentication.

❹ The `ca_cert` field indicates the pathname of the CA certificate file. This file is needed to verify the server certificate.

❺ This field contains the parameters for the first phase of authentication, the TLS tunnel. According to the authentication server used, specify a specific label for authentication. Most of the time, the label will be “client EAP encryption” which is set by using `peaplabel=0`. More information can be found in wpa_supplicant.conf(5).

❻ This field specifies the authentication protocol used in the encrypted TLS tunnel. In the case of PEAP, it is `auth=MSCHAPV2`.

Add the following to `/etc/rc.conf`:

```
wlans_ath0="wlan0"
ifconfig_wlan0="WPA DHCP"
```

Then, bring up the interface:

```
# service netif start
Starting wpa_supplicant.
DHCPREQUEST on wlan0 to 255.255.255.255 port 67 interval 7
DHCPREQUEST on wlan0 to 255.255.255.255 port 67 interval 15
DHCPREQUEST on wlan0 to 255.255.255.255 port 67 interval 21
DHCPACK from 192.168.0.20
bound to 192.168.0.254 -- renewal in 300 seconds.
wlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
      ether 00:11:95:d5:43:62
      inet 192.168.0.254 netmask 0xffffff00 broadcast 192.168.0.255
      media: IEEE 802.11 Wireless Ethernet DS/11Mbps mode 11g
      status: associated
      ssid freebsdap channel 1 (2412 Mhz 11g) bssid 00:11:95:c3:0d:ac
      country US ecm authmode WPA2/802.11i privacy ON deftxkey UNDEF
      AES-CCM 3:128-bit txpower 21.5 bmiss 7 scanvalid 450 bgscan
      bgscanintvl 300 bgscanidle 250 roam:rssi 7 roam:rate 5 protmode CTS
      wme burst roaming MANUAL
```

32.3.3.1.4 WEP

Wired Equivalent Privacy (WEP) is part of the original 802.11 standard. There is no authentication mechanism, only a weak form of access control which is easily cracked.

WEP can be set up using ifconfig(8):

```
# ifconfig wlan0 create wlandev ath0
# ifconfig wlan0 inet 192.168.1.100 netmask 255.255.255.0 \
     ssid my_net wepmode on weptxkey 3 wepkey 3:0x3456789012
```

- The `weptxkey` specifies which WEP key will be used in the transmission. This example uses the third key. This must match the setting on the access point. When unsure which key is used by the access point, try `1` (the first key) for this value.
- The `wepkey` selects one of the WEP keys. It should be in the format *`index:key`*. Key `1` is used by default; the index only needs to be set when using a key other than the first key.

> **Note:** Replace the `0x3456789012` with the key configured for use on the access point.

Refer to ifconfig(8) for further information.

The wpa_supplicant(8) facility can be used to configure a wireless interface with WEP. The example above can be set up by adding the following lines to `/etc/wpa_supplicant.conf`:

```
network={
  ssid="my_net"
  key_mgmt=NONE
  wep_key3=3456789012
  wep_tx_keyidx=3
}
```

Then:

```
# wpa_supplicant -i wlan0 -c /etc/wpa_supplicant.conf
Trying to associate with 00:13:46:49:41:76 (SSID='dlinkap' freq=2437 MHz)
Associated with 00:13:46:49:41:76
```

32.3.4 Ad-hoc Mode

IBSS mode, also called ad-hoc mode, is designed for point to point connections. For example, to establish an ad-hoc network between the machines `A` and `B`, choose two IP addresses and a SSID.

On `A`:

```
# ifconfig wlan0 create wlandev ath0 wlanmode adhoc
# ifconfig wlan0 inet 192.168.0.1 netmask 255.255.255.0 ssid freebsdap
# ifconfig wlan0
  wlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
         ether 00:11:95:c3:0d:ac
         inet 192.168.0.1 netmask 0xffffff00 broadcast 192.168.0.255
         media: IEEE 802.11 Wireless Ethernet autoselect mode 11g <adhoc>
         status: running
         ssid freebsdap channel 2 (2417 Mhz 11g) bssid 02:11:95:c3:0d:ac
         country US ecm authmode OPEN privacy OFF txpower 21.5 scanvalid 60
         protmode CTS wme burst
```

The `adhoc` parameter indicates that the interface is running in IBSS mode.

`B` should now be able to detect `A`:

```
# ifconfig wlan0 create wlandev ath0 wlanmode adhoc
# ifconfig wlan0 up scan
  SSID/MESH ID    BSSID              CHAN RATE   S:N     INT CAPS
  freebsdap       02:11:95:c3:0d:ac    2   54M -64:-96  100 IS   WME
```

The `I` in the output confirms that `A` is in ad-hoc mode. Now, configure `B` with a different IP address:

```
# ifconfig wlan0 inet 192.168.0.2 netmask 255.255.255.0 ssid freebsdap
# ifconfig wlan0
  wlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
         ether 00:11:95:d5:43:62
         inet 192.168.0.2 netmask 0xffffff00 broadcast 192.168.0.255
```

```
        media: IEEE 802.11 Wireless Ethernet autoselect mode 11g <adhoc>
        status: running
        ssid freebsdap channel 2 (2417 Mhz 11g) bssid 02:11:95:c3:0d:ac
        country US ecm authmode OPEN privacy OFF txpower 21.5 scanvalid 60
        protmode CTS wme burst
```

Both A and B are now ready to exchange information.

32.3.5 FreeBSD Host Access Points

FreeBSD can act as an Access Point (AP) which eliminates the need to buy a hardware AP or run an ad-hoc network. This can be particularly useful when a FreeBSD machine is acting as a gateway to another network such as the Internet.

32.3.5.1 Basic Settings

Before configuring a FreeBSD machine as an AP, the kernel must be configured with the appropriate networking support for the wireless card as well as the security protocols being used. For more details, see Section 32.3.2.

> **Note:** The NDIS driver wrapper for Windows drivers does not currently support AP operation. Only native FreeBSD wireless drivers support AP mode.

Once wireless networking support is loaded, check if the wireless device supports the host-based access point mode, also known as hostap mode:

```
# ifconfig wlan0 create wlandev ath0
# ifconfig wlan0 list caps
drivercaps=6f85edc1<STA,FF,TURBOP,IBSS,HOSTAP,AHDEMO,TXPMGT,SHSLOT,SHPREAMBLE,MONITOR,MBSS,WPA1,W
cryptocaps=1f<WEP,TKIP,AES,AES_CCM,TKIPMIC>
```

This output displays the card's capabilities. The HOSTAP word confirms that this wireless card can act as an AP. Various supported ciphers are also listed: WEP, TKIP, and AES. This information indicates which security protocols can be used on the AP.

The wireless device can only be put into hostap mode during the creation of the network pseudo-device, so a previously created device must be destroyed first:

```
# ifconfig wlan0 destroy
```

then regenerated with the correct option before setting the other parameters:

```
# ifconfig wlan0 create wlandev ath0 wlanmode hostap
# ifconfig wlan0 inet 192.168.0.1 netmask 255.255.255.0 ssid freebsdap mode 11g channel 1
```

Use ifconfig(8) again to see the status of the wlan0 interface:

```
# ifconfig wlan0
  wlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
         ether 00:11:95:c3:0d:ac
         inet 192.168.0.1 netmask 0xffffff00 broadcast 192.168.0.255
```

```
        media: IEEE 802.11 Wireless Ethernet autoselect mode 11g <hostap>
        status: running
        ssid freebsdap channel 1 (2412 Mhz 11g) bssid 00:11:95:c3:0d:ac
        country US ecm authmode OPEN privacy OFF txpower 21.5 scanvalid 60
        protmode CTS wme burst dtimperiod 1 -dfs
```

The `hostap` parameter indicates the interface is running in the host-based access point mode.

The interface configuration can be done automatically at boot time by adding the following lines to `/etc/rc.conf`:

```
wlans_ath0="wlan0"
create_args_wlan0="wlanmode hostap"
ifconfig_wlan0="inet 192.168.0.1 netmask 255.255.255.0 ssid freebsdap mode 11g channel 1"
```

32.3.5.2 Host-based Access Point Without Authentication or Encryption

Although it is not recommended to run an AP without any authentication or encryption, this is a simple way to check if the AP is working. This configuration is also important for debugging client issues.

Once the AP is configured, initiate a scan from another wireless machine to find the AP:

```
# ifconfig wlan0 create wlandev ath0
# ifconfig wlan0 up scan
SSID/MESH ID    BSSID              CHAN RATE   S:N     INT CAPS
freebsdap       00:11:95:c3:0d:ac    1   54M -66:-96  100 ES   WME
```

The client machine found the AP and can be associated with it:

```
# ifconfig wlan0 inet 192.168.0.2 netmask 255.255.255.0 ssid freebsdap
# ifconfig wlan0
  wlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
         ether 00:11:95:d5:43:62
         inet 192.168.0.2 netmask 0xffffff00 broadcast 192.168.0.255
         media: IEEE 802.11 Wireless Ethernet OFDM/54Mbps mode 11g
         status: associated
         ssid freebsdap channel 1 (2412 Mhz 11g) bssid 00:11:95:c3:0d:ac
         country US ecm authmode OPEN privacy OFF txpower 21.5 bmiss 7
         scanvalid 60 bgscan bgscanintvl 300 bgscanidle 250 roam:rssi 7
         roam:rate 5 protmode CTS wme burst
```

32.3.5.3 WPA Host-based Access Point

This section focuses on setting up a FreeBSD AP using the WPA security protocol. More details regarding WPA and the configuration of WPA-based wireless clients can be found in Section 32.3.3.1.3.

The hostapd(8) daemon is used to deal with client authentication and key management on the WPA-enabled AP.

The following configuration operations are performed on the FreeBSD machine acting as the AP. Once the AP is correctly working, hostapd(8) should be automatically enabled at boot with the following line in `/etc/rc.conf`:

```
hostapd_enable="YES"
```

Before trying to configure hostapd(8), first configure the basic settings introduced in Section 32.3.5.1.

32.3.5.3.1 WPA-PSK

WPA-PSK is intended for small networks where the use of a backend authentication server is not possible or desired.

The configuration is done in `/etc/hostapd.conf`:

```
interface=wlan0 ❶
debug=1 ❷
ctrl_interface=/var/run/hostapd ❸
ctrl_interface_group=wheel ❹
ssid=freebsdap ❺
wpa=1 ❻
wpa_passphrase=freebsdmall ❼
wpa_key_mgmt=WPA-PSK ❽
wpa_pairwise=CCMP TKIP ❾
```

❶ This field indicates the wireless interface used for the AP.

❷ This field sets the level of verbosity during the execution of hostapd(8). A value of `1` represents the minimal level.

❸ The `ctrl_interface` field gives the pathname of the directory used by hostapd(8) to store its domain socket files for the communication with external programs such as hostapd_cli(8). The default value is used in this example.

❹ The `ctrl_interface_group` line sets the group which is allowed to access the control interface files.

❺ This field sets the network name.

❻ The `wpa` field enables WPA and specifies which WPA authentication protocol will be required. A value of `1` configures the AP for WPA-PSK.

❼ The `wpa_passphrase` field contains the ASCII passphrase for WPA authentication.

> **Warning:** Always use strong passwords that are sufficiently long and made from a rich alphabet so that they will not be easily guessed or attacked.

❽ The `wpa_key_mgmt` line refers to the key management protocol to use. This example sets WPA-PSK.

❾ The `wpa_pairwise` field indicates the set of accepted encryption algorithms by the AP. In this example, both TKIP (WPA) and CCMP (WPA2) ciphers are accepted. The CCMP cipher is an alternative to TKIP and is strongly preferred when possible. TKIP should be used solely for stations incapable of doing CCMP.

The next step is to start hostapd(8):

```
# service hostapd forcestart

# ifconfig wlan0
  wlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 2290
         inet 192.168.0.1 netmask 0xffffff00 broadcast 192.168.0.255
         inet6 fe80::211:95ff:fec3:dac%ath0 prefixlen 64 scopeid 0x4
         ether 00:11:95:c3:0d:ac
         media: IEEE 802.11 Wireless Ethernet autoselect mode 11g <hostap>
         status: associated
         ssid freebsdap channel 1 bssid 00:11:95:c3:0d:ac
```

```
        authmode WPA2/802.11i privacy MIXED deftxkey 2 TKIP 2:128-bit txpowmax 36 protmode CTS d
```

Once the AP is running, the clients can associate with it. See Section 32.3.3.1.3 for more details. It is possible to see the stations associated with the AP using `ifconfig wlan0 list sta`.

32.3.5.4 WEP Host-based Access Point

It is not recommended to use WEP for setting up an AP since there is no authentication mechanism and the encryption is easily cracked. Some legacy wireless cards only support WEP and these cards will only support an AP without authentication or encryption.

The wireless device can now be put into hostap mode and configured with the correct SSID and IP address:

```
# ifconfig wlan0 create wlandev ath0 wlanmode hostap
# ifconfig wlan0 inet 192.168.0.1 netmask 255.255.255.0 \
 ssid freebsdap wepmode on weptxkey 3 wepkey 3:0x3456789012 mode 11g
```

- The `weptxkey` indicates which WEP key will be used in the transmission. This example uses the third key as key numbering starts with `1`. This parameter must be specified in order to encrypt the data.
- The `wepkey` sets the selected WEP key. It should be in the format *`index:key`*. If the index is not given, key `1` is set. The index needs to be set when using keys other than the first key.

Use ifconfig(8) to see the status of the `wlan0` interface:

```
# ifconfig wlan0
  wlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
         ether 00:11:95:c3:0d:ac
         inet 192.168.0.1 netmask 0xffffff00 broadcast 192.168.0.255
         media: IEEE 802.11 Wireless Ethernet autoselect mode 11g <hostap>
         status: running
         ssid freebsdap channel 4 (2427 Mhz 11g) bssid 00:11:95:c3:0d:ac
         country US ecm authmode OPEN privacy ON deftxkey 3 wepkey 3:40-bit
         txpower 21.5 scanvalid 60 protmode CTS wme burst dtimperiod 1 -dfs
```

From another wireless machine, it is now possible to initiate a scan to find the AP:

```
# ifconfig wlan0 create wlandev ath0
# ifconfig wlan0 up scan
SSID            BSSID              CHAN RATE  S:N   INT CAPS
freebsdap       00:11:95:c3:0d:ac    1   54M 22:1   100 EPS
```

In this example, the client machine found the AP and can associate with it using the correct parameters. See Section 32.3.3.1.4 for more details.

32.3.6 Using Both Wired and Wireless Connections

A wired connection provides better performance and reliability, while a wireless connection provides flexibility and mobility. Laptop users typically want to roam seamlessly between the two types of connections.

On FreeBSD, it is possible to combine two or even more network interfaces together in a “failover” fashion. This type of configuration uses the most preferred and available connection from a group of network interfaces, and the operating system switches automatically when the link state changes.

Link aggregation and failover is covered in Section 32.6 and an example for using both wired and wireless connections is provided at Example 32-3.

32.3.7 Troubleshooting

This section describes a number of steps to help troubleshoot common wireless networking problems.

- If the access point is not listed when scanning, check that the configuration has not limited the wireless device to a limited set of channels.
- If the device cannot associate with an access point, verify that the configuration matches the settings on the access point. This includes the authentication scheme and any security protocols. Simplify the configuration as much as possible. If using a security protocol such as WPA or WEP, configure the access point for open authentication and no security to see if traffic will pass.
- Once the system can associate with the access point, diagnose the security configuration using tools like ping(8).

  Debugging support is provided by wpa_supplicant(8). Try running this utility manually with the `-dd` option and look at the system logs.
- There are many lower-level debugging tools. Debugging messages can be enabled in the 802.11 protocol support layer using wlandebug(8). On a FreeBSD system prior to FreeBSD 9.1, this program can be found in `/usr/src/tools/tools/net80211`. For example, to enable console messages related to scanning for access points and the 802.11 protocol handshakes required to arrange communication:

  ```
  # wlandebug -i ath0 +scan+auth+debug+assoc
    net.wlan.0.debug: 0 => 0xc80000<assoc,auth,scan>
  ```

  Many useful statistics are maintained by the 802.11 layer and `wlanstats`, found in `/usr/src/tools/tools/net80211`, will dump this information. These statistics should display all errors identified by the 802.11 layer. However, some errors are identified in the device drivers that lie below the 802.11 layer so they may not show up. To diagnose device-specific problems, refer to the drivers’ documentation.

If the above information does not help to clarify the problem, submit a problem report and include output from the above tools.

32.4 Bluetooth

Written by Pav Lucistnik.

32.4.1 Introduction

Bluetooth is a wireless technology for creating personal networks operating in the 2.4 GHz unlicensed band, with a range of 10 meters. Networks are usually formed ad-hoc from portable devices such as cellular phones, handhelds and laptops. Unlike Wi-Fi wireless technology, Bluetooth offers higher level service profiles, such as FTP-like file servers, file pushing, voice transport, serial line emulation, and more.

The Bluetooth stack in FreeBSD is implemented using the netgraph(4) framework. A broad variety of Bluetooth USB dongles is supported by ng_ubt(4). Broadcom BCM2033 based Bluetooth devices are supported by the ubtbcmfw(4) and ng_ubt(4) drivers. The 3Com Bluetooth PC Card 3CRWB60-A is supported by the ng_bt3c(4) driver. Serial and UART based Bluetooth devices are supported by sio(4), ng_h4(4) and hcseriald(8). This section describes the use of a USB Bluetooth dongle.

32.4.2 Plugging in the Device

By default, Bluetooth device drivers are available as kernel modules. Before attaching a device, load the driver into the kernel:

```
# kldload ng_ubt
```

If the Bluetooth device is present in the system during system startup, load the module from `/boot/loader.conf`:

```
ng_ubt_load="YES"
```

Plug in the USB dongle. Output similar to the following will appear on the console and in the system log:

```
ubt0: vendor 0x0a12 product 0x0001, rev 1.10/5.25, addr 2
ubt0: Interface 0 endpoints: interrupt=0x81, bulk-in=0x82, bulk-out=0x2
ubt0: Interface 1 (alt.config 5) endpoints: isoc-in=0x83, isoc-out=0x3,
      wMaxPacketSize=49, nframes=6, buffer size=294
```

To start and stop the Bluetooth stack, use service(8). It is a good idea to stop the stack before unplugging the device. When starting the stack, the output should be similar to the following:

```
# service bluetooth start ubt0
BD_ADDR: 00:02:72:00:d4:1a
Features: 0xff 0xff 0xf 00 00 00 00 00
<3-Slot> <5-Slot> <Encryption> <Slot offset>
<Timing accuracy> <Switch> <Hold mode> <Sniff mode>
<Park mode> <RSSI> <Channel quality> <SCO link>
<HV2 packets> <HV3 packets> <u-law log> <A-law log> <CVSD>
<Paging scheme> <Power control> <Transparent SCO data>
Max. ACL packet size: 192 bytes
Number of ACL packets: 8
Max. SCO packet size: 64 bytes
Number of SCO packets: 8
```

32.4.3 Host Controller Interface (HCI)

The Host Controller Interface (HCI) provides a command interface to the baseband controller and link manager as well as access to hardware status and control registers. This interface provides a uniform method for accessing Bluetooth baseband capabilities. The HCI layer on the host exchanges data and commands with the HCI firmware on the Bluetooth hardware. The Host Controller Transport Layer (physical bus) driver provides both HCI layers with the ability to exchange information.

A single netgraph node of type *hci* is created for a single Bluetooth device. The HCI node is normally connected to the downstream Bluetooth device driver node and the upstream L2CAP node. All HCI operations must be performed

on the HCI node and not on the device driver node. The default name for the HCI node is “devicehci”. For more details, refer to ng_hci(4).

One of the most common tasks is discovery of Bluetooth devices in RF proximity. This operation is called *inquiry*. Inquiry and other HCI related operations are done using hccontrol(8). The example below shows how to find out which Bluetooth devices are in range. The list of devices should be displayed in a few seconds. Note that a remote device will only answer the inquiry if it is set to *discoverable* mode.

```
% hccontrol -n ubt0hci inquiry
Inquiry result, num_responses=1
Inquiry result #0
       BD_ADDR: 00:80:37:29:19:a4
       Page Scan Rep. Mode: 0x1
       Page Scan Period Mode: 00
       Page Scan Mode: 00
       Class: 52:02:04
       Clock offset: 0x78ef
Inquiry complete. Status: No error [00]
```

The `BD_ADDR` is the unique address of a Bluetooth device, similar to the MAC address of a network card. This address is needed for further communication with a device. It is possible to assign a human readable name to a BD_ADDR. Information regarding the known Bluetooth hosts is contained in `/etc/bluetooth/hosts`. The following example shows how to obtain the human readable name that was assigned to the remote device:

```
% hccontrol -n ubt0hci remote_name_request 00:80:37:29:19:a4
BD_ADDR: 00:80:37:29:19:a4
Name: Pav’s T39
```

If an inquiry is performed on a remote Bluetooth device, it will find the computer as “your.host.name (ubt0)”. The name assigned to the local device can be changed at any time.

The Bluetooth system provides a point-to-point connection between two Bluetooth units, or a point-to-multipoint connection which is shared among several Bluetooth devices. The following example shows how to obtain the list of active baseband connections for the local device:

```
% hccontrol -n ubt0hci read_connection_list
Remote BD_ADDR    Handle Type Mode Role Encrypt Pending Queue State
00:80:37:29:19:a4     41  ACL    0 MAST    NONE       0     0 OPEN
```

A *connection handle* is useful when termination of the baseband connection is required, though it is normally not required to do this by hand. The stack will automatically terminate inactive baseband connections.

```
# hccontrol -n ubt0hci disconnect 41
Connection handle: 41
Reason: Connection terminated by local host [0x16]
```

Type `hccontrol help` for a complete listing of available HCI commands. Most of the HCI commands do not require superuser privileges.

32.4.4 Logical Link Control and Adaptation Protocol (L2CAP)

The Logical Link Control and Adaptation Protocol (L2CAP) provides connection-oriented and connectionless data services to upper layer protocols with protocol multiplexing capability and segmentation and reassembly operation. L2CAP permits higher level protocols and applications to transmit and receive L2CAP data packets up to 64 kilobytes in length.

L2CAP is based around the concept of *channels*. A channel is a logical connection on top of a baseband connection. Each channel is bound to a single protocol in a many-to-one fashion. Multiple channels can be bound to the same protocol, but a channel cannot be bound to multiple protocols. Each L2CAP packet received on a channel is directed to the appropriate higher level protocol. Multiple channels can share the same baseband connection.

A single netgraph node of type *l2cap* is created for a single Bluetooth device. The L2CAP node is normally connected to the downstream Bluetooth HCI node and upstream Bluetooth socket nodes. The default name for the L2CAP node is “devicel2cap”. For more details refer to ng_l2cap(4).

A useful command is l2ping(8), which can be used to ping other devices. Some Bluetooth implementations might not return all of the data sent to them, so `0 bytes` in the following example is normal.

```
# l2ping -a 00:80:37:29:19:a4
0 bytes from 0:80:37:29:19:a4 seq_no=0 time=48.633 ms result=0
0 bytes from 0:80:37:29:19:a4 seq_no=1 time=37.551 ms result=0
0 bytes from 0:80:37:29:19:a4 seq_no=2 time=28.324 ms result=0
0 bytes from 0:80:37:29:19:a4 seq_no=3 time=46.150 ms result=0
```

The l2control(8) utility is used to perform various operations on L2CAP nodes. This example shows how to obtain the list of logical connections (channels) and the list of baseband connections for the local device:

```
% l2control -a 00:02:72:00:d4:1a read_channel_list
L2CAP channels:
Remote BD_ADDR     SCID/ DCID   PSM  IMTU/ OMTU State
00:07:e0:00:0b:ca    66/   64     3   132/  672 OPEN
% l2control -a 00:02:72:00:d4:1a read_connection_list
L2CAP connections:
Remote BD_ADDR    Handle Flags Pending State
00:07:e0:00:0b:ca     41 O           0 OPEN
```

Another diagnostic tool is btsockstat(1). It is similar to netstat(1), but for Bluetooth network-related data structures. The example below shows the same logical connection as l2control(8) above.

```
% btsockstat
Active L2CAP sockets
PCB      Recv-Q Send-Q Local address/PSM       Foreign address   CID   State
c2afe900      0      0 00:02:72:00:d4:1a/3     00:07:e0:00:0b:ca 66    OPEN
Active RFCOMM sessions
L2PCB    PCB      Flag MTU   Out-Q DLCs State
c2afe900 c2b53380 1    127   0     Yes  OPEN
Active RFCOMM sockets
PCB      Recv-Q Send-Q Local address     Foreign address   Chan DLCI State
c2e8bc80      0    250 00:02:72:00:d4:1a 00:07:e0:00:0b:ca 3    6    OPEN
```

32.4.5 RFCOMM Protocol

The RFCOMM protocol provides emulation of serial ports over the L2CAP protocol. The protocol is based on the ETSI standard TS 07.10. RFCOMM is a simple transport protocol, with additional provisions for emulating the 9 circuits of RS-232 (EIATIA-232-E) serial ports. RFCOMM supports up to 60 simultaneous connections (RFCOMM channels) between two Bluetooth devices.

For the purposes of RFCOMM, a complete communication path involves two applications running on the communication endpoints with a communication segment between them. RFCOMM is intended to cover applications that make use of the serial ports of the devices in which they reside. The communication segment is a direct connect Bluetooth link from one device to another.

RFCOMM is only concerned with the connection between the devices in the direct connect case, or between the device and a modem in the network case. RFCOMM can support other configurations, such as modules that communicate via Bluetooth wireless technology on one side and provide a wired interface on the other side.

In FreeBSD, RFCOMM is implemented at the Bluetooth sockets layer.

32.4.6 Pairing of Devices

By default, Bluetooth communication is not authenticated, and any device can talk to any other device. A Bluetooth device, such as a cellular phone, may choose to require authentication to provide a particular service. Bluetooth authentication is normally done with a *PIN code*, an ASCII string up to 16 characters in length. The user is required to enter the same PIN code on both devices. Once the user has entered the PIN code, both devices will generate a *link key*. After that, the link key can be stored either in the devices or in a persistent storage. Next time, both devices will use the previously generated link key. This procedure is called *pairing*. Note that if the link key is lost by either device, the pairing must be repeated.

The hcsecd(8) daemon is responsible for handling Bluetooth authentication requests. The default configuration file is `/etc/bluetooth/hcsecd.conf`. An example section for a cellular phone with the PIN code arbitrarily set to “1234” is shown below:

```
device {
        bdaddr  00:80:37:29:19:a4;
        name    "Pav's T39";
        key     nokey;
        pin     "1234";
      }
```

The only limitation on PIN codes is length. Some devices, such as Bluetooth headsets, may have a fixed PIN code built in. The `-d` switch forces hcsecd(8) to stay in the foreground, so it is easy to see what is happening. Set the remote device to receive pairing and initiate the Bluetooth connection to the remote device. The remote device should indicate that pairing was accepted and request the PIN code. Enter the same PIN code listed in `hcsecd.conf`. Now the computer and the remote device are paired. Alternatively, pairing can be initiated on the remote device.

The following line can be added to `/etc/rc.conf` to configure hcsecd(8) to start automatically on system start:

```
hcsecd_enable="YES"
```

The following is a sample of the hcsecd(8) daemon output:

```
hcsecd[16484]: Got Link_Key_Request event from 'ubt0hci', remote bdaddr 0:80:37:29:19:a4
hcsecd[16484]: Found matching entry, remote bdaddr 0:80:37:29:19:a4, name 'Pav's T39', link key d
```

```
hcsecd[16484]: Sending Link_Key_Negative_Reply to 'ubt0hci' for remote bdaddr 0:80:37:29:19:a4
hcsecd[16484]: Got PIN_Code_Request event from 'ubt0hci', remote bdaddr 0:80:37:29:19:a4
hcsecd[16484]: Found matching entry, remote bdaddr 0:80:37:29:19:a4, name 'Pav's T39', PIN code ex
hcsecd[16484]: Sending PIN_Code_Reply to 'ubt0hci' for remote bdaddr 0:80:37:29:19:a4
```

32.4.7 Service Discovery Protocol (SDP)

The Service Discovery Protocol (SDP) provides the means for client applications to discover the existence of services provided by server applications as well as the attributes of those services. The attributes of a service include the type or class of service offered and the mechanism or protocol information needed to utilize the service.

SDP involves communication between a SDP server and a SDP client. The server maintains a list of service records that describe the characteristics of services associated with the server. Each service record contains information about a single service. A client may retrieve information from a service record maintained by the SDP server by issuing a SDP request. If the client, or an application associated with the client, decides to use a service, it must open a separate connection to the service provider in order to utilize the service. SDP provides a mechanism for discovering services and their attributes, but it does not provide a mechanism for utilizing those services.

Normally, a SDP client searches for services based on some desired characteristics of the services. However, there are times when it is desirable to discover which types of services are described by an SDP server's service records without any prior information about the services. This process of looking for any offered services is called *browsing*.

The Bluetooth SDP server, sdpd(8), and command line client, sdpcontrol(8), are included in the standard FreeBSD installation. The following example shows how to perform a SDP browse query.

```
% sdpcontrol -a 00:01:03:fc:6e:ec browse
Record Handle: 00000000
Service Class ID List:
        Service Discovery Server (0x1000)
Protocol Descriptor List:
        L2CAP (0x0100)
                Protocol specific parameter #1: u/int/uuid16 1
                Protocol specific parameter #2: u/int/uuid16 1

Record Handle: 0x00000001
Service Class ID List:
        Browse Group Descriptor (0x1001)

Record Handle: 0x00000002
Service Class ID List:
        LAN Access Using PPP (0x1102)
Protocol Descriptor List:
        L2CAP (0x0100)
        RFCOMM (0x0003)
                Protocol specific parameter #1: u/int8/bool 1
Bluetooth Profile Descriptor List:
        LAN Access Using PPP (0x1102) ver. 1.0
```

Note that each service has a list of attributes, such as the RFCOMM channel. Depending on the service, the user might need to make note of some of the attributes. Some Bluetooth implementations do not support service browsing and may return an empty list. In this case, it is possible to search for the specific service. The example below shows how to search for the OBEX Object Push (OPUSH) service:

```
% sdpcontrol -a 00:01:03:fc:6e:ec search OPUSH
```

Offering services on FreeBSD to Bluetooth clients is done with the sdpd(8) server. The following line can be added to `/etc/rc.conf`:

```
sdpd_enable="YES"
```

Then the sdpd(8) daemon can be started with:

```
# service sdpd start
```

The local server application that wants to provide Bluetooth service to the remote clients will register service with the local SDP daemon. An example of such an application is rfcomm_pppd(8). Once started, it will register the Bluetooth LAN service with the local SDP daemon.

The list of services registered with the local SDP server can be obtained by issuing a SDP browse query via the local control channel:

```
# sdpcontrol -l browse
```

32.4.8 Dial-Up Networking and Network Access with PPP Profiles

The Dial-Up Networking (DUN) profile is mostly used with modems and cellular phones. The scenarios covered by this profile are the following:

- Use of a cellular phone or modem by a computer as a wireless modem for connecting to a dial-up Internet access server, or for using other dial-up services.
- Use of a cellular phone or modem by a computer to receive data calls.

Network access with a PPP profile can be used in the following situations:

- LAN access for a single Bluetooth device.
- LAN access for multiple Bluetooth devices.
- PC to PC connection using PPP networking over serial cable emulation.

In FreeBSD, these profiles are implemented with ppp(8) and the rfcomm_pppd(8) wrapper which converts a RFCOMM Bluetooth connection into something PPP can use. Before a profile can be used, a new PPP label must be created in `/etc/ppp/ppp.conf`. Consult rfcomm_pppd(8) for examples.

In the following example, rfcomm_pppd(8) is used to open a RFCOMM connection to a remote device with a BD_ADDR of `00:80:37:29:19:a4` on a DUN RFCOMM channel. The actual RFCOMM channel number will be obtained from the remote device via SDP. It is possible to specify the RFCOMM channel by hand, and in this case rfcomm_pppd(8) will not perform the SDP query. Use sdpcontrol(8) to find out the RFCOMM channel on the remote device.

```
# rfcomm_pppd -a 00:80:37:29:19:a4 -c -C dun -l rfcomm-dialup
```

In order to provide network access with the PPP LAN service, sdpd(8) must be running and a new entry for LAN clients must be created in `/etc/ppp/ppp.conf`. Consult rfcomm_pppd(8) for examples. Finally, start the RFCOMM PPP server on a valid RFCOMM channel number. The RFCOMM PPP server will automatically register

the Bluetooth LAN service with the local SDP daemon. The example below shows how to start the RFCOMM PPP server.

```
# rfcomm_pppd -s -C 7 -l rfcomm-server
```

32.4.9 OBEX Object Push (OPUSH) Profile

OBEX is a widely used protocol for simple file transfers between mobile devices. Its main use is in infrared communication, where it is used for generic file transfers between notebooks or PDAs, and for sending business cards or calendar entries between cellular phones and other devices with PIM applications.

The OBEX server and client are implemented as a third-party package, **obexapp**, which is available as `comms/obexapp` package or port.

The OBEX client is used to push and/or pull objects from the OBEX server. An object can, for example, be a business card or an appointment. The OBEX client can obtain the RFCOMM channel number from the remote device via SDP. This can be done by specifying the service name instead of the RFCOMM channel number. Supported service names are: IrMC, FTRN, and OPUSH. It is also possible to specify the RFCOMM channel as a number. Below is an example of an OBEX session where the device information object is pulled from the cellular phone, and a new object, the business card, is pushed into the phone's directory.

```
% obexapp -a 00:80:37:29:19:a4 -C IrMC
obex> get telecom/devinfo.txt devinfo-t39.txt
Success, response: OK, Success (0x20)
obex> put new.vcf
Success, response: OK, Success (0x20)
obex> di
Success, response: OK, Success (0x20)
```

In order to provide the OPUSH service, sdpd(8) must be running and a root folder, where all incoming objects will be stored, must be created. The default path to the root folder is `/var/spool/obex`. Finally, start the OBEX server on a valid RFCOMM channel number. The OBEX server will automatically register the OPUSH service with the local SDP daemon. The example below shows how to start the OBEX server.

```
# obexapp -s -C 10
```

32.4.10 Serial Port Profile

The Serial Port Profile (SPP) allows Bluetooth devices to perform serial cable emulation. This profile allows legacy applications to use Bluetooth as a cable replacement, through a virtual serial port abstraction.

In FreeBSD, rfcomm_sppd(1) implements SPP and a pseudo tty is used as a virtual serial port abstraction. The example below shows how to connect to a remote device serial port service. A RFCOMM channel does not have to be specified as rfcomm_sppd(1) can obtain it from the remote device via SDP. To override this, specify a RFCOMM channel on the command line.

```
# rfcomm_sppd -a 00:07:E0:00:0B:CA -t /dev/ttyp6
rfcomm_sppd[94692]: Starting on /dev/ttyp6...
```

Once connected, the pseudo tty can be used as serial port:

```
# cu -l ttyp6
```

32.4.11 Troubleshooting

32.4.11.1 A Remote Device Cannot Connect

Some older Bluetooth devices do not support role switching. By default, when FreeBSD is accepting a new connection, it tries to perform a role switch and become master. Devices, which do not support this will not be able to connect. Since role switching is performed when a new connection is being established, it is not possible to ask the remote device if it supports role switching. There is a HCI option to disable role switching on the local side:

```
# hccontrol -n ubt0hci write_node_role_switch 0
```

32.4.11.2 Displaying Bluetooth Packets

Use the third-party package **hcidump**, which is available as a `comms/hcidump` package or port. This utility is similar to tcpdump(1) and can be used to display the contents of Bluetooth packets on the terminal and to dump the Bluetooth packets to a file.

32.5 Bridging

Written by Andrew Thompson.

32.5.1 Introduction

It is sometimes useful to divide one physical network, such as an Ethernet segment, into two separate network segments without having to create IP subnets and use a router to connect the segments together. A device that connects two networks together in this fashion is called a “bridge”. A FreeBSD system with two network interface cards can act as a bridge.

The bridge works by learning the MAC layer (Ethernet) addresses of the devices on each of its network interfaces. It forwards traffic between two networks only when the source and destination are on different networks.

In many respects, a bridge is like an Ethernet switch with very few ports.

32.5.2 Situations Where Bridging Is Appropriate

There are many common situations in which a bridge is used today.

32.5.2.1 Connecting Networks

The basic operation of a bridge is to join two or more network segments together. There are many reasons to use a host based bridge over plain networking equipment such as cabling constraints, firewalling, or connecting pseudo

networks such as a virtual machine interface. A bridge can also connect a wireless interface running in hostap mode to a wired network and act as an access point.

32.5.2.2 Filtering/Traffic Shaping Firewall

A common situation is where firewall functionality is needed without routing or Network Address Translation (NAT).

An example is a small company that is connected via DSL or ISDN to an ISP. There are thirteen globally-accessible IP addresses from the ISP and ten computers on the network. In this situation, using a router-based firewall is difficult because of subnetting issues.

A bridge-based firewall can be configured and dropped into the path just downstream of the DSL or ISDN router without any IP numbering issues.

32.5.2.3 Network Tap

A bridge can join two network segments and be used to inspect all Ethernet frames that pass between them using bpf(4) and tcpdump(1) on the bridge interface or by sending a copy of all frames out an additional interface known as a span port.

32.5.2.4 Layer 2 VPN

Two Ethernet networks can be joined across an IP link by bridging the networks to an EtherIP tunnel or a tap(4) based solution such as **OpenVPN**.

32.5.2.5 Layer 2 Redundancy

A network can be connected together with multiple links and use the Spanning Tree Protocol STP to block redundant paths. For an Ethernet network to function properly, only one active path can exist between two devices. STP will detect loops and put the redundant links into a blocked state. Should one of the active links fail, STP will calculate a different tree and enable one of the blocked paths to restore connectivity to all points in the network.

32.5.3 Kernel Configuration

This section covers the if_bridge(4) implementation. A netgraph bridging driver is also available, and is described in ng_bridge(4).

In FreeBSD, if_bridge(4) is a kernel module which is automatically loaded by ifconfig(8) when creating a bridge interface. It is also possible to compile the bridge in to the kernel by adding `device if_bridge` to a custom kernel configuration file.

Packet filtering can be used with any firewall package that hooks in via the pfil(9) framework. The firewall can be loaded as a module or compiled into the kernel.

The bridge can be used as a traffic shaper with altq(4) or dummynet(4).

32.5.4 Enabling the Bridge

The bridge is created using interface cloning. To create a bridge use ifconfig(8):

```
# ifconfig bridge create
bridge0
# ifconfig bridge0
bridge0: flags=8802<BROADCAST,SIMPLEX,MULTICAST> metric 0 mtu 1500
        ether 96:3d:4b:f1:79:7a
        id 00:00:00:00:00:00 priority 32768 hellotime 2 fwddelay 15
        maxage 20 holdcnt 6 proto rstp maxaddr 100 timeout 1200
        root id 00:00:00:00:00:00 priority 0 ifcost 0 port 0
```

When a bridge interface is created, it is automatically assigned a randomly generated Ethernet address. The `maxaddr` and `timeout` parameters control how many MAC addresses the bridge will keep in its forwarding table and how many seconds before each entry is removed after it is last seen. The other parameters control how STP operates.

Next, add the member network interfaces to the bridge. For the bridge to forward packets, all member interfaces and the bridge need to be up:

```
# ifconfig bridge0 addm fxp0 addm fxp1 up
# ifconfig fxp0 up
# ifconfig fxp1 up
```

The bridge is now forwarding Ethernet frames between `fxp0` and `fxp1`. Add the following lines to `/etc/rc.conf` so the bridge is created at startup:

```
cloned_interfaces="bridge0"
ifconfig_bridge0="addm fxp0 addm fxp1 up"
ifconfig_fxp0="up"
ifconfig_fxp1="up"
```

If the bridge host needs an IP address, the correct place to set this is on the bridge interface itself rather than one of the member interfaces. This can be set statically or via DHCP:

```
# ifconfig bridge0 inet 192.168.0.1/24
```

It is also possible to assign an IPv6 address to a bridge interface.

32.5.5 Firewalling

When packet filtering is enabled, bridged packets will pass through the filter inbound on the originating interface on the bridge interface, and outbound on the appropriate interfaces. Either stage can be disabled. When direction of the packet flow is important, it is best to firewall on the member interfaces rather than the bridge itself.

The bridge has several configurable settings for passing non-IP and IP packets, and layer2 firewalling with ipfw(8). See if_bridge(4) for more information.

32.5.6 Spanning Tree

The bridge driver implements the Rapid Spanning Tree Protocol (RSTP or 802.1w) with backwards compatibility with legacy STP. STP is used to detect and remove loops in a network topology. RSTP provides faster convergence

than legacy STP, the protocol will exchange information with neighboring switches to quickly transition to forwarding without creating loops. FreeBSD supports RSTP and STP as operating modes, with RSTP being the default mode.

STP can be enabled on member interfaces using ifconfig(8). For a bridge with `fxp0` and `fxp1` as the current interfaces, enable STP with:

```
# ifconfig bridge0 stp fxp0 stp fxp1
bridge0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
        ether d6:cf:d5:a0:94:6d
        id 00:01:02:4b:d4:50 priority 32768 hellotime 2 fwddelay 15
        maxage 20 holdcnt 6 proto rstp maxaddr 100 timeout 1200
        root id 00:01:02:4b:d4:50 priority 32768 ifcost 0 port 0
        member: fxp0 flags=1c7<LEARNING,DISCOVER,STP,AUTOEDGE,PTP,AUTOPTP>
                port 3 priority 128 path cost 200000 proto rstp
                role designated state forwarding
        member: fxp1 flags=1c7<LEARNING,DISCOVER,STP,AUTOEDGE,PTP,AUTOPTP>
                port 4 priority 128 path cost 200000 proto rstp
                role designated state forwarding
```

This bridge has a spanning tree ID of `00:01:02:4b:d4:50` and a priority of `32768`. As the `root id` is the same, it indicates that this is the root bridge for the tree.

Another bridge on the network also has STP enabled:

```
bridge0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
        ether 96:3d:4b:f1:79:7a
        id 00:13:d4:9a:06:7a priority 32768 hellotime 2 fwddelay 15
        maxage 20 holdcnt 6 proto rstp maxaddr 100 timeout 1200
        root id 00:01:02:4b:d4:50 priority 32768 ifcost 400000 port 4
        member: fxp0 flags=1c7<LEARNING,DISCOVER,STP,AUTOEDGE,PTP,AUTOPTP>
                port 4 priority 128 path cost 200000 proto rstp
                role root state forwarding
        member: fxp1 flags=1c7<LEARNING,DISCOVER,STP,AUTOEDGE,PTP,AUTOPTP>
                port 5 priority 128 path cost 200000 proto rstp
                role designated state forwarding
```

The line `root id 00:01:02:4b:d4:50 priority 32768 ifcost 400000 port 4` shows that the root bridge is `00:01:02:4b:d4:50` and has a path cost of `400000` from this bridge. The path to the root bridge is via `port 4` which is `fxp0`.

32.5.7 Advanced Bridging

32.5.7.1 Reconstruct Traffic Flows

The bridge supports monitor mode, where the packets are discarded after bpf(4) processing and are not processed or forwarded further. This can be used to multiplex the input of two or more interfaces into a single bpf(4) stream. This is useful for reconstructing the traffic for network taps that transmit the RX/TX signals out through two separate interfaces.

To read the input from four network interfaces as one stream:

```
# ifconfig bridge0 addm fxp0 addm fxp1 addm fxp2 addm fxp3 monitor up
# tcpdump -i bridge0
```

32.5.7.2 Span Ports

A copy of every Ethernet frame received by the bridge will be transmitted out a designated span port. The number of span ports configured on a bridge is unlimited, but if an interface is designated as a span port, it cannot also be used as a regular bridge port. This is most useful for snooping a bridged network passively on another host connected to one of the span ports of the bridge.

To send a copy of all frames out the interface named `fxp4`:

```
# ifconfig bridge0 span fxp4
```

32.5.7.3 Private Interfaces

A private interface does not forward any traffic to any other port that is also a private interface. The traffic is blocked unconditionally so no Ethernet frames will be forwarded, including ARP. If traffic needs to be selectively blocked, a firewall should be used instead.

32.5.7.4 Sticky Interfaces

If a bridge member interface is marked as sticky, dynamically learned address entries are treated at static once entered into the forwarding cache. Sticky entries are never aged out of the cache or replaced, even if the address is seen on a different interface. This gives the benefit of static address entries without the need to pre-populate the forwarding table. Clients learned on a particular segment of the bridge can not roam to another segment.

Another example of using sticky addresses is to combine the bridge with VLANs to create a router where customer networks are isolated without wasting IP address space. Consider that `CustomerA` is on `vlan100` and `CustomerB` is on `vlan101`. The bridge has the address `192.168.0.1` and is also an Internet router.

```
# ifconfig bridge0 addm vlan100 sticky vlan100 addm vlan101 sticky vlan101
# ifconfig bridge0 inet 192.168.0.1/24
```

In this example, both clients see `192.168.0.1` as their default gateway. Since the bridge cache is sticky, one host can not spoof the MAC address of the other customer in order to intercept their traffic.

Any communication between the VLANs can be blocked using a firewall or, as seen in this example, private interfaces:

```
# ifconfig bridge0 private vlan100 private vlan101
```

The customers are completely isolated from each other and the full `/24` address range can be allocated without subnetting.

32.5.7.5 Address Limits

The number of unique source MAC addresses behind an interface can be limited. Once the limit is reached, packets with unknown source addresses are dropped until an existing host cache entry expires or is removed.

The following example sets the maximum number of Ethernet devices for `CustomerA` on `vlan100` to 10:

```
# ifconfig bridge0 ifmaxaddr vlan100 10
```

32.5.7.6 SNMP Monitoring

The bridge interface and STP parameters can be monitored via bsnmpd(1) which is included in the FreeBSD base system. The exported bridge MIBs conform to the IETF standards so any SNMP client or monitoring package can be used to retrieve the data.

On the bridge, uncomment the `begemotSnmpdModulePath."bridge" = "/usr/lib/snmp_bridge.so"` line from `/etc/snmp.config` and start bsnmpd(1). Other configuration, such as community names and access lists, may need to be modified. See bsnmpd(1) and snmp_bridge(3) for more information.

The following examples use the **Net-SNMP** software (`net-mgmt/net-snmp`) to query a bridge from a client system. The `net-mgmt/bsnmptools` port can also be used. From the SNMP client which is running **Net-SNMP**, add the following lines to `$HOME/.snmp/snmp.conf` in order to import the bridge MIB definitions:

```
mibdirs +/usr/share/snmp/mibs
mibs +BRIDGE-MIB:RSTP-MIB:BEGEMOT-MIB:BEGEMOT-BRIDGE-MIB
```

To monitor a single bridge using the IETF BRIDGE-MIB (RFC4188):

```
% snmpwalk -v 2c -c public bridge1.example.com mib-2.dot1dBridge
BRIDGE-MIB::dot1dBaseBridgeAddress.0 = STRING: 66:fb:9b:6e:5c:44
BRIDGE-MIB::dot1dBaseNumPorts.0 = INTEGER: 1 ports
BRIDGE-MIB::dot1dStpTimeSinceTopologyChange.0 = Timeticks: (189959) 0:31:39.59 centi-seconds
BRIDGE-MIB::dot1dStpTopChanges.0 = Counter32: 2
BRIDGE-MIB::dot1dStpDesignatedRoot.0 = Hex-STRING: 80 00 00 01 02 4B D4 50
...
BRIDGE-MIB::dot1dStpPortState.3 = INTEGER: forwarding(5)
BRIDGE-MIB::dot1dStpPortEnable.3 = INTEGER: enabled(1)
BRIDGE-MIB::dot1dStpPortPathCost.3 = INTEGER: 200000
BRIDGE-MIB::dot1dStpPortDesignatedRoot.3 = Hex-STRING: 80 00 00 01 02 4B D4 50
BRIDGE-MIB::dot1dStpPortDesignatedCost.3 = INTEGER: 0
BRIDGE-MIB::dot1dStpPortDesignatedBridge.3 = Hex-STRING: 80 00 00 01 02 4B D4 50
BRIDGE-MIB::dot1dStpPortDesignatedPort.3 = Hex-STRING: 03 80
BRIDGE-MIB::dot1dStpPortForwardTransitions.3 = Counter32: 1
RSTP-MIB::dot1dStpVersion.0 = INTEGER: rstp(2)
```

The `dot1dStpTopChanges.0` value is two, indicating that the STP bridge topology has changed twice. A topology change means that one or more links in the network have changed or failed and a new tree has been calculated. The `dot1dStpTimeSinceTopologyChange.0` value will show when this happened.

To monitor multiple bridge interfaces, the private BEGEMOT-BRIDGE-MIB can be used:

```
% snmpwalk -v 2c -c public bridge1.example.com
enterprises.fokus.begemot.begemotBridge
BEGEMOT-BRIDGE-MIB::begemotBridgeBaseName."bridge0" = STRING: bridge0
BEGEMOT-BRIDGE-MIB::begemotBridgeBaseName."bridge2" = STRING: bridge2
BEGEMOT-BRIDGE-MIB::begemotBridgeBaseAddress."bridge0" = STRING: e:ce:3b:5a:9e:13
BEGEMOT-BRIDGE-MIB::begemotBridgeBaseAddress."bridge2" = STRING: 12:5e:4d:74:d:fc
BEGEMOT-BRIDGE-MIB::begemotBridgeBaseNumPorts."bridge0" = INTEGER: 1
```

```
BEGEMOT-BRIDGE-MIB::begemotBridgeBaseNumPorts."bridge2" = INTEGER: 1
...
BEGEMOT-BRIDGE-MIB::begemotBridgeStpTimeSinceTopologyChange."bridge0" = Timeticks: (116927) 0:19:1
BEGEMOT-BRIDGE-MIB::begemotBridgeStpTimeSinceTopologyChange."bridge2" = Timeticks: (82773) 0:13:47
BEGEMOT-BRIDGE-MIB::begemotBridgeStpTopChanges."bridge0" = Counter32: 1
BEGEMOT-BRIDGE-MIB::begemotBridgeStpTopChanges."bridge2" = Counter32: 1
BEGEMOT-BRIDGE-MIB::begemotBridgeStpDesignatedRoot."bridge0" = Hex-STRING: 80 00 00 40 95 30 5E 31
BEGEMOT-BRIDGE-MIB::begemotBridgeStpDesignatedRoot."bridge2" = Hex-STRING: 80 00 00 50 8B B8 C6 A9
```

To change the bridge interface being monitored via the `mib-2.dot1dBridge` subtree:

```
% snmpset -v 2c -c private bridge1.example.com
BEGEMOT-BRIDGE-MIB::begemotBridgeDefaultBridgeIf.0 s bridge2
```

32.6 Link Aggregation and Failover

Written by Andrew Thompson.

32.6.1 Introduction

The lagg(4) interface allows aggregation of multiple network interfaces as one virtual interface for the purpose of providing fault-tolerance and high-speed links.

32.6.2 Operating Modes

The following operating modes are supported by lagg(4):

Failover

Sends and receives traffic only through the master port. If the master port becomes unavailable, the next active port is used. The first interface added is the master port and any interfaces added after that are used as failover devices. If failover to a non-master port occurs, the original port will become master when it becomes available again.

Cisco® Fast EtherChannel®

Cisco Fast EtherChannel (FEC) is a static setup and does not negotiate aggregation with the peer or exchange frames to monitor the link. If the switch supports LACP, that should be used instead.

FEC balances outgoing traffic across the active ports based on hashed protocol header information and accepts incoming traffic from any active port. The hash includes the Ethernet source and destination address and, if available, the VLAN tag, and the IPv4 or IPv6 source and destination address.

LACP

The IEEE 802.3ad Link Aggregation Control Protocol (LACP) and the Marker Protocol. LACP will negotiate a set of aggregable links with the peer in to one or more Link Aggregated Groups (LAGs). Each LAG is composed of ports of the same speed, set to full-duplex operation. The traffic will be balanced across the ports

in the LAG with the greatest total speed. In most cases, there will only be one LAG which contains all ports. In the event of changes in physical connectivity, LACP will quickly converge to a new configuration.

LACP balances outgoing traffic across the active ports based on hashed protocol header information and accepts incoming traffic from any active port. The hash includes the Ethernet source and destination address and, if available, the VLAN tag, and the IPv4 or IPv6 source and destination address.

Loadbalance

This is an alias of *FEC* mode.

Round-robin

Distributes outgoing traffic using a round-robin scheduler through all active ports and accepts incoming traffic from any active port. This mode violates Ethernet frame ordering and should be used with caution.

32.6.3 Examples

Example 32-1. LACP Aggregation with a Cisco® Switch

This example connects two interfaces on a FreeBSD machine to the switch as a single load balanced and fault tolerant link. More interfaces can be added to increase throughput and fault tolerance. Frame ordering is mandatory on Ethernet links and any traffic between two stations always flows over the same physical link, limiting the maximum speed to that of one interface. The transmit algorithm attempts to use as much information as it can to distinguish different traffic flows and balance across the available interfaces.

On the Cisco switch, add the *FastEthernet0/1* and *FastEthernet0/2* interfaces to channel group *1*:

```
interface FastEthernet0/1
 channel-group 1 mode active
 channel-protocol lacp
!
interface FastEthernet0/2
 channel-group 1 mode active
 channel-protocol lacp
```

Create the lagg(4) interface using *fxp0* and *fxp1*, and bring the interfaces up with the IP address of *10.0.0.3/24*:

```
# ifconfig fxp0 up
# ifconfig fxp1 up
# ifconfig lagg0 create
# ifconfig lagg0 up laggproto lacp laggport fxp0 laggport fxp1 10.0.0.3/24
```

View the interface status by running:

```
# ifconfig lagg0
```

Ports marked as *ACTIVE* are part of the active aggregation group that has been negotiated with the remote switch. Traffic will be transmitted and received through active ports. Use the verbose output of ifconfig(8) to view the LAG identifiers.

```
lagg0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
        options=8<VLAN_MTU>
        ether 00:05:5d:71:8d:b8
        media: Ethernet autoselect
        status: active
```

```
        laggproto lacp
        laggport: fxp1 flags=1c<ACTIVE,COLLECTING,DISTRIBUTING>
        laggport: fxp0 flags=1c<ACTIVE,COLLECTING,DISTRIBUTING>
```

To see the port status on the Cisco switch, use **`show lacp neighbor`**:

```
switch# show lacp neighbor
Flags:  S - Device is requesting Slow LACPDUs
        F - Device is requesting Fast LACPDUs
        A - Device is in Active mode       P - Device is in Passive mode

Channel group 1 neighbors

Partner's information:

                  LACP port                        Oper    Port     Port
Port      Flags   Priority  Dev ID         Age     Key     Number   State
Fa0/1     SA      32768     0005.5d71.8db8  29s    0x146   0x3      0x3D
Fa0/2     SA      32768     0005.5d71.8db8  29s    0x146   0x4      0x3D
```

For more detail, type **`show lacp neighbor detail`**.

To retain this configuration across reboots, the following entries can be added to `/etc/rc.conf`:

```
ifconfig_fxp0="up"
ifconfig_fxp1="up"
cloned_interfaces="lagg0"
ifconfig_lagg0="laggproto lacp laggport fxp0 laggport fxp1 10.0.0.3/24"
```

Example 32-2. Failover Mode

Failover mode can be used to switch over to a secondary interface if the link is lost on the master interface. To configure failover mode, first bring the underlying physical interfaces up. Then, create the lagg(4) interface, using *`fxp0`* as the master interface and *`fxp1`* as the secondary interface, and assign an IP address of *`10.0.0.15/24`*:

```
# ifconfig fxp0 up
# ifconfig fxp1 up
# ifconfig lagg0 create
# ifconfig lagg0 up laggproto failover laggport fxp0 laggport fxp1 10.0.0.15/24
```

The interface should now look something like this:

```
# ifconfig lagg0
lagg0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
        options=8<VLAN_MTU>
        ether 00:05:5d:71:8d:b8
        inet 10.0.0.15 netmask 0xffffff00 broadcast 10.0.0.255
        media: Ethernet autoselect
        status: active
        laggproto failover
        laggport: fxp1 flags=0<>
        laggport: fxp0 flags=5<MASTER,ACTIVE>
```

Traffic will be transmitted and received on *`fxp0`*. If the link is lost on *`fxp0`*, *`fxp1`* will become the active link. If the link is restored on the master interface, it will once again become the active link.

To retain this configuration across reboots, the following entries can be added to `/etc/rc.conf`:

```
ifconfig_fxp0="up"
ifconfig_fxp1="up"
cloned_interfaces="lagg0"
ifconfig_lagg0="laggproto failover laggport fxp0 laggport fxp1 10.0.0.15/24"
```

Example 32-3. Failover Mode Between Wired and Wireless Interfaces

For laptop users, it is usually desirable to configure the wireless device as a secondary interface, which is used when the wired connection is not available. With lagg(4), it is possible to use one IP address, prefer the wired connection for both performance and security reasons, while maintaining the ability to transfer data over the wireless connection.

In this setup, override the underlying wireless interface's MAC address to match that of the lagg(4), which is inherited from the wired interface.

In this example, the wired interface, *bge0*, is the master, and the wireless interface, *wlan0*, is the failover interface. The *wlan0* device was created from *iwn0*, which will be configured with the wired connection's MAC address. The first step is to determine the MAC address of the wired interface:

```
# ifconfig bge0
bge0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
       options=19b<RXCSUM,TXCSUM,VLAN_MTU,VLAN_HWTAGGING,VLAN_HWCSUM,TSO4>
       ether 00:21:70:da:ae:37
       inet6 fe80::221:70ff:feda:ae37%bge0 prefixlen 64 scopeid 0x2
       nd6 options=29<PERFORMNUD,IFDISABLED,AUTO_LINKLOCAL>
       media: Ethernet autoselect (1000baseT <full-duplex>)
       status: active
```

Replace *bge0* to match the system's interface name. The `ether` line will contain the MAC address of the wired interface. Now, change the MAC address of the underlying wireless interface:

```
# ifconfig iwn0 ether 00:21:70:da:ae:37
```

Bring the wireless interface up, but do not set an IP address:

```
# ifconfig wlan0 create wlandev iwn0 ssid my_router up
```

Bring the *bge0* interface up. Create the lagg(4) interface with *bge0* as master, and failover to *wlan0*:

```
# ifconfig bge0 up
# ifconfig lagg0 create
# ifconfig lagg0 up laggproto failover laggport bge0 laggport wlan0
```

The interface will now look something like this:

```
# ifconfig lagg0
lagg0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
        options=8<VLAN_MTU>
        ether 00:21:70:da:ae:37
        media: Ethernet autoselect
        status: active
        laggproto failover
        laggport: wlan0 flags=0<>
        laggport: bge0 flags=5<MASTER,ACTIVE>
```

Then, start the DHCP client to obtain an IP address:

```
# dhclient lagg0
```

To retain this configuration across reboots, the following entries can be added to `/etc/rc.conf`:

```
ifconfig_bge0="up"
ifconfig_iwn0="ether 00:21:70:da:ae:37"
wlans_iwn0="wlan0"
ifconfig_wlan0="WPA"
cloned_interfaces="lagg0"
ifconfig_lagg0="laggproto failover laggport bge0 laggport wlan0 DHCP"
```

32.7 Diskless Operation

Updated by Jean-François Dockès. Reorganized and enhanced by Alex Dupre.

A FreeBSD machine can boot over the network and operate without a local disk, using file systems mounted from an NFS server. No system modification is necessary, beyond standard configuration files. Such a system is relatively easy to set up because all the necessary elements are readily available:

The Intel Preboot eXecution Environment (PXE) can be used to load the kernel over the network. It provides a form of smart boot ROM built into some networking cards or motherboards. See pxeboot(8) for more details.

A sample script (`/usr/share/examples/diskless/clone_root`) eases the creation and maintenance of the workstation's root file system on the server. The script will probably require a little customization.

Standard system startup files exist in `/etc` to detect and support a diskless system startup.

Swapping, if needed, can be done either to an NFS file or to a local disk.

There are many ways to set up diskless workstations. Many elements are involved, and most can be customized to suit local taste. The following will describe variations on the setup of a complete system, emphasizing simplicity and compatibility with the standard FreeBSD startup scripts. The system described has the following characteristics:

- The diskless workstations use a shared, read-only `/` and `/usr`.

  The root file system is a copy of a standard FreeBSD root, with some configuration files overridden by ones specific to diskless operation or, possibly, to the workstation they belong to.

  The parts of the root which have to be writable are overlaid with md(4) file systems. Any changes will be lost when the system reboots.

  Caution: As described, this system is insecure. It should live in a protected area of a network and be untrusted by other hosts.

32.7.1 Background Information

Setting up diskless workstations is both relatively straightforward and prone to errors. These are sometimes difficult to diagnose for a number of reasons. For example:

- Compile time options may determine different behaviors at runtime.
- Error messages are often cryptic or totally absent.

In this context, having some knowledge of the background mechanisms involved is useful to solve the problems that may arise.

Several operations need to be performed for a successful bootstrap:

- The machine needs to obtain initial parameters such as its IP address, executable filename, server name, and root path. This is done using the DHCP or BOOTP protocols. DHCP is a compatible extension of BOOTP, and uses the same port numbers and basic packet format. It is possible to configure a system to use only BOOTP and bootpd(8) is included in the base FreeBSD system.
- DHCP has a number of advantages over BOOTP such as nicer configuration files and support for PXE. This section describes mainly a DHCP configuration, with equivalent examples using bootpd(8) when possible. The sample configuration uses **ISC DHCP** which is available in the Ports Collection.
- The machine needs to transfer one or several programs to local memory. Either TFTP or NFS are used. The choice between TFTP and NFS is a compile time option in several places. A common source of error is to specify filenames for the wrong protocol. TFTP typically transfers all files from a single directory on the server and expects filenames relative to this directory. NFS needs absolute file paths.
- The possible intermediate bootstrap programs and the kernel need to be initialized and executed. PXE loads pxeboot(8), which is a modified version of the FreeBSD third stage loader, loader(8). The third stage loader will obtain most parameters necessary to system startup and leave them in the kernel environment before transferring control. It is possible to use a `GENERIC` kernel in this case.
- Finally, the machine needs to access its file systems using NFS.

Refer to diskless(8) for more information.

32.7.2 Setup Instructions

32.7.2.1 Configuration Using ISC DHCP

The **ISC DHCP** server can answer both BOOTP and DHCP requests.

ISC DHCP is not part of the base system. Install the `net/isc-dhcp42-server` port or package.

Once **ISC DHCP** is installed, edit its configuration file, `/usr/local/etc/dhcpd.conf`. Here follows a commented example for PXE host `corbieres`:

```
default-lease-time 600;
max-lease-time 7200;
authoritative;

option domain-name "example.com";
option domain-name-servers 192.168.4.1;
option routers 192.168.4.1;

subnet 192.168.4.0 netmask 255.255.255.0 {
  use-host-decl-names on; ❶
  option subnet-mask 255.255.255.0;
  option broadcast-address 192.168.4.255;

  host corbieres {
```

```
    hardware ethernet 00:02:b3:27:62:df;
    fixed-address corbieres.example.com;
    next-server 192.168.4.4; ❷
    filename "pxeboot"; ❸
    option root-path "192.168.4.4:/data/misc/diskless"; ❹
  }
}
```

❶ This option tells **dhcpd** to send the value in the `host` declarations as the hostname for the diskless host. An alternate way would be to add an `option host-name` *`corbieres`* inside the `host` declarations.

❷ The `next-server` directive designates the TFTP or NFS server to use for loading loader(8) or the kernel file. The default is to use the same host as the DHCP server.

❸ The `filename` directive defines the file that PXE will load for the next execution step. It must be specified according to the transfer method used. PXE uses TFTP, which is why a relative filename is used here. Also, PXE loads `pxeboot`, not the kernel. There are other interesting possibilities, like loading `pxeboot` from a FreeBSD CD-ROM `/boot` directory. Since pxeboot(8) can load a `GENERIC` kernel, it is possible to use PXE to boot from a remote CD-ROM.

❹ The `root-path` option defines the path to the root file system, in usual NFS notation. When using PXE, it is possible to leave off the host's IP address as long as the BOOTP kernel option is not enabled. The NFS server will then be the same as the TFTP one.

32.7.2.2 Booting with PXE

By default, pxeboot(8) loads the kernel via NFS. It can be compiled to use TFTP instead by specifying the `LOADER_TFTP_SUPPORT` option in `/etc/make.conf`. See the comments in `/usr/share/examples/etc/make.conf` for instructions.

There are two other `make.conf` options which may be useful for setting up a serial console diskless machine: `BOOT_PXELDR_PROBE_KEYBOARD`, and `BOOT_PXELDR_ALWAYS_SERIAL`.

To use PXE when the machine starts, select the `Boot from network` option in the BIOS setup or type a function key during system initialization.

32.7.2.3 Configuring the TFTP and NFS Servers

If PXE is configured to use TFTP, enable tftpd(8) on the file server:

1. Create a directory from which tftpd(8) will serve the files, such as `/tftpboot`.
2. Add this line to `/etc/inetd.conf`:

```
tftp    dgram   udp     wait    root    /usr/libexec/tftpd      tftpd -l -s /tftpboot
```

> **Note:** Some PXE versions require the TCP version of TFTP. In this case, add a second line, replacing `dgram udp` with `stream tcp`.

3. Tell inetd(8) to reread its configuration file. Add `inetd_enable="YES"` to `/etc/rc.conf` in order for this command to execute correctly:

   ```
   # service inetd restart
   ```

Place `tftpboot` anywhere on the server. Make sure that the location is set in both `/etc/inetd.conf` and `/usr/local/etc/dhcpd.conf`.

Enable NFS and export the appropriate file system on the NFS server.

1. Add this line to `/etc/rc.conf`:

   ```
   nfs_server_enable="YES"
   ```

2. Export the file system where the diskless root directory is located by adding the following to `/etc/exports`. Adjust the mount point and replace *`corbieres`* with the names of the diskless workstations:

   ```
   /data/misc -alldirs -ro margaux corbieres
   ```

3. Tell mountd(8) to reread its configuration file. If NFS is enabled in `/etc/rc.conf`, it is recommended to reboot instead.

   ```
   # service mountd restart
   ```

32.7.2.4 Building a Diskless Kernel

When using PXE, building a custom kernel with the following options is not strictly necessary. These options cause more DHCP requests to be issued during kernel startup, with a small risk of inconsistency between the new values and those retrieved by pxeboot(8) in some special cases. The advantage is that the host name will be set. Otherwise, set the host name in a client-specific `/etc/rc.conf`.

```
options     BOOTP          # Use BOOTP to obtain IP address/hostname
options     BOOTP_NFSROOT  # NFS mount root file system using BOOTP info
```

The custom kernel can also include `BOOTP_NFSV3`, `BOOT_COMPAT` and `BOOTP_WIRED_TO`. Refer to `NOTES` for descriptions of these options.

These option names are historical and slightly misleading as they actually enable indifferent use of DHCP and BOOTP inside the kernel.

Build the custom kernel, using the instructions in Chapter 9, and copy it to the place specified in `/usr/local/etc/dhcpd.conf`.

32.7.2.5 Preparing the Root File System

Create a root file system for the diskless workstations in the location listed as `root-path` in `/usr/local/etc/dhcpd.conf`.

32.7.2.5.1 Using `make world` to Populate Root

This method is quick and will install a complete virgin system, not just the root file system, into `DESTDIR`. Execute the following script:

```
#!/bin/sh
```

```
export DESTDIR=/data/misc/diskless
mkdir -p ${DESTDIR}
cd /usr/src; make buildworld && make buildkernel
make installworld && make installkernel
cd /usr/src/etc; make distribution
```

Once done, customize `/etc/rc.conf` and `/etc/fstab` placed into `DESTDIR` according to the system's requirements.

32.7.2.6 Configuring Swap

If needed, a swap file located on the server can be accessed via NFS.

32.7.2.6.1 NFS Swap

The kernel does not support enabling NFS swap at boot time. Swap must be enabled by the startup scripts, by mounting a writable file system and creating and enabling a swap file. To create a swap file:

```
# dd if=/dev/zero of=/path/to/swapfile bs=1k count=1 oseek=100000
```

To enable the swap file, add the following line to `/etc/rc.conf`:

```
swapfile=/path/to/swapfile
```

32.7.2.7 Miscellaneous Issues

32.7.2.7.1 Running with a Read-only `/usr`

If the diskless workstation is configured to run **Xorg**, adjust the **XDM** configuration file as it puts the error log on `/usr` by default.

32.7.2.7.2 Using a Non-FreeBSD Server

When the server for the root file system is not running FreeBSD, create the root file system on a FreeBSD machine, then copy it to its destination, using tar(1) or cpio(1).

In this situation, there are sometimes problems with the special files in `/dev`, due to differing major/minor integer sizes. A solution to this problem is to export a directory from the non-FreeBSD server, mount this directory onto a FreeBSD machine, and use devfs(5) to allocate device nodes transparently for the user.

32.8 PXE Booting with an NFS Root File System

Written by Craig Rodrigues.

The Intel Preboot eXecution Environment (PXE) allows booting the operating system over the network. PXE support is usually provided in the BIOS where it can be enabled in the BIOS settings which enable booting from the network. A fully functioning PXE setup also requires properly configured DHCP and TFTP servers.

When the host computer boots, it receives information over DHCP about where to obtain the initial boot loader via TFTP. After the host computer receives this information, it downloads the boot loader via TFTP and then executes the boot loader. This is documented in section 2.2.1 of the Preboot Execution Environment (PXE) Specification (http://download.intel.com/design/archives/wfm/downloads/pxespec.pdf). In FreeBSD, the boot loader retrieved during the PXE process is `/boot/pxeboot`. After `/boot/pxeboot` executes, the FreeBSD kernel is loaded and the rest of the FreeBSD bootup sequence proceeds. Refer to Chapter 13 for more information about the FreeBSD booting process.

32.8.1 Setting Up the chroot(8) Environment for the NFS Root File System

1. Choose a directory which will have a FreeBSD installation which will be NFS mountable. For example, a directory such as `/b/tftpboot/FreeBSD/install` can be used.

   ```
   # export NFSROOTDIR=/b/tftpboot/FreeBSD/install
   # mkdir -p ${NFSROOTDIR}
   ```

2. Enable the NFS server by following the instructions in Section 30.3.2.

3. Export the directory via NFS by adding the following to `/etc/exports`:

   ```
   /b -ro -alldirs
   ```

4. Restart the NFS server:

   ```
   # service nfsd restart
   ```

5. Enable inetd(8) by following the steps outlined in Section 30.2.2.

6. Add the following line to `/etc/inetd.conf`:

   ```
   tftp dgram udp wait root /usr/libexec/tftpd tftpd -l -s /b/tftpboot
   ```

7. Restart inetd(8):

   ```
   # service inetd restart
   ```

8. Rebuild the FreeBSD kernel and userland (Section 25.7):

   ```
   # cd /usr/src
   # make buildworld
   # make buildkernel
   ```

9. Install FreeBSD into the directory mounted over NFS:

   ```
   # make installworld DESTDIR=${NFSROOTDIR}
   # make installkernel DESTDIR=${NFSROOTDIR}
   # make distribution DESTDIR=${NFSROOTDIR}
   ```

10. Test that the TFTP server works and can download the boot loader which will be obtained via PXE:

    ```
    # tftp localhost
    tftp> get FreeBSD/install/boot/pxeboot
    ```

```
Received 264951 bytes in 0.1 seconds
```

11. Edit `${NFSROOTDIR}/etc/fstab` and create an entry to mount the root file system over NFS:

```
# Device                                          Mountpoint    FSType   Options  Dump Pass
myhost.example.com:/b/tftpboot/FreeBSD/install       /         nfs      ro        0    0
```

Replace *`myhost.example.com`* with the hostname or IP address of the NFS server. In this example, the root file system is mounted read-only in order to prevent NFS clients from potentially deleting the contents of the root file system.

12. Set the root password in the chroot(8) environment:

```
# chroot ${NFSROOTDIR}
# passwd
```

This sets the root password for client machines which are PXE booting.

13. Enable ssh(1) root logins for client machines which are PXE booting by editing `${NFSROOTDIR}/etc/ssh/sshd_config` and enabling `PermitRootLogin`. This option is documented in sshd_config(5).

14. Perform other customizations of the chroot(8) environment in ${NFSROOTDIR}. These customizations could include things like adding packages with pkg_add(1), editing the password file with vipw(8), or editing amd.conf(5) maps for automounting. For example:

```
# chroot ${NFSROOTDIR}
# pkg_add -r bash
```

32.8.2 Configuring Memory File Systems Used by `/etc/rc.initdiskless`

When booting from an NFS root volume, `/etc/rc` detects the NFS boot and runs `/etc/rc.initdiskless`. Read the comments in this script to understand what is going on. In this case, `/etc` and `/var` need to be memory backed file systems so that these directories are writable but the NFS root directory is read-only:

```
# chroot ${NFSROOTDIR}
# mkdir -p conf/base
# tar -c -v -f conf/base/etc.cpio.gz --format cpio --gzip etc
# tar -c -v -f conf/base/var.cpio.gz --format cpio --gzip var
```

When the system boots, memory file systems for `/etc` and `/var` will be created and mounted and the contents of the `cpio.gz` files will be copied into them.

32.8.3 Setting up the DHCP Server

PXE requires a TFTP and a DHCP server to be set up. The DHCP server does not need to be the same machine as the TFTP server, but it needs to be accessible in the network.

1. Install the DHCP server by following the instructions documented at Section 30.6.7. Make sure that `/etc/rc.conf` and `/usr/local/etc/dhcpd.conf` are correctly configured.

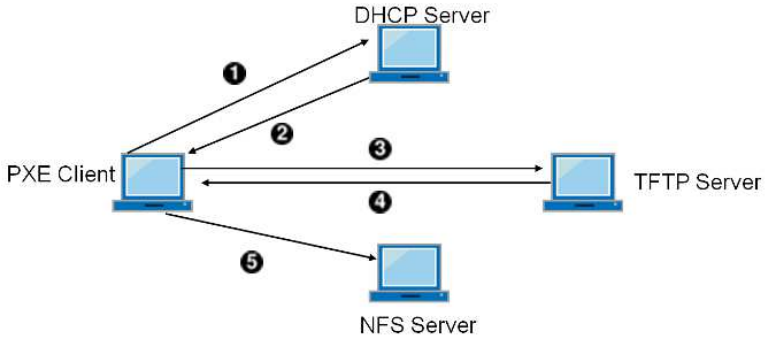
2. In `/usr/local/etc/dhcpd.conf`, configure the `next-server`, `filename`, and `option root-path` settings to specify the TFTP server IP address, the path to `/boot/pxeboot` in TFTP, and the path to the NFS root file system. Here is a sample `dhcpd.conf` setup:

```
subnet 192.168.0.0 netmask 255.255.255.0 {
   range 192.168.0.2 192.168.0.3 ;
   option subnet-mask 255.255.255.0 ;
   option routers 192.168.0.1 ;
   option broadcast-address 192.168.0.255 ;
   option domain-name-server 192.168.35.35, 192.168.35.36 ;
   option domain-name "example.com";

   # IP address of TFTP server
   next-server 192.168.0.1 ;

   # path of boot loader obtained
   # via tftp
   filename "FreeBSD/install/boot/pxeboot" ;

   # pxeboot boot loader will try to NFS mount this directory for root FS
   option root-path "192.168.0.1:/b/tftpboot/FreeBSD/install/" ;

}
```

32.8.4 Configuring the PXE Client and Debugging Connection Problems

1. When the client machine boots up, enter the BIOS configuration menu. Configure the BIOS to boot from the network. If all previous configuration steps are correct, everything should "just work".
2. Use the `net/wireshark` package or port to debug the network traffic involved during the PXE booting process, as illustrated in the diagram below. In Section 32.8.3, an example configuration is shown where the DHCP, TFTP, and NFS servers are on the same machine. However, these servers can be on separate machines.

Figure 32-1. PXE Booting Process with NFS Root Mount



❶ Client broadcasts a `DHCPDISCOVER` message.

❷ The DHCP server responds with the IP address, `next-server`, `filename`, and `root-path` values.

❸ The client sends a TFTP request to `next-server`, asking to retrieve `filename`.

❹ The TFTP server responds and sends `filename` to client.

❺ The client executes `filename`, which is pxeboot(8), which then loads the kernel. When the kernel executes, the root file system specified by `root-path` is mounted over NFS.

3. Make sure that the `pxeboot` file can be retrieved by TFTP. On the TFTP server, read `/var/log/xferlog` to ensure that the `pxeboot` file is being retrieved from the correct location. To test this example configuration:

```
# tftp 192.168.0.1
tftp> get FreeBSD/install/boot/pxeboot
Received 264951 bytes in 0.1 seconds
```

   The `BUGS` sections in tftpd(8) and tftp(1) document some limitations with TFTP.

4. Make sure that the root file system can be mounted via NFS. To test this example configuration:

```
# mount -t nfs 192.168.0.1:/b/tftpboot/FreeBSD/install /mnt
```

5. Read the code in `src/sys/boot/i386/libi386/pxe.c` to understand how the `pxeboot` loader sets variables like `boot.nfsroot.server` and `boot.nfsroot.path`. These variables are then used in the NFS diskless root mount code in `src/sys/nfsclient/nfs_diskless.c`.

6. Read pxeboot(8) and loader(8).

32.9 Network Address Translation

Contributed by Chern Lee.

32.9.1 Overview

FreeBSD's Network Address Translation (NAT) daemon, natd(8), accepts incoming raw IP packets, changes the source to the local machine, and injects these packets back into the outgoing IP packet stream. The source IP address and port are changed such that when data is received back, it is able to determine the original location of the data and forward it back to its original requester.

The most common use of NAT is to perform what is commonly known as Internet Connection Sharing.

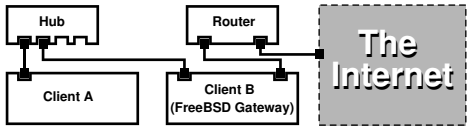
32.9.2 Setup

Due to the diminishing IP address space in IPv4 and the increased number of users on high-speed consumer lines such as cable or DSL, people are increasingly in need of an Internet Connection Sharing solution. The ability to connect several computers online through one connection and IP address makes natd(8) a reasonable choice.

Most commonly, a user has a machine connected to a cable or DSL line with one IP address and wishes to use this one connected computer to provide Internet access to several more over a LAN.

To do this, the FreeBSD machine connected to the Internet must act as a gateway. This gateway machine must have two NICs: one connects to the Internet router and the other connects to a LAN. All the machines on the LAN are connected through a hub or switch.

> **Note:** There are many ways to get a LAN connected to the Internet through a FreeBSD gateway. This example will only cover a gateway with at least two NICs.



A setup like this is commonly used to share an Internet connection. One of the LAN machines is connected to the Internet and the rest of the machines access the Internet through that “gateway” machine.

32.9.3 Boot Loader Configuration

The kernel features for natd(8) are not enabled in the `GENERIC` kernel, but they can be loaded at boot time by adding a couple of options to `/boot/loader.conf`:

```
ipfw_load="YES"
ipdivert_load="YES"
```

Additionally, the `net.inet.ip.fw.default_to_accept` tunable option should be set to `1`:

```
net.inet.ip.fw.default_to_accept="1"
```

> **Note:** It is a good idea to set this option during the first attempts to setup a firewall and NAT gateway. This sets the default policy of ipfw(8) to be more permissive than the default `deny ip from any to any`, making it slightly more difficult to get locked out of the system right after a reboot.

32.9.4 Kernel Configuration

When modules are not an option or if it is preferable to build all the required features into a custom kernel, the following options must be in the custom kernel configuration file:

```
options IPFIREWALL
options IPDIVERT
```

Additionally, the following may also be suitable:

```
options IPFIREWALL_DEFAULT_TO_ACCEPT
options IPFIREWALL_VERBOSE
```

32.9.5 System Startup Configuration

To enable firewall and NAT support at boot time, the following must be in `/etc/rc.conf`:

```
gateway_enable="YES" ❶
firewall_enable="YES" ❷
firewall_type="OPEN" ❸
natd_enable="YES"
natd_interface="fxp0" ❹
natd_flags="" ❺
```

❶ Sets up the machine to act as a gateway. Running `sysctl net.inet.ip.forwarding=1` would have the same effect.

❷ Enables the firewall rules in `/etc/rc.firewall` at boot.

❸ This specifies a predefined firewall ruleset that allows anything in. See `/etc/rc.firewall` for additional types.

❹ Indicates which interface to forward packets through. This is the interface that is connected to the Internet.

❺ Any additional configuration options passed to natd(8) on boot.

These `/etc/rc.conf` options will run `natd -interface fxp0` at boot. This can also be run manually after boot.

> **Note:** It is also possible to use a configuration file for natd(8) when there are too many options to pass. In this case, the configuration file must be defined by adding the following line to `/etc/rc.conf`:
>
> ```
> natd_flags="-f /etc/natd.conf"
> ```
>
> A list of configuration options, one per line, can be added to `/etc/natd.conf`. For example:
>
> ```
> redirect_port tcp 192.168.0.2:6667 6667
> redirect_port tcp 192.168.0.3:80 80
> ```
>
> For more information about this configuration file, consult natd(8).

Each machine and interface behind the LAN should be assigned IP addresses in the private network space, as defined by RFC 1918 (ftp://ftp.isi.edu/in-notes/rfc1918.txt), and have a default gateway of the natd(8) machine's internal IP address.

For example, client `A` and `B` behind the LAN have IP addresses of `192.168.0.2` and `192.168.0.3`, while the natd(8) machine's LAN interface has an IP address of `192.168.0.1`. The default gateway of clients `A` and `B` must be set to that of the natd(8) machine, `192.168.0.1`. The natd(8) machine's external Internet interface does not require any special modification for natd(8) to work.

32.9.6 Port Redirection

The drawback with natd(8) is that the LAN clients are not accessible from the Internet. Clients on the LAN can make outgoing connections to the world but cannot receive incoming ones. This presents a problem if trying to run Internet services on one of the LAN client machines. A simple way around this is to redirect selected Internet ports on the natd(8) machine to a LAN client.

For example, an IRC server runs on client `A` and a web server runs on client `B`. For this to work properly, connections received on ports 6667 (IRC) and 80 (HTTP) must be redirected to the respective machines.

The syntax for `-redirect_port` is as follows:

```
-redirect_port proto targetIP:targetPORT[-targetPORT]
               [aliasIP:]aliasPORT[-aliasPORT]
               [remoteIP[:remotePORT[-remotePORT]]]
```

In the above example, the argument should be:

```
-redirect_port tcp 192.168.0.2:6667 6667
-redirect_port tcp 192.168.0.3:80 80
```

This redirects the proper TCP ports to the LAN client machines.

Port ranges over individual ports can be indicated with `-redirect_port`. For example, *`tcp 192.168.0.2:2000-3000 2000-3000`* would redirect all connections received on ports 2000 to 3000 to ports 2000 to 3000 on client `A`.

These options can be used when directly running natd(8), placed within the `natd_flags=""` option in `/etc/rc.conf`, or passed via a configuration file.

For further configuration options, consult natd(8)

32.9.7 Address Redirection

Address redirection is useful if more than one IP address is available. Each LAN client can be assigned its own external IP address by natd(8), which will then rewrite outgoing packets from the LAN clients with the proper external IP address and redirects all traffic incoming on that particular IP address back to the specific LAN client. This is also known as static NAT. For example, if IP addresses `128.1.1.1`, `128.1.1.2`, and `128.1.1.3` are available, `128.1.1.1` can be used as the natd(8) machine's external IP address, while `128.1.1.2` and `128.1.1.3` are forwarded back to LAN clients `A` and `B`.

The `-redirect_address` syntax is as follows:

```
-redirect_address localIP publicIP
```

localIP	The internal IP address of the LAN client.
publicIP	The external IP address corresponding to the LAN client.

In the example, this argument would read:

```
-redirect_address 192.168.0.2 128.1.1.2
-redirect_address 192.168.0.3 128.1.1.3
```

Like `-redirect_port`, these arguments are placed within the `natd_flags=""` option of `/etc/rc.conf`, or passed via a configuration file. With address redirection, there is no need for port redirection since all data received on a particular IP address is redirected.

The external IP addresses on the natd(8) machine must be active and aliased to the external interface. Refer to rc.conf(5) for details.

32.10 IPv6

Originally Written by Aaron Kaplan. Restructured and Added by Tom Rhodes. Extended by Brad Davis.

IPv6, also known as IPng "IP next generation", is the new version of the well known IP protocol, also known as IPv4. FreeBSD includes the KAME (http://www.kame.net/) IPv6 reference implementation. FreeBSD comes with everything needed to use IPv6. This section focuses on getting IPv6 configured and running.

In the early 1990s, people became aware of the rapidly diminishing address space of IPv4. Given the expansion rate of the Internet, there were two major concerns:

- Running out of addresses. Today this is not so much of a concern, since RFC1918 private address space (`10.0.0.0/8`, `172.16.0.0/12`, and `192.168.0.0/16`) and NAT are being employed.
- Router table entries were getting too large. This is still a concern today.

IPv6 deals with these and many other issues by providing the following:

- 128 bit address space which allows for 340,282,366,920,938,463,463,374,607,431,768,211,456 addresses. This means there are approximately 6.67 * 10^27 IPv6 addresses per square meter on the planet.
- Routers only store network aggregation addresses in their routing tables, thus reducing the average space of a routing table to 8192 entries.

There are other useful features of IPv6:

- Address autoconfiguration (RFC2462 (http://www.ietf.org/rfc/rfc2462.txt)).
- Anycast addresses ("one-out-of many").
- Mandatory multicast addresses.
- IPsec (IP security).
- Simplified header structure.
- Mobile IP.
- IPv6-to-IPv4 transition mechanisms.

For more information see:

- KAME.net (http://www.kame.net)

32.10.1 Background on IPv6 Addresses

There are different types of IPv6 addresses: unicast, anycast, and multicast.

Unicast addresses are the well known addresses. A packet sent to a unicast address arrives at the interface belonging to the address.

Anycast addresses are syntactically indistinguishable from unicast addresses but they address a group of interfaces. The packet destined for an anycast address will arrive at the nearest (in router metric) interface. Anycast addresses may only be used by routers.

Multicast addresses identify a group of interfaces. A packet destined for a multicast address will arrive at all interfaces belonging to the multicast group.

Note: The IPv4 broadcast address, usually `xxx.xxx.xxx.255`, is expressed by multicast addresses in IPv6.

Table 32-2. Reserved IPv6 Addresses

IPv6 address	Prefixlength (Bits)	Description	Notes
`::`	128 bits	unspecified	Equivalent to `0.0.0.0` in IPv4.

IPv6 address	Prefixlength (Bits)	Description	Notes
`::1`	128 bits	loopback address	Equivalent to `127.0.0.1` in IPv4.
`::00:xx:xx:xx:xx`	96 bits	embedded IPv4	The lower 32 bits are the compatible IPv4 address.
`::ff:xx:xx:xx:xx`	96 bits	IPv4 mapped IPv6 address	The lower 32 bits are the IPv4 address for hosts which do not support IPv6.
`fe80::` - `feb::`	10 bits	link-local	Equivalent to the loopback address in IPv4.
`fec0::` - `fef::`	10 bits	site-local	
`ff::`	8 bits	multicast	
`001` (base 2)	3 bits	global unicast	All global unicast addresses are assigned from this pool. The first 3 bits are “001”.

32.10.2 Reading IPv6 Addresses

The canonical form is represented as: `x:x:x:x:x:x:x:x`, with each “x” being a 16 bit hex value. For example: `FEBC:A574:382B:23C1:AA49:4592:4EFE:9982`.

Often an address will have long substrings of all zeros. One such substring per address can be abbreviated by “::”. Also, up to three leading “0”s per hex quad can be omitted. For example, `fe80::1` corresponds to the canonical form `fe80:0000:0000:0000:0000:0000:0000:0001`.

A third form is to write the last 32 bit part in the well known (decimal) IPv4 style with dots (“.”) as separators. For example, `2002::10.0.0.1` corresponds to the hexadecimal canonical representation `2002:0000:0000:0000:0000:0000:0a00:0001`, which in turn is equivalent to `2002::a00:1`.

Here is a sample entry from ifconfig(8):

```
# ifconfig

rl0: flags=8943<UP,BROADCAST,RUNNING,PROMISC,SIMPLEX,MULTICAST> mtu 1500
         inet 10.0.0.10 netmask 0xffffff00 broadcast 10.0.0.255
         inet6 fe80::200:21ff:fe03:8e1%rl0 prefixlen 64 scopeid 0x1
         ether 00:00:21:03:08:e1
         media: Ethernet autoselect (100baseTX )
         status: active
```

`fe80::200:21ff:fe03:8e1%rl0` is an auto configured link-local address. It is generated from the MAC address as part of the auto configuration.

For further information on the structure of IPv6 addresses, see RFC3513 (http://www.ietf.org/rfc/rfc3513.txt).

32.10.3 Getting Connected

Currently, there are four ways to connect to other IPv6 hosts and networks:

- Contact an Internet Service Provider to see if they offer IPv6.
- SixXS (http://www.sixxs.net) offers tunnels with end-points all around the globe.
- Tunnel via 6-to-4 as described in RFC3068 (http://www.ietf.org/rfc/rfc3068.txt).
- Use the `net/freenet6` port for a dial-up connection.

32.10.4 DNS in the IPv6 World

There used to be two types of DNS records for IPv6. The IETF has declared AAAA records as the current standard.

Using AAAA records is straightforward. Assign the hostname to the IPv6 address in the primary zone DNS file:

```
MYHOSTNAME            AAAA     MYIPv6ADDR
```

Current versions of named(8) and `dns/djbdns` support AAAA records.

32.10.5 Applying the Needed Changes to `/etc/rc.conf`

32.10.5.1 IPv6 Client Settings

These settings configure a machine on a LAN which acts as a client, not a router. To instruct rtsol(8) to autoconfigure the interface on boot on FreeBSD 9.*x* and later, add this line to `rc.conf`:

```
ipv6_prefer="YES"
```

For FreeBSD 8.*x*, add:

```
ipv6_enable="YES"
```

To statically assign the IPv6 address, `2001:471:1f11:251:290:27ff:fee0:2093`, to `fxp0`, add the following for FreeBSD 9.*x*:

```
ifconfig_fxp0_ipv6="inet6 2001:471:1f11:251:290:27ff:fee0:2093 prefixlen 64"
```

> **Note:** Be sure to change *`prefixlen 64`* to the appropriate value for the subnet.

For FreeBSD 8*x*, add:

```
ipv6_ifconfig_fxp0="2001:471:1f11:251:290:27ff:fee0:2093"
```

To assign a default router of `2001:471:1f11:251::1`, add the following to `/etc/rc.conf`:

```
ipv6_defaultrouter="2001:471:1f11:251::1"
```

32.10.5.2 IPv6 Router/Gateway Settings

This section demonstrates how to take the directions from a tunnel provider and convert it into settings that will persist through reboots. To restore the tunnel on startup, add the following lines to `/etc/rc.conf`.

The first entry lists the generic tunneling interfaces to be configured. This example configures one interface, `gif0`:

```
gif_interfaces="gif0"
```

To configure that interface with a local endpoint of *`MY_IPv4_ADDR`* to a remote endpoint of *`REMOTE_IPv4_ADDR`*:

```
gifconfig_gif0="MY_IPv4_ADDR REMOTE_IPv4_ADDR"
```

To apply the IPv6 address that has been assigned for use as the IPv6 tunnel endpoint, add the following line for FreeBSD 9.*x* and later:

```
ifconfig_gif0_ipv6="inet6 MY_ASSIGNED_IPv6_TUNNEL_ENDPOINT_ADDR"
```

For FreeBSD 8.*x*, add:

```
ipv6_ifconfig_gif0="MY_ASSIGNED_IPv6_TUNNEL_ENDPOINT_ADDR"
```

Then, set the default route for IPv6. This is the other side of the IPv6 tunnel:

```
ipv6_defaultrouter="MY_IPv6_REMOTE_TUNNEL_ENDPOINT_ADDR"
```

32.10.5.3 IPv6 Tunnel Settings

If the server is to route IPv6 between the rest of the network and the world, the following `/etc/rc.conf` setting will also be needed:

```
ipv6_gateway_enable="YES"
```

32.10.6 Router Advertisement and Host Auto Configuration

This section demonstrates how to setup rtadvd(8) to advertise the IPv6 default route.

To enable rtadvd(8), add the following to `/etc/rc.conf`:

```
rtadvd_enable="YES"
```

It is important to specify the interface on which to do IPv6 router solicitation. For example, to tell rtadvd(8) to use `fxp0`:

```
rtadvd_interfaces="fxp0"
```

Next, create the configuration file, `/etc/rtadvd.conf` as seen in this example:

```
fxp0:\
       :addrs#1:addr="2001:471:1f11:246::":prefixlen#64:tc=ether:
```

Replace `fxp0` with the interface to be used and `2001:471:1f11:246::` with the prefix of the allocation.

For a dedicated `/64` subnet, nothing else needs to be changed. Otherwise, change the `prefixlen#` to the correct value.

32.10.7 IPv6 and IPv6 Address Mapping

When IPv6 is enabled on a server, there may be a need to enable IPv4 mapped IPv6 address communication. This compatibility option allows for IPv4 addresses to be represented as IPv6 addresses. Permitting IPv6 applications to communicate with IPv4 and vice versa may be a security issue.

This option may not be required in most cases and is available only for compatibility. This option will allow IPv6-only applications to work with IPv4 in a dual stack environment. This is most useful for third party applications which may not support an IPv6-only environment. To enable this feature, add the following to `/etc/rc.conf`:

```
ipv6_ipv4mapping="YES"
```

Reviewing the information in RFC 3493, section 3.6 and 3.7 as well as RFC 4038 section 4.2 may be useful to some adminstrators.

32.11 Asynchronous Transfer Mode (ATM)

Contributed by Harti Brandt.

32.11.1 Configuring Classical IP over ATM

Classical IP over ATM (CLIP) is the simplest method to use Asynchronous Transfer Mode (ATM) with IP. It can be used with Switched Virtual Circuits (SVCs) and with Permanent Virtual Circuits (PVCs). This section describes how to set up a network based on PVCs.

32.11.1.1 Fully Meshed Configurations

The first method to set up a CLIP with PVCs is to connect each machine to each other machine in the network via a dedicated PVC. While this is simple to configure, it becomes impractical for a large number of machines. The following example supposes four machines in the network, each connected to the ATM network with an ATM adapter card. The first step is the planning of the IP addresses and the ATM connections between the machines. This example uses the following:

Host	IP Address
`hostA`	`192.168.173.1`
`hostB`	`192.168.173.2`
`hostC`	`192.168.173.3`
`hostD`	`192.168.173.4`

To build a fully meshed net, one ATM connection is needed between each pair of machines:

Machines	VPI.VCI couple
`hostA` - `hostB`	0.100

Machines	VPI.VCI couple
hostA - hostC	0.101
hostA - hostD	0.102
hostB - hostC	0.103
hostB - hostD	0.104
hostC - hostD	0.105

The Virtual Path Identifier VPI and Virtual Channel Identifier VCI values at each end of the connection may differ, but for simplicity, this example assumes they are the same. Next, configure the ATM interfaces on each host:

```
hostA# ifconfig hatm0 192.168.173.1 up
hostB# ifconfig hatm0 192.168.173.2 up
hostC# ifconfig hatm0 192.168.173.3 up
hostD# ifconfig hatm0 192.168.173.4 up
```

This example assumes that the ATM interface is `hatm0` on all hosts. Next, the PVCs need to be configured on `hostA`. This should already be configured on the ATM switch; consult the manual for the switch on how to do this.

```
hostA# atmconfig natm add 192.168.173.2 hatm0 0 100 llc/snap ubr
hostA# atmconfig natm add 192.168.173.3 hatm0 0 101 llc/snap ubr
hostA# atmconfig natm add 192.168.173.4 hatm0 0 102 llc/snap ubr

hostB# atmconfig natm add 192.168.173.1 hatm0 0 100 llc/snap ubr
hostB# atmconfig natm add 192.168.173.3 hatm0 0 103 llc/snap ubr
hostB# atmconfig natm add 192.168.173.4 hatm0 0 104 llc/snap ubr

hostC# atmconfig natm add 192.168.173.1 hatm0 0 101 llc/snap ubr
hostC# atmconfig natm add 192.168.173.2 hatm0 0 103 llc/snap ubr
hostC# atmconfig natm add 192.168.173.4 hatm0 0 105 llc/snap ubr

hostD# atmconfig natm add 192.168.173.1 hatm0 0 102 llc/snap ubr
hostD# atmconfig natm add 192.168.173.2 hatm0 0 104 llc/snap ubr
hostD# atmconfig natm add 192.168.173.3 hatm0 0 105 llc/snap ubr
```

Other traffic contracts besides `ubr` can be used if the ATM adapter supports it. In this case, the name of the traffic contract is followed by the parameters of the traffic. Help for the atmconfig(8) tool can be obtained with:

```
# atmconfig help natm add
```

Refer to atmconfig(8) for more information.

The same configuration can also be done via `/etc/rc.conf`. These lines configure `hostA`:

```
network_interfaces="lo0 hatm0"
ifconfig_hatm0="inet 192.168.173.1 up"
natm_static_routes="hostB hostC hostD"
route_hostB="192.168.173.2 hatm0 0 100 llc/snap ubr"
route_hostC="192.168.173.3 hatm0 0 101 llc/snap ubr"
route_hostD="192.168.173.4 hatm0 0 102 llc/snap ubr"
```

The current state of all CLIP routes can be obtained with:

```
hostA# atmconfig natm show
```

32.12 Common Address Redundancy Protocol (CARP)

Contributed by Tom Rhodes.

The Common Address Redundancy Protocol (CARP) allows multiple hosts to share the same IP address. In some configurations, this may be used for availability or load balancing. Hosts may use separate IP addresses, as in the example provided here.

To enable support for CARP, the FreeBSD kernel can be rebuilt as described in Chapter 9 with the following option:

```
device  carp
```

Alternatively, the `if_carp.ko` module can be loaded at boot time. Add the following line to `/boot/loader.conf`:

```
if_carp_load="YES"
```

CARP functionality should now be available and may be tuned via several sysctl(8) variables:

OID	Description
`net.inet.carp.allow`	Accept incoming CARP packets. Enabled by default.
`net.inet.carp.preempt`	This option downs all of the CARP interfaces on the host when one goes down. Disabled by default.
`net.inet.carp.log`	A value of `0` disables any logging. A value of `1` enables logging of bad CARP packets. Values greater than `1` enable logging of state changes for the CARP interfaces. The default value is `1`.
`net.inet.carp.arpbalance`	Balance local network traffic using ARP. Disabled by default.
`net.inet.carp.suppress_preempt`	A read-only variable showing the status of preemption suppression. Preemption can be suppressed if the link on an interface is down. A value of `0` means that preemption is not suppressed. Every problem increments this variable.

The CARP devices themselves may be created using ifconfig(8):

```
# ifconfig carp0 create
```

In a real environment, each interface has a unique identification number known as a Virtual Host IDentification (VHID) which is used to distinguish the host on the network.

32.12.1 Using CARP for Server Availability

One use of CARP is to provide server availability. This example configures failover support for three hosts, all with unique IP addresses and providing the same web content. These machines act in conjunction with a Round Robin

DNS configuration. The failover machine has two additional CARP interfaces, one for each of the content server's IP addresses. When a failure occurs, the failover server will pick up the failed machine's IP address. This means that the failure should go completely unnoticed by the user. The failover server requires identical content and services as the other content servers it is expected to pick up load for.

The two machines should be configured identically other than their hostnames and VHIDs. This example calls these machines `hosta.example.org` and `hostb.example.org` respectively. First, the required lines for a CARP configuration have to be added to `/etc/rc.conf`. Here are the lines for `hosta.example.org`:

```
hostname="hosta.example.org"
ifconfig_fxp0="inet 192.168.1.3 netmask 255.255.255.0"
cloned_interfaces="carp0"
ifconfig_carp0="vhid 1 pass testpass 192.168.1.50/24"
```

On `hostb.example.org`, use the following lines:

```
hostname="hostb.example.org"
ifconfig_fxp0="inet 192.168.1.4 netmask 255.255.255.0"
cloned_interfaces="carp0"
ifconfig_carp0="vhid 2 pass testpass 192.168.1.51/24"
```

> **Note:** It is very important that the passwords, specified by the `pass` option to ifconfig(8), are identical. The `carp` devices will only listen to and accept advertisements from machines with the correct password. The VHID must also be unique for each machine.

The third machine, `provider.example.org`, should be prepared so that it may handle failover from either host. This machine will require two `carp` devices, one to handle each host. The appropriate `/etc/rc.conf` configuration lines will be similar to the following:

```
hostname="provider.example.org"
ifconfig_fxp0="inet 192.168.1.5 netmask 255.255.255.0"
cloned_interfaces="carp0 carp1"
ifconfig_carp0="vhid 1 advskew 100 pass testpass 192.168.1.50/24"
ifconfig_carp1="vhid 2 advskew 100 pass testpass 192.168.1.51/24"
```

Having the two `carp` devices will allow `provider.example.org` to notice and pick up the IP address of either machine, should it stop responding.

> **Note:** The default FreeBSD kernel *may* have preemption enabled. If so, `provider.example.org` may not relinquish the IP address back to the original content server. In this case, an administrator may have to manually force the IP back to the master. The following command should be issued on `provider.example.org`:
>
> ```
> # ifconfig carp0 down && ifconfig carp0 up
> ```
>
> This should be done on the `carp` interface which corresponds to the correct host.

At this point, CARP should be enabled and available for testing. For testing, either networking has to be restarted or the machines rebooted.

More information is available in carp(4).

V. Appendices

Appendix A. Obtaining FreeBSD

A.1 CDROM and DVD Publishers

A.1.1 CD and DVD Sets

FreeBSD CD and DVD sets are available from many online retailers:

- FreeBSD Mall, Inc.
  2420 Sand Creek Rd C-1 #347
  Brentwood,
  CA
  94513
  USA
  Phone: +1 925 240-6652
  Fax: +1 925 674-0821
  Email: <info@freebsdmall.com>
  WWW: http://www.freebsdmall.com/
- Dr. Hinner EDV
  Kochelseestr. 11
  D-81371 München
  Germany
  Phone: (0177) 428 419 0
  WWW: http://www.hinner.de/linux/freebsd.html
- Linux Distro UK
  42 Wharfedale Road
  Margate
  CT9 2TB
  United Kingdom
  WWW: https://linux-distro.co.uk/
- The Linux Emporium
  The Techno Centre, Puma Way
  Parkside
  CV1 2TT
  United Kingdom
  Phone: +44 (0)247 615 8121
  Fax: +44 1491 837016
  WWW: http://www.linuxemporium.co.uk/products/bsd/
- LinuxCenter.Ru
  Galernaya Street, 55
  Saint-Petersburg
  190000
  Russia
  Phone: +7-812-3125208

Email: <info@linuxcenter.ru>
WWW: http://linuxcenter.ru/shop/freebsd

A.2 FTP Sites

The official sources for FreeBSD are available via anonymous FTP from a worldwide set of mirror sites. The site ftp://ftp.FreeBSD.org/pub/FreeBSD/ is well connected and allows a large number of connections to it, but you are probably better off finding a “closer” mirror site (especially if you decide to set up some sort of mirror site).

Additionally, FreeBSD is available via anonymous FTP from the following mirror sites. If you choose to obtain FreeBSD via anonymous FTP, please try to use a site near you. The mirror sites listed as “Primary Mirror Sites” typically have the entire FreeBSD archive (all the currently available versions for each of the architectures) but you will probably have faster download times from a site that is in your country or region. The regional sites carry the most recent versions for the most popular architecture(s) but might not carry the entire FreeBSD archive. All sites provide access via anonymous FTP but some sites also provide access via other methods. The access methods available for each site are provided in parentheses after the hostname.

Central Servers, Primary Mirror Sites, Armenia, Australia, Austria, Brazil, Canada, China, Czech Republic, Denmark, Estonia, Finland, France, Germany, Greece, Hong Kong, Iceland, Ireland, Israel, Italy, Japan, Korea, Latvia, Lithuania, Netherlands, New Zealand, Norway, Poland, Portugal, Romania, Russia, Saudi Arabia, Slovak Republic, Slovenia, South Africa, Spain, Sweden, Switzerland, Taiwan, Turkey, Ukraine, United Kingdom, USA.

(as of UTC)

Central Servers

- ftp://ftp.FreeBSD.org/pub/FreeBSD/ (ftp / ftpv6 / http (http://ftp.FreeBSD.org/pub/FreeBSD/) / httpv6 (http://ftp.FreeBSD.org/pub/FreeBSD/))

Primary Mirror Sites

In case of problems, please contact the hostmaster <mirror-admin@FreeBSD.org> for this domain.

- ftp://ftp1.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp2.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp3.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp4.FreeBSD.org/pub/FreeBSD/ (ftp / ftpv6 / http (http://ftp4.FreeBSD.org/pub/FreeBSD/) / httpv6 (http://ftp4.FreeBSD.org/pub/FreeBSD/))
- ftp://ftp5.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp6.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp7.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp8.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp9.FreeBSD.org/pub/FreeBSD/ (ftp)

- ftp://ftp10.FreeBSD.org/pub/FreeBSD/ (ftp / ftpv6 / http (http://ftp10.FreeBSD.org/pub/FreeBSD/) / httpv6 (http://ftp10.FreeBSD.org/pub/FreeBSD/))
- ftp://ftp11.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp12.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp13.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp14.FreeBSD.org/pub/FreeBSD/ (ftp / http (http://ftp14.FreeBSD.org/pub/FreeBSD/))

Armenia

In case of problems, please contact the hostmaster <hostmaster@am.FreeBSD.org> for this domain.

- ftp://ftp1.am.FreeBSD.org/pub/FreeBSD/ (ftp / http (http://ftp1.am.FreeBSD.org/pub/FreeBSD/) / rsync)

Australia

In case of problems, please contact the hostmaster <hostmaster@au.FreeBSD.org> for this domain.

- ftp://ftp.au.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp2.au.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp3.au.FreeBSD.org/pub/FreeBSD/ (ftp)

Austria

In case of problems, please contact the hostmaster <hostmaster@at.FreeBSD.org> for this domain.

- ftp://ftp.at.FreeBSD.org/pub/FreeBSD/ (ftp / ftpv6 / http (http://ftp.at.FreeBSD.org/pub/FreeBSD/) / httpv6 (http://ftp.at.FreeBSD.org/pub/FreeBSD/))

Brazil

In case of problems, please contact the hostmaster <hostmaster@br.FreeBSD.org> for this domain.

- ftp://ftp.br.FreeBSD.org/pub/FreeBSD/ (ftp / http (http://ftp.br.FreeBSD.org/pub/FreeBSD/))
- ftp://ftp2.br.FreeBSD.org/pub/FreeBSD/ (ftp / http (http://ftp2.br.FreeBSD.org/))
- ftp://ftp3.br.FreeBSD.org/pub/FreeBSD/ (ftp / rsync)
- ftp://ftp4.br.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp5.br.FreeBSD.org
- ftp://ftp6.br.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp7.br.FreeBSD.org/pub/FreeBSD/ (ftp)

Canada

In case of problems, please contact the hostmaster <hostmaster@ca.FreeBSD.org> for this domain.

- ftp://ftp.ca.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp2.ca.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp3.ca.FreeBSD.org/pub/FreeBSD/ (ftp)

China

In case of problems, please contact the hostmaster <hostmaster@cn.FreeBSD.org> for this domain.

- ftp://ftp.cn.FreeBSD.org/pub/FreeBSD/ (ftp)

Czech Republic

In case of problems, please contact the hostmaster <hostmaster@cz.FreeBSD.org> for this domain.

- ftp://ftp.cz.FreeBSD.org/pub/FreeBSD/ (ftp / ftpv6 (ftp://ftp.cz.FreeBSD.org/pub/FreeBSD/) / http (http://ftp.cz.FreeBSD.org/pub/FreeBSD/) / httpv6 (http://ftp.cz.FreeBSD.org/pub/FreeBSD/) / rsync / rsyncv6)
- ftp://ftp2.cz.FreeBSD.org/pub/FreeBSD/ (ftp / http (http://ftp2.cz.FreeBSD.org/pub/FreeBSD/))

Denmark

In case of problems, please contact the hostmaster <hostmaster@dk.FreeBSD.org> for this domain.

- ftp://ftp.dk.FreeBSD.org/pub/FreeBSD/ (ftp / ftpv6 / http (http://ftp.dk.FreeBSD.org/pub/FreeBSD/) / httpv6 (http://ftp.dk.FreeBSD.org/pub/FreeBSD/))

Estonia

In case of problems, please contact the hostmaster <hostmaster@ee.FreeBSD.org> for this domain.

- ftp://ftp.ee.FreeBSD.org/pub/FreeBSD/ (ftp)

Finland

In case of problems, please contact the hostmaster <hostmaster@fi.FreeBSD.org> for this domain.

- ftp://ftp.fi.FreeBSD.org/pub/FreeBSD/ (ftp)

France

In case of problems, please contact the hostmaster <hostmaster@fr.FreeBSD.org> for this domain.

- ftp://ftp.fr.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp1.fr.FreeBSD.org/pub/FreeBSD/ (ftp / http (http://ftp1.fr.FreeBSD.org/pub/FreeBSD/) / rsync)

- ftp://ftp2.fr.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp3.fr.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp4.fr.FreeBSD.org/pub/FreeBSD/ (ftp / ftpv6 (ftp://ftp4.fr.FreeBSD.org/pub/FreeBSD/) / http (http://ftp4.fr.FreeBSD.org/pub/FreeBSD/) / httpv6 (http://ftp4.fr.FreeBSD.org/pub/FreeBSD/) / rsync (rsync://ftp4.fr.FreeBSD.org/FreeBSD/) / rsyncv6 (rsync://ftp4.fr.FreeBSD.org/FreeBSD/))
- ftp://ftp5.fr.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp6.fr.FreeBSD.org/pub/FreeBSD/ (ftp / rsync)
- ftp://ftp7.fr.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp8.fr.FreeBSD.org/pub/FreeBSD/ (ftp)

Germany

In case of problems, please contact the hostmaster <de-bsd-hubs@de.FreeBSD.org> for this domain.

- ftp://ftp.de.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp1.de.FreeBSD.org/freebsd/ (ftp / http (http://www1.de.FreeBSD.org/freebsd/) / rsync (rsync://rsync3.de.FreeBSD.org/freebsd/))
- ftp://ftp2.de.FreeBSD.org/pub/FreeBSD/ (ftp / http (http://ftp2.de.FreeBSD.org/pub/FreeBSD/) / rsync)
- ftp://ftp3.de.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp4.de.FreeBSD.org/FreeBSD/ (ftp / http (http://ftp4.de.FreeBSD.org/pub/FreeBSD/) / rsync)
- ftp://ftp5.de.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp6.de.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp7.de.FreeBSD.org/pub/FreeBSD/ (ftp / http (http://ftp7.de.FreeBSD.org/pub/FreeBSD/))
- ftp://ftp8.de.FreeBSD.org/pub/FreeBSD/ (ftp)

Greece

In case of problems, please contact the hostmaster <hostmaster@gr.FreeBSD.org> for this domain.

- ftp://ftp.gr.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp2.gr.FreeBSD.org/pub/FreeBSD/ (ftp)

Hong Kong

- ftp://ftp.hk.FreeBSD.org/pub/FreeBSD/ (ftp)

Iceland

In case of problems, please contact the hostmaster <hostmaster@is.FreeBSD.org> for this domain.

- ftp://ftp.is.FreeBSD.org/pub/FreeBSD/ (ftp / rsync)

Ireland

In case of problems, please contact the hostmaster <hostmaster@ie.FreeBSD.org> for this domain.

- ftp://ftp.ie.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp2.ie.FreeBSD.org/pub/FreeBSD/ (ftp / http (http://ftp2.ie.FreeBSD.org/pub/FreeBSD/) / rsync)
- ftp://ftp3.ie.FreeBSD.org/pub/FreeBSD/ (ftp / http (http://ftp3.ie.FreeBSD.org/pub/FreeBSD/) / rsync)

Israel

In case of problems, please contact the hostmaster <hostmaster@il.FreeBSD.org> for this domain.

- ftp://ftp.il.FreeBSD.org/pub/FreeBSD/ (ftp / ftpv6)

Italy

In case of problems, please contact the hostmaster <hostmaster@it.FreeBSD.org> for this domain.

- ftp://ftp.it.FreeBSD.org/pub/FreeBSD/ (ftp)

Japan

In case of problems, please contact the hostmaster <hostmaster@jp.FreeBSD.org> for this domain.

- ftp://ftp.jp.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp2.jp.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp3.jp.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp4.jp.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp5.jp.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp6.jp.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp7.jp.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp8.jp.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp9.jp.FreeBSD.org/pub/FreeBSD/ (ftp)

Korea

In case of problems, please contact the hostmaster <hostmaster@kr.FreeBSD.org> for this domain.

- ftp://ftp.kr.FreeBSD.org/pub/FreeBSD/ (ftp / rsync)

- ftp://ftp2.kr.FreeBSD.org/pub/FreeBSD/ (ftp / http (http://ftp2.kr.FreeBSD.org/pub/FreeBSD/))

Latvia

In case of problems, please contact the hostmaster <hostmaster@lv.FreeBSD.org> for this domain.

- ftp://ftp.lv.FreeBSD.org/pub/FreeBSD/ (ftp / http (http://ftp.lv.FreeBSD.org/pub/FreeBSD/))
- ftp://ftp2.lv.FreeBSD.org/pub/FreeBSD/ (ftp)

Lithuania

In case of problems, please contact the hostmaster <hostmaster@lt.FreeBSD.org> for this domain.

- ftp://ftp.lt.FreeBSD.org/pub/FreeBSD/ (ftp / http (http://ftp.lt.FreeBSD.org/pub/FreeBSD/))

Netherlands

In case of problems, please contact the hostmaster <hostmaster@nl.FreeBSD.org> for this domain.

- ftp://ftp.nl.FreeBSD.org/pub/FreeBSD/ (ftp / http (http://ftp.nl.FreeBSD.org/os/FreeBSD/) / rsync)
- ftp://ftp2.nl.FreeBSD.org/pub/FreeBSD/ (ftp)

New Zealand

- ftp://ftp.nz.FreeBSD.org/pub/FreeBSD/ (ftp / http (http://ftp.nz.FreeBSD.org/pub/FreeBSD/))

Norway

In case of problems, please contact the hostmaster <hostmaster@no.FreeBSD.org> for this domain.

- ftp://ftp.no.FreeBSD.org/pub/FreeBSD/ (ftp / rsync)
- ftp://ftp3.no.FreeBSD.org/pub/FreeBSD/ (ftp)

Poland

In case of problems, please contact the hostmaster <hostmaster@pl.FreeBSD.org> for this domain.

- ftp://ftp.pl.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp2.pl.FreeBSD.org/pub/FreeBSD/ (ftp / ftpv6 (ftp://ftp2.pl.FreeBSD.org/pub/FreeBSD/) / http (http://ftp2.pl.FreeBSD.org/pub/FreeBSD/) / httpv6 (http://ftp2.pl.FreeBSD.org/pub/FreeBSD/) / rsync / rsyncv6)

Portugal

In case of problems, please contact the hostmaster <hostmaster@pt.FreeBSD.org> for this domain.

- ftp://ftp.pt.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp2.pt.FreeBSD.org/pub/freebsd/ (ftp)
- ftp://ftp4.pt.FreeBSD.org/pub/ISO/FreeBSD/ (ftp)

Romania

In case of problems, please contact the hostmaster <hostmaster@ro.FreeBSD.org> for this domain.

- ftp://ftp.ro.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp1.ro.FreeBSD.org/pub/FreeBSD/ (ftp / ftpv6 / http (http://ftp1.ro.FreeBSD.org/pub/FreeBSD/) / httpv6 (http://ftp1.ro.FreeBSD.org/pub/FreeBSD/))

Russia

In case of problems, please contact the hostmaster <hostmaster@ru.FreeBSD.org> for this domain.

- ftp://ftp.ru.FreeBSD.org/pub/FreeBSD/ (ftp / http (http://ftp.ru.FreeBSD.org/FreeBSD/) / rsync)
- ftp://ftp2.ru.FreeBSD.org/pub/FreeBSD/ (ftp / http (http://ftp2.ru.FreeBSD.org/pub/FreeBSD/) / rsync)
- ftp://ftp3.ru.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp4.ru.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp5.ru.FreeBSD.org/pub/FreeBSD/ (ftp / http (http://ftp5.ru.FreeBSD.org/pub/FreeBSD/) / rsync)
- ftp://ftp6.ru.FreeBSD.org/pub/FreeBSD/ (ftp)

Saudi Arabia

In case of problems, please contact the hostmaster <ftpadmin@isu.net.sa> for this domain.

- ftp://ftp.isu.net.sa/pub/ftp.freebsd.org/ (ftp)

Slovak Republic

In case of problems, please contact the hostmaster <hostmaster@sk.FreeBSD.org> for this domain.

- ftp://ftp.sk.FreeBSD.org/pub/FreeBSD/ (ftp / ftpv6 (ftp://ftp.sk.FreeBSD.org/pub/FreeBSD/) / http (http://ftp.sk.FreeBSD.org/pub/FreeBSD/) / httpv6 (http://ftp.sk.FreeBSD.org/pub/FreeBSD/) / rsync / rsyncv6)
- ftp://ftp2.sk.FreeBSD.org/pub/FreeBSD/ (ftp / ftpv6 (ftp://ftp2.sk.FreeBSD.org/pub/FreeBSD/) / http (http://ftp2.sk.FreeBSD.org/pub/FreeBSD/) / httpv6 (http://ftp2.sk.FreeBSD.org/pub/FreeBSD/))

Slovenia

In case of problems, please contact the hostmaster <hostmaster@si.FreeBSD.org> for this domain.

- ftp://ftp.si.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp2.si.FreeBSD.org/pub/FreeBSD/ (ftp)

South Africa

In case of problems, please contact the hostmaster <hostmaster@za.FreeBSD.org> for this domain.

- ftp://ftp.za.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp2.za.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp3.za.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp4.za.FreeBSD.org/pub/FreeBSD/ (ftp)

Spain

In case of problems, please contact the hostmaster <hostmaster@es.FreeBSD.org> for this domain.

- ftp://ftp.es.FreeBSD.org/pub/FreeBSD/ (ftp / http (http://ftp.es.FreeBSD.org/pub/FreeBSD/))
- ftp://ftp2.es.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp3.es.FreeBSD.org/pub/FreeBSD/ (ftp)

Sweden

In case of problems, please contact the hostmaster <hostmaster@se.FreeBSD.org> for this domain.

- ftp://ftp.se.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp2.se.FreeBSD.org/pub/FreeBSD/ (ftp / rsync (rsync://ftp2.se.FreeBSD.org/))
- ftp://ftp3.se.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp4.se.FreeBSD.org/pub/FreeBSD/ (ftp / ftpv6 (ftp://ftp4.se.FreeBSD.org/pub/FreeBSD/) / http (http://ftp4.se.FreeBSD.org/pub/FreeBSD/) / httpv6 (http://ftp4.se.FreeBSD.org/pub/FreeBSD/) / rsync (rsync://ftp4.se.FreeBSD.org/pub/FreeBSD/) / rsyncv6 (rsync://ftp4.se.FreeBSD.org/pub/FreeBSD/))
- ftp://ftp5.se.FreeBSD.org/pub/FreeBSD/ (ftp / http (http://ftp5.se.FreeBSD.org/) / rsync)
- ftp://ftp6.se.FreeBSD.org/pub/FreeBSD (ftp / http (http://ftp6.se.FreeBSD.org/pub/FreeBSD))

Switzerland

In case of problems, please contact the hostmaster <hostmaster@ch.FreeBSD.org> for this domain.

- ftp://ftp.ch.FreeBSD.org/pub/FreeBSD/ (ftp / http (http://ftp.ch.FreeBSD.org/pub/FreeBSD/))

Taiwan

In case of problems, please contact the hostmaster <hostmaster@tw.FreeBSD.org> for this domain.

- ftp://ftp.tw.FreeBSD.org/pub/FreeBSD/ (ftp / ftpv6 (ftp://ftp.tw.FreeBSD.org/pub/FreeBSD/) / rsync / rsyncv6)
- ftp://ftp2.tw.FreeBSD.org/pub/FreeBSD/ (ftp / ftpv6 (ftp://ftp2.tw.FreeBSD.org/pub/FreeBSD/) / http (http://ftp2.tw.FreeBSD.org/pub/FreeBSD/) / httpv6 (http://ftp2.tw.FreeBSD.org/pub/FreeBSD/) / rsync / rsyncv6)
- ftp://ftp3.tw.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp4.tw.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp5.tw.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp6.tw.FreeBSD.org/pub/FreeBSD/ (ftp / http (http://ftp6.tw.FreeBSD.org/) / rsync)
- ftp://ftp7.tw.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp8.tw.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp9.tw.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp10.tw.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp11.tw.FreeBSD.org/pub/FreeBSD/ (ftp / http (http://ftp11.tw.FreeBSD.org/FreeBSD/))
- ftp://ftp12.tw.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp13.tw.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp14.tw.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp15.tw.FreeBSD.org/pub/FreeBSD/ (ftp)

Turkey

- ftp://ftp.tr.FreeBSD.org/pub/FreeBSD/ (ftp / http (http://ftp.tr.FreeBSD.org/pub/FreeBSD/) / rsync)
- ftp://ftp2.tr.FreeBSD.org/pub/FreeBSD/ (ftp / rsync)

Ukraine

- ftp://ftp.ua.FreeBSD.org/pub/FreeBSD/ (ftp / http (http://ftp.ua.FreeBSD.org/pub/FreeBSD/))
- ftp://ftp2.ua.FreeBSD.org/pub/FreeBSD/ (ftp / http (http://ftp2.ua.FreeBSD.org/pub/FreeBSD/))
- ftp://ftp7.ua.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp8.ua.FreeBSD.org/pub/FreeBSD/ (ftp / http (http://ftp8.ua.FreeBSD.org/FreeBSD/))
- ftp://ftp11.ua.FreeBSD.org/pub/FreeBSD/ (ftp)

United Kingdom

In case of problems, please contact the hostmaster <hostmaster@uk.FreeBSD.org> for this domain.

- ftp://ftp.uk.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp2.uk.FreeBSD.org/pub/FreeBSD/ (ftp / http (http://ftp2.uk.FreeBSD.org/) / rsync)
- ftp://ftp3.uk.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp4.uk.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp5.uk.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp6.uk.FreeBSD.org/pub/FreeBSD/ (ftp)

USA

In case of problems, please contact the hostmaster <hostmaster@us.FreeBSD.org> for this domain.

- ftp://ftp1.us.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp2.us.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp3.us.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp4.us.FreeBSD.org/pub/FreeBSD/ (ftp / ftpv6 / http (http://ftp4.us.FreeBSD.org/pub/FreeBSD/) / httpv6 (http://ftp4.us.FreeBSD.org/pub/FreeBSD/))
- ftp://ftp5.us.FreeBSD.org/pub/FreeBSD/ (ftp / rsync)
- ftp://ftp6.us.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp7.us.FreeBSD.org/pub/FreeBSD/ (ftp / http (http://ftp7.us.FreeBSD.org/pub/FreeBSD/) / rsync)
- ftp://ftp8.us.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp9.us.FreeBSD.org
- ftp://ftp10.us.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp11.us.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp12.us.FreeBSD.org/pub/FreeBSD/ (ftp / rsync)
- ftp://ftp13.us.FreeBSD.org/pub/FreeBSD/ (ftp / http (http://ftp13.us.FreeBSD.org/pub/FreeBSD/) / rsync)
- ftp://ftp14.us.FreeBSD.org/pub/FreeBSD/ (ftp / http (http://ftp14.us.FreeBSD.org/pub/FreeBSD/))
- ftp://ftp15.us.FreeBSD.org/pub/FreeBSD/ (ftp)

A.3 Anonymous CVS (Deprecated)

A.3.1 Warning

Warning: CVS has been deprecated by the project, and its use is not recommended. Subversion (http://www.FreeBSD.org/doc/en_US.ISO8859-1/books/handbook/svn.html) should be used instead.

A.4 Using CTM

CTM is a method for keeping a remote directory tree in sync with a central one. It has been developed for usage with FreeBSD's source trees, though other people may find it useful for other purposes as time goes by. Little, if any, documentation currently exists at this time on the process of creating deltas, so contact the ctm-users (http://lists.FreeBSD.org/mailman/listinfo/ctm-users) mailing list for more information and if you wish to use **CTM** for other things.

A.4.1 Why Should I Use CTM?

CTM will give you a local copy of the FreeBSD source trees. There are a number of "flavors" of the tree available. Whether you wish to track the entire CVS tree or just one of the branches, **CTM** can provide you the information. If you are an active developer on FreeBSD, but have lousy or non-existent TCP/IP connectivity, or simply wish to have the changes automatically sent to you, **CTM** was made for you. You will need to obtain up to three deltas per day for the most active branches. However, you should consider having them sent by automatic email. The sizes of the updates are always kept as small as possible. This is typically less than 5K, with an occasional (one in ten) being 10-50K and every now and then a large 100K+ or more coming around.

You will also need to make yourself aware of the various caveats related to working directly from the development sources rather than a pre-packaged release. This is particularly true if you choose the "current" sources. It is recommended that you read Staying current with FreeBSD.

A.4.2 What Do I Need to Use CTM?

You will need two things: The **CTM** program, and the initial deltas to feed it (to get up to "current" levels).

The **CTM** program has been part of FreeBSD ever since version 2.0 was released, and lives in `/usr/src/usr.sbin/ctm` if you have a copy of the source available.

The "deltas" you feed **CTM** can be had two ways, FTP or email. If you have general FTP access to the Internet then the following FTP sites support access to **CTM**:

ftp://ftp.FreeBSD.org/pub/FreeBSD/CTM/

or see section mirrors.

FTP the relevant directory and fetch the `README` file, starting from there.

If you wish to get your deltas via email:

Subscribe to one of the **CTM** distribution lists. ctm-src-cur (http://lists.FreeBSD.org/mailman/listinfo/ctm-src-cur) supports the entire Subversion tree. ctm-src-cur (http://lists.FreeBSD.org/mailman/listinfo/ctm-src-cur) supports the head of the development branch. ctm-src-9 (http://lists.FreeBSD.org/mailman/listinfo/ctm-src-9) supports the 9.X release branch, etc.. (If you do not know how to subscribe yourself to a list, click on the list name above or go to http://lists.FreeBSD.org/mailman/listinfo and click on the list that you wish to subscribe to. The list page should contain all of the necessary subscription instructions.)

When you begin receiving your **CTM** updates in the mail, you may use the `ctm_rmail` program to unpack and apply them. You can actually use the `ctm_rmail` program directly from a entry in `/etc/aliases` if you want to have the process run in a fully automated fashion. Check the `ctm_rmail` manual page for more details.

> **Note:** No matter what method you use to get the **CTM** deltas, you should subscribe to the ctm-announce (http://lists.FreeBSD.org/mailman/listinfo/ctm-announce) mailing list. In the future, this will be the only place where announcements concerning the operations of the **CTM** system will be posted. Click on the list name above and follow the instructions to subscribe to the list.

A.4.3 Using CTM for the First Time

Before you can start using **CTM** deltas, you will need to get to a starting point for the deltas produced subsequently to it.

First you should determine what you already have. Everyone can start from an "empty" directory. You must use an initial "Empty" delta to start off your **CTM** supported tree. At some point it is intended that one of these "started" deltas be distributed on the CD for your convenience, however, this does not currently happen.

Since the trees are many tens of megabytes, you should prefer to start from something already at hand. If you have a -RELEASE CD, you can copy or extract an initial source from it. This will save a significant transfer of data.

You can recognize these "starter" deltas by the `X` appended to the number (`src-cur.3210XEmpty.gz` for instance). The designation following the `X` corresponds to the origin of your initial "seed". `Empty` is an empty directory. As a rule a base transition from `Empty` is produced every 100 deltas. By the way, they are large! 70 to 80 Megabytes of `gzip`'d data is common for the `XEmpty` deltas.

Once you have picked a base delta to start from, you will also need all deltas with higher numbers following it.

A.4.4 Using CTM in Your Daily Life

To apply the deltas, simply say:

```
# cd /where/ever/you/want/the/stuff
# ctm -v -v /where/you/store/your/deltas/src-xxx.*
```

CTM understands deltas which have been put through `gzip`, so you do not need to `gunzip` them first, this saves disk space.

Unless it feels very secure about the entire process, **CTM** will not touch your tree. To verify a delta you can also use the `-c` flag and **CTM** will not actually touch your tree; it will merely verify the integrity of the delta and see if it would apply cleanly to your current tree.

There are other options to **CTM** as well, see the manual pages or look in the sources for more information.

That is really all there is to it. Every time you get a new delta, just run it through **CTM** to keep your sources up to date.

Do not remove the deltas if they are hard to download again. You just might want to keep them around in case something bad happens. Even if you only have floppy disks, consider using `fdwrite` to make a copy.

A.4.5 Keeping Your Local Changes

As a developer one would like to experiment with and change files in the source tree. **CTM** supports local modifications in a limited way: before checking for the presence of a file `foo`, it first looks for `foo.ctm`. If this file exists, **CTM** will operate on it instead of `foo`.

This behavior gives us a simple way to maintain local changes: simply copy the files you plan to modify to the corresponding file names with a `.ctm` suffix. Then you can freely hack the code, while **CTM** keeps the `.ctm` file up-to-date.

A.4.6 Other Interesting CTM Options

A.4.6.1 Finding Out Exactly What Would Be Touched by an Update

You can determine the list of changes that **CTM** will make on your source repository using the `-l` option to **CTM**.

This is useful if you would like to keep logs of the changes, pre- or post- process the modified files in any manner, or just are feeling a tad paranoid.

A.4.6.2 Making Backups Before Updating

Sometimes you may want to backup all the files that would be changed by a **CTM** update.

Specifying the `-B backup-file` option causes **CTM** to backup all files that would be touched by a given **CTM** delta to `backup-file`.

A.4.6.3 Restricting the Files Touched by an Update

Sometimes you would be interested in restricting the scope of a given **CTM** update, or may be interested in extracting just a few files from a sequence of deltas.

You can control the list of files that **CTM** would operate on by specifying filtering regular expressions using the `-e` and `-x` options.

For example, to extract an up-to-date copy of `lib/libc/Makefile` from your collection of saved **CTM** deltas, run the commands:

```
# cd /where/ever/you/want/to/extract/it/
# ctm -e '^lib/libc/Makefile' ~ctm/src-xxx.*
```

For every file specified in a **CTM** delta, the `-e` and `-x` options are applied in the order given on the command line. The file is processed by **CTM** only if it is marked as eligible after all the `-e` and `-x` options are applied to it.

A.4.7 Future Plans for CTM

Tons of them:

- Use some kind of authentication into the **CTM** system, so as to allow detection of spoofed **CTM** updates.
- Clean up the options to **CTM**, they became confusing and counter intuitive.

A.4.8 Miscellaneous Stuff

There is a sequence of deltas for the `ports` collection too, but interest has not been all that high yet.

A.4.9 CTM Mirrors

CTM/FreeBSD is available via anonymous FTP from the following mirror sites. If you choose to obtain **CTM** via anonymous FTP, please try to use a site near you.

In case of problems, please contact the ctm-users (http://lists.FreeBSD.org/mailman/listinfo/ctm-users) mailing list.

California, Bay Area, official source

- ftp://ftp.FreeBSD.org/pub/FreeBSD/development/CTM/

South Africa, backup server for old deltas

- ftp://ftp.za.FreeBSD.org/pub/FreeBSD/CTM/

Taiwan/R.O.C.

- ftp://ctm.tw.FreeBSD.org/pub/FreeBSD/development/CTM/
- ftp://ctm2.tw.FreeBSD.org/pub/FreeBSD/development/CTM/
- ftp://ctm3.tw.FreeBSD.org/pub/FreeBSD/development/CTM/

If you did not find a mirror near to you or the mirror is incomplete, try to use a search engine such as alltheweb (http://www.alltheweb.com/).

A.5 Using Subversion

A.5.1 Introduction

As of July 2012, FreeBSD uses Subversion (http://subversion.apache.org/) (*svn*) as the primary version control system for storing all of FreeBSD's source code, documentation, and the Ports Collection.

> **Note:** Subversion is generally a developer tool. Most users should use FreeBSD Update to update the FreeBSD base system, and Portsnap to update the FreeBSD Ports Collection.

In **Subversion**, URLs are used to designate a repository, taking the form of *protocol://hostname/path*. Mirrors may support different protocols as specified below. The first component of the path is the FreeBSD repository to access. There are three different repositories, `base` for the FreeBSD base system source code, `ports` for the Ports Collection, and `doc` for documentation. For example, the URL `svn://svn0.us-east.FreeBSD.org/ports/head/` specifies the main branch of the ports repository on the `svn0.us-east.FreeBSD.org` mirror, using the `svn` protocol.

A.5.2 Installation

Subversion must be installed before it can be used to check out the contents of any of the repositories. If a copy of the ports tree is already present, one can install **Subversion** like this:

```
# cd /usr/ports/devel/subversion
# make install clean
```

If the ports tree is not available, **Subversion** can be installed as a package:

```
# pkg_add -r subversion
```

If **pkgng** is being used to manage packages, **Subversion** can be installed with it instead:

```
# pkg install devel/subversion
```

A.5.3 Running Subversion

The `svn` command is used to fetch a clean copy of the sources into a local directory. The files in this directory are called a *local working copy*.

> **Warning:** If the local directory already exists but was not created by `svn`, rename or delete it before the checkout. Checkout over an existing non-`svn` directory can cause conflicts between the existing files and those brought in from the repository.

A checkout from a given repository is performed with a command like this:

```
# svn checkout svn-mirror/repository/branch lwcdir
```

where:

- *`svn-mirror`* is a URL for one of the Subversion mirror sites.
- *`repository`* is one of the Project repositories, i.e., `base`, `ports`, or `doc`.
- *`branch`* depends on the repository used. `ports` and `doc` are mostly updated in the `head` branch, while `base` maintains the latest version of -CURRENT under `head` and the respective latest versions of the -STABLE branches under `stable/8` (for 8.*x*) and `stable/9` (9.*x*).
- *`lwcdir`* is the target directory where the contents of the specified branch should be placed. This is usually `/usr/ports` for `ports`, `/usr/src` for `base`, and `/usr/doc` for `doc`.

This example checks out the Ports Collection from the western US repository using the HTTPS protocol, placing the local working copy in `/usr/ports`. If `/usr/ports` is already present but was not created by `svn`, remember to rename or delete it before the checkout.

```
# svn checkout https://svn0.us-west.FreeBSD.org/ports/head /usr/ports
```

Because the initial checkout has to download the full branch of the remote repository, it can take a while. Please be patient.

After the initial checkout, the local working copy can be updated by running:

```
# svn update lwcdir
```

To update `/usr/ports` created in the example above, use:

```
# svn update /usr/ports
```

The update is much quicker than a checkout, only transferring files that have changed.

An alternate way of updating the local working copy after checkout is provided by the `Makefile` in the `/usr/ports`, `/usr/src`, and `/usr/doc` directories. Set `SVN_UPDATE` and use the `update` target. For example, to update `/usr/src`:

```
# cd /usr/src
# make update SVN_UPDATE=yes
```

A.5.4 For More Information

For other information about using **Subversion**, please see the “Subversion Book”, titled Version Control with Subversion (http://svnbook.red-bean.com/), or the Subversion Documentation (http://subversion.apache.org/docs/).

A.6 Subversion Mirror Sites

All mirrors carry all repositories.

The master FreeBSD **Subversion** server, `svn.FreeBSD.org`, is publicly accessible, read-only. That may change in the future, so users are encouraged to use one of the official mirrors. To view the FreeBSD **Subversion** repositories through a browser, use http://svnweb.FreeBSD.org/.

> **Note:** The FreeBSD svn mirror network is still in its early days, and will likely change. Do not count on this list of mirrors being static. In particular, the SSL certificates of the servers will likely change at some point.

Name	Protocols	Location	SSL fingerprint

Name	Protocols	Location	SSL fingerprint
svn0.us-west.FreeBSD.org	SVN, http (http://svn0.us-west.FreeBSD.org/), https (https://svn0.us-west.FreeBSD.org/)	USA, California	SHA1 79:35:8F:CA:6D:34:D9:30:44:D1:00:AF:33:4D:E6:11:44:4D:15:EC
svn0.us-east.FreeBSD.org	SVN, http (http://svn0.us-east.FreeBSD.org/), https (https://svn0.us-east.FreeBSD.org/)	USA, New Jersey	SHA1 06:D1:23:DE:5E:7A:F7:2B:7A:7E:74:95:5F:54:8D:5C:B0:D6:2E:8F

HTTPS is the preferred protocol, providing protection against another computer pretending to be the FreeBSD mirror (commonly known as a “man in the middle” attack) or otherwise trying to send bad content to the end user.

On the first connection to an HTTPS mirror, the user will be asked to verify the server *fingerprint*:

```
Error validating server certificate for 'https://svn0.us-west.freebsd.org:443':
 - The certificate is not issued by a trusted authority. Use the
   fingerprint to validate the certificate manually!
Certificate information:
 - Hostname: svnmir.ysv.FreeBSD.org
 - Valid: from Fri, 24 Aug 2012 22:04:04 GMT until Sat, 24 Aug 2013 22:04:04 GMT
 - Issuer: clusteradm, FreeBSD.org, CA, US
 - Fingerprint: 79:35:8f:ca:6d:34:d9:30:44:d1:00:af:33:4d:e6:11:44:4d:15:ec
(R)eject, accept (t)emporarily or accept (p)ermanently?
```

Compare the fingerprint shown to those listed in the table above. If the fingerprint matches, the server security certificate can be accepted temporarily or permanently. A temporary certificate will expire after a single session with the server, and the verification step will be repeated on the next connection. Accepting the certificate permanently will store the authentication credentials in `~/.subversion/auth/` and the user will not be asked to verify the fingerprint again until the certificate expires.

If HTTPS cannot be used due to firewall or other problems, `SVN` is the next choice, with slightly faster transfers. When neither can be used, use HTTP.

A.7 Using CVSup (Deprecated)

A.7.1 Introduction

Warning: `cvsup` has been deprecated by the project, and its use is not recommended. **Subversion** should be

used instead.

CVSup is a software package for distributing and updating source trees from a master CVS repository on a remote server host. The FreeBSD sources are maintained in a CVS repository on a central development machine in California. With **CVSup**, FreeBSD users can easily keep their own source trees up to date.

CVSup uses the so-called *pull* model of updating. Under the pull model, each client asks the server for updates, if and when they are wanted. The server waits passively for update requests from its clients. Thus all updates are instigated by the client. The server never sends unsolicited updates. Users must either run the **CVSup** client manually to get an update, or they must set up a `cron` job to run it automatically on a regular basis.

The term **CVSup**, capitalized just so, refers to the entire software package. Its main components are the client `cvsup` which runs on each user's machine, and the server `cvsupd` which runs at each of the FreeBSD mirror sites.

> **Note:** The **csup** utility is a rewrite of the **CVSup** software in C. Its biggest advantage is, that it is faster and does not depend on the Modula-3 language, thus you do not need to install it as a requirement. Moreover you can use it out-of-the-box, since it is included in the base system. If you decided to use **csup**, just skip the steps on the installation of **CVSup** and substitute the references of **CVSup** with **csup** while following the remainder of this article.

A.7.2 Installation

The easiest way to install **CVSup** is to use the precompiled `net/cvsup` package from the FreeBSD packages collection. If you prefer to build **CVSup** from source, you can use the `net/cvsup` port instead. But be forewarned: the `net/cvsup` port depends on the Modula-3 system, which takes a substantial amount of time and disk space to download and build.

> **Note:** If you are going to be using **CVSup** on a machine which will not have **Xorg** installed, such as a server, be sure to use the port which does not include the **CVSup** GUI, `net/cvsup-without-gui`.

A.7.3 CVSup Configuration

CVSup's operation is controlled by a configuration file called the `supfile`. There are some sample `supfiles` in the directory `/usr/share/examples/cvsup/`.

The information in a `supfile` answers the following questions for **CVSup**:

- Which files do you want to receive?
- Which versions of them do you want?
- Where do you want to get them from?
- Where do you want to put them on your own machine?
- Where do you want to put your status files?

In the following sections, we will construct a typical `supfile` by answering each of these questions in turn. First, we describe the overall structure of a `supfile`.

A `supfile` is a text file. Comments begin with # and extend to the end of the line. Lines that are blank and lines that contain only comments are ignored.

Each remaining line describes a set of files that the user wishes to receive. The line begins with the name of a "collection", a logical grouping of files defined by the server. The name of the collection tells the server which files you want. After the collection name come zero or more fields, separated by white space. These fields answer the questions listed above. There are two types of fields: flag fields and value fields. A flag field consists of a keyword standing alone, e.g., `delete` or `compress`. A value field also begins with a keyword, but the keyword is followed without intervening white space by = and a second word. For example, `release=cvs` is a value field.

A `supfile` typically specifies more than one collection to receive. One way to structure a `supfile` is to specify all of the relevant fields explicitly for each collection. However, that tends to make the `supfile` lines quite long, and it is inconvenient because most fields are the same for all of the collections in a `supfile`. **CVSup** provides a defaulting mechanism to avoid these problems. Lines beginning with the special pseudo-collection name `*default` can be used to set flags and values which will be used as defaults for the subsequent collections in the `supfile`. A default value can be overridden for an individual collection, by specifying a different value with the collection itself. Defaults can also be changed or augmented in mid-supfile by additional `*default` lines.

With this background, we will now proceed to construct a `supfile` for receiving and updating the main source tree of FreeBSD-CURRENT.

- Which files do you want to receive?

  The files available via **CVSup** are organized into named groups called "collections". The collections that are available are described in the following section. In this example, we wish to receive the entire main source tree for the FreeBSD system. There is a single large collection `src-all` which will give us all of that. As a first step toward constructing our `supfile`, we simply list the collections, one per line (in this case, only one line):

  ```
  src-all
  ```

- Which version(s) of them do you want?

  With **CVSup**, you can receive virtually any version of the sources that ever existed. That is possible because the **cvsupd** server works directly from the CVS repository, which contains all of the versions. You specify which one of them you want using the `tag=` and `date=` value fields.

  > **Warning:** Be very careful to specify any `tag=` fields correctly. Some tags are valid only for certain collections of files. If you specify an incorrect or misspelled tag, **CVSup** will delete files which you probably do not want deleted. In particular, use *only* `tag=.` for the `ports-*` collections.

  The `tag=` field names a symbolic tag in the repository. There are two kinds of tags, revision tags and branch tags. A revision tag refers to a specific revision. Its meaning stays the same from day to day. A branch tag, on the other hand, refers to the latest revision on a given line of development, at any given time. Because a branch tag does not refer to a specific revision, it may mean something different tomorrow than it means today.

  Section A.8 contains branch tags that users might be interested in. When specifying a tag in **CVSup**'s configuration file, it must be preceded with `tag=` (`RELENG_8` will become `tag=RELENG_8`). Keep in mind that only the `tag=.` is relevant for the Ports Collection.

> **Warning:** Be very careful to type the tag name exactly as shown. **CVSup** cannot distinguish between valid and invalid tags. If you misspell the tag, **CVSup** will behave as though you had specified a valid tag which happens to refer to no files at all. It will delete your existing sources in that case.

When you specify a branch tag, you normally receive the latest versions of the files on that line of development. If you wish to receive some past version, you can do so by specifying a date with the `date=` value field. The cvsup(1) manual page explains how to do that.

For our example, we wish to receive FreeBSD-CURRENT. We add this line at the beginning of our `supfile`:

```
*default tag=.
```

There is an important special case that comes into play if you specify neither a `tag=` field nor a `date=` field. In that case, you receive the actual RCS files directly from the server's CVS repository, rather than receiving a particular version. Developers generally prefer this mode of operation. By maintaining a copy of the repository itself on their systems, they gain the ability to browse the revision histories and examine past versions of files. This gain is achieved at a large cost in terms of disk space, however.

- Where do you want to get them from?

  We use the `host=` field to tell `cvsup` where to obtain its updates. Any of the CVSup mirror sites will do, though you should try to select one that is close to you in cyberspace. In this example we will use a fictional FreeBSD distribution site, `cvsup99.FreeBSD.org`:

  ```
  *default host=cvsup99.FreeBSD.org
  ```

  You will need to change the host to one that actually exists before running **CVSup**. On any particular run of `cvsup`, you can override the host setting on the command line, with `-h` *`hostname`*.

- Where do you want to put them on your own machine?

  The `prefix=` field tells `cvsup` where to put the files it receives. In this example, we will put the source files directly into our main source tree, `/usr/src`. The `src` directory is already implicit in the collections we have chosen to receive, so this is the correct specification:

  ```
  *default prefix=/usr
  ```

- Where should `cvsup` maintain its status files?

  The **CVSup** client maintains certain status files in what is called the "base" directory. These files help **CVSup** to work more efficiently, by keeping track of which updates you have already received. We will use the standard base directory, `/var/db`:

  ```
  *default base=/var/db
  ```

  If your base directory does not already exist, now would be a good time to create it. The `cvsup` client will refuse to run if the base directory does not exist.

- Miscellaneous `supfile` settings:

  There is one more line of boiler plate that normally needs to be present in the `supfile`:

  ```
  *default release=cvs delete use-rel-suffix compress
  ```

  `release=cvs` indicates that the server should get its information out of the main FreeBSD CVS repository. This is virtually always the case, but there are other possibilities which are beyond the scope of this discussion.

`delete` gives **CVSup** permission to delete files. You should always specify this, so that **CVSup** can keep your source tree fully up-to-date. **CVSup** is careful to delete only those files for which it is responsible. Any extra files you happen to have will be left strictly alone.

`use-rel-suffix` is ... arcane. If you really want to know about it, see the cvsup(1) manual page. Otherwise, just specify it and do not worry about it.

`compress` enables the use of gzip-style compression on the communication channel. If your network link is T1 speed or faster, you probably should not use compression. Otherwise, it helps substantially.

- Putting it all together:

  Here is the entire `supfile` for our example:

```
*default tag=.
*default host=cvsup99.FreeBSD.org
*default prefix=/usr
*default base=/var/db
*default release=cvs delete use-rel-suffix compress

src-all
```

A.7.3.1 The `refuse` File

As mentioned above, **CVSup** uses a *pull method*. Basically, this means that you connect to the **CVSup** server, and it says, “Here is what you can download from me...”, and your client responds “OK, I will take this, this, this, and this.” In the default configuration, the **CVSup** client will take every file associated with the collection and tag you chose in the configuration file. In order to download a partial tree, use the `refuse` file.

The `refuse` file tells **CVSup** that it should not take every single file from a collection; in other words, it tells the client to *refuse* certain files from the server. The `refuse` file can be found (or, if you do not yet have one, should be placed) in *`base`*`/sup/`. *`base`* is defined in your `supfile`; our defined *`base`* is `/var/db`, which means that by default the `refuse` file is `/var/db/sup/refuse`.

The `refuse` file has a very simple format; it simply contains the names of files or directories that you do not wish to download. For example:

```
bin/
usr.bin/
```

Users who are on slow links or pay by the minute for their Internet connection will be able to save time as they will no longer need to download files that they will never use. For more information on `refuse` files and other neat features of **CVSup**, please view its manual page.

A.7.4 Running CVSup

You are now ready to try an update. The command line for doing this is quite simple:

```
# cvsup supfile
```

where *`supfile`* is of course the name of the `supfile` you have just created. Assuming you are running under X11, `cvsup` will display a GUI window with some buttons to do the usual things. Press the go button, and watch it run.

Since you are updating your actual `/usr/src` tree in this example, you will need to run the program as `root` so that `cvsup` has the permissions it needs to update your files. Having just created your configuration file, and having never used this program before, that might understandably make you nervous. There is an easy way to do a trial run without touching your precious files. Just create an empty directory somewhere convenient, and name it as an extra argument on the command line:

```
# mkdir /var/tmp/dest
# cvsup supfile /var/tmp/dest
```

The directory you specify will be used as the destination directory for all file updates. **CVSup** will examine your usual files in `/usr/src`, but it will not modify or delete any of them. Any file updates will instead land in `/var/tmp/dest/usr/src`. **CVSup** will also leave its base directory status files untouched when run this way. The new versions of those files will be written into the specified directory. As long as you have read access to `/usr/src`, you do not even need to be `root` to perform this kind of trial run.

If you are not running X11 or if you just do not like GUIs, you should add a couple of options to the command line when you run `cvsup`:

```
# cvsup -g -L 2 supfile
```

The `-g` tells **CVSup** not to use its GUI. This is automatic if you are not running X11, but otherwise you have to specify it.

The `-L 2` tells **CVSup** to print out the details of all the file updates it is doing. There are three levels of verbosity, from `-L 0` to `-L 2`. The default is 0, which means total silence except for error messages.

There are plenty of other options available. For a brief list of them, type `cvsup -H`. For more detailed descriptions, see the manual page.

Once you are satisfied with the way updates are working, you can arrange for regular runs of **CVSup** using cron(8). Obviously, you should not let **CVSup** use its GUI when running it from cron(8).

A.7.5 CVSup File Collections

The file collections available via **CVSup** are organized hierarchically. There are a few large collections, and they are divided into smaller sub-collections. Receiving a large collection is equivalent to receiving each of its sub-collections. The hierarchical relationships among collections are reflected by the use of indentation in the list below.

The most commonly used collection is `src-all`.

`cvs-all release=cvs`

The main FreeBSD CVS repository, including the cryptography code.

`distrib release=cvs`

Files related to the distribution and mirroring of FreeBSD.

`projects-all release=cvs`

Sources for the FreeBSD projects repository.

`src-all release=cvs`

The main FreeBSD sources, including the cryptography code.

`src-base release=cvs`

Miscellaneous files at the top of `/usr/src`.

`src-bin release=cvs`

User utilities that may be needed in single-user mode (`/usr/src/bin`).

`src-cddl release=cvs`

Utilities and libraries covered by the CDDL license (`/usr/src/cddl`).

`src-contrib release=cvs`

Utilities and libraries from outside the FreeBSD project, used relatively unmodified (`/usr/src/contrib`).

`src-crypto release=cvs`

Cryptography utilities and libraries from outside the FreeBSD project, used relatively unmodified (`/usr/src/crypto`).

`src-eBones release=cvs`

Kerberos and DES (`/usr/src/eBones`). Not used in current releases of FreeBSD.

`src-etc release=cvs`

System configuration files (`/usr/src/etc`).

`src-games release=cvs`

Games (`/usr/src/games`).

`src-gnu release=cvs`

Utilities covered by the GNU Public License (`/usr/src/gnu`).

`src-include release=cvs`

Header files (`/usr/src/include`).

`src-kerberos5 release=cvs`

Kerberos5 security package (`/usr/src/kerberos5`).

`src-kerberosIV release=cvs`

KerberosIV security package (`/usr/src/kerberosIV`).

`src-lib release=cvs`

Libraries (`/usr/src/lib`).

`src-libexec release=cvs`

System programs normally executed by other programs (`/usr/src/libexec`).

`src-release release=cvs`

Files required to produce a FreeBSD release (`/usr/src/release`).

`src-rescue release=cvs`

Statically linked programs for emergency recovery; see rescue(8) (`/usr/src/rescue`).

`src-sbin release=cvs`

System utilities for single-user mode (`/usr/src/sbin`).

`src-secure release=cvs`

Cryptographic libraries and commands (`/usr/src/secure`).

`src-share release=cvs`

Files that can be shared across multiple systems (`/usr/src/share`).

`src-sys release=cvs`

The kernel (`/usr/src/sys`).

`src-sys-crypto release=cvs`

Kernel cryptography code (`/usr/src/sys/crypto`).

`src-tools release=cvs`

Various tools for the maintenance of FreeBSD (`/usr/src/tools`).

`src-usrbin release=cvs`

User utilities (`/usr/src/usr.bin`).

`src-usrsbin release=cvs`

System utilities (`/usr/src/usr.sbin`).

`distrib release=self`

The **CVSup** server's own configuration files. Used by **CVSup** mirror sites.

`gnats release=current`

The GNATS bug-tracking database.

`mail-archive release=current`

FreeBSD mailing list archive.

A.7.6 For More Information

For the **CVSup** FAQ and other information about **CVSup**, see The CVSup Home Page (http://www.cvsup.org).

Most FreeBSD-related discussion of **CVSup** takes place on the FreeBSD technical discussions mailing list (http://lists.FreeBSD.org/mailman/listinfo/freebsd-hackers). New versions of the software are announced there, as well as on the FreeBSD announcements mailing list (http://lists.FreeBSD.org/mailman/listinfo/freebsd-announce).

For questions or bug reports about **CVSup** take a look at the CVSup FAQ (http://www.cvsup.org/faq.html#bugreports).

A.7.7 CVSup Sites

CVSup servers for FreeBSD are running at the following sites:

Central Servers, Primary Mirror Sites, Armenia, Australia, Austria, Brazil, Canada, China, Costa Rica, Czech Republic, Denmark, Estonia, Finland, France, Germany, Greece, Iceland, Ireland, Israel, Italy, Japan, Korea, Kuwait, Kyrgyzstan, Latvia, Lithuania, Netherlands, New Zealand, Norway, Philippines, Poland, Portugal, Romania, Russia, San Marino, Slovak Republic, Slovenia, South Africa, Spain, Sweden, Switzerland, Taiwan, Thailand, Turkey, Ukraine, United Kingdom, USA.

(as of UTC)

Central Servers

- cvsup.FreeBSD.org

Primary Mirror Sites

- cvsup1.FreeBSD.org
- cvsup2.FreeBSD.org
- cvsup3.FreeBSD.org
- cvsup4.FreeBSD.org
- cvsup5.FreeBSD.org
- cvsup6.FreeBSD.org
- cvsup7.FreeBSD.org
- cvsup8.FreeBSD.org
- cvsup9.FreeBSD.org
- cvsup10.FreeBSD.org
- cvsup11.FreeBSD.org
- cvsup12.FreeBSD.org
- cvsup13.FreeBSD.org

- cvsup14.FreeBSD.org
- cvsup15.FreeBSD.org
- cvsup16.FreeBSD.org
- cvsup18.FreeBSD.org

Armenia

- cvsup1.am.FreeBSD.org

Australia

- cvsup.au.FreeBSD.org

Austria

- cvsup.at.FreeBSD.org

Brazil

- cvsup.br.FreeBSD.org
- cvsup2.br.FreeBSD.org
- cvsup3.br.FreeBSD.org
- cvsup4.br.FreeBSD.org
- cvsup5.br.FreeBSD.org

Canada

- cvsup1.ca.FreeBSD.org

China

- cvsup.cn.FreeBSD.org
- cvsup2.cn.FreeBSD.org

Costa Rica

- cvsup1.cr.FreeBSD.org

Czech Republic

- cvsup.cz.FreeBSD.org

Denmark

- cvsup.dk.FreeBSD.org
- cvsup2.dk.FreeBSD.org

Estonia

- cvsup.ee.FreeBSD.org

Finland

- cvsup.fi.FreeBSD.org
- cvsup2.fi.FreeBSD.org

France

- cvsup.fr.FreeBSD.org
- cvsup1.fr.FreeBSD.org
- cvsup2.fr.FreeBSD.org
- cvsup3.fr.FreeBSD.org
- cvsup4.fr.FreeBSD.org
- cvsup5.fr.FreeBSD.org
- cvsup8.fr.FreeBSD.org

Germany

- cvsup.de.FreeBSD.org
- cvsup2.de.FreeBSD.org
- cvsup3.de.FreeBSD.org
- cvsup4.de.FreeBSD.org
- cvsup5.de.FreeBSD.org
- cvsup6.de.FreeBSD.org
- cvsup7.de.FreeBSD.org
- cvsup8.de.FreeBSD.org

Greece

- cvsup.gr.FreeBSD.org
- cvsup2.gr.FreeBSD.org

Iceland

- cvsup.is.FreeBSD.org

Ireland

- cvsup.ie.FreeBSD.org
- cvsup2.ie.FreeBSD.org

Israel

- cvsup.il.FreeBSD.org

Italy

- cvsup.it.FreeBSD.org

Japan

- cvsup.jp.FreeBSD.org
- cvsup2.jp.FreeBSD.org
- cvsup3.jp.FreeBSD.org
- cvsup4.jp.FreeBSD.org
- cvsup5.jp.FreeBSD.org
- cvsup6.jp.FreeBSD.org

Korea

- cvsup.kr.FreeBSD.org
- cvsup2.kr.FreeBSD.org
- cvsup3.kr.FreeBSD.org

Kuwait

- cvsup1.kw.FreeBSD.org

Kyrgyzstan

- cvsup.kg.FreeBSD.org

Latvia

- cvsup.lv.FreeBSD.org
- cvsup2.lv.FreeBSD.org

Lithuania

- cvsup.lt.FreeBSD.org
- cvsup2.lt.FreeBSD.org
- cvsup3.lt.FreeBSD.org

Netherlands

- cvsup.nl.FreeBSD.org
- cvsup2.nl.FreeBSD.org
- cvsup3.nl.FreeBSD.org

New Zealand

- cvsup.nz.FreeBSD.org

Norway

- cvsup.no.FreeBSD.org

Philippines

- cvsup1.ph.FreeBSD.org

Poland

- cvsup.pl.FreeBSD.org
- cvsup2.pl.FreeBSD.org
- cvsup3.pl.FreeBSD.org

Portugal

- cvsup.pt.FreeBSD.org
- cvsup2.pt.FreeBSD.org
- cvsup3.pt.FreeBSD.org

Romania

- cvsup.ro.FreeBSD.org
- cvsup1.ro.FreeBSD.org
- cvsup2.ro.FreeBSD.org
- cvsup3.ro.FreeBSD.org

Russia

- cvsup.ru.FreeBSD.org
- cvsup2.ru.FreeBSD.org
- cvsup3.ru.FreeBSD.org
- cvsup4.ru.FreeBSD.org
- cvsup5.ru.FreeBSD.org
- cvsup6.ru.FreeBSD.org
- cvsup7.ru.FreeBSD.org

San Marino

- cvsup.sm.FreeBSD.org

Slovak Republic

- cvsup.sk.FreeBSD.org

Slovenia

- cvsup.si.FreeBSD.org
- cvsup2.si.FreeBSD.org

South Africa

- cvsup.za.FreeBSD.org

- cvsup2.za.FreeBSD.org

Spain

- cvsup.es.FreeBSD.org
- cvsup2.es.FreeBSD.org
- cvsup3.es.FreeBSD.org

Sweden

- cvsup.se.FreeBSD.org
- cvsup2.se.FreeBSD.org

Switzerland

- cvsup.ch.FreeBSD.org

Taiwan

- cvsup.tw.FreeBSD.org
- cvsup3.tw.FreeBSD.org
- cvsup4.tw.FreeBSD.org
- cvsup5.tw.FreeBSD.org
- cvsup6.tw.FreeBSD.org
- cvsup7.tw.FreeBSD.org
- cvsup8.tw.FreeBSD.org
- cvsup9.tw.FreeBSD.org
- cvsup10.tw.FreeBSD.org
- cvsup11.tw.FreeBSD.org
- cvsup12.tw.FreeBSD.org
- cvsup13.tw.FreeBSD.org
- cvsup14.tw.FreeBSD.org

Thailand

- cvsup.th.FreeBSD.org

Turkey

- cvsup.tr.FreeBSD.org
- cvsup2.tr.FreeBSD.org

Ukraine

- cvsup3.ua.FreeBSD.org
- cvsup5.ua.FreeBSD.org
- cvsup6.ua.FreeBSD.org

United Kingdom

- cvsup.uk.FreeBSD.org
- cvsup2.uk.FreeBSD.org
- cvsup3.uk.FreeBSD.org
- cvsup4.uk.FreeBSD.org

USA

- cvsup1.us.FreeBSD.org
- cvsup2.us.FreeBSD.org
- cvsup3.us.FreeBSD.org
- cvsup4.us.FreeBSD.org
- cvsup5.us.FreeBSD.org
- cvsup6.us.FreeBSD.org
- cvsup7.us.FreeBSD.org
- cvsup8.us.FreeBSD.org
- cvsup9.us.FreeBSD.org
- cvsup10.us.FreeBSD.org

- cvsup11.us.FreeBSD.org
- cvsup12.us.FreeBSD.org
- cvsup13.us.FreeBSD.org
- cvsup14.us.FreeBSD.org
- cvsup15.us.FreeBSD.org
- cvsup16.us.FreeBSD.org
- cvsup18.us.FreeBSD.org

A.8 CVS Tags

Warning: CVS has been deprecated by the project, and its use is not recommended. **Subversion** should be used instead.

When obtaining or updating sources using **cvs** or **CVSup**, a revision tag must be specified. A revision tag refers to either a particular line of FreeBSD development, or a specific point in time. The first type are called “branch tags”, and the second type are called “release tags”.

A.8.1 Branch Tags

All of these, with the exception of `HEAD` (which is always a valid tag), only apply to the `src/` tree. The `ports/`, `doc/`, and `www/` trees are not branched.

HEAD

Symbolic name for the main line, or FreeBSD-CURRENT. Also the default when no revision is specified.

In **CVSup**, this tag is represented by a `.` (not punctuation, but a literal `.` character).

Note: In CVS, this is the default when no revision tag is specified. It is usually *not* a good idea to checkout or update to CURRENT sources on a STABLE machine, unless that is your intent.

RELENG_9

The line of development for FreeBSD-9.X, also known as FreeBSD 9-STABLE

RELENG_9_1

The release branch for FreeBSD-9.1, used only for security advisories and other critical fixes.

RELENG_9_0

The release branch for FreeBSD-9.0, used only for security advisories and other critical fixes.

RELENG_8

The line of development for FreeBSD-8.X, also known as FreeBSD 8-STABLE

RELENG_8_3

The release branch for FreeBSD-8.3, used only for security advisories and other critical fixes.

RELENG_8_2

The release branch for FreeBSD-8.2, used only for security advisories and other critical fixes.

RELENG_8_1

The release branch for FreeBSD-8.1, used only for security advisories and other critical fixes.

RELENG_8_0

The release branch for FreeBSD-8.0, used only for security advisories and other critical fixes.

RELENG_7

The line of development for FreeBSD-7.X, also known as FreeBSD 7-STABLE

RELENG_7_4

The release branch for FreeBSD-7.4, used only for security advisories and other critical fixes.

RELENG_7_3

The release branch for FreeBSD-7.3, used only for security advisories and other critical fixes.

RELENG_7_2

The release branch for FreeBSD-7.2, used only for security advisories and other critical fixes.

RELENG_7_1

The release branch for FreeBSD-7.1, used only for security advisories and other critical fixes.

RELENG_7_0

The release branch for FreeBSD-7.0, used only for security advisories and other critical fixes.

RELENG_6

The line of development for FreeBSD-6.X, also known as FreeBSD 6-STABLE

RELENG_6_4

The release branch for FreeBSD-6.4, used only for security advisories and other critical fixes.

RELENG_6_3

The release branch for FreeBSD-6.3, used only for security advisories and other critical fixes.

RELENG_6_2

The release branch for FreeBSD-6.2, used only for security advisories and other critical fixes.

RELENG_6_1

The release branch for FreeBSD-6.1, used only for security advisories and other critical fixes.

RELENG_6_0

The release branch for FreeBSD-6.0, used only for security advisories and other critical fixes.

RELENG_5

The line of development for FreeBSD-5.X, also known as FreeBSD 5-STABLE.

RELENG_5_5

The release branch for FreeBSD-5.5, used only for security advisories and other critical fixes.

RELENG_5_4

The release branch for FreeBSD-5.4, used only for security advisories and other critical fixes.

RELENG_5_3

The release branch for FreeBSD-5.3, used only for security advisories and other critical fixes.

RELENG_5_2

The release branch for FreeBSD-5.2 and FreeBSD-5.2.1, used only for security advisories and other critical fixes.

RELENG_5_1

The release branch for FreeBSD-5.1, used only for security advisories and other critical fixes.

RELENG_5_0

The release branch for FreeBSD-5.0, used only for security advisories and other critical fixes.

RELENG_4

The line of development for FreeBSD-4.X, also known as FreeBSD 4-STABLE.

RELENG_4_11

The release branch for FreeBSD-4.11, used only for security advisories and other critical fixes.

RELENG_4_10

The release branch for FreeBSD-4.10, used only for security advisories and other critical fixes.

RELENG_4_9

The release branch for FreeBSD-4.9, used only for security advisories and other critical fixes.

RELENG_4_8

The release branch for FreeBSD-4.8, used only for security advisories and other critical fixes.

RELENG_4_7

The release branch for FreeBSD-4.7, used only for security advisories and other critical fixes.

RELENG_4_6

The release branch for FreeBSD-4.6 and FreeBSD-4.6.2, used only for security advisories and other critical fixes.

RELENG_4_5

The release branch for FreeBSD-4.5, used only for security advisories and other critical fixes.

RELENG_4_4

The release branch for FreeBSD-4.4, used only for security advisories and other critical fixes.

RELENG_4_3

The release branch for FreeBSD-4.3, used only for security advisories and other critical fixes.

RELENG_3

The line of development for FreeBSD-3.X, also known as 3.X-STABLE.

RELENG_2_2

The line of development for FreeBSD-2.2.X, also known as 2.2-STABLE. This branch is mostly obsolete.

A.8.2 Release Tags

These tags refer to a specific point in time when a particular version of FreeBSD was released. The release engineering process is documented in more detail by the Release Engineering Information (http://www.FreeBSD.org/releng/) and Release Process (http://www.FreeBSD.org/doc/en_US.ISO8859-1/articles/releng/release-proc.html) documents. The `src` tree uses tag names that start with `RELENG_` tags. The `ports` and `doc` trees use tags whose names begin with `RELEASE` tags. Finally, the `www` tree is not tagged with any special name for releases.

RELENG_9_1_0_RELEASE

FreeBSD 9.1

RELENG_9_0_0_RELEASE

FreeBSD 9.0

RELENG_8_3_0_RELEASE

FreeBSD 8.3

RELENG_8_2_0_RELEASE

FreeBSD 8.2

RELENG_8_1_0_RELEASE

FreeBSD 8.1

RELENG_8_0_0_RELEASE
: FreeBSD 8.0

RELENG_7_4_0_RELEASE
: FreeBSD 7.4

RELENG_7_3_0_RELEASE
: FreeBSD 7.3

RELENG_7_2_0_RELEASE
: FreeBSD 7.2

RELENG_7_1_0_RELEASE
: FreeBSD 7.1

RELENG_7_0_0_RELEASE
: FreeBSD 7.0

RELENG_6_4_0_RELEASE
: FreeBSD 6.4

RELENG_6_3_0_RELEASE
: FreeBSD 6.3

RELENG_6_2_0_RELEASE
: FreeBSD 6.2

RELENG_6_1_0_RELEASE
: FreeBSD 6.1

RELENG_6_0_0_RELEASE
: FreeBSD 6.0

RELENG_5_5_0_RELEASE
: FreeBSD 5.5

RELENG_5_4_0_RELEASE
: FreeBSD 5.4

RELENG_4_11_0_RELEASE
: FreeBSD 4.11

RELENG_5_3_0_RELEASE
: FreeBSD 5.3

RELENG_4_10_0_RELEASE

: FreeBSD 4.10

RELENG_5_2_1_RELEASE

: FreeBSD 5.2.1

RELENG_5_2_0_RELEASE

: FreeBSD 5.2

RELENG_4_9_0_RELEASE

: FreeBSD 4.9

RELENG_5_1_0_RELEASE

: FreeBSD 5.1

RELENG_4_8_0_RELEASE

: FreeBSD 4.8

RELENG_5_0_0_RELEASE

: FreeBSD 5.0

RELENG_4_7_0_RELEASE

: FreeBSD 4.7

RELENG_4_6_2_RELEASE

: FreeBSD 4.6.2

RELENG_4_6_1_RELEASE

: FreeBSD 4.6.1

RELENG_4_6_0_RELEASE

: FreeBSD 4.6

RELENG_4_5_0_RELEASE

: FreeBSD 4.5

RELENG_4_4_0_RELEASE

: FreeBSD 4.4

RELENG_4_3_0_RELEASE

: FreeBSD 4.3

RELENG_4_2_0_RELEASE

: FreeBSD 4.2

RELENG_4_1_1_RELEASE
: FreeBSD 4.1.1

RELENG_4_1_0_RELEASE
: FreeBSD 4.1

RELENG_4_0_0_RELEASE
: FreeBSD 4.0

RELENG_3_5_0_RELEASE
: FreeBSD-3.5

RELENG_3_4_0_RELEASE
: FreeBSD-3.4

RELENG_3_3_0_RELEASE
: FreeBSD-3.3

RELENG_3_2_0_RELEASE
: FreeBSD-3.2

RELENG_3_1_0_RELEASE
: FreeBSD-3.1

RELENG_3_0_0_RELEASE
: FreeBSD-3.0

RELENG_2_2_8_RELEASE
: FreeBSD-2.2.8

RELENG_2_2_7_RELEASE
: FreeBSD-2.2.7

RELENG_2_2_6_RELEASE
: FreeBSD-2.2.6

RELENG_2_2_5_RELEASE
: FreeBSD-2.2.5

RELENG_2_2_2_RELEASE
: FreeBSD-2.2.2

RELENG_2_2_1_RELEASE
: FreeBSD-2.2.1

RELENG_2_2_0_RELEASE

FreeBSD-2.2.0

A.9 rsync Sites

The following sites make FreeBSD available through the rsync protocol. The **rsync** utility works in much the same way as the rcp(1) command, but has more options and uses the rsync remote-update protocol which transfers only the differences between two sets of files, thus greatly speeding up the synchronization over the network. This is most useful if you are a mirror site for the FreeBSD FTP server, or the CVS repository. The **rsync** suite is available for many operating systems, on FreeBSD, see the `net/rsync` port or use the package.

Czech Republic

rsync://ftp.cz.FreeBSD.org/

Available collections:

- ftp: A partial mirror of the FreeBSD FTP server.
- FreeBSD: A full mirror of the FreeBSD FTP server.

Netherlands

rsync://ftp.nl.FreeBSD.org/

Available collections:

- FreeBSD: A full mirror of the FreeBSD FTP server.

Russia

rsync://ftp.mtu.ru/

Available collections:

- FreeBSD: A full mirror of the FreeBSD FTP server.
- FreeBSD-gnats: The GNATS bug-tracking database.
- FreeBSD-Archive: The mirror of FreeBSD Archive FTP server.

Sweden

rsync://ftp4.se.freebsd.org/

Available collections:

- FreeBSD: A full mirror of the FreeBSD FTP server.

Taiwan

rsync://ftp.tw.FreeBSD.org/

rsync://ftp2.tw.FreeBSD.org/

rsync://ftp6.tw.FreeBSD.org/

Available collections:

- FreeBSD: A full mirror of the FreeBSD FTP server.

United Kingdom

rsync://rsync.mirrorservice.org/

Available collections:

- ftp.freebsd.org: A full mirror of the FreeBSD FTP server.

United States of America

rsync://ftp-master.FreeBSD.org/

This server may only be used by FreeBSD primary mirror sites.

Available collections:

- FreeBSD: The master archive of the FreeBSD FTP server.
- acl: The FreeBSD master ACL list.

rsync://ftp13.FreeBSD.org/

Available collections:

- FreeBSD: A full mirror of the FreeBSD FTP server.

Appendix B. Bibliography

While the manual pages provide the definitive reference for individual pieces of the FreeBSD operating system, they are notorious for not illustrating how to put the pieces together to make the whole operating system run smoothly. For this, there is no substitute for a good book on UNIX system administration and a good users' manual.

B.1 Books & Magazines Specific to FreeBSD

International books & Magazines:

- Using FreeBSD (http://jdli.tw.FreeBSD.org/publication/book/freebsd2/index.htm) (in Traditional Chinese), published by Drmaster (http://www.drmaster.com.tw/), 1997. ISBN 9-578-39435-7.
- FreeBSD Unleashed (Simplified Chinese translation), published by China Machine Press (http://www.hzbook.com/). ISBN 7-111-10201-0.
- FreeBSD From Scratch Second Edition (in Simplified Chinese), published by China Machine Press. ISBN 7-111-10286-X.
- FreeBSD Handbook Second Edition (Simplified Chinese translation), published by Posts & Telecom Press (http://www.ptpress.com.cn/). ISBN 7-115-10541-3.
- FreeBSD & Windows (in Simplified Chinese), published by China Railway Publishing House (http://www.tdpress.com/). ISBN 7-113-03845-X
- FreeBSD Internet Services HOWTO (in Simplified Chinese), published by China Railway Publishing House. ISBN 7-113-03423-3
- FreeBSD (in Japanese), published by CUTT. ISBN 4-906391-22-2 C3055 P2400E.
- Complete Introduction to FreeBSD (http://www.shoeisha.com/book/Detail.asp?bid=650) (in Japanese), published by Shoeisha Co., Ltd (http://www.shoeisha.co.jp/). ISBN 4-88135-473-6 P3600E.
- Personal UNIX Starter Kit FreeBSD (http://www.ascii.co.jp/pb/book1/shinkan/detail/1322785.html) (in Japanese), published by ASCII (http://www.ascii.co.jp/). ISBN 4-7561-1733-3 P3000E.
- FreeBSD Handbook (Japanese translation), published by ASCII (http://www.ascii.co.jp/). ISBN 4-7561-1580-2 P3800E.
- FreeBSD mit Methode (in German), published by Computer und Literatur Verlag (http://www.cul.de)/Vertrieb Hanser, 1998. ISBN 3-932311-31-0.
- FreeBSD de Luxe (http://www.mitp.de/vmi/mitp/detail/pWert/1343/) (in German), published by Verlag Modere Industrie (http://www.mitp.de), 2003. ISBN 3-8266-1343-0.
- FreeBSD Install and Utilization Manual (http://www.pc.mycom.co.jp/FreeBSD/install-manual.html) (in Japanese), published by Mainichi Communications Inc. (http://www.pc.mycom.co.jp/), 1998. ISBN 4-8399-0112-0.
- Onno W Purbo, Dodi Maryanto, Syahrial Hubbany, Widjil Widodo *Building Internet Server with FreeBSD (http://maxwell.itb.ac.id/)* (in Indonesia Language), published by Elex Media Komputindo (http://www.elexmedia.co.id/).
- Absolute BSD: The Ultimate Guide to FreeBSD (Traditional Chinese translation), published by GrandTech Press (http://www.grandtech.com.tw/), 2003. ISBN 986-7944-92-5.

- The FreeBSD 6.0 Book (http://www.twbsd.org/cht/book/) (in Traditional Chinese), published by Drmaster, 2006. ISBN 9-575-27878-X.

English language books & Magazines:

- Absolute FreeBSD, 2nd Edition: The Complete Guide to FreeBSD (http://www.absoluteFreeBSD.com/), published by No Starch Press (http://www.nostarch.com/), 2007. ISBN: 978-1-59327-151-0
- The Complete FreeBSD (http://www.freebsdmall.com/cgi-bin/fm/bsdcomp), published by O'Reilly (http://www.oreilly.com/), 2003. ISBN: 0596005164
- The FreeBSD Corporate Networker's Guide (http://www.freebsd-corp-net-guide.com/), published by Addison-Wesley (http://www.awl.com/aw/), 2000. ISBN: 0201704811
- FreeBSD: An Open-Source Operating System for Your Personal Computer (http://andrsn.stanford.edu/FreeBSD/introbook/), published by The Bit Tree Press, 2001. ISBN: 0971204500
- Teach Yourself FreeBSD in 24 Hours, published by Sams (http://www.samspublishing.com/), 2002. ISBN: 0672324245
- FreeBSD 6 Unleashed, published by Sams (http://www.samspublishing.com/), 2006. ISBN: 0672328755
- FreeBSD: The Complete Reference, published by McGrawHill (http://books.mcgraw-hill.com), 2003. ISBN: 0072224096
- BSD Magazine (http://www.bsdmag.org), published by Software Press Sp. z o.o. SK. ISSN 1898-9144

B.2 Users' Guides

- Ohio State University has written a UNIX Introductory Course (http://www.cs.duke.edu/csl/docs/unix_course/) which is available online in HTML and PostScript format.

  An Italian translation (http://www.FreeBSD.org/doc/it_IT.ISO8859-15/books/unix-introduction/index.html) of this document is available as part of the FreeBSD Italian Documentation Project.
- Jpman Project, Japan FreeBSD Users Group (http://www.jp.FreeBSD.org/). FreeBSD User's Reference Manual (http://www.pc.mycom.co.jp/FreeBSD/urm.html) (Japanese translation). Mainichi Communications Inc. (http://www.pc.mycom.co.jp/), 1998. ISBN4-8399-0088-4 P3800E.
- Edinburgh University (http://www.ed.ac.uk/) has written an Online Guide (http://unixhelp.ed.ac.uk/) for newcomers to the UNIX environment.

B.3 Administrators' Guides

- Jpman Project, Japan FreeBSD Users Group (http://www.jp.FreeBSD.org/). FreeBSD System Administrator's Manual (http://www.pc.mycom.co.jp/FreeBSD/sam.html) (Japanese translation). Mainichi Communications Inc. (http://www.pc.mycom.co.jp/), 1998. ISBN4-8399-0109-0 P3300E.
- Dreyfus, Emmanuel. Cahiers de l'Admin: BSD (http://www.eyrolles.com/Informatique/Livre/9782212114638/) 2nd Ed. (in French), Eyrolles, 2004. ISBN 2-212-11463-X

B.4 Programmers' Guides

- Computer Systems Research Group, UC Berkeley. *4.4BSD Programmer's Reference Manual*. O'Reilly & Associates, Inc., 1994. ISBN 1-56592-078-3
- Computer Systems Research Group, UC Berkeley. *4.4BSD Programmer's Supplementary Documents*. O'Reilly & Associates, Inc., 1994. ISBN 1-56592-079-1
- Harbison, Samuel P. and Steele, Guy L. Jr. *C: A Reference Manual*. 4th ed. Prentice Hall, 1995. ISBN 0-13-326224-3
- Kernighan, Brian and Dennis M. Ritchie. *The C Programming Language*. 2nd Ed. PTR Prentice Hall, 1988. ISBN 0-13-110362-8
- Lehey, Greg. *Porting UNIX Software*. O'Reilly & Associates, Inc., 1995. ISBN 1-56592-126-7
- Plauger, P. J. *The Standard C Library*. Prentice Hall, 1992. ISBN 0-13-131509-9
- Spinellis, Diomidis. *Code Reading: The Open Source Perspective* (http://www.spinellis.gr/codereading/). Addison-Wesley, 2003. ISBN 0-201-79940-5
- Spinellis, Diomidis. *Code Quality: The Open Source Perspective* (http://www.spinellis.gr/codequality/). Addison-Wesley, 2006. ISBN 0-321-16607-8
- Stevens, W. Richard and Stephen A. Rago. *Advanced Programming in the UNIX Environment*. 2nd Ed. Reading, Mass. : Addison-Wesley, 2005. ISBN 0-201-43307-9
- Stevens, W. Richard. *UNIX Network Programming*. 2nd Ed, PTR Prentice Hall, 1998. ISBN 0-13-490012-X

B.5 Operating System Internals

- Andleigh, Prabhat K. *UNIX System Architecture*. Prentice-Hall, Inc., 1990. ISBN 0-13-949843-5
- Jolitz, William. "Porting UNIX to the 386". *Dr. Dobb's Journal*. January 1991-July 1992.
- Leffler, Samuel J., Marshall Kirk McKusick, Michael J Karels and John Quarterman *The Design and Implementation of the 4.3BSD UNIX Operating System*. Reading, Mass. : Addison-Wesley, 1989. ISBN 0-201-06196-1
- Leffler, Samuel J., Marshall Kirk McKusick, *The Design and Implementation of the 4.3BSD UNIX Operating System: Answer Book*. Reading, Mass. : Addison-Wesley, 1991. ISBN 0-201-54629-9
- McKusick, Marshall Kirk, Keith Bostic, Michael J Karels, and John Quarterman. *The Design and Implementation of the 4.4BSD Operating System*. Reading, Mass. : Addison-Wesley, 1996. ISBN 0-201-54979-4

  (Chapter 2 of this book is available online (http://www.FreeBSD.org/doc/en_US.ISO8859-1/books/design-44bsd/book.html) as part of the FreeBSD Documentation Project.)
- Marshall Kirk McKusick, George V. Neville-Neil *The Design and Implementation of the FreeBSD Operating System*. Boston, Mass. : Addison-Wesley, 2004. ISBN 0-201-70245-2
- Stevens, W. Richard. *TCP/IP Illustrated, Volume 1: The Protocols*. Reading, Mass. : Addison-Wesley, 1996. ISBN 0-201-63346-9

- Schimmel, Curt. *Unix Systems for Modern Architectures*. Reading, Mass. : Addison-Wesley, 1994. ISBN 0-201-63338-8
- Stevens, W. Richard. *TCP/IP Illustrated, Volume 3: TCP for Transactions, HTTP, NNTP and the UNIX Domain Protocols*. Reading, Mass. : Addison-Wesley, 1996. ISBN 0-201-63495-3
- Vahalia, Uresh. *UNIX Internals -- The New Frontiers.* Prentice Hall, 1996. ISBN 0-13-101908-2
- Wright, Gary R. and W. Richard Stevens. *TCP/IP Illustrated, Volume 2: The Implementation*. Reading, Mass. : Addison-Wesley, 1995. ISBN 0-201-63354-X

B.6 Security Reference

- Cheswick, William R. and Steven M. Bellovin. *Firewalls and Internet Security: Repelling the Wily Hacker*. Reading, Mass. : Addison-Wesley, 1995. ISBN 0-201-63357-4
- Garfinkel, Simson. *PGP Pretty Good Privacy* O'Reilly & Associates, Inc., 1995. ISBN 1-56592-098-8

B.7 Hardware Reference

- Anderson, Don and Tom Shanley. *Pentium Processor System Architecture*. 2nd Ed. Reading, Mass. : Addison-Wesley, 1995. ISBN 0-201-40992-5
- Ferraro, Richard F. *Programmer's Guide to the EGA, VGA, and Super VGA Cards*. 3rd ed. Reading, Mass. : Addison-Wesley, 1995. ISBN 0-201-62490-7
- Intel Corporation publishes documentation on their CPUs, chipsets and standards on their developer web site (http://developer.intel.com/), usually as PDF files.
- Shanley, Tom. *80486 System Architecture*. 3rd ed. Reading, Mass. : Addison-Wesley, 1995. ISBN 0-201-40994-1
- Shanley, Tom. *ISA System Architecture*. 3rd ed. Reading, Mass. : Addison-Wesley, 1995. ISBN 0-201-40996-8
- Shanley, Tom. *PCI System Architecture*. 4th ed. Reading, Mass. : Addison-Wesley, 1999. ISBN 0-201-30974-2
- Van Gilluwe, Frank. *The Undocumented PC*, 2nd Ed. Reading, Mass: Addison-Wesley Pub. Co., 1996. ISBN 0-201-47950-8
- Messmer, Hans-Peter. *The Indispensable PC Hardware Book*, 4th Ed. Reading, Mass : Addison-Wesley Pub. Co., 2002. ISBN 0-201-59616-4

B.8 UNIX History

- Lion, John *Lion's Commentary on UNIX, 6th Ed. With Source Code*. ITP Media Group, 1996. ISBN 1573980137
- Raymond, Eric S. *The New Hacker's Dictionary, 3rd edition*. MIT Press, 1996. ISBN 0-262-68092-0. Also known as the Jargon File (http://www.catb.org/~esr/jargon/html/index.html)

- Salus, Peter H. *A quarter century of UNIX*. Addison-Wesley Publishing Company, Inc., 1994. ISBN 0-201-54777-5
- Simon Garfinkel, Daniel Weise, Steven Strassmann. *The UNIX-HATERS Handbook*. IDG Books Worldwide, Inc., 1994. ISBN 1-56884-203-1. Out of print, but available online (http://www.simson.net/ref/ugh.pdf).
- Don Libes, Sandy Ressler *Life with UNIX* — special edition. Prentice-Hall, Inc., 1989. ISBN 0-13-536657-7
- *The BSD family tree*. http://www.FreeBSD.org/cgi/cvsweb.cgi/src/share/misc/bsd-family-tree or `/usr/share/misc/bsd-family-tree` on a FreeBSD machine.
- *Networked Computer Science Technical Reports Library*. http://www.ncstrl.org/
- *Old BSD releases from the Computer Systems Research group (CSRG)*. http://www.mckusick.com/csrg/: The 4CD set covers all BSD versions from 1BSD to 4.4BSD and 4.4BSD-Lite2 (but not 2.11BSD, unfortunately). The last disk also holds the final sources plus the SCCS files.

B.9 Magazines and Journals

- *The C/C++ Users Journal*. R&D Publications Inc. ISSN 1075-2838
- *Sys Admin — The Journal for UNIX System Administrators* Miller Freeman, Inc., ISSN 1061-2688
- *freeX — Das Magazin für Linux - BSD - UNIX* (in German) Computer- und Literaturverlag GmbH, ISSN 1436-7033

Appendix C. Resources on the Internet

The rapid pace of FreeBSD progress makes print media impractical as a means of following the latest developments. Electronic resources are the best, if not often the only, way to stay informed of the latest advances. Since FreeBSD is a volunteer effort, the user community itself also generally serves as a "technical support department" of sorts, with electronic mail, web forums, and USENET news being the most effective way of reaching that community.

The most important points of contact with the FreeBSD user community are outlined below. Please send other resources not mentioned here to the FreeBSD documentation project mailing list (http://lists.FreeBSD.org/mailman/listinfo/freebsd-doc) so that they may also be included.

C.1 Mailing Lists

The mailing lists are the most direct way of addressing questions or opening a technical discussion to a concentrated FreeBSD audience. There are a wide variety of lists on a number of different FreeBSD topics. Sending questions to the most appropriate mailing list will invariably assure a faster and more accurate response.

The charters for the various lists are given at the bottom of this document. *Please read the charter before joining or sending mail to any list*. Most list subscribers receive many hundreds of FreeBSD related messages every day, and the charters and rules for use are meant to keep the signal-to-noise ratio of the lists high. To do less would see the mailing lists ultimately fail as an effective communications medium for the Project.

> **Note:** *To test the ability to send email to FreeBSD lists, send a test message to freebsd-test (http://lists.FreeBSD.org/mailman/listinfo/freebsd-test).* Please do not send test messages to any other list.

When in doubt about what list to post a question to, see How to get best results from the FreeBSD-questions mailing list (http://www.FreeBSD.org/doc/en_US.ISO8859-1/articles/freebsd-questions).

Before posting to any list, please learn about how to best use the mailing lists, such as how to help avoid frequently-repeated discussions, by reading the Mailing List Frequently Asked Questions (http://www.FreeBSD.org/doc/en_US.ISO8859-1/articles/mailing-list-faq) (FAQ) document.

Archives are kept for all of the mailing lists and can be searched using the FreeBSD World Wide Web server (http://www.FreeBSD.org/search/index.html). The keyword searchable archive offers an excellent way of finding answers to frequently asked questions and should be consulted before posting a question. Note that this also means that messages sent to FreeBSD mailing lists are archived in perpetuity. When protecting privacy is a concern, consider using a disposable secondary email address and posting only public information.

C.1.1 List Summary

General lists: The following are general lists which anyone is free (and encouraged) to join:

List	Purpose
freebsd-advocacy (http://lists.FreeBSD.org/mailman/listinfo/freebsd-advocacy)	FreeBSD Evangelism

List	Purpose
freebsd-announce (http://lists.FreeBSD.org/mailman/listinfo/freebsd-announce)	Important events and Project milestones (moderated)
freebsd-arch (http://lists.FreeBSD.org/mailman/listinfo/freebsd-arch)	Architecture and design discussions
freebsd-bugbusters (http://lists.FreeBSD.org/mailman/listinfo/freebsd-bugbusters)	Discussions pertaining to the maintenance of the FreeBSD problem report database and related tools
freebsd-bugs (http://lists.FreeBSD.org/mailman/listinfo/freebsd-bugs)	Bug reports
freebsd-chat (http://lists.FreeBSD.org/mailman/listinfo/freebsd-chat)	Non-technical items related to the FreeBSD community
freebsd-chromium (http://lists.FreeBSD.org/mailman/listinfo/freebsd-chromium)	FreeBSD-specific Chromium issues
freebsd-current (http://lists.FreeBSD.org/mailman/listinfo/freebsd-current)	Discussion concerning the use of FreeBSD-CURRENT
freebsd-isp (http://lists.FreeBSD.org/mailman/listinfo/freebsd-isp)	Issues for Internet Service Providers using FreeBSD
freebsd-jobs (http://lists.FreeBSD.org/mailman/listinfo/freebsd-jobs)	FreeBSD employment and consulting opportunities
freebsd-questions (http://lists.FreeBSD.org/mailman/listinfo/freebsd-questions)	User questions and technical support
freebsd-security-notifications (http://lists.FreeBSD.org/mailman/listinfo/freebsd-security-notifications)	Security notifications (moderated)
freebsd-stable (http://lists.FreeBSD.org/mailman/listinfo/freebsd-stable)	Discussion concerning the use of FreeBSD-STABLE
freebsd-test (http://lists.FreeBSD.org/mailman/listinfo/freebsd-test)	Where to send test messages instead of to one of the actual lists

Technical lists: The following lists are for technical discussion. Read the charter for each list carefully before joining or sending mail to one as there are firm guidelines for their use and content.

List	Purpose
freebsd-acpi (http://lists.FreeBSD.org/mailman/listinfo/freebsd-acpi)	ACPI and power management development
freebsd-afs (http://lists.FreeBSD.org/mailman/listinfo/freebsd-afs)	Porting AFS to FreeBSD
freebsd-aic7xxx (http://lists.FreeBSD.org/mailman/listinfo/aic7xxx)	Developing drivers for the Adaptec AIC 7xxx
freebsd-amd64 (http://lists.FreeBSD.org/mailman/listinfo/freebsd-amd64)	Porting FreeBSD to AMD64 systems (moderated)
freebsd-apache (http://lists.FreeBSD.org/mailman/listinfo/freebsd-apache)	Discussion about **Apache** related ports
freebsd-arm (http://lists.FreeBSD.org/mailman/listinfo/freebsd-arm)	Porting FreeBSD to ARM® processors
freebsd-atm (http://lists.FreeBSD.org/mailman/listinfo/freebsd-atm)	Using ATM networking with FreeBSD
freebsd-bluetooth (http://lists.FreeBSD.org/mailman/listinfo/freebsd-bluetooth)	Using Bluetooth technology in FreeBSD
freebsd-cluster (http://lists.FreeBSD.org/mailman/listinfo/freebsd-cluster)	Using FreeBSD in a clustered environment
freebsd-cvsweb (http://lists.FreeBSD.org/mailman/listinfo/freebsd-cvsweb)	CVSweb maintenance
freebsd-database (http://lists.FreeBSD.org/mailman/listinfo/freebsd-database)	Discussing database use and development under FreeBSD
freebsd-desktop (http://lists.FreeBSD.org/mailman/listinfo/freebsd-desktop)	Using and improving FreeBSD on the desktop
freebsd-doc (http://lists.FreeBSD.org/mailman/listinfo/freebsd-doc)	Creating FreeBSD related documents
freebsd-drivers (http://lists.FreeBSD.org/mailman/listinfo/freebsd-drivers)	Writing device drivers for FreeBSD

List	Purpose
freebsd-dtrace (http://lists.FreeBSD.org/mailman/listinfo/freebsd-dtrace)	Using and working on DTrace in FreeBSD
freebsd-eclipse (http://lists.FreeBSD.org/mailman/listinfo/freebsd-eclipse)	FreeBSD users of Eclipse IDE, tools, rich client applications and ports.
freebsd-embedded (http://lists.FreeBSD.org/mailman/listinfo/freebsd-embedded)	Using FreeBSD in embedded applications
freebsd-eol (http://lists.FreeBSD.org/mailman/listinfo/freebsd-eol)	Peer support of FreeBSD-related software that is no longer supported by the FreeBSD Project.
freebsd-emulation (http://lists.FreeBSD.org/mailman/listinfo/freebsd-emulation)	Emulation of other systems such as Linux/MS-DOS/Windows
freebsd-firewire (http://lists.FreeBSD.org/mailman/listinfo/freebsd-firewire)	FreeBSD FireWire (iLink, IEEE 1394) technical discussion
freebsd-fs (http://lists.FreeBSD.org/mailman/listinfo/freebsd-fs)	File systems
freebsd-gecko (http://lists.FreeBSD.org/mailman/listinfo/freebsd-gecko)	**Gecko Rendering Engine** issues
freebsd-geom (http://lists.FreeBSD.org/mailman/listinfo/freebsd-geom)	GEOM-specific discussions and implementations
freebsd-gnome (http://lists.FreeBSD.org/mailman/listinfo/freebsd-gnome)	Porting **GNOME** and **GNOME** applications
freebsd-hackers (http://lists.FreeBSD.org/mailman/listinfo/freebsd-hackers)	General technical discussion
freebsd-hardware (http://lists.FreeBSD.org/mailman/listinfo/freebsd-hardware)	General discussion of hardware for running FreeBSD

List	Purpose
freebsd-i18n (http://lists.FreeBSD.org/mailman/listinfo/freebsd-i18n)	FreeBSD Internationalization
freebsd-ia32 (http://lists.FreeBSD.org/mailman/listinfo/freebsd-ia32)	FreeBSD on the IA-32 (Intel x86) platform
freebsd-ia64 (http://lists.FreeBSD.org/mailman/listinfo/freebsd-ia64)	Porting FreeBSD to Intel's upcoming IA64 systems
freebsd-infiniband (http://lists.FreeBSD.org/mailman/listinfo/freebsd-infiniband)	Infiniband on FreeBSD
freebsd-ipfw (http://lists.FreeBSD.org/mailman/listinfo/freebsd-ipfw)	Technical discussion concerning the redesign of the IP firewall code
freebsd-isdn (http://lists.FreeBSD.org/mailman/listinfo/freebsd-isdn)	ISDN developers
freebsd-jail (http://lists.FreeBSD.org/mailman/listinfo/freebsd-jail)	Discussion about the jail(8) facility
freebsd-java (http://lists.FreeBSD.org/mailman/listinfo/freebsd-java)	Java developers and people porting JDKs to FreeBSD
freebsd-lfs (http://lists.FreeBSD.org/mailman/listinfo/freebsd-lfs)	Porting LFS to FreeBSD
freebsd-mips (http://lists.FreeBSD.org/mailman/listinfo/freebsd-mips)	Porting FreeBSD to MIPS®
freebsd-mobile (http://lists.FreeBSD.org/mailman/listinfo/freebsd-mobile)	Discussions about mobile computing
freebsd-mono (http://lists.FreeBSD.org/mailman/listinfo/freebsd-mono)	Mono and C# applications on FreeBSD
freebsd-mozilla (http://lists.FreeBSD.org/mailman/listinfo/freebsd-mozilla)	Porting **Mozilla** to FreeBSD
freebsd-multimedia (http://lists.FreeBSD.org/mailman/listinfo/freebsd-multimedia)	Multimedia applications
freebsd-new-bus (http://lists.FreeBSD.org/mailman/listinfo/freebsd-new-bus)	Technical discussions about bus architecture

List	Purpose
freebsd-net (http://lists.FreeBSD.org/mailman/listinfo/freebsd-net)	Networking discussion and TCP/IP source code
freebsd-numerics (http://lists.FreeBSD.org/mailman/listinfo/freebsd-numerics)	Discussions of high quality implementation of libm functions
freebsd-office (http://lists.FreeBSD.org/mailman/listinfo/freebsd-office)	Office applications on FreeBSD
freebsd-performance (http://lists.FreeBSD.org/mailman/listinfo/freebsd-performance)	Performance tuning questions for high performance/load installations
freebsd-perl (http://lists.FreeBSD.org/mailman/listinfo/freebsd-perl)	Maintenance of a number of Perl-related ports
freebsd-pf (http://lists.FreeBSD.org/mailman/listinfo/freebsd-pf)	Discussion and questions about the packet filter firewall system
freebsd-platforms (http://lists.FreeBSD.org/mailman/listinfo/freebsd-platforms)	Concerning ports to non Intel architecture platforms
freebsd-ports (http://lists.FreeBSD.org/mailman/listinfo/freebsd-ports)	Discussion of the Ports Collection
freebsd-ports-announce (http://lists.FreeBSD.org/mailman/listinfo/freebsd-ports-announce)	Important news and instructions about the Ports Collection (moderated)
freebsd-ports-bugs (http://lists.FreeBSD.org/mailman/listinfo/freebsd-ports-bugs)	Discussion of the ports bugs/PRs
freebsd-ppc (http://lists.FreeBSD.org/mailman/listinfo/freebsd-ppc)	Porting FreeBSD to the PowerPC
freebsd-proliant (http://lists.FreeBSD.org/mailman/listinfo/freebsd-proliant)	Technical discussion of FreeBSD on HP ProLiant server platforms
freebsd-python (http://lists.FreeBSD.org/mailman/listinfo/freebsd-python)	FreeBSD-specific Python issues
freebsd-rc (http://lists.FreeBSD.org/mailman/listinfo/freebsd-rc)	Discussion related to the `rc.d` system and its development

List	Purpose
freebsd-realtime (http://lists.FreeBSD.org/mailman/listinfo/freebsd-realtime)	Development of realtime extensions to FreeBSD
freebsd-ruby (http://lists.FreeBSD.org/mailman/listinfo/freebsd-ruby)	FreeBSD-specific Ruby discussions
freebsd-scsi (http://lists.FreeBSD.org/mailman/listinfo/freebsd-scsi)	The SCSI subsystem
freebsd-security (http://lists.FreeBSD.org/mailman/listinfo/freebsd-security)	Security issues affecting FreeBSD
freebsd-small (http://lists.FreeBSD.org/mailman/listinfo/freebsd-small)	Using FreeBSD in embedded applications (obsolete; use freebsd-embedded (http://lists.FreeBSD.org/mailman/listinfo/freebsd-embedded) instead)
freebsd-snapshots (http://lists.FreeBSD.org/mailman/listinfo/freebsd-snapshots)	FreeBSD Development Snapshot Announcements
freebsd-sparc64 (http://lists.FreeBSD.org/mailman/listinfo/freebsd-sparc64)	Porting FreeBSD to SPARC® based systems
freebsd-standards (http://lists.FreeBSD.org/mailman/listinfo/freebsd-standards)	FreeBSD's conformance to the C99 and the POSIX standards
freebsd-sysinstall (http://lists.FreeBSD.org/mailman/listinfo/freebsd-sysinstall)	sysinstall(8) development
freebsd-tcltk (http://lists.FreeBSD.org/mailman/listinfo/freebsd-tcltk)	FreeBSD-specific Tcl/Tk discussions
freebsd-testing (http://lists.FreeBSD.org/mailman/listinfo/freebsd-testing)	Testing on FreeBSD
freebsd-tex (http://lists.FreeBSD.org/mailman/listinfo/freebsd-tex)	Porting TEX and its applications to FreeBSD
freebsd-threads (http://lists.FreeBSD.org/mailman/listinfo/freebsd-threads)	Threading in FreeBSD

List	Purpose
freebsd-tilera (http://lists.FreeBSD.org/mailman/listinfo/freebsd-tilera)	Porting FreeBSD to the Tilera family of CPUs
freebsd-tokenring (http://lists.FreeBSD.org/mailman/listinfo/freebsd-tokenring)	Support Token Ring in FreeBSD
freebsd-toolchain (http://lists.FreeBSD.org/mailman/listinfo/freebsd-toolchain)	Maintenance of FreeBSD's integrated toolchain
freebsd-usb (http://lists.FreeBSD.org/mailman/listinfo/freebsd-usb)	Discussing FreeBSD support for USB
freebsd-virtualization (http://lists.FreeBSD.org/mailman/listinfo/freebsd-virtualization)	Discussion of various virtualization techniques supported by FreeBSD
freebsd-vuxml (http://lists.FreeBSD.org/mailman/listinfo/freebsd-vuxml)	Discussion on VuXML infrastructure
freebsd-x11 (http://lists.FreeBSD.org/mailman/listinfo/freebsd-x11)	Maintenance and support of X11 on FreeBSD
freebsd-xen (http://lists.FreeBSD.org/mailman/listinfo/freebsd-xen)	Discussion of the FreeBSD port to Xen™ — implementation and usage
freebsd-xfce (http://lists.FreeBSD.org/mailman/listinfo/freebsd-xfce)	**XFCE** for FreeBSD — porting and maintaining
freebsd-zope (http://lists.FreeBSD.org/mailman/listinfo/freebsd-zope)	**Zope** for FreeBSD — porting and maintaining

Limited lists: The following lists are for more specialized (and demanding) audiences and are probably not of interest to the general public. It is also a good idea to establish a presence in the technical lists before joining one of these limited lists in order to understand the communications etiquette involved.

List	Purpose
freebsd-hubs (http://lists.FreeBSD.org/mailman/listinfo/freebsd-hubs)	People running mirror sites (infrastructural support)
freebsd-user-groups (http://lists.FreeBSD.org/mailman/listinfo/freebsd-user-groups)	User group coordination
freebsd-wip-status (http://lists.FreeBSD.org/mailman/listinfo/freebsd-wip-status)	FreeBSD Work-In-Progress Status

List	Purpose
freebsd-wireless (http://lists.FreeBSD.org/mailman/listinfo/freebsd-wireless)	Discussions of 802.11 stack, tools, device driver development

Digest lists: All of the above lists are available in a digest format. Once subscribed to a list, the digest options can be changed in the account options section.

SVN lists: The following lists are for people interested in seeing the log messages for changes to various areas of the source tree. They are *Read-Only* lists and should not have mail sent to them.

List	Source area	Area Description (source for)
svn-doc-all (http://lists.FreeBSD.org/mailman/listinfo/svn-doc-all)	`/usr/doc`	All changes to the doc Subversion repository (except for `user`, `projects` and `translations`)
svn-doc-head (http://lists.FreeBSD.org/mailman/listinfo/svn-doc-head)	`/usr/doc`	All changes to the “head” branch of the doc Subversion repository
svn-doc-projects (http://lists.FreeBSD.org/mailman/listinfo/svn-doc-projects)	`/usr/doc/projects`	All changes to the `projects` area of the doc Subversion repository
svn-doc-svnadmin (http://lists.FreeBSD.org/mailman/listinfo/svn-doc-svnadmin)	`/usr/doc`	All changes to the administrative scripts, hooks, and other configuration data of the doc Subversion repository
svn-ports-all (http://lists.FreeBSD.org/mailman/listinfo/svn-ports-all)	`/usr/ports`	All changes to the ports Subversion repository
svn-ports-head (http://lists.FreeBSD.org/mailman/listinfo/svn-ports-head)	`/usr/ports`	All changes to the “head” branch of the ports Subversion repository
svn-ports-svnadmin (http://lists.FreeBSD.org/mailman/listinfo/svn-ports-svnadmin)	`/usr/ports`	All changes to the administrative scripts, hooks, and other configuration data of the ports Subversion repository
svn-src-all (http://lists.FreeBSD.org/mailman/listinfo/svn-src-all)	`/usr/src`	All changes to the src Subversion repository (except for `user` and `projects`)

List	Source area	Area Description (source for)
svn-src-head (http://lists.FreeBSD.org/mailman/listinfo/svn-src-head)	`/usr/src`	All changes to the "head" branch of the src Subversion repository (the FreeBSD-CURRENT branch)
svn-src-projects (http://lists.FreeBSD.org/mailman/listinfo/svn-src-projects)	`/usr/projects`	All changes to the `projects` area of the src Subversion repository
svn-src-release (http://lists.FreeBSD.org/mailman/listinfo/svn-src-release)	`/usr/src`	All changes to the `releases` area of the src Subversion repository
svn-src-releng (http://lists.FreeBSD.org/mailman/listinfo/svn-src-releng)	`/usr/src`	All changes to the `releng` branches of the src Subversion repository (the security / release engineering branches)
svn-src-stable (http://lists.FreeBSD.org/mailman/listinfo/svn-src-stable)	`/usr/src`	All changes to the all stable branches of the src Subversion repository
svn-src-stable-6 (http://lists.FreeBSD.org/mailman/listinfo/svn-src-stable-6)	`/usr/src`	All changes to the `stable/6` branch of the src Subversion repository
svn-src-stable-7 (http://lists.FreeBSD.org/mailman/listinfo/svn-src-stable-7)	`/usr/src`	All changes to the `stable/7` branch of the src Subversion repository
svn-src-stable-8 (http://lists.FreeBSD.org/mailman/listinfo/svn-src-stable-8)	`/usr/src`	All changes to the `stable/8` branch of the src Subversion repository
svn-src-stable-9 (http://lists.FreeBSD.org/mailman/listinfo/svn-src-stable-9)	`/usr/src`	All changes to the `stable/9` branch of the src Subversion repository
svn-src-stable-other (http://lists.FreeBSD.org/mailman/listinfo/svn-src-stable-other)	`/usr/src`	All changes to the older `stable` branches of the src Subversion repository
svn-src-svnadmin (http://lists.FreeBSD.org/mailman/listinfo/svn-src-svnadmin)	`/usr/src`	All changes to the administrative scripts, hooks, and other configuration data of the src Subversion repository

List	Source area	Area Description (source for)
svn-src-user (http://lists.FreeBSD.org/mailman/listinfo/svn-src-user)	`/usr/src`	All changes to the experimental `user` area of the src Subversion repository
svn-src-vendor (http://lists.FreeBSD.org/mailman/listinfo/svn-src-vendor)	`/usr/src`	All changes to the vendor work area of the src Subversion repository

C.1.2 How to Subscribe

To subscribe to a list, click the list name at http://lists.FreeBSD.org/mailman/listinfo. The page that is displayed should contain all of the necessary subscription instructions for that list.

To actually post to a given list, send mail to <*listname*`@FreeBSD.org`>. It will then be redistributed to mailing list members world-wide.

To unsubscribe from a list, click on the URL found at the bottom of every email received from the list. It is also possible to send an email to <*listname*`-unsubscribe@FreeBSD.org`> to unsubscribe.

It is important to keep discussion in the technical mailing lists on a technical track. To only receive important announcements, instead join the FreeBSD announcements mailing list (http://lists.FreeBSD.org/mailman/listinfo/freebsd-announce), which is intended for infrequent traffic.

C.1.3 List Charters

All FreeBSD mailing lists have certain basic rules which must be adhered to by anyone using them. Failure to comply with these guidelines will result in two (2) written warnings from the FreeBSD Postmaster <`postmaster@FreeBSD.org`>, after which, on a third offense, the poster will removed from all FreeBSD mailing lists and filtered from further posting to them. We regret that such rules and measures are necessary at all, but today's Internet is a pretty harsh environment, it would seem, and many fail to appreciate just how fragile some of its mechanisms are.

Rules of the road:

- The topic of any posting should adhere to the basic charter of the list it is posted to. If the list is about technical issues, the posting should contain technical discussion. Ongoing irrelevant chatter or flaming only detracts from the value of the mailing list for everyone on it and will not be tolerated. For free-form discussion on no particular topic, the FreeBSD chat mailing list (http://lists.FreeBSD.org/mailman/listinfo/freebsd-chat) is freely available and should be used instead.
- No posting should be made to more than 2 mailing lists, and only to 2 when a clear and obvious need to post to both lists exists. For most lists, there is already a great deal of subscriber overlap and except for the most esoteric mixes (say "-stable & -scsi"), there really is no reason to post to more than one list at a time. If a message is received with multiple mailing lists on the `Cc` line, trim the `Cc` line before replying. *The person who replies is still responsible for cross-posting, no matter who the originator might have been.*

- Personal attacks and profanity (in the context of an argument) are not allowed, and that includes users and developers alike. Gross breaches of netiquette, like excerpting or reposting private mail when permission to do so was not and would not be forthcoming, are frowned upon but not specifically enforced. *However*, there are also very few cases where such content would fit within the charter of a list and it would therefore probably rate a warning (or ban) on that basis alone.
- Advertising of non-FreeBSD related products or services is strictly prohibited and will result in an immediate ban if it is clear that the offender is advertising by spam.

Individual list charters:

freebsd-acpi (http://lists.FreeBSD.org/mailman/listinfo/freebsd-acpi)

ACPI and power management development

freebsd-afs (http://lists.FreeBSD.org/mailman/listinfo/freebsd-afs)

Andrew File System

This list is for discussion on porting and using AFS from CMU/Transarc

freebsd-announce (http://lists.FreeBSD.org/mailman/listinfo/freebsd-announce)

Important events / milestones

This is the mailing list for people interested only in occasional announcements of significant FreeBSD events. This includes announcements about snapshots and other releases. It contains announcements of new FreeBSD capabilities. It may contain calls for volunteers etc. This is a low volume, strictly moderated mailing list.

freebsd-arch (http://lists.FreeBSD.org/mailman/listinfo/freebsd-arch)

Architecture and design discussions

This list is for discussion of the FreeBSD architecture. Messages will mostly be kept strictly technical in nature. Examples of suitable topics are:

- How to re-vamp the build system to have several customized builds running at the same time.
- What needs to be fixed with VFS to make Heidemann layers work.
- How do we change the device driver interface to be able to use the same drivers cleanly on many buses and architectures.
- How to write a network driver.

freebsd-bluetooth (http://lists.FreeBSD.org/mailman/listinfo/freebsd-bluetooth)

Bluetooth in FreeBSD

This is the forum where FreeBSD's Bluetooth users congregate. Design issues, implementation details, patches, bug reports, status reports, feature requests, and all matters related to Bluetooth are fair game.

freebsd-bugbusters (http://lists.FreeBSD.org/mailman/listinfo/freebsd-bugbusters)

Coordination of the Problem Report handling effort

The purpose of this list is to serve as a coordination and discussion forum for the Bugmeister, his Bugbusters, and any other parties who have a genuine interest in the PR database. This list is not for discussions about specific bugs, patches or PRs.

freebsd-bugs (http://lists.FreeBSD.org/mailman/listinfo/freebsd-bugs)

Bug reports

This is the mailing list for reporting bugs in FreeBSD. Whenever possible, bugs should be submitted using the send-pr(1) command or the WEB interface (http://www.FreeBSD.org/send-pr.html) to it.

freebsd-chat (http://lists.FreeBSD.org/mailman/listinfo/freebsd-chat)

Non technical items related to the FreeBSD community

This list contains the overflow from the other lists about non-technical, social information. It includes discussion about whether Jordan looks like a toon ferret or not, whether or not to type in capitals, who is drinking too much coffee, where the best beer is brewed, who is brewing beer in their basement, and so on. Occasional announcements of important events (such as upcoming parties, weddings, births, new jobs, etc) can be made to the technical lists, but the follow ups should be directed to this -chat list.

freebsd-chromium (http://lists.FreeBSD.org/mailman/listinfo/freebsd-chromium)

FreeBSD-specific Chromium issues

This is a list for the discussion of Chromium support for FreeBSD. This is a technical list to discuss development and installation of Chromium.

freebsd-core

FreeBSD core team

This is an internal mailing list for use by the core members. Messages can be sent to it when a serious FreeBSD-related matter requires arbitration or high-level scrutiny.

freebsd-current (http://lists.FreeBSD.org/mailman/listinfo/freebsd-current)

Discussions about the use of FreeBSD-CURRENT

This is the mailing list for users of FreeBSD-CURRENT. It includes warnings about new features coming out in -CURRENT that will affect the users, and instructions on steps that must be taken to remain -CURRENT. Anyone running “CURRENT” must subscribe to this list. This is a technical mailing list for which strictly technical content is expected.

freebsd-cvsweb (http://lists.FreeBSD.org/mailman/listinfo/freebsd-cvsweb)

FreeBSD CVSweb Project

Technical discussions about use, development and maintenance of FreeBSD-CVSweb.

freebsd-desktop (http://lists.FreeBSD.org/mailman/listinfo/freebsd-desktop)

Using and improving FreeBSD on the desktop

This is a forum for discussion of FreeBSD on the desktop. It is primarily a place for desktop porters and users to discuss issues and improve FreeBSD’s desktop support.

freebsd-doc (http://lists.FreeBSD.org/mailman/listinfo/freebsd-doc)

Documentation Project

This mailing list is for the discussion of issues and projects related to the creation of documentation for FreeBSD. The members of this mailing list are collectively referred to as "The FreeBSD Documentation Project". It is an open list; feel free to join and contribute!

freebsd-drivers (http://lists.FreeBSD.org/mailman/listinfo/freebsd-drivers)

Writing device drivers for FreeBSD

This is a forum for technical discussions related to device drivers on FreeBSD. It is primarily a place for device driver writers to ask questions about how to write device drivers using the APIs in the FreeBSD kernel.

freebsd-dtrace (http://lists.FreeBSD.org/mailman/listinfo/freebsd-dtrace)

Using and working on DTrace in FreeBSD

DTrace is an integrated component of FreeBSD that provides a framework for understanding the kernel as well as user space programs at run time. The mailing list is an archived discussion for developers of the code as well as those using it.

freebsd-eclipse (http://lists.FreeBSD.org/mailman/listinfo/freebsd-eclipse)

FreeBSD users of Eclipse IDE, tools, rich client applications and ports.

The intention of this list is to provide mutual support for everything to do with choosing, installing, using, developing and maintaining the Eclipse IDE, tools, rich client applications on the FreeBSD platform and assisting with the porting of Eclipse IDE and plugins to the FreeBSD environment.

The intention is also to facilitate exchange of information between the Eclipse community and the FreeBSD community to the mutual benefit of both.

Although this list is focused primarily on the needs of Eclipse users it will also provide a forum for those who would like to develop FreeBSD specific applications using the Eclipse framework.

freebsd-embedded (http://lists.FreeBSD.org/mailman/listinfo/freebsd-embedded)

Using FreeBSD in embedded applications

This list discusses topics related to using FreeBSD in embedded systems. This is a technical mailing list for which strictly technical content is expected. For the purpose of this list, embedded systems are those computing devices which are not desktops and which usually serve a single purpose as opposed to being general computing environments. Examples include, but are not limited to, all kinds of phone handsets, network equipment such as routers, switches and PBXs, remote measuring equipment, PDAs, Point Of Sale systems, and so on.

freebsd-emulation (http://lists.FreeBSD.org/mailman/listinfo/freebsd-emulation)

Emulation of other systems such as Linux/MS-DOS/Windows

This is a forum for technical discussions related to running programs written for other operating systems on FreeBSD.

freebsd-eol (http://lists.FreeBSD.org/mailman/listinfo/freebsd-eol)

Peer support of FreeBSD-related software that is no longer supported by the FreeBSD Project.

This list is for those interested in providing or making use of peer support of FreeBSD-related software for which the FreeBSD Project no longer provides official support in the form of security advisories and patches.

freebsd-firewire (http://lists.FreeBSD.org/mailman/listinfo/freebsd-firewire)

FireWire (iLink, IEEE 1394)

This is a mailing list for discussion of the design and implementation of a FireWire (aka IEEE 1394 aka iLink) subsystem for FreeBSD. Relevant topics specifically include the standards, bus devices and their protocols, adapter boards/cards/chips sets, and the architecture and implementation of code for their proper support.

freebsd-fs (http://lists.FreeBSD.org/mailman/listinfo/freebsd-fs)

File systems

Discussions concerning FreeBSD filesystems. This is a technical mailing list for which strictly technical content is expected.

freebsd-gecko (http://lists.FreeBSD.org/mailman/listinfo/freebsd-gecko)

Gecko Rendering Engine

This is a forum about **Gecko** applications using FreeBSD.

Discussion centers around Gecko Ports applications, their installation, their development and their support within FreeBSD.

freebsd-geom (http://lists.FreeBSD.org/mailman/listinfo/freebsd-geom)

GEOM

Discussions specific to GEOM and related implementations. This is a technical mailing list for which strictly technical content is expected.

freebsd-gnome (http://lists.FreeBSD.org/mailman/listinfo/freebsd-gnome)

GNOME

Discussions concerning The **GNOME** Desktop Environment for FreeBSD systems. This is a technical mailing list for which strictly technical content is expected.

freebsd-infiniband (http://lists.FreeBSD.org/mailman/listinfo/freebsd-infiniband)

Infiniband on FreeBSD

Technical mailing list discussing Infiniband, OFED, and OpenSM on FreeBSD.

freebsd-ipfw (http://lists.FreeBSD.org/mailman/listinfo/freebsd-ipfw)

IP Firewall

This is the forum for technical discussions concerning the redesign of the IP firewall code in FreeBSD. This is a technical mailing list for which strictly technical content is expected.

freebsd-ia64 (http://lists.FreeBSD.org/mailman/listinfo/freebsd-ia64)

Porting FreeBSD to IA64

This is a technical mailing list for individuals actively working on porting FreeBSD to the IA-64 platform from Intel, to bring up problems or discuss alternative solutions. Individuals interested in following the technical discussion are also welcome.

freebsd-isdn (http://lists.FreeBSD.org/mailman/listinfo/freebsd-isdn)

ISDN Communications

This is the mailing list for people discussing the development of ISDN support for FreeBSD.

freebsd-java (http://lists.FreeBSD.org/mailman/listinfo/freebsd-java)

Java Development

This is the mailing list for people discussing the development of significant Java applications for FreeBSD and the porting and maintenance of JDKs.

freebsd-jobs (http://lists.FreeBSD.org/mailman/listinfo/freebsd-jobs)

Jobs offered and sought

This is a forum for posting employment notices specifically related to FreeBSD and resumes from those seeking FreeBSD-related employment. This is *not* a mailing list for general employment issues since adequate forums for that already exist elsewhere.

Note that this list, like other `FreeBSD.org` mailing lists, is distributed worldwide. Be clear about the geographic location and the extent to which telecommuting or assistance with relocation is available.

Email should use open formats only — preferably plain text, but basic Portable Document Format (PDF), HTML, and a few others are acceptable to many readers. Closed formats such as Microsoft Word (`.doc`) will be rejected by the mailing list server.

freebsd-kde (https://mail.kde.org/mailman/listinfo/kde-freebsd)

KDE

Discussions concerning **KDE** on FreeBSD systems. This is a technical mailing list for which strictly technical content is expected.

freebsd-hackers (http://lists.FreeBSD.org/mailman/listinfo/freebsd-hackers)

Technical discussions

This is a forum for technical discussions related to FreeBSD. This is the primary technical mailing list. It is for individuals actively working on FreeBSD, to bring up problems or discuss alternative solutions. Individuals interested in following the technical discussion are also welcome. This is a technical mailing list for which strictly technical content is expected.

freebsd-hardware (http://lists.FreeBSD.org/mailman/listinfo/freebsd-hardware)

General discussion of FreeBSD hardware

General discussion about the types of hardware that FreeBSD runs on, various problems and suggestions concerning what to buy or avoid.

freebsd-hubs (http://lists.FreeBSD.org/mailman/listinfo/freebsd-hubs)

Mirror sites

Announcements and discussion for people who run FreeBSD mirror sites.

freebsd-isp (http://lists.FreeBSD.org/mailman/listinfo/freebsd-isp)

Issues for Internet Service Providers

This mailing list is for discussing topics relevant to Internet Service Providers (ISPs) using FreeBSD. This is a technical mailing list for which strictly technical content is expected.

freebsd-mono (http://lists.FreeBSD.org/mailman/listinfo/freebsd-mono)

Mono and C# applications on FreeBSD

This is a list for discussions related to the Mono development framework on FreeBSD. This is a technical mailing list. It is for individuals actively working on porting Mono or C# applications to FreeBSD, to bring up problems or discuss alternative solutions. Individuals interested in following the technical discussion are also welcome.

freebsd-office (http://lists.FreeBSD.org/mailman/listinfo/freebsd-office)

Office applications on FreeBSD

Discussion centers around office applications, their installation, their development and their support within FreeBSD.

freebsd-ops-announce (http://lists.FreeBSD.org/mailman/listinfo/freebsd-ops-announce)

Project Infrastructure Announcements

This is the mailing list for people interested in changes and issues related to the FreeBSD.org Project infrastructure.

This moderated list is strictly for announcements: no replies, requests, discussions, or opinions.

freebsd-performance (http://lists.FreeBSD.org/mailman/listinfo/freebsd-performance)

Discussions about tuning or speeding up FreeBSD

This mailing list exists to provide a place for hackers, administrators, and/or concerned parties to discuss performance related topics pertaining to FreeBSD. Acceptable topics includes talking about FreeBSD installations that are either under high load, are experiencing performance problems, or are pushing the limits of FreeBSD. Concerned parties that are willing to work toward improving the performance of FreeBSD are highly encouraged to subscribe to this list. This is a highly technical list ideally suited for experienced FreeBSD users, hackers, or administrators interested in keeping FreeBSD fast, robust, and scalable. This list is not a question-and-answer list that replaces reading through documentation, but it is a place to make contributions or inquire about unanswered performance related topics.

freebsd-pf (http://lists.FreeBSD.org/mailman/listinfo/freebsd-pf)

Discussion and questions about the packet filter firewall system

Discussion concerning the packet filter (pf) firewall system in terms of FreeBSD. Technical discussion and user questions are both welcome. This list is also a place to discuss the ALTQ QoS framework.

freebsd-pkg (http://lists.FreeBSD.org/mailman/listinfo/freebsd-pkg)

Binary package management and package tools discussion

Discussion of all aspects of managing FreeBSD systems by using binary packages to install software, including binary package toolkits and formats, their development and support within FreeBSD, package repository management, and 3rd party packages.

Note that discussion of ports which fail to generate packages correctly should generally be considered as ports problems, and so inappropriate for this list.

freebsd-platforms (http://lists.FreeBSD.org/mailman/listinfo/freebsd-platforms)

Porting to Non Intel platforms

Cross-platform FreeBSD issues, general discussion and proposals for non Intel FreeBSD ports. This is a technical mailing list for which strictly technical content is expected.

freebsd-ports (http://lists.FreeBSD.org/mailman/listinfo/freebsd-ports)

Discussion of "ports"

Discussions concerning FreeBSD's "ports collection" (`/usr/ports`), ports infrastructure, and general ports coordination efforts. This is a technical mailing list for which strictly technical content is expected.

freebsd-ports-announce (http://lists.FreeBSD.org/mailman/listinfo/freebsd-ports-announce)

Important news and instructions about the FreeBSD "Ports Collection"

Important news for developers, porters, and users of the "Ports Collection" (`/usr/ports`), including architecture/infrastructure changes, new capabilities, critical upgrade instructions, and release engineering information. This is a low-volume mailing list, intended for announcements.

freebsd-ports-bugs (http://lists.FreeBSD.org/mailman/listinfo/freebsd-ports-bugs)

Discussion of "ports" bugs

Discussions concerning problem reports for FreeBSD's "ports collection" (`/usr/ports`), proposed ports, or modifications to ports. This is a technical mailing list for which strictly technical content is expected.

freebsd-proliant (http://lists.FreeBSD.org/mailman/listinfo/freebsd-proliant)

Technical discussion of FreeBSD on HP ProLiant server platforms

This mailing list is to be used for the technical discussion of the usage of FreeBSD on HP ProLiant servers, including the discussion of ProLiant-specific drivers, management software, configuration tools, and BIOS updates. As such, this is the primary place to discuss the hpasmd, hpasmcli, and hpacucli modules.

freebsd-python (http://lists.FreeBSD.org/mailman/listinfo/freebsd-python)

Python on FreeBSD

This is a list for discussions related to improving Python-support on FreeBSD. This is a technical mailing list. It is for individuals working on porting Python, its 3rd party modules and **Zope** stuff to FreeBSD. Individuals interested in following the technical discussion are also welcome.

freebsd-questions (http://lists.FreeBSD.org/mailman/listinfo/freebsd-questions)

User questions

This is the mailing list for questions about FreeBSD. Do not send "how to" questions to the technical lists unless the question is quite technical.

freebsd-ruby (http://lists.FreeBSD.org/mailman/listinfo/freebsd-ruby)

FreeBSD-specific Ruby discussions

This is a list for discussions related to the Ruby support on FreeBSD. This is a technical mailing list. It is for individuals working on Ruby ports, 3rd party libraries and frameworks.

Individuals interested in the technical discussion are also welcome.

freebsd-scsi (http://lists.FreeBSD.org/mailman/listinfo/freebsd-scsi)

SCSI subsystem

This is the mailing list for people working on the SCSI subsystem for FreeBSD. This is a technical mailing list for which strictly technical content is expected.

freebsd-security (http://lists.FreeBSD.org/mailman/listinfo/freebsd-security)

Security issues

FreeBSD computer security issues (DES, Kerberos, known security holes and fixes, etc). This is a technical mailing list for which strictly technical discussion is expected. Note that this is not a question-and-answer list, but that contributions (BOTH question AND answer) to the FAQ are welcome.

freebsd-security-notifications (http://lists.FreeBSD.org/mailman/listinfo/freebsd-security-notifications)

Security Notifications

Notifications of FreeBSD security problems and fixes. This is not a discussion list. The discussion list is FreeBSD-security.

freebsd-small (http://lists.FreeBSD.org/mailman/listinfo/freebsd-small)

Using FreeBSD in embedded applications

This list discusses topics related to unusually small and embedded FreeBSD installations. This is a technical mailing list for which strictly technical content is expected.

> **Note:** This list has been obsoleted by freebsd-embedded (http://lists.FreeBSD.org/mailman/listinfo/freebsd-embedded).

freebsd-snapshots (http://lists.FreeBSD.org/mailman/listinfo/freebsd-snapshots)

FreeBSD Development Snapshot Announcements

This list provides notifications about the availability of new FreeBSD development snapshots for the head/ and stable/ branches.

freebsd-stable (http://lists.FreeBSD.org/mailman/listinfo/freebsd-stable)

Discussions about the use of FreeBSD-STABLE

This is the mailing list for users of FreeBSD-STABLE. It includes warnings about new features coming out in -STABLE that will affect the users, and instructions on steps that must be taken to remain -STABLE. Anyone

running “STABLE” should subscribe to this list. This is a technical mailing list for which strictly technical content is expected.

freebsd-standards (http://lists.FreeBSD.org/mailman/listinfo/freebsd-standards)

C99 & POSIX Conformance

This is a forum for technical discussions related to FreeBSD Conformance to the C99 and the POSIX standards.

freebsd-testing (http://lists.FreeBSD.org/mailman/listinfo/freebsd-testing)

Testing on FreeBSD

Technical mailing list discussing testing on FreeBSD, including ATF/Kyua, test build infrastructure, port tests to FreeBSD from other operating systems (NetBSD, ...), etc.

freebsd-tex (http://lists.FreeBSD.org/mailman/listinfo/freebsd-tex)

Porting TEX and its applications to FreeBSD

This is a technical mailing list for discussions related to TeX and its applications on FreeBSD. It is for individuals actively working on porting TeX to FreeBSD, to bring up problems or discuss alternative solutions. Individuals interested in following the technical discussion are also welcome.

freebsd-toolchain (http://lists.FreeBSD.org/mailman/listinfo/freebsd-toolchain)

Maintenance of FreeBSD’s integrated toolchain

This is the mailing list for discussions related to the maintenance of the toolchain shipped with FreeBSD. This could include the state of Clang and GCC, but also pieces of software such as assemblers, linkers and debuggers.

freebsd-usb (http://lists.FreeBSD.org/mailman/listinfo/freebsd-usb)

Discussing FreeBSD support for USB

This is a mailing list for technical discussions related to FreeBSD support for USB.

freebsd-user-groups (http://lists.FreeBSD.org/mailman/listinfo/freebsd-user-groups)

User Group Coordination List

This is the mailing list for the coordinators from each of the local area Users Groups to discuss matters with each other and a designated individual from the Core Team. This mail list should be limited to meeting synopsis and coordination of projects that span User Groups.

freebsd-virtualization (http://lists.FreeBSD.org/mailman/listinfo/freebsd-virtualization)

Discussion of various virtualization techniques supported by FreeBSD

A list to discuss the various virtualization techniques supported by FreeBSD. On one hand the focus will be on the implementation of the basic functionality as well as adding new features. On the other hand users will have a forum to ask for help in case of problems or to discuss their use cases.

freebsd-wip-status (http://lists.FreeBSD.org/mailman/listinfo/freebsd-wip-status)

FreeBSD Work-In-Progress Status

This mailing list can be used by developers to announce the creation and progress of FreeBSD related work. Messages will be moderated. It is suggested to send the message "To:" a more topical FreeBSD list and only

"BCC:" this list. This way the WIP can also be discussed on the topical list, as no discussion is allowed on this list.

Look inside the archives for examples of suitable messages.

An editorial digest of the messages to this list might be posted to the FreeBSD website every few months as part of the Status Reports [1]. Past reports are archived.

freebsd-wireless (http://lists.FreeBSD.org/mailman/listinfo/freebsd-wireless)

Discussions of 802.11 stack, tools device driver development

The FreeBSD-wireless list focuses on 802.11 stack (sys/net80211), device driver and tools development. This includes bugs, new features and maintenance.

freebsd-xen (http://lists.FreeBSD.org/mailman/listinfo/freebsd-xen)

Discussion of the FreeBSD port to Xen — implementation and usage

A list that focuses on the FreeBSD Xen port. The anticipated traffic level is small enough that it is intended as a forum for both technical discussions of the implementation and design details as well as administrative deployment issues.

freebsd-xfce (http://lists.FreeBSD.org/mailman/listinfo/freebsd-xfce)

XFCE

This is a forum for discussions related to bring the **XFCE** environment to FreeBSD. This is a technical mailing list. It is for individuals actively working on porting **XFCE** to FreeBSD, to bring up problems or discuss alternative solutions. Individuals interested in following the technical discussion are also welcome.

freebsd-zope (http://lists.FreeBSD.org/mailman/listinfo/freebsd-zope)

Zope

This is a forum for discussions related to bring the **Zope** environment to FreeBSD. This is a technical mailing list. It is for individuals actively working on porting **Zope** to FreeBSD, to bring up problems or discuss alternative solutions. Individuals interested in following the technical discussion are also welcome.

C.1.4 Filtering on the Mailing Lists

The FreeBSD mailing lists are filtered in multiple ways to avoid the distribution of spam, viruses, and other unwanted emails. The filtering actions described in this section do not include all those used to protect the mailing lists.

Only certain types of attachments are allowed on the mailing lists. All attachments with a MIME content type not found in the list below will be stripped before an email is distributed on the mailing lists.

- application/octet-stream
- application/pdf
- application/pgp-signature
- application/x-pkcs7-signature
- message/rfc822
- multipart/alternative

- multipart/related
- multipart/signed
- text/html
- text/plain
- text/x-diff
- text/x-patch

> **Note:** Some of the mailing lists might allow attachments of other MIME content types, but the above list should be applicable for most of the mailing lists.

If an email contains both an HTML and a plain text version, the HTML version will be removed. If an email contains only an HTML version, it will be converted to plain text.

C.2 Usenet Newsgroups

In addition to two FreeBSD specific newsgroups, there are many others in which FreeBSD is discussed or are otherwise relevant to FreeBSD users.

C.2.1 BSD Specific Newsgroups

- comp.unix.bsd.freebsd.announce (news:comp.unix.bsd.freebsd.announce)
- comp.unix.bsd.freebsd.misc (news:comp.unix.bsd.freebsd.misc)
- de.comp.os.unix.bsd (news:de.comp.os.unix.bsd) (German)
- fr.comp.os.bsd (news:fr.comp.os.bsd) (French)
- it.comp.os.freebsd (news:it.comp.os.freebsd) (Italian)

C.2.2 Other UNIX Newsgroups of Interest

- comp.unix (news:comp.unix)
- comp.unix.questions (news:comp.unix.questions)
- comp.unix.admin (news:comp.unix.admin)
- comp.unix.programmer (news:comp.unix.programmer)
- comp.unix.shell (news:comp.unix.shell)
- comp.unix.user-friendly (news:comp.unix.user-friendly)
- comp.security.unix (news:comp.security.unix)
- comp.sources.unix (news:comp.sources.unix)

- comp.unix.advocacy (news:comp.unix.advocacy)
- comp.unix.misc (news:comp.unix.misc)
- comp.unix.bsd (news:comp.unix.bsd)

C.2.3 X Window System

- comp.windows.x.i386unix (news:comp.windows.x.i386unix)
- comp.windows.x (news:comp.windows.x)
- comp.windows.x.apps (news:comp.windows.x.apps)
- comp.windows.x.announce (news:comp.windows.x.announce)
- comp.windows.x.intrinsics (news:comp.windows.x.intrinsics)
- comp.windows.x.motif (news:comp.windows.x.motif)
- comp.windows.x.pex (news:comp.windows.x.pex)
- comp.emulators.ms-windows.wine (news:comp.emulators.ms-windows.wine)

C.3 World Wide Web Servers

C.3.1 Forums, Blogs, and Social Networks

- The FreeBSD Forums (http://forums.freebsd.org/) provide a web based discussion forum for FreeBSD questions and technical discussion.
- Planet FreeBSD (http://planet.freebsdish.org/) offers an aggregation feed of dozens of blogs written by FreeBSD developers. Many developers use this to post quick notes about what they are working on, new patches, and other works in progress.
- The BSDConferences YouTube Channel (http://www.youtube.com/bsdconferences) provides a collection of high quality videos from BSD Conferences around the world. This is a great way to watch key developers give presentations about new work in FreeBSD.

C.3.2 Official Mirrors

Central Servers, Armenia, Australia, Austria, Belgium, Brazil, Canada, China, Costa Rica, Czech Republic, Denmark, Estonia, Finland, France, Germany, Greece, Hong Kong, Iceland, Italy, Japan, Korea, Kuwait, Kyrgyzstan, Latvia, Lithuania, Netherlands, Norway, Philippines, Portugal, Romania, Russia, San Marino, Slovak Republic, Slovenia, South Africa, Spain, Sweden, Switzerland, Taiwan, Thailand, Turkey, Ukraine, United Kingdom, USA.

(as of UTC)

-

Central Servers

- http://www.FreeBSD.org/

- Armenia
  - http://www1.am.FreeBSD.org/ (IPv6)

- Australia
  - http://www.au.FreeBSD.org/
  - http://www2.au.FreeBSD.org/

- Austria
  - http://www.at.FreeBSD.org/ (IPv6)

- Belgium
  - http://freebsd.unixtech.be/

- Brazil
  - http://www.br.FreeBSD.org/ (IPv6)
  - http://www2.br.FreeBSD.org/www.freebsd.org/
  - http://www3.br.FreeBSD.org/

- Canada
  - http://www.ca.FreeBSD.org/
  - http://www2.ca.FreeBSD.org/

- China
  - http://www.cn.FreeBSD.org/

- Costa Rica
  - http://www1.cr.FreeBSD.org/
- Czech Republic
  - http://www.cz.FreeBSD.org/ (IPv6)
- Denmark
  - http://www.dk.FreeBSD.org/ (IPv6)
- Estonia
  - http://www.ee.FreeBSD.org/
- Finland
  - http://www.fi.FreeBSD.org/
  - http://www2.fi.FreeBSD.org/
- France
  - http://www.fr.FreeBSD.org/
  - http://www1.fr.FreeBSD.org/
- Germany
  - http://www.de.FreeBSD.org/
- Greece
  - http://www.gr.FreeBSD.org/
-

Hong Kong

- http://www.hk.FreeBSD.org/

- Iceland
  - http://www.is.FreeBSD.org/

- Italy
  - http://www.it.FreeBSD.org/
  - http://www.gufi.org/mirrors/www.freebsd.org/data/

- Japan
  - http://www.jp.FreeBSD.org/www.FreeBSD.org/ (IPv6)

- Korea
  - http://www.kr.FreeBSD.org/
  - http://www2.kr.FreeBSD.org/

- Kuwait
  - http://www.kw.FreeBSD.org/

- Kyrgyzstan
  - http://www.kg.FreeBSD.org/

- Latvia
  - http://www.lv.FreeBSD.org/
  - http://www2.lv.FreeBSD.org/

-

Lithuania

- http://www.lt.FreeBSD.org/

- Netherlands
  - http://www.nl.FreeBSD.org/
  - http://www2.nl.FreeBSD.org/

- Norway
  - http://www.no.FreeBSD.org/

- Philippines
  - http://www.FreeBSD.org.ph/

- Portugal
  - http://www.pt.FreeBSD.org/
  - http://www1.pt.FreeBSD.org/
  - http://www4.pt.FreeBSD.org/
  - http://www5.pt.FreeBSD.org/

- Romania
  - http://www.ro.FreeBSD.org/
  - http://www1.ro.FreeBSD.org/
  - http://www2.ro.FreeBSD.org/
  - http://www3.ro.FreeBSD.org/

- Russia
  - http://www.ru.FreeBSD.org/
  - http://www2.ru.FreeBSD.org/
  - http://www3.ru.FreeBSD.org/

- http://www4.ru.FreeBSD.org/
- http://www5.ru.FreeBSD.org/

- San Marino
  - http://www.sm.FreeBSD.org/

- Slovak Republic
  - http://www.sk.FreeBSD.org/

- Slovenia
  - http://www.si.FreeBSD.org/
  - http://www2.si.FreeBSD.org/

- South Africa
  - http://www.za.FreeBSD.org/
  - http://www2.za.FreeBSD.org/

- Spain
  - http://www.es.FreeBSD.org/
  - http://www2.es.FreeBSD.org/
  - http://www3.es.FreeBSD.org/

- Sweden
  - http://www.se.FreeBSD.org/
  - http://www2.se.FreeBSD.org/

- Switzerland
  - http://www.ch.FreeBSD.org/ (IPv6)

- http://www2.ch.FreeBSD.org/ (IPv6)

- Taiwan
  - http://www.tw.FreeBSD.org/ (IPv6)
  - http://www2.tw.FreeBSD.org/
  - http://www3.tw.FreeBSD.org/
  - http://www4.tw.FreeBSD.org/
  - http://www5.tw.FreeBSD.org/ (IPv6)
  - http://www6.tw.FreeBSD.org/
  - http://www7.tw.FreeBSD.org/

- Thailand
  - http://www.th.FreeBSD.org/

- Turkey
  - http://www.tr.FreeBSD.org/
  - http://www2.tr.FreeBSD.org/
  - http://www3.tr.FreeBSD.org/ (IPv6)

- Ukraine
  - http://www.ua.FreeBSD.org/
  - http://www2.ua.FreeBSD.org/
  - http://www5.ua.FreeBSD.org/
  - http://www4.ua.FreeBSD.org/

- United Kingdom
  - http://www1.uk.FreeBSD.org/
  - http://www3.uk.FreeBSD.org/

-

USA

- http://www2.us.FreeBSD.org/
- http://www5.us.FreeBSD.org/ (IPv6)

C.4 Email Addresses

The following user groups provide FreeBSD related email addresses for their members. The listed administrator reserves the right to revoke the address if it is abused in any way.

Domain	Facilities	User Group	Administrator
ukug.uk.FreeBSD.org	Forwarding only	<ukfreebsd@uk.FreeBSD.org>	Lee Johnston <lee@uk.FreeBSD.org>

Notes

1. http://www.freebsd.org/news/status/

Appendix D. PGP Keys

In case you need to verify a signature or send encrypted email to one of the officers or developers a number of keys are provided here for your convenience. A complete keyring of `FreeBSD.org` users is available for download from http://www.FreeBSD.org/doc/pgpkeyring.txt.

D.1 Officers

D.1.1 Security Officer Team <security-officer@FreeBSD.org>

```
pub  1024D/CA6CDFB2 2002-08-27 FreeBSD Security Officer <security-officer@FreeBSD.org>
     Key fingerprint = C374 0FC5 69A6 FBB1 4AED  B131 15D6 8804 CA6C DFB2
sub  2048g/A3071809 2002-08-27
```

D.1.2 Core Team Secretary <core-secretary@FreeBSD.org>

```
pub   2048R/2CA49776 2012-07-23
      Key fingerprint = 89F6 C031 B4E3 D472 E4CE  8372 4D58 FDCD 2CA4 9776
uid                  FreeBSD Core Team Secretary <core-secretary@freebsd.org>
sub   2048R/BBAD1C98 2012-07-23
```

D.1.3 Ports Management Team Secretary <portmgr-secretary@FreeBSD.org>

```
pub   2048R/BBC4D7D5 2012-07-24
      Key fingerprint = FB37 45C8 6F15 E8ED AC81  32FC D829 4EC3 BBC4 D7D5
uid                  FreeBSD Ports Management Team Secretary <portmgr-secretary@FreeBSD.org>
sub   2048R/5F65CFE7 2012-07-24
```

D.2 Core Team Members

D.2.1 Thomas Abthorpe <tabthorpe@FreeBSD.org>

```
pub   2048R/A473C990 2010-05-28
      Key fingerprint = D883 2D7C EB78 944A 69FC  36A6 D937 1097 A473 C990
uid                  Thomas Abthorpe (FreeBSD Committer) <tabthorpe@FreeBSD.org>
uid                  Thomas Abthorpe <tabthorpe@abthorpe.org>
uid                  Thomas Abthorpe <tabthorpe@goodking.ca>
uid                  Thomas Abthorpe <tabthorpe@goodking.org>
uid                  Thomas Abthorpe <thomas@goodking.ca>
sub   2048R/8CA60EE0 2010-05-28
```

D.2.2 Gavin Atkinson <gavin@FreeBSD.org>

```
pub   1024D/A093262B 2005-02-18
      Key fingerprint = 313A A79F 697D 3A5C 216A  EDF5 935D EF44 A093 262B
uid                  Gavin Atkinson <gavin@16squared.co.uk>
uid                  Gavin Atkinson (FreeBSD key) <gavin@FreeBSD.org>
uid                  Gavin Atkinson (Work e-mail) <ga9@york.ac.uk>
uid                  Gavin Atkinson <gavin.atkinson@ury.york.ac.uk>
sub   2048g/58F40B3D 2005-02-18
```

D.2.3 John Baldwin <jhb@FreeBSD.org>

```
pub  1024R/C10A874D 1999-01-13 John Baldwin <jbaldwin@weather.com>
     Key fingerprint = 43 33 1D 37 72 B1 EF 5B  9B 5F 39 F8 BD C1 7C B5
uid                            John Baldwin <john@baldwin.cx>
uid                            John Baldwin <jhb@FreeBSD.org>
uid                            John Baldwin <jobaldwi@vt.edu>
```

D.2.4 Konstantin Belousov <kib@FreeBSD.org>

```
pub   4096R/C1BCAD41 2012-11-17
      Key fingerprint = 7DE0 3388 64AC 53C3 7B88  3A79 90C2 B92B C1BC AD41
uid                  Konstantin Belousov <kib@FreeBSD.org>
uid                  Konstantin Belousov <kostikbel@gmail.com>
uid                  Konstantin Belousov <kib@kib.kiev.ua>
sub   4096R/3BBC8F64 2012-11-17
```

D.2.5 David Chisnall <theraven@FreeBSD.org>

```
pub   4096R/65C4F55D 2012-11-28
      Key fingerprint = 3E8F 5E9F 7586 F090 AC2C  58C2 BA06 FF14 65C4 F55D
uid                  David Chisnall <theraven@FreeBSD.org>
sub   4096R/04B2A21D 2012-11-28
```

D.2.6 Hiroki Sato <hrs@FreeBSD.org>

```
pub   1024D/2793CF2D 2001-06-12
      Key fingerprint = BDB3 443F A5DD B3D0 A530  FFD7 4F2C D3D8 2793 CF2D
uid                  Hiroki Sato <hrs@allbsd.org>
uid                  Hiroki Sato <hrs@eos.ocn.ne.jp>
uid                  Hiroki Sato <hrs@ring.gr.jp>
uid                  Hiroki Sato <hrs@FreeBSD.org>
uid                  Hiroki Sato <hrs@jp.FreeBSD.org>
uid                  Hiroki Sato <hrs@vlsi.ee.noda.tus.ac.jp>
uid                  Hiroki Sato <hrs@jp.NetBSD.org>
uid                  Hiroki Sato <hrs@NetBSD.org>
```

```
uid                  Hiroki Sato <hrs@ec.ss.titech.ac.jp>
uid                  Hiroki Sato <hrs@ieee.org>
uid                  Hiroki Sato <hrs@acm.org>
uid                  Hiroki Sato <hrs@bsdconsulting.co.jp>
uid                  Hiroki Sato <hrs@bsdresearch.org>
uid                  Hiroki Sato <hrs@ec.ce.titech.ac.jp>
sub   1024g/8CD251FF 2001-06-12
```

D.2.7 Peter Wemm <peter@FreeBSD.org>

```
pub  1024D/7277717F 2003-12-14 Peter Wemm <peter@wemm.org>
     Key fingerprint = 622B 2282 E92B 3BAB 57D1  A417 1512 AE52 7277 717F
uid                            Peter Wemm <peter@FreeBSD.ORG>
sub  1024g/8B40D9D1 2003-12-14
pub  1024R/D89CE319 1995-04-02 Peter Wemm <peter@netplex.com.au>
     Key fingerprint = 47 05 04 CA 4C EE F8 93  F6 DB 02 92 6D F5 58 8A
uid                            Peter Wemm <peter@perth.dialix.oz.au>
uid                            Peter Wemm <peter@haywire.dialix.com>
```

D.2.8 Martin Wilke <miwi@FreeBSD.org>

```
pub   1024D/B1E6FCE9 2009-01-31
      Key fingerprint = C022 7D60 F598 8188 2635  0F6E 74B2 4884 B1E6 FCE9
uid                  Martin Wilke <miwi@FreeBSD.org>
sub   4096g/096DA69D 2009-01-31
```

D.3 Developers

D.3.1 Ariff Abdullah <ariff@FreeBSD.org>

```
pub   1024D/C5304CDA 2005-10-01
      Key fingerprint = 5C7C 6BF4 8293 DE76 27D9  FD57 96BF 9D78 C530 4CDA
uid                  Ariff Abdullah <skywizard@MyBSD.org.my>
uid                  Ariff Abdullah <ariff@MyBSD.org.my>
uid                  Ariff Abdullah <ariff@FreeBSD.org>
sub   2048g/8958C1D3 2005-10-01
```

D.3.2 Thomas Abthorpe <tabthorpe@FreeBSD.org>

```
pub   2048R/A473C990 2010-05-28
      Key fingerprint = D883 2D7C EB78 944A 69FC  36A6 D937 1097 A473 C990
uid                  Thomas Abthorpe (FreeBSD Committer) <tabthorpe@FreeBSD.org>
uid                  Thomas Abthorpe <tabthorpe@abthorpe.org>
uid                  Thomas Abthorpe <tabthorpe@goodking.ca>
```

```
uid                  Thomas Abthorpe <tabthorpe@goodking.org>
uid                  Thomas Abthorpe <thomas@goodking.ca>
sub   2048R/8CA60EE0 2010-05-28
```

D.3.3 Eitan Adler <eadler@FreeBSD.org>

```
pub   4096R/8FC8196C 2011-02-11
      Key fingerprint = 49C7 29DF E09C 0FC7 A1C4  6ECB A338 A6FC 8FC8 196C
uid                  Eitan Adler <lists@eitanadler.com>
sub   4096R/18763D51 2011-02-11
sub   4096R/DAB9CF9B 2011-02-11
```

D.3.4 Shaun Amott <shaun@FreeBSD.org>

```
pub   1024D/6B387A9A 2001-03-19
      Key fingerprint = B506 E6C7 74A1 CC11 9A23  5C13 9268 5D08 6B38 7A9A
uid                  Shaun Amott <shaun@inerd.com>
uid                  Shaun Amott <shaun@FreeBSD.org>
sub   2048g/26FA8703 2001-03-19
sub   2048R/7FFF5151 2005-11-06
sub   2048R/27C54137 2005-11-06
```

D.3.5 Henrik Brix Andersen <brix@FreeBSD.org>

```
pub   1024D/54E278F8 2003-04-09
      Key fingerprint = 7B63 EF32 7831 A704 220D  7E61 BFE4 387E 54E2 78F8
uid                  Henrik Brix Andersen <henrik@brixandersen.dk>
uid                  Henrik Brix Andersen <brix@FreeBSD.org>
uid                  Henrik Brix Andersen <hbn@terma.com>
uid                  Henrik Brix Andersen <brix@osaa.dk>
sub   1024g/3B13C209 2003-04-09
```

D.3.6 Matthias Andree <mandree@FreeBSD.org>

```
pub   1024D/052E7D95 2003-08-28
      Key fingerprint = FDD0 0C43 6E33 07E1 0758  C6A8 BE61 8339 052E 7D95
uid                  Matthias Andree <mandree@freebsd.org>
uid                  Matthias Andree <matthias.andree@gmx.de>
sub   1536g/E65A83DA 2003-08-28
```

D.3.7 Will Andrews <will@FreeBSD.org>

```
pub  1024D/F81672C5 2000-05-22 Will Andrews (Key for official matters) <will@FreeBSD.org>
     Key fingerprint = 661F BBF7 9F5D 3D02 C862  5F6C 178E E274 F816 72C5
uid                            Will Andrews <will@physics.purdue.edu>
uid                            Will Andrews <will@puck.firepipe.net>
uid                            Will Andrews <will@c-60.org>
uid                            Will Andrews <will@csociety.org>
uid                            Will Andrews <will@csociety.ecn.purdue.edu>
uid                            Will Andrews <will@telperion.openpackages.org>
sub  1024g/55472804 2000-05-22
```

D.3.8 Dimitry Andric <dim@FreeBSD.org>

```
pub   1024D/2E2096A3 1997-11-17
      Key fingerprint = 7AB4 62D2 CE35 FC6D 4239  4FCD B05E A30A 2E20 96A3
uid                  Dimitry Andric <dimitry@andric.com>
uid                  Dimitry Andric <dim@xs4all.nl>
uid                  Dimitry Andric <dimitry.andric@tomtom.com>
uid                  [jpeg image of size 5132]
uid                  Dimitry Andric <dim@nah6.com>
uid                  Dimitry Andric <dim@FreeBSD.org>
sub   4096g/6852A5C5 1997-11-17
```

D.3.9 Eric Anholt <anholt@FreeBSD.org>

```
pub   1024D/6CF0EAF7 2003-09-08
      Key fingerprint = 76FE 2475 820B B75F DCA4  0F3E 1D47 6F60 6CF0 EAF7
uid                  Eric Anholt <eta@lclark.edu>
uid                  Eric Anholt <anholt@FreeBSD.org>
sub   1024g/80B404C1 2003-09-08
```

D.3.10 Marcus von Appen <mva@FreeBSD.org>

```
pub   1024D/B267A647 2009-02-14
      Key fingerprint = C7CC 1853 D8C5 E580 7795  B654 8BAF 3F12 B267 A647
uid                  Marcus von Appen <freebsd@sysfault.org>
uid                  Marcus von Appen <mva@freebsd.org>
sub   2048g/D34A3BAF 2009-02-14
```

D.3.11 Marcelo Araujo <araujo@FreeBSD.org>

```
pub   1024D/53E4CFA8 2007-04-27
      Key fingerprint = 9D6A 2339 925C 4F61 ED88  ED8B A2FC 4977 53E4 CFA8
uid                  Marcelo Araujo (Ports Committer) <araujo@FreeBSD.org>
sub   2048g/63CC012D 2007-04-27
```

D.3.12 Mathieu Arnold <mat@FreeBSD.org>

```
pub   1024D/FE6D850F 2005-04-25
      Key fingerprint = 2771 11F4 0A7E 73F9 ADDD  A542 26A4 7C6A FE6D 850F
uid                  Mathieu Arnold <mat@FreeBSD.org>
uid                  Mathieu Arnold <mat@mat.cc>
uid                  Mathieu Arnold <mat@cpan.org>
uid                  Mathieu Arnold <m@absolight.fr>
uid                  Mathieu Arnold <m@absolight.net>
uid                  Mathieu Arnold <mat@club-internet.fr>
uid                  Mathieu Arnold <marnold@april.org>
uid                  Mathieu Arnold <paypal@mat.cc>
sub   2048g/EAD18BD9 2005-04-25
```

D.3.13 Takuya ASADA <syuu@FreeBSD.org>

```
pub   2048R/43788F78 2012-11-21
      Key fingerprint = 31CE 242E 6F4F F24F EEF4  D9BB 0890 2C5F 4378 8F78
uid                  Takuya ASADA <syuu@freebsd.org>
sub   2048R/A87B0906 2012-11-21
```

D.3.14 Satoshi Asami <asami@FreeBSD.org>

```
pub  1024R/1E08D889 1997-07-23 Satoshi Asami <asami@cs.berkeley.edu>
     Key fingerprint = EB 3C 68 9E FB 6C EB 3F  DB 2E 0F 10 8F CE 79 CA
uid                           Satoshi Asami <asami@FreeBSD.ORG>
```

D.3.15 Gavin Atkinson <gavin@FreeBSD.org>

```
pub   1024D/A093262B 2005-02-18
      Key fingerprint = 313A A79F 697D 3A5C 216A  EDF5 935D EF44 A093 262B
uid                  Gavin Atkinson <gavin@16squared.co.uk>
uid                  Gavin Atkinson (FreeBSD key) <gavin@FreeBSD.org>
uid                  Gavin Atkinson (Work e-mail) <ga9@york.ac.uk>
uid                  Gavin Atkinson <gavin.atkinson@ury.york.ac.uk>
sub   2048g/58F40B3D 2005-02-18
```

D.3.16 Joseph S. Atkinson <jsa@FreeBSD.org>

```
pub   2048R/21AA7B06 2010-07-14
      Key fingerprint = 5B38 63B0 9CCA 12BE 3919  9412 CC9D FC84 21AA 7B06
uid                  Joseph S. Atkinson <jsa@FreeBSD.org>
uid                  Joseph S. Atkinson <jsa.bsd@gmail.com>
uid                  Joseph S. Atkinson <jsa@wickedmachine.net>
sub   2048R/5601C3E3 2010-07-14
```

D.3.17 Philippe Audeoud <jadawin@FreeBSD.org>

```
pub   1024D/C835D40E 2005-04-13
      Key fingerprint = D090 8C96 3612 15C9 4E3E  7A4A E498 FC2B C835 D40E
uid                  Philippe Audeoud <jadawin@tuxaco.net>
uid                  Philippe Audeoud <philippe@tuxaco.net>
uid                  Philippe Audeoud <philippe.audeoud@sitadelle.com>
uid                  Philippe Audeoud <jadawin@freebsd.org>
sub   2048g/EF8EA329 2005-04-13
```

D.3.18 Timur I. Bakeyev <timur@FreeBSD.org>

```
pub   1024D/60BA1F47 2002-04-27
      Key fingerprint = 84BF EAD1 607D 362F 210E  69B3 0BF0 6412 60BA 1F47
uid                  Timur I. Bakeyev (BaT) <timur@bat.ru>
uid                  Timur I. Bakeyev <timur@gnu.org>
uid                  Timur I. Bakeyev (BaT) <bat@cpan.org>
uid                  Timur I. Bakeyev (BaT) <timur@FreeBSD.org>
uid                  Timur I. Bakeyev (BaT) <timur@gnome.org>
uid                  Timur I. Bakeyev <timur@gnome.org>
sub   2048g/8A5B0042 2002-04-27
```

D.3.19 Glen Barber <gjb@FreeBSD.org>

```
pub   2048R/A0B946A3 2010-08-03 [expires: 2017-04-25]
      Key fingerprint = 78B3 42BA 26C7 B2AC 681E  A7BE 524F 0C37 A0B9 46A3
uid                  Glen Barber <gjb@FreeBSD.org>
uid                  Glen Barber <glen.j.barber@gmail.com>
uid                  Glen Barber <gjb@glenbarber.us>
sub   2048R/6C0527E5 2010-08-03
```

D.3.20 Nick Barkas <snb@FreeBSD.org>

```
pub   2048R/DDADB9DC 2010-07-27
      Key fingerprint = B678 6ECB 303D F580 A050  098F BDFF 4F3D DDAD B9DC
uid                  S. Nicholas Barkas <snb@freebsd.org>
sub   2048R/36E181FB 2010-07-27
sub   2048R/BDA4BED3 2010-07-29
sub   2048R/782A8737 2010-07-29
```

D.3.21 Simon Barner <barner@FreeBSD.org>

```
pub   1024D/EBADA82A 2000-11-10
      Key fingerprint = 67D1 3562 9A2F 3177 E46A  35ED 0A49 FEFD EBAD A82A
uid                  Simon Barner <barner@FreeBSD.org>
uid                  Simon Barner <barner@in.tum.de>
```

```
uid                  Simon Barner <barner@informatik.tu-muenchen.de>
uid                  Simon Barner <barner@gmx.de>
sub   2048g/F63052DE 2000-11-10
```

D.3.22 Artem Belevich <art@FreeBSD.org>

```
pub   2048R/9ED4C836 2011-03-28
      Key fingerprint = 7400 D541 07ED 3DF3 3E97  F2D5 8BDF 101C 9ED4 C836
uid                  Artem Belevich <artemb@gmail.com>
uid                  Artem Belevich <art@freebsd.org>
sub   2048R/55B0E4EB 2011-03-28
```

D.3.23 Anton Berezin <tobez@FreeBSD.org>

```
pub  1024D/7A7BA3C0 2000-05-25 Anton Berezin <tobez@catpipe.net>
     Key fingerprint = CDD8 560C 174B D8E5 0323  83CE 22CA 584C 7A7B A3C0
uid                            Anton Berezin <tobez@tobez.org>
uid                            Anton Berezin <tobez@FreeBSD.org>
sub  1024g/ADC71E87 2000-05-25
```

D.3.24 Damien Bergamini <damien@FreeBSD.org>

```
pub   2048R/D129F093 2005-03-02
      Key fingerprint = D3AB 28C3 1A4A E219 3145  54FE 220A 7486 D129 F093
uid                  Damien Bergamini <damien.bergamini@free.fr>
uid                  Damien Bergamini <damien@FreeBSD.org>
sub   2048R/9FBA73A4 2005-03-02
```

D.3.25 Tim Bishop <tdb@FreeBSD.org>

```
pub   1024D/5AE7D984 2000-10-07
      Key fingerprint = 1453 086E 9376 1A50 ECF6  AE05 7DCE D659 5AE7 D984
uid                  Tim Bishop <tim@bishnet.net>
uid                  Tim Bishop <T.D.Bishop@kent.ac.uk>
uid                  Tim Bishop <tdb@i-scream.org>
uid                  Tim Bishop <tdb@FreeBSD.org>
sub   4096g/7F886031 2000-10-07
```

D.3.26 Grzegorz Blach <gblach@FreeBSD.org>

```
pub   2048R/D25B0682 2012-11-03 [expires: 2014-11-03]
      Key fingerprint = 225B 941C A886 05C6 1C87  9C03 DE72 593D D25B 0682
uid                  Grzegorz Blach <gblach@FreeBSD.org>
sub   2048R/5DE28719 2012-11-03 [expires: 2014-11-03]
```

D.3.27 Martin Blapp <mbr@FreeBSD.org>

```
pub  1024D/D300551E 2001-12-20 Martin Blapp <mb@imp.ch>
     Key fingerprint = B434 53FC C87C FE7B 0A18  B84C 8686 EF22 D300 551E
sub  1024g/998281C8 2001-12-20
```

D.3.28 Warren Block <wblock@FreeBSD.org>

```
pub   2048R/A1F360A3 2011-09-14
      Key fingerprint = 3A44 4DEC B304 5191 8A41  C317 5117 4BB6 A1F3 60A3
uid                  Warren Block <wblock@FreeBSD.org>
uid                  Warren Block <wblock@wonkity.com>
sub   2048R/51F483F3 2011-09-14
```

D.3.29 Vitaly Bogdanov <bvs@FreeBSD.org>

```
pub  1024D/B32017F7 2005-10-02 Vitaly Bogdanov <gad@gad.glazov.net>
     Key fingerprint = 402E B8E4 53CB 22FF BE62  AE35 A0BF B077 B320 17F7
uid                            Vitaly Bogdanov <bvs@freebsd.org>
sub  1024g/0E88C62E 2005-10-02
```

D.3.30 Roman Bogorodskiy <novel@FreeBSD.org>

```
pub   2048R/08C2226A 2010-12-03
      Key fingerprint = 8BA4 DF2A D14F 99B6 37E0  0070 C96D 5FFE 08C2 226A
uid                  Roman Bogorodskiy <bogorodskiy@gmail.com>
uid                  Roman Bogorodskiy <novel@FreeBSD.org>
uid                  Roman Bogorodskiy <rbogorodskiy@apache.org>
uid                  Roman Bogorodskiy <rbogorodskiy@griddynamics.com>
sub   2048R/EC4ED237 2010-12-03
```

D.3.31 Renato Botelho <garga@FreeBSD.org>

```
pub   4096R/9F625790 2012-11-28 [expires: 2017-11-27]
      Key fingerprint = E3DA 9B2A 6160 99CB 4B31  7641 F1F0 E7A1 9F62 5790
uid                  Renato Botelho (FreeBSD) <garga@FreeBSD.org>
uid                  Renato Botelho (Personal) <rbgarga@gmail.com>
uid                  Renato Botelho (FreeBSD) <garga.bsd@gmail.com>
sub   4096R/473CC82A 2012-11-28 [expires: 2017-11-27]
```

D.3.32 Alexander Botero-Lowry <alexbl@FreeBSD.org>

```
pub   1024D/12A95A7B 2006-09-13
      Key fingerprint = D0C3 47F8 AE87 C829 0613  3586 24DF F52B 12A9 5A7B
uid                  Alexander Botero-Lowry <alexbl@FreeBSD.org>
sub   2048g/CA287923 2006-09-13
```

D.3.33 Sofian Brabez <sbz@FreeBSD.org>

```
pub   1024D/2487E57E 2011-03-15 [expires: 2016-03-14]
      Key fingerprint = 05BA DC7E F628 DE3F B241  BFBB 7363 51F4 2487 E57E
uid                  Sofian Brabez <sbrabez@gmail.com>
uid                  Sofian Brabez <sbz@FreeBSD.org>
uid                  Sofian Brabez <sbz@6dev.net>
```

D.3.34 Edson Brandi <ebrandi@FreeBSD.org>

```
pub   3072R/FFD3035B 2012-11-26 [expires: 2017-11-25]
      Key fingerprint = 443B 5363 564F 06C3 EA54  9482 209E 9B54 FFD3 035B
uid                  Edson Brandi <ebrandi@FreeBSD.org>
uid                  Edson Brandi <ebrandi@fugspbr.org>
uid                  Edson Brandi <ebrandi@ebrandi.eti.br>
uid                  Edson Brandi <edson.brandi@gmail.com>
uid                  Edson Brandi <ebrandi@primeirospassos.org>
uid                  Edson Brandi <ebrandi@gmail.com>
uid                  Edson Brandi <ebrandi@fug.com.br>
uid                  Edson Brandi <contato@edsonbrandi.com>
uid                  Edson Brandi (Born 1977-08-14 in S. S. DA GRAMA, SP - Brazil)
sub   3072R/A34B8175 2012-11-26 [expires: 2013-11-26]
sub   3072R/4EB0E0EA 2012-11-26 [expires: 2013-11-26]
sub   3072R/89917E73 2012-11-26 [expires: 2013-11-26]
```

D.3.35 Hartmut Brandt <harti@FreeBSD.org>

```
pub  1024D/5920099F 2003-01-29 Hartmut Brandt <brandt@fokus.fraunhofer.de>
     Key fingerprint = F60D 09A0 76B7 31EE 794B  BB91 082F 291D 5920 099F
uid                           Hartmut Brandt <harti@freebsd.org>
sub  1024g/21D30205 2003-01-29
```

D.3.36 Oliver Braun <obraun@FreeBSD.org>

```
pub  1024D/EF25B1BA 2001-05-06 Oliver Braun <obraun@unsane.org>
     Key fingerprint = 6A3B 042A 732E 17E4 B6E7  3EAF C0B1 6B7D EF25 B1BA
uid                            Oliver Braun <obraun@obraun.net>
uid                            Oliver Braun <obraun@freebsd.org>
uid                            Oliver Braun <obraun@haskell.org>
```

```
sub  1024g/09D28582 2001-05-06
```

D.3.37 Max Brazhnikov <makc@FreeBSD.org>

```
pub   1024D/ACB3CD12 2008-08-18
      Key fingerprint = 4BAA 200E 720A 0BD1 7BB0  9DFD FBD9 08C2 ACB3 CD12
uid                  Max Brazhnikov <makc@FreeBSD.org>
uid                  Max Brazhnikov <makc@issp.ac.ru>
sub   1024g/5FAA4088 2008-08-18
```

D.3.38 Jonathan M. Bresler <jmb@FreeBSD.org>

```
pub  1024R/97E638DD 1996-06-05 Jonathan M. Bresler <jmb@Bresler.org>
     Key fingerprint = 31 57 41 56 06 C1 40 13  C5 1C E3 E5 DC 62 0E FB
uid                            Jonathan M. Bresler <jmb@FreeBSD.ORG>
uid                            Jonathan M. Bresler
uid                            Jonathan M. Bresler <Jonathan.Bresler@USi.net>
uid                            Jonathan M. Bresler <jmb@Frb.GOV>
```

D.3.39 Antoine Brodin <antoine@FreeBSD.org>

```
pub   1024D/50CC2671 2008-02-03
      Key fingerprint = F3F7 72F0 9C4C 9E56 4BE9  44EA 1B80 31F3 50CC 2671
uid                  Antoine Brodin <antoine@FreeBSD.org>
sub   2048g/6F4AFBE5 2008-02-03
```

D.3.40 Diane Bruce <db@FreeBSD.org>

```
pub   2048R/8E9CAA7B 2012-05-16
      Key fingerprint = 8B08 E022 705D 0083 64C4  5E60 5148 0C74 8E9C AA7B
uid                  Diane Bruce <db@db.net>
uid                  Diane Bruce <db@FreeBSD.org>
sub   2048R/932E5985 2012-05-16
```

D.3.41 Christian Brüffer <brueffer@FreeBSD.org>

```
pub  1024D/A0ED982D 2002-10-14 Christian Brueffer <chris@unixpages.org>
     Key fingerprint = A5C8 2099 19FF AACA F41B  B29B 6C76 178C A0ED 982D
uid                            Christian Brueffer <brueffer@hitnet.rwth-aachen.de>
uid                            Christian Brueffer <brueffer@FreeBSD.org>
sub  4096g/1DCC100F 2002-10-14
```

D.3.42 Markus Brüffer <markus@FreeBSD.org>

```
pub   1024D/78F8A8D4 2002-10-21
      Key fingerprint = 3F9B EBE8 F290 E5CC 1447  8760 D48D 1072 78F8 A8D4
uid                  Markus Brueffer <markus@brueffer.de>
uid                  Markus Brueffer <buff@hitnet.rwth-aachen.de>
uid                  Markus Brueffer <mbrueffer@mi.rwth-aachen.de>
uid                  Markus Brueffer <markus@FreeBSD.org>
sub   4096g/B7E5C7B6 2002-10-21
```

D.3.43 Sean Bruno <sbruno@FreeBSD.org>

```
pub   2048R/08E81687 2012-10-15
      Key fingerprint = B9F9 138F 349C D3B2 2AA4  1398 1909 45DC 08E8 1687
uid                  Sean Bruno (clusteradm and developer key) <sbruno@freebsd.org>
sub   2048R/BCC23981 2012-10-15
```

D.3.44 Oleg Bulyzhin <oleg@FreeBSD.org>

```
pub   1024D/78CE105F 2004-02-06
      Key fingerprint = 98CC 3E66 26DE 50A8 DBC4  EB27 AF22 DCEF 78CE 105F
uid                  Oleg Bulyzhin <oleg@FreeBSD.org>
uid                  Oleg Bulyzhin <oleg@rinet.ru>
sub   1024g/F747C159 2004-02-06
```

D.3.45 Michael Bushkov <bushman@FreeBSD.org>

```
pub   1024D/F694C6E4 2007-03-11 [expires: 2008-03-10]
      Key fingerprint = 4278 4392 BF6B 2864 C48E  0FA9 7216 C73C F694 C6E4
uid                  Michael Bushkov <bushman@rsu.ru>
uid                  Michael Bushkov <bushman@freebsd.org>
sub   2048g/5A783997 2007-03-11 [expires: 2008-03-10]
```

D.3.46 Jayachandran C. <jchandra@FreeBSD.org>

```
pub   1024D/3316E465 2010-05-19
      Key fingerprint = 320B DB08 4FE3 BCFD 60AF  E4DB F486 015F 3316 E465
uid                  Jayachandran C. <jchandra@freebsd.org>
sub   2048g/1F7755F9 2010-05-19
```

D.3.47 Jesus R. Camou <jcamou@FreeBSD.org>

```
pub   1024D/C2161947 2005-03-01
      Key fingerprint = 274C B265 48EC 42AE A2CA  47D9 7D98 588A C216 1947
uid                  Jesus R. Camou <jcamou@FreeBSD.org>
sub   2048g/F8D2A8DF 2005-03-01
```

D.3.48 José Alonso Cárdenas Márquez <acm@FreeBSD.org>

```
pub   1024D/9B21BC19 2006-07-18
      Key fingerprint = 4156 2EAC A11C 9651 713B  3FC1 195F D4A8 9B21 BC19
uid                  Jose Alonso Cardenas Marquez <acm@FreeBSD.org>
sub   2048g/ADA16C52 2006-07-18
```

D.3.49 Pietro Cerutti <gahr@FreeBSD.org>

```
pub   1024D/9571F78E 2006-05-17
      Key fingerprint = 1203 92B5 3919 AF84 9B97  28D6 C0C2 6A98 9571 F78E
uid                  Pietro Cerutti <gahr@gahr.ch>
uid                  Pietro Cerutti (The FreeBSD Project) <gahr@FreeBSD.org>
sub   2048g/F24227D5 2006-05-17 [expires: 2011-05-16]
```

D.3.50 Dmitry Chagin <dchagin@FreeBSD.org>

```
pub   1024D/738EFCED 2009-02-27
      Key fingerprint = 3F3F 8B87 CE09 9E10 3606  6ACA D2DD 936F 738E FCED
uid                  Dmitry Chagin <dchagin@freebsd.org>
uid                  Dmitry Chagin (dchagin key) <chagin.dmitry@gmail.com>
sub   2048g/6A3FDFF9 2009-02-27
```

D.3.51 Hye-Shik Chang <perky@FreeBSD.org>

```
pub  1024D/CFDB4BA4 1999-04-23 Hye-Shik Chang <perky@FreeBSD.org>
     Key fingerprint = 09D9 57D6 58BA 44DD CAEC  71CD 0D65 2C59 CFDB 4BA4
uid                           Hye-Shik Chang <hyeshik@gmail.com>
sub  1024g/A94A8ED1 1999-04-23
```

D.3.52 Jonathan Chen <jon@FreeBSD.org>

```
pub  1024D/2539468B 1999-10-11 Jonathan Chen <jon@spock.org>
     Key fingerprint = EE31 CDA1 A105 C8C9 5365  3DB5 C2FC 86AA 2539 468B
uid                            Jonathan Chen <jon@freebsd.org>
uid                            Jonathan Chen <chenj@rpi.edu>
uid                            Jonathan Chen <spock@acm.rpi.edu>
```

```
uid                           Jonathan Chen <jon@cs.rpi.edu>
sub  3072g/B81EF1DB 1999-10-11
```

D.3.53 Jonathan Anderson <jonathan@FreeBSD.org>

```
pub   1024D/E3BBCA48 2006-06-17
      Key fingerprint = D7C6 9096 874F 707E 48F8  FAB7 22A6 6E53 E3BB CA48
uid                  Jonathan Anderson <jonathan@FreeBSD.org>
uid                  Jonathan Anderson <jonathan.anderson@ieee.org>
uid                  Jonathan Anderson <anderson@engr.mun.ca>
uid                  Jonathan Anderson <jonathan.anderson@mun.ca>
sub   2048g/A703650D 2006-06-17
```

D.3.54 Fukang Chen <loader@FreeBSD.org>

```
pub   4096R/6BD4DDE6 2012-10-26
      Key fingerprint = A33E 88AB D358 DA49 59A6  B263 A9A2 599C 6BD4 DDE6
uid                  loader <loader@FreeBSD.org>
uid                  loader <loader@FreeBSDMall.com>
sub   4096R/1036D26C 2012-10-26
```

D.3.55 Luoqi Chen <luoqi@FreeBSD.org>

```
pub  1024D/2926F3BE 2002-02-22 Luoqi Chen <luoqi@FreeBSD.org>
     Key fingerprint = B470 A815 5917 D9F4 37F3  CE2A 4D75 3BD1 2926 F3BE
uid                           Luoqi Chen <luoqi@bricore.com>
uid                           Luoqi Chen <lchen@onetta.com>
sub  1024g/5446EB72 2002-02-22
```

D.3.56 Andrey A. Chernov <ache@FreeBSD.org>

```
pub   1024D/964474DD 2006-12-26
      Key fingerprint = 0F63 1B61 D76D AA23 1591  EA09 560E 582B 9644 74DD
uid                  Andrey Chernov <ache@freebsd.org>
uid                  [jpeg image of size 4092]
sub   2048g/08331894 2006-12-26
```

D.3.57 Alexander V. Chernikov <melifaro@FreeBSD.org>

```
pub   1024D/2675AB69 2008-02-17
      Key fingerprint = 00D2 E063 2FB0 2990 C602  50FD C1C2 7889 2675 AB69
uid                  Alexander V. Chernikov <melifaro@yandex-team.ru>
uid                  Alexander V. Chernikov <melifaro@ipfw.ru>
uid                  Alexander V. Chernikov <melifaro@freebsd.org>
```

```
sub   4096g/BC64F40C 2008-02-17
```

D.3.58 Sean Chittenden <seanc@FreeBSD.org>

```
pub  1024D/EE278A28 2004-02-08 Sean Chittenden <sean@chittenden.org>
     Key fingerprint = E41F F441 7E91 6CBA 1844  65CF B939 3C78 EE27 8A28
sub  2048g/55321853 2004-02-08
```

D.3.59 Junho CHOI <cjh@FreeBSD.org>

```
pub  1024D/E60260F5 2002-10-14 CHOI Junho (Work) <cjh@wdb.co.kr>
     Key fingerprint = 1369 7374 A45F F41A F3C0  07E3 4A01 C020 E602 60F5
uid                            CHOI Junho (Personal) <cjh@kr.FreeBSD.org>
uid                            CHOI Junho (FreeBSD) <cjh@FreeBSD.org>
sub  1024g/04A4FDD8 2002-10-14
```

D.3.60 Crist J. Clark <cjc@FreeBSD.org>

```
pub  1024D/FE886AD3 2002-01-25 Crist J. Clark <cjclark@jhu.edu>
     Key fingerprint = F04E CCD7 3834 72C2 707F  0A8F 259F 8F4B FE88 6AD3
uid                            Crist J. Clark <cjclark@alum.mit.edu>
uid                            Crist J. Clark <cjc@freebsd.org>
sub  1024g/9B6BAB99 2002-01-25
```

D.3.61 Joe Marcus Clarke <marcus@FreeBSD.org>

```
pub  1024D/FE14CF87 2002-03-04 Joe Marcus Clarke (FreeBSD committer address) <marcus@FreeBSD.org>
     Key fingerprint = CC89 6407 73CC 0286 28E4  AFB9 6F68 8F8A FE14 CF87
uid                            Joe Marcus Clarke <marcus@marcuscom.com>
sub  1024g/B9ACE4D2 2002-03-04
```

D.3.62 Nik Clayton <nik@FreeBSD.org>

```
pub  1024D/2C37E375 2000-11-09 Nik Clayton <nik@freebsd.org>
     Key fingerprint = 15B8 3FFC DDB4 34B0 AA5F  94B7 93A8 0764 2C37 E375
uid                            Nik Clayton <nik@slashdot.org>
uid                            Nik Clayton <nik@crf-consulting.co.uk>
uid                            Nik Clayton <nik@ngo.org.uk>
uid                            Nik Clayton <nik@bsdi.com>
sub  1024g/769E298A 2000-11-09
```

D.3.63 Benjamin Close <benjsc@FreeBSD.org>

```
pub   1024D/4842B5B4 2002-04-10
      Key fingerprint = F00D C83D 5F7E 5561 DF91  B74D E602 CAA3 4842 B5B4
uid                  Benjamin Simon Close <Benjamin.Close@clearchain.com>
uid                  Benjamin Simon Close <benjsc@FreeBSD.org>
uid                  Benjamin Simon Close <benjsc@clearchain.com>
sub   2048g/3FA8A57E 2002-04-10
```

D.3.64 Tijl Coosemans <tijl@FreeBSD.org>

```
pub   2048D/20A0B62B 2010-07-13
      Key fingerprint = 39AA F580 6B44 5161 9F86  ED49 7E80 92D8 20A0 B62B
uid                  Tijl Coosemans <tijl@coosemans.org>
uid                  Tijl Coosemans <tijl@freebsd.org>
sub   2048g/7D71BA74 2010-07-13
```

D.3.65 Raphael Kubo da Costa <rakuco@FreeBSD.org>

```
pub   4096R/18DCEED6 2011-10-03
      Key fingerprint = 6911 54FE BA6E 6106 5789  7099 8DD0 7D21 18DC EED6
uid                  Raphael Kubo da Costa (Personal key) <rakuco@FreeBSD.org>
```

D.3.66 Alan L. Cox <alc@FreeBSD.org>

```
pub   2048R/33E2893B 2013-06-15
      Key fingerprint = FC7C 93FD 2C2C ABA5 C1D1  3E74 8513 043C 33E2 893B
uid                  Alan Cox <alc@FreeBSD.org>
uid                  Alan Cox <alc@cs.rice.edu>
uid                  Alan Cox <alc@rice.edu>
sub   2048R/693757AA 2013-06-15
```

D.3.67 Bruce Cran <brucec@FreeBSD.org>

```
pub   2048R/6AF6F99E 2010-01-29
      Key fingerprint = 9A3C AE57 2706 B0E3 4B8A  8374 5787 A72B 6AF6 F99E
uid                  Bruce Cran <brucec@FreeBSD.org>
uid                  Bruce Cran <bruce@cran.org.uk>
sub   2048R/1D665CEE 2010-01-29
```

D.3.68 Frederic Culot <culot@FreeBSD.org>

```
pub   1024D/34876C5B 2006-08-26
      Key fingerprint = 50EE CE94 E43E BA85 CB67  262B B739 1A26 3487 6C5B
uid                  Frederic Culot <culot@FreeBSD.org>
uid                  Frederic Culot <frederic@culot.org>
sub   2048g/F1EF901F 2006-08-26
```

D.3.69 Aaron Dalton <aaron@FreeBSD.org>

```
pub   1024D/8811D2A4 2006-06-21 [expires: 2011-06-20]
      Key fingerprint = 8DE0 3CBB 3692 992F 53EF  ACC7 BE56 0A4D 8811 D2A4
uid                  Aaron Dalton <aaron@freebsd.org>
sub   2048g/304EE8E5 2006-06-21 [expires: 2011-06-20]
```

D.3.70 Baptiste Daroussin <bapt@FreeBSD.org>

```
pub   1024D/49A4E84C 2008-11-19
      Key fingerprint = A14B A5FC B860 86DE 73E2  B24C F244 ED31 49A4 E84C
uid                  Baptiste Daroussin <bapt@etoilebsd.net>
uid                  Baptiste Daroussin <baptiste.daroussin@gmail.com>
uid                  Baptiste Daroussin <bapt@FreeBSD.org>
sub   2048g/54AB46B4 2008-11-19
```

D.3.71 Ceri Davies <ceri@FreeBSD.org>

```
pub   1024D/34B7245F 2002-03-08
      Key fingerprint = 9C88 EB05 A908 1058 A4AE  9959 A1C7 DCC1 34B7 245F
uid                  Ceri Davies <ceri@submonkey.net>
uid                  Ceri Davies <ceri@FreeBSD.org>
uid                  Ceri Davies <ceri@opensolaris.org>
sub   1024g/0C482CBC 2002-03-08
```

D.3.72 Brad Davis <brd@FreeBSD.org>

```
pub   1024D/ED0A754D 2005-05-14 [expires: 2014-02-21]
      Key fingerprint = 5DFD D1A6 BEEE A6D4 B3F5  4236 D362 3291 ED0A 754D
uid                  Brad Davis <so14k@so14k.com>
uid                  Brad Davis <brd@FreeBSD.org>
sub   2048g/1F29D404 2005-05-14 [expires: 2014-02-21]
```

D.3.73 Pawel Jakub Dawidek <pjd@FreeBSD.org>

```
pub  1024D/B1293F34 2004-02-02 Pawel Jakub Dawidek <Pawel@Dawidek.net>
     Key fingerprint = A3A3 5B4D 9CF9 2312 0783  1B1D 168A EF5D B129 3F34
uid                            Pawel Jakub Dawidek <pjd@FreeBSD.org>
uid                            Pawel Jakub Dawidek <pjd@FreeBSD.pl>
sub  2048g/3EEC50A7 2004-02-02 [expires: 2006-02-01]
```

D.3.74 Brian S. Dean <bsd@FreeBSD.org>

```
pub  1024D/723BDEE9 2002-01-23 Brian S. Dean <bsd@FreeBSD.org>
     Key fingerprint = EF49 7ABE 47ED 91B3 FC3D  7EA5 4D90 2FF7 723B DEE9
sub  1024g/4B02F876 2002-01-23
```

D.3.75 Carl Delsey <carl@FreeBSD.org>

```
pub   4096R/FB3B5D38 2013-01-15
      Key fingerprint = F0E5 3849 C6C3 668B 68A3  BCC7 6031 E963 FB3B 5D38
uid                  Carl Delsey <carl@FreeBSD.org>
sub   4096R/256F29D3 2013-01-15
```

D.3.76 Vasil Dimov <vd@FreeBSD.org>

```
pub   1024D/F6C1A420 2004-12-08
      Key fingerprint = B1D5 04C6 26CC 0D20 9525  14B8 170E 923F F6C1 A420
uid                  Vasil Dimov <vd@FreeBSD.org>
uid                  Vasil Dimov <vd@datamax.bg>
sub   4096g/A0148C94 2004-12-08
```

D.3.77 Roman Divacky <rdivacky@FreeBSD.org>

```
pub   1024D/3DC2044C 2006-11-15
      Key fingerprint = 6B61 25CA 49BC AAC5 21A9  FA7A 2D51 23E8 3DC2 044C
uid                  Roman Divacky <rdivacky@freebsd.org>
sub   2048g/39BDCE16 2006-11-15
```

D.3.78 Alexey Dokuchaev <danfe@FreeBSD.org>

```
pub  1024D/3C060B44 2004-08-23 Alexey Dokuchaev <danfe@FreeBSD.org>
     Key fingerprint = D970 08A4 922C 8D63 0C19  8D27 F421 76EE 3C06 0B44
sub  1024g/70BAE967 2004-08-23
```

D.3.79 Dima Dorfman <dd@FreeBSD.org>

```
pub   1024D/69FAE582 2001-09-04
      Key fingerprint = B340 8338 7DA3 4D61 7632  098E 0730 055B 69FA E582
uid                  Dima Dorfman <dima@trit.org>
uid                  Dima Dorfman <dima@unixfreak.org>
uid                  Dima Dorfman <dd@freebsd.org>
sub   2048g/65AF3B89 2003-08-19 [expires: 2005-08-18]
sub   2048g/8DB0CF2C 2005-05-29 [expires: 2007-05-29]
```

D.3.80 Bryan Drewery <bdrewery@FreeBSD.org>

```
pub   4096R/3C9B0CF9 2012-04-06 [expires: 2017-04-05]
      Key fingerprint = 36FE BE99 2F52 80DF 4811  362A 6E78 2AC0 3C9B 0CF9
uid                  Bryan Drewery <bryan@shatow.net>
uid                  Bryan Drewery <bdrewery@gmail.com>
uid                  Bryan Drewery <bryan@xzibition.com>
uid                  Bryan Drewery <bdrewery@FreeBSD.org>
sub   4096R/9E2CE2D3 2012-04-06 [expires: 2017-04-05]
```

D.3.81 Olivier Duchateau <olivierd@FreeBSD.org>

```
pub   2048R/22431859 2012-05-28 [expires: 2017-05-27]
      Key fingerprint = C057 112A 4A27 B5F2 CD8F  6C9A FC5A 0167 2243 1859
uid                  Olivier Duchateau <duchateau.olivier@gmail.com>
sub   2048R/63A85BDF 2012-05-28 [expires: 2017-05-27]
```

D.3.82 Bruno Ducrot <bruno@FreeBSD.org>

```
pub   1024D/7F463187 2000-12-29
      Key fingerprint = 7B79 E1D6 F5A1 6614 792F  D906 899B 4D28 7F46 3187
uid                  Ducrot Bruno (Poup Master) <ducrot@poupinou.org>
sub   1024g/40282874 2000-12-29
```

D.3.83 Alex Dupre <ale@FreeBSD.org>

```
pub  1024D/CE5F554D 1999-06-27 Alex Dupre <sysadmin@alexdupre.com>
     Key fingerprint = DE23 02EA 5927 D5A9 D793  2BA2 8115 E9D8 CE5F 554D
uid                           Alex Dupre <ale@FreeBSD.org>
uid                           [jpeg image of size 5544]
uid                           Alex Dupre <ICQ:5431856>
sub  2048g/FD5E2D21 1999-06-27
```

D.3.84 Peter Edwards <peadar@FreeBSD.org>

```
pub  1024D/D80B4B3F 2004-03-01 Peter Edwards <peadar@FreeBSD.org>
     Key fingerprint = 7A8A 9756 903E BEF2 4D9E  3C94 EE52 52F7 D80B 4B3F
uid                           Peter Edwards <pmedwards@eircom.net>
```

D.3.85 Daniel Eischen <deischen@FreeBSD.org>

```
pub   4096R/7D15560B 2012-11-17
      Key fingerprint = 0039 2133 69CA 14D3 236A  E331 361A 68B2 7D15 560B
uid                  Daniel Eischen <deischen@FreeBSD.org>
sub   4096R/A51F81F7 2012-11-17
```

D.3.86 Josef El-Rayes <josef@FreeBSD.org>

```
pub  2048R/A79DB53C 2004-01-04 Josef El-Rayes <josef@FreeBSD.org>
     Key fingerprint = 58EB F5B7 2AB9 37FE 33C8  716B 59C5 22D9 A79D B53C
uid                           Josef El-Rayes <josef@daemon.li>
```

D.3.87 Lars Engels <lme@FreeBSD.org>

```
pub   1024D/C0F769F8 2004-08-27
      Key fingerprint = 17FC 08E1 5E09 BD21 489E  2050 29CE 75DA C0F7 69F8
uid                  Lars Engels <lars.engels@0x20.net>
sub   1024g/8AD5BF9D 2004-08-27
```

D.3.88 Udo Erdelhoff <ue@FreeBSD.org>

```
pub  1024R/E74FA871 1994-07-19 Udo Erdelhoff <uer@de.uu.net>
     Key fingerprint = 8C B1 80 CA 2C 52 73 81  FB A7 B4 03 C5 32 C8 67
uid                           Udo Erdelhoff <ue@nathan.ruhr.de>
uid                           Udo Erdelhoff <ue@freebsd.org>
uid                           Udo Erdelhoff <uerdelho@eu.uu.net>
uid                           Udo Erdelhoff <uerdelho@uu.net>
```

D.3.89 Ruslan Ermilov <ru@FreeBSD.org>

```
pub  1024D/996E145E 2004-06-02 Ruslan Ermilov (FreeBSD) <ru@FreeBSD.org>
     Key fingerprint = 274E D201 71ED 11F6 9CCB  0194 A917 E9CC 996E 145E
uid                           Ruslan Ermilov (FreeBSD Ukraine) <ru@FreeBSD.org.ua>
uid                           Ruslan Ermilov (IPNet) <ru@ip.net.ua>
sub  1024g/557E3390 2004-06-02 [expires: 2007-06-02]
```

D.3.90 Lukas Ertl <le@FreeBSD.org>

```
pub  1024D/F10D06CB 2000-11-23 Lukas Ertl <le@FreeBSD.org>
     Key fingerprint = 20CD C5B3 3A1D 974E 065A  B524 5588 79A9 F10D 06CB
uid                            Lukas Ertl <a9404849@unet.univie.ac.at>
uid                            Lukas Ertl <l.ertl@univie.ac.at>
uid                            Lukas Ertl <le@univie.ac.at>
sub  1024g/5960CE8E 2000-11-23
```

D.3.91 Brendan Fabeny <bf@FreeBSD.org>

```
pub   2048R/9806EBC1 2010-06-08 [expires: 2012-06-07]
      Key fingerprint = 2075 ADD3 7634 A4F9 5357  D934 08E7 06D9 9806 EBC1
uid                  b. f. <bf@freebsd.org>
sub   2048R/1CD0AD79 2010-06-08 [expires: 2012-06-07]
```

D.3.92 Guido Falsi <madpilot@FreeBSD.org>

```
pub   2048R/56CBD293 2012-04-12
      Key fingerprint = F317 2057 E17E 4E3A 3DA5  9E1D 1AE6 860E 56CB D293
uid                  Guido Falsi <madpilot@FreeBSD.org>
uid                  Guido Falsi <mad@madpilot.net>
sub   2048R/1F9772C5 2012-04-12
```

D.3.93 Rong-En Fan <rafan@FreeBSD.org>

```
pub   1024D/86FD8C68 2004-06-04
      Key fingerprint = DC9E 5B4D 2DDA D5C7 B6F8  6E69 D78E 1091 86FD 8C68
uid                  Rong-En Fan <rafan@infor.org>
uid                  Rong-En Fan <rafan@csie.org>
uid                  Rong-En Fan <rafan@FreeBSD.org>
sub   2048g/42A8637E 2009-01-25 [expires: 2012-07-08]
```

D.3.94 Stefan Farfeleder <stefanf@FreeBSD.org>

```
pub  1024D/8BEFD15F 2004-03-14 Stefan Farfeleder <stefan@fafoe.narf.at>
     Key fingerprint = 4220 FE60 A4A1 A490 5213  27A6 319F 8B28 8BEF D15F
uid                            Stefan Farfeleder <stefanf@complang.tuwien.ac.at>
uid                            Stefan Farfeleder <stefanf@FreeBSD.org>
uid                            Stefan Farfeleder <stefanf@ten15.org>
sub  2048g/418753E9 2004-03-14 [expires: 2007-03-14]
```

D.3.95 Babak Farrokhi <farrokhi@FreeBSD.org>

```
pub   1024D/7C810476 2005-12-22
      Key fingerprint = AABD 388F A207 58B4 2EE3  5DFD 4FC1 32C3 7C81 0476
uid                  Babak Farrokhi <farrokhi@FreeBSD.org>
uid                  Babak Farrokhi <babak@farrokhi.net>
sub   2048g/2A5F93C7 2005-12-22
```

D.3.96 Chris D. Faulhaber <jedgar@FreeBSD.org>

```
pub  1024D/FE817A50 2000-12-20 Chris D. Faulhaber <jedgar@FreeBSD.org>
     Key fingerprint = A47D A838 9216 F921 A456  54FF 39B6 86E0 FE81 7A50
uid                            Chris D. Faulhaber <jedgar@fxp.org>
sub  2048g/93452698 2000-12-20
```

D.3.97 Mark Felder <feld@FreeBSD.org>

```
pub   2048R/E64C94FE 2013-06-25
      Key fingerprint = 71ED 6A7F F4D7 430A BDF3  A180 BF01 619F E64C 94FE
uid                  Mark Felder <feld@freebsd.org>
sub   2048R/FDC20CA9 2013-06-25
```

D.3.98 Brian F. Feldman <green@FreeBSD.org>

```
pub  1024D/41C13DE3 2000-01-11 Brian Fundakowski Feldman <green@FreeBSD.org>
     Key fingerprint = 6A32 733A 1BF6 E07B 5B8D  AE14 CC9D DCA2 41C1 3DE3
sub  1024g/A98B9FCC 2000-01-11 [expires: 2001-01-10]

pub  1024D/773905D6 2000-09-02 Brian Fundakowski Feldman <green@FreeBSD.org>
     Key fingerprint = FE23 7481 91EA 5E58 45EA  6A01 B552 B043 7739 05D6
sub  2048g/D2009B98 2000-09-02
```

D.3.99 Mário Sérgio Fujikawa Ferreira <lioux@FreeBSD.org>

```
pub   1024D/75A63712 2006-02-23 [expires: 2007-02-23]
      Key fingerprint = 42F2 2F74 8EF9 5296 898F  C981 E9CF 463B 75A6 3712
uid                  Mario Sergio Fujikawa Ferreira (lioux) <lioux@FreeBSD.org>
uid                  Mario Sergio Fujikawa Ferreira <lioux@uol.com.br>
sub   4096g/BB7D80F2 2006-02-23 [expires: 2007-02-23]
```

D.3.100 Matthew Fleming <mdf@FreeBSD.org>

```
pub   2048R/A783DAA2 2012-11-22 [expires: 2016-11-22]
      Key fingerprint = 773F E069 BE98 CE96 4AC6  B8AB 1A1B 255E A783 DAA2
uid                  Matthew D Fleming <mdf356@gmail.com>
uid                  Matthew D Fleming <mdf@FreeBSD.org>
sub   2048R/4015B7AA 2012-11-22 [expires: 2016-11-22]
```

D.3.101 Tony Finch <fanf@FreeBSD.org>

```
pub  1024D/84C71B6E 2002-05-03 Tony Finch <dot@dotat.at>
     Key fingerprint = 199C F25B 2679 6D04 63C5  2159 FFC0 F14C 84C7 1B6E
uid                            Tony Finch <fanf@FreeBSD.org>
uid                            Tony Finch <fanf@apache.org>
uid                            Tony Finch <fanf2@cam.ac.uk>
sub  2048g/FD101E8B 2002-05-03
```

D.3.102 Marc Fonvieille <blackend@FreeBSD.org>

```
pub  1024D/4F8E74E8 2004-12-25 Marc Fonvieille <blackend@FreeBSD.org>
     Key fingerprint = 55D3 4883 4A04 828A A139  A5CF CD0F 51C0 4F8E 74E8
uid                            Marc Fonvieille <marc@blackend.org>
uid                            Marc Fonvieille <marc@freebsd-fr.org>
sub  1024g/37AD4E7D 2004-12-25
```

D.3.103 Pete Fritchman <petef@FreeBSD.org>

```
pub  1024D/74B91CFD 2001-01-30 Pete Fritchman <petef@FreeBSD.org>
     Key fingerprint = 9A9F 8A13 DB0D 7777 8D8E  1CB2 C5C9 A08F 74B9 1CFD
uid                            Pete Fritchman <petef@databits.net>
uid                            Pete Fritchman <petef@csh.rit.edu>
sub  1024g/0C02AF0C 2001-01-30
```

D.3.104 Bernhard Fröhlich <decke@FreeBSD.org>

```
pub   1024D/CF5840D4 2008-01-07 [expires: 2015-05-05]
      Key fingerprint = 47F6 BDF1 DF9E 81E2 2C54  8A06 E796 7A5A CF58 40D4
uid                  Bernhard Fröhlich <decke@FreeBSD.org>
uid                  Bernhard Fröhlich <decke@bluelife.at>
sub   2048g/4E51CE79 2008-01-07
```

D.3.105 Bill Fumerola <billf@FreeBSD.org>

```
pub  1024D/7F868268 2000-12-07 Bill Fumerola (FreeBSD Developer) <billf@FreeBSD.org>
     Key fingerprint = 5B2D 908E 4C2B F253 DAEB  FC01 8436 B70B 7F86 8268
uid                            Bill Fumerola (Security Yahoo) <fumerola@yahoo-inc.com>
sub  1024g/43980DA9 2000-12-07
```

D.3.106 Andriy Gapon <avg@FreeBSD.org>

```
pub   2048R/A651FE2F 2009-02-16
      Key fingerprint = F234 4D58 DEFF 5E3A 4E0F  13BC 74A5 2D27 A651 FE2F
uid                  Andriy Gapon (FreeBSD) <avg@FreeBSD.org>
uid                  Andriy Gapon (FreeBSD) <avg@freebsd.org>
uid                  Andriy Gapon (FreeBSD) <avg@icyb.net.ua>
sub   4096R/F9A4D312 2009-02-16
```

D.3.107 Beat Gätzi <beat@FreeBSD.org>

```
pub   1024D/774249DB 2009-01-28 [expires: 2014-01-27]
      Key fingerprint = C410 3187 5B29 DD02 745F  0890 40C5 BCF7 7742 49DB
uid                  Beat Gaetzi <beat@FreeBSD.org>
sub   2048g/173CFFCA 2009-01-28 [expires: 2014-01-27]
```

D.3.108 Daniel Geržo <danger@FreeBSD.org>

```
pub   1024D/DA913352 2007-08-30 [expires: 2008-08-29]
      Key fingerprint = 7372 3F15 F839 AFF5 4052  CAC7 1ADA C204 DA91 3352
uid                  Daniel Gerzo <gerzo@rulez.sk>
uid                  Daniel Gerzo <danger@rulez.sk>
uid                  Daniel Gerzo (The FreeBSD Project) <danger@FreeBSD.org>
uid                  Daniel Gerzo (Micronet, a.s.) <gerzo@micronet.sk>
sub   2048g/C5D57BDC 2007-08-30 [expires: 2008-08-29]
```

D.3.109 Simon J. Gerraty <sjg@FreeBSD.org>

```
pub   1024D/B6CC76BF 2002-06-12
      Key fingerprint = F3BA D6CB E1F8 02EA 705F  BCAD 6125 F840 B6CC 76BF
uid                  Simon J. Gerraty <sjg@crufty.net>
uid                  Simon J. Gerraty <sjg@juniper.net>
uid                  Simon J. Gerraty <sjg@NetBSD.org>
uid                  Simon J. Gerraty <sjg@FreeBSD.org>
sub   1024g/D94B72B9 2002-06-12
```

D.3.110 Justin T. Gibbs <gibbs@FreeBSD.org>

```
pub   2048R/45A4FC2F 2012-02-10
      Key fingerprint = B98A C3AB 412B 094B D6FE  E713 FA5A 1E30 45A4 FC2F
uid                  Justin T. Gibbs <gibbs@FreeBSD.org>
uid                  Justin T. Gibbs <gibbs@FreeBSDFoundation.org>
uid                  Justin T. Gibbs <gibbs@scsiguy.com>
sub   2048R/AF6927F8 2012-02-10
```

D.3.111 Pedro Giffuni <pfg@FreeBSD.org>

```
pub   2048D/422BDFE4 2011-12-06
      Key fingerprint = A12B 7C6B 54C0 921B C64F  7B35 58DF 6813 422B DFE4
uid                  Pedro Giffuni (FreeBSD key signature) <pfg@FreeBSD.org>
sub   2048g/43A91DE0 2011-12-06
```

D.3.112 Palle Girgensohn <girgen@FreeBSD.org>

```
pub   2048R/4A6BAAAD 2012-02-23 [expires: 2016-02-23]
      Key fingerprint = BD8C 332C E630 31D6 2FDB  80BD 5FF2 A161 4A6B AAAD
uid                  Palle Girgensohn <girgen@pingpong.net>
uid                  [jpeg image of size 8260]
uid                  Palle Girgensohn <girgen@FreeBSD.org>
sub   2048R/6BC41243 2012-02-23 [expires: 2016-02-23]
```

D.3.113 Philip M. Gollucci <pgollucci@FreeBSD.org>

```
pub   1024D/DB9B8C1C 2008-04-15
      Key fingerprint = B90B FBC3 A3A1 C71A 8E70  3F8C 75B8 8FFB DB9B 8C1C
uid                  Philip M. Gollucci (FreeBSD Foundation) <pgollucci@freebsd.org>
uid                  Philip M. Gollucci (Riderway Inc.) <pgollucci@riderway.com>
uid                  Philip M. Gollucci <pgollucci@p6m7g8.com>
uid                  Philip M. Gollucci (ASF) <pgollucci@apache.org>
sub   2048g/73943732 2008-04-15
```

D.3.114 Daichi GOTO <daichi@FreeBSD.org>

```
pub  1024D/09EBADD6 2002-09-25 Daichi GOTO <daichi@freebsd.org>
     Key fingerprint = 620A 9A34 57FB 5E93 0828  28C7 C360 C6ED 09EB ADD6
sub  1024g/F0B1F1CA 2002-09-25
```

D.3.115 Marcus Alves Grando <mnag@FreeBSD.org>

```
pub   1024D/CDCC273F 2005-09-15 [expires: 2010-09-14]
      Key fingerprint = 57F9 DEC1 5BBF 06DE 44A5  9A4A 8BEE 5F3A CDCC 273F
uid                  Marcus Alves Grando <marcus@sbh.eng.br>
uid                  Marcus Alves Grando <marcus@corp.grupos.com.br>
uid                  Marcus Alves Grando <mnag@FreeBSD.org>
sub   2048g/698AC00C 2005-09-15 [expires: 2010-09-14]
```

D.3.116 Peter Grehan <grehan@FreeBSD.org>

```
pub  1024D/EA45EA7D 2004-07-13 Peter Grehan <grehan@freebsd.org>
     Key fingerprint = 84AD 73DC 370E 15CA 7556  43C8 F5C8 4450 EA45 EA7D
sub  2048g/0E122D70 2004-07-13
```

D.3.117 Jamie Gritton <jamie@FreeBSD.org>

```
pub   1024D/8832CB7F 2009-01-29
      Key fingerprint = 34F8 1E62 C7A5 7CB9 A91F  7864 8C5A F85E 8832 CB7F
uid                  James Gritton <jamie@FreeBSD.org>
sub   2048g/94E3594D 2009-01-29
```

D.3.118 William Grzybowski <wg@FreeBSD.org>

```
pub   2048R/CFC460C5 2012-09-28
      Key fingerprint = FC40 5CD8 0879 7F50 0036  D924 D9F7 8B27 CFC4 60C5
uid                  William Grzybowski (FreeBSD) <wg@freebsd.org>
uid                  William Grzybowski <william88@gmail.com>
sub   2048R/05577997 2012-09-28
```

D.3.119 Barbara Guida <bar@FreeBSD.org>

```
pub   2048R/3DF5F750 2012-11-13
      Key fingerprint = D367 F6C8 2A5F 2921 70D2  B446 27DD 6FD6 3DF5 F750
uid                  Barbara Guida <bar@FreeBSD.org>
uid                  Barbara Guida <barbara.freebsd@gmail.com>
sub   2048R/1DF7506C 2012-11-13
```

D.3.120 John-Mark Gurney <jmg@FreeBSD.org>

```
pub   1024D/6D3FA396 2011-03-03 [expires: 2016-03-01]
      Key fingerprint = 54BA 873B 6515 3F10 9E88  9322 9CB1 8F74 6D3F A396
uid                  John-Mark Gurney <jmg@FreeBSD.org>
uid                  John-Mark Gurney <jmg@funkthat.com>
```

```
sub   4096g/0A4C095E 2011-03-03 [expires: 2016-03-01]
```

D.3.121 Mateusz Guzik <mjg@FreeBSD.org>

```
pub   2048R/21489259 2012-06-03
      Key fingerprint = 3A9F 25FF ABF6 BB23 5C70  C61B 96D3 5178 2148 9259
uid                  Mateusz Guzik <mjg@freebsd.org>
sub   2048R/EA19FE8D 2012-06-03
```

D.3.122 Jason E. Hale <jhale@FreeBSD.org>

```
pub   3072D/8F2E5907 2012-09-07
      Key fingerprint = 009C 54BF 32D0 F373 8126  C8A1 D8DD 2CA4 8F2E 5907
uid                  Jason E. Hale <jhale@FreeBSD.org>
uid                  Jason E. Hale <bsdkaffee@gmail.com>
sub   4096g/7081A001 2012-09-07
```

D.3.123 Daniel Harris <dannyboy@FreeBSD.org>

```
pub  1024D/84D0D7E7 2001-01-15 Daniel Harris <dannyboy@worksforfood.com>
     Key fingerprint = 3C61 B8A1 3F09 D194 3259  7173 6C63 DA04 84D0 D7E7
uid                            Daniel Harris <dannyboy@freebsd.org>
uid                            Daniel Harris <dh@askdh.com>
uid                            Daniel Harris <dh@wordassault.com>
sub  1024g/9DF0231A 2001-01-15
```

D.3.124 Daniel Hartmeier <dhartmei@FreeBSD.org>

```
pub  1024R/6A3A7409 1994-08-15 Daniel Hartmeier <dhartmei@freebsd.org>
     Key fingerprint = 13 7E 9A F3 36 82 09 FE  FD 57 B8 5C 2B 81 7E 1F
```

D.3.125 Olli Hauer <ohauer@FreeBSD.org>

```
pub   2048R/5D008F1A 2010-07-26
      Key fingerprint = E9EE C9A5 EB4C BD29 74D7  9178 E56E 06B3 5D00 8F1A
uid                  olli hauer <ohauer@FreeBSD.org>
uid                  olli hauer <ohauer@gmx.de>
sub   2048R/5E25776E 2010-07-26
```

D.3.126 Emanuel Haupt <ehaupt@FreeBSD.org>

```
pub   3072D/329A273C 2012-11-17 [expires: 2013-11-17]
      Key fingerprint = 920C A49A 5A23 F9E3 4EB0  4387 AB90 5C56 329A 273C
uid                  Emanuel Haupt <ehaupt@FreeBSD.org>
sub   3072g/70183B96 2012-11-17 [expires: 2013-11-17]
```

D.3.127 John Hay <jhay@FreeBSD.org>

```
pub  2048R/A9275B93 2000-05-10 John Hay <jhay@icomtek.csir.co.za>
     Key fingerprint = E7 95 F4 B9 D4 A7 49 6A  83 B9 77 49 28 9E 37 70
uid                            John Hay <jhay@mikom.csir.co.za>
uid                            Thawte Freemail Member <jhay@mikom.csir.co.za>
uid                            John Hay <jhay@csir.co.za>
uid                            John Hay <jhay@FreeBSD.ORG>
```

D.3.128 Sheldon Hearn <sheldonh@FreeBSD.org>

```
pub  1024D/74A06ACD 2002-06-20 Sheldon Hearn <sheldonh@starjuice.net>
     Key fingerprint = 01A3 EF91 9C5A 3633 4E01  8085 A462 57F1 74A0 6ACD
sub  1536g/C42F8AC8 2002-06-20
```

D.3.129 Mike Heffner <mikeh@FreeBSD.org>

```
pub  1024D/CDECBF99 2001-02-02 Michael Heffner <mheffner@novacoxmail.com>
     Key fingerprint = AFAB CCEB 68C7 573F 5110  9285 1689 1942 CDEC BF99
uid                            Michael Heffner <mheffner@vt.edu>
uid                            Michael Heffner <mikeh@FreeBSD.org>
uid                            Michael Heffner <spock@techfour.net>
uid                            Michael Heffner (ACM sysadmin) <mheffner@acm.vt.edu>
sub  1024g/3FE83FB5 2001-02-02
```

D.3.130 Martin Heinen <mheinen@FreeBSD.org>

```
pub  1024D/116C5C85 2002-06-17 Martin Heinen <mheinen@freebsd.org>
     Key fingerprint = C898 3FCD EEA0 17ED BEA9  564D E5A6 AFF2 116C 5C85
uid                            Martin Heinen <martin@sumuk.de>
sub  1024g/EA67506B 2002-06-17
```

D.3.131 Niels Heinen <niels@FreeBSD.org>

```
pub  1024D/5FE39B80 2004-12-06 Niels Heinen <niels.heinen@ubizen.com>
     Key fingerprint = 75D8 4100 CF5B 3280 543F  930C 613E 71AA 5FE3 9B80
uid                            Niels Heinen <niels@defaced.be>
```

```
uid                           Niels Heinen <niels@heinen.ws>
uid                           Niels Heinen <niels@FreeBSD.org>
sub  2048g/057F4DA7 2004-12-06
```

D.3.132 Jaakko Heinonen <jh@FreeBSD.org>

```
pub   1024D/53CCB781 2009-10-01 [expires: 2014-09-30]
      Key fingerprint = 3AED A2B6 B63D D771 1AFD  25FA DFDF 5B89 53CC B781
uid                  Jaakko Heinonen (FreeBSD) <jh@FreeBSD.org>
sub   4096g/BB97397E 2009-10-01 [expires: 2014-09-30]
```

D.3.133 Jason Helfman <jgh@FreeBSD.org>

```
pub   2048R/4150D3DC 2011-12-18 [expires: 2021-12-15]
      Key fingerprint = 8E0D C457 9A0F C91C 23F3  0454 2059 9A63 4150 D3DC
uid                  Jason Helfman <jgh@FreeBSD.org>
sub   2048R/695B1B92 2011-12-18 [expires: 2021-12-15]
```

D.3.134 Guy Helmer <ghelmer@FreeBSD.org>

```
pub   2048R/8F1CEBC4 2012-05-22
      Key fingerprint = 483E 9E6C C644 2520 C9FE  4E87 9989 CCAF 8F1C EBC4
uid                  Guy Helmer <guy.helmer@palisadesystems.com>
uid                  Guy Helmer <guy.helmer@gmail.com>
uid                  Guy Helmer <ghelmer@freebsd.org>
sub   2048R/2073E3F8 2012-05-22

pub  1024R/35F4ED2D 1997-01-26 Guy G. Helmer <ghelmer@freebsd.org>
     Key fingerprint = A2 59 4B 92 02 5B 9E B1  B9 4E 2E 03 29 D5 DC 3A
uid                           Guy G. Helmer <ghelmer@cs.iastate.edu>
uid                           Guy G. Helmer <ghelmer@palisadesys.com>
```

D.3.135 Maxime Henrion <mux@FreeBSD.org>

```
pub  1024D/881D4806 2003-01-09 Maxime Henrion <mux@FreeBSD.org>
     Key fingerprint = 81F1 BE2D 12F1 184A 77E4  ACD0 5563 7614 881D 4806
sub  2048g/D0B510C0 2003-01-09
```

D.3.136 Wen Heping <wen@FreeBSD.org>

```
pub   2048R/A03F07DA 2012-12-10
      Key fingerprint = 0258 F2C7 C123 E627 9E14  B4BA 270F 30AA A03F 07DA
uid                  Wen Heping (wen) <wen@FreeBSD.org>
sub   2048R/CFC8D6A9 2012-12-10
```

D.3.137 Dennis Herrmann <dhn@FreeBSD.org>

```
pub   4096R/F7CDCAA1 2012-08-26
      Key fingerprint = 0587 E730 68A6 2646 A991  505D CD9B 3A87 F7CD CAA1
uid                  Dennis ’dhn’ Herrmann (Everybody wants to go to heaven, but nobody wants to d
sub   4096R/0A6D554F 2012-08-26
```

D.3.138 Justin Hibbits <jhibbits@FreeBSD.org>

```
pub   2048R/37BE2DB9 2011-12-01
      Key fingerprint = 8A12 7064 4F3D 339A 191D  AD52 30C7 858E 37BE 2DB9
uid                  Justin Hibbits <chmeeedalf@gmail.com>
uid                  Justin Hibbits <jhibbits@freebsd.org>
uid                  Justin Hibbits <jrh29@alumni.cwru.edu>
sub   2048R/A8DA156F 2011-12-01
```

D.3.139 Peter Holm <pho@FreeBSD.org>

```
pub   1024D/CF244E81 2008-11-17
      Key fingerprint = BE9B 32D8 89F1 F285 00E4  E4C5 EF3F B4B5 CF24 4E81
uid                  Peter Holm <pho@FreeBSD.org>
sub   2048g/E20A409F 2008-11-17
```

D.3.140 Michael L. Hostbaek <mich@FreeBSD.org>

```
pub  1024D/0F55F6BE 2001-08-07 Michael L. Hostbaek <mich@freebsdcluster.org>
     Key fingerprint = 4D62 9396 B19F 38D3 5C99  1663 7B0A 5212 0F55 F6BE
uid                           Michael L. Hostbaek <mich@freebsdcluster.dk>
uid                           Michael L. Hostbaek <mich@icommerce-france.com>
uid                           Micahel L. Hostbaek <mich@freebsd.dk>
uid                           Michael L. Hostbaek <mich@the-lab.org>
uid                           Michael L. Hostbaek <mich@freebsd.org>
sub  1024g/8BE4E30F 2001-08-07
```

D.3.141 Po-Chuan Hsieh <sunpoet@FreeBSD.org>

```
pub   4096R/CC57E36B 2010-09-21
      Key fingerprint = 8AD8 68F2 7D2B 0A10 7E9B  8CC0 DC44 247E CC57 E36B
uid                  Po-Chuan Hsieh (FreeBSD) <sunpoet@FreeBSD.org>
uid                  Po-Chuan Hsieh (sunpoet) <sunpoet@sunpoet.net>
sub   4096R/ADE9E203 2010-09-21
```

D.3.142 Li-Wen Hsu <lwhsu@FreeBSD.org>

```
pub   1024D/2897B228 2005-01-16
      Key fingerprint = B6F7 170A 6DC6 5D1A BD4B  D86A 416B 0E39 2897 B228
uid                  Li-wen Hsu <lwhsu@lwhsu.org>
uid                  Li-wen Hsu <lwhsu@lwhsu.ckefgisc.org>
uid                  Li-wen Hsu <lwhsu@lwhsu.csie.net>
uid                  Li-wen Hsu <lwhsu@ckefgisc.org>
uid                  Li-wen Hsu <lwhsu@csie.nctu.edu.tw>
uid                  Li-wen Hsu <lwhsu@ccca.nctu.edu.tw>
uid                  Li-wen Hsu <lwhsu@iis.sinica.edu.tw>
uid                  Li-wen Hsu <lwhsu@cs.nctu.edu.tw>
uid                  Li-Wen Hsu <lwhsu@FreeBSD.org>
sub   2048g/16F82238 2005-01-16
```

D.3.143 Howard F. Hu <foxfair@FreeBSD.org>

```
pub  1024D/4E9BCA59 2003-09-01 Foxfair Hu <foxfair@FreeBSD.org>
     Key fingerprint = 280C A846 CA1B CAC9 DDCF  F4CB D553 4BD5 4E9B CA59
uid                           Foxfair Hu <foxfair@drago.fomokka.net>
uid                           Howard Hu <howardhu@yahoo-inc.com>
sub  1024g/3356D8C1 2003-09-01
```

D.3.144 Chin-San Huang <chinsan@FreeBSD.org>

```
pub   1024D/350EECFA 2006-10-04
      Key fingerprint = 1C4D 0C9E 0E68 DB74 0688  CE43 D2A5 3F82 350E ECFA
uid                  Chin-San Huang (lab) <chinsan@chinsan2.twbbs.org>
uid                  Chin-San Huang (FreeBSD committer) <chinsan@FreeBSD.org>
uid                  Chin-San Huang (Gmail) <chinsan.tw@gmail.com>
sub   2048g/35F75A30 2006-10-04
```

D.3.145 Davide Italiano <davide@FreeBSD.org>

```
pub   2048R/4CB47484 2012-01-17
      Key fingerprint = B5C9 77F5 1E67 D110 8D19  7587 EB95 EA82 4CB4 7484
uid                  Davide Italiano <davide@FreeBSD.org>
sub   2048R/91F7443D 2012-01-17
```

D.3.146 Jordan K. Hubbard <jkh@FreeBSD.org>

```
pub  1024R/8E542D5D 1996-04-04 Jordan K. Hubbard <jkh@FreeBSD.org>
     Key fingerprint = 3C F2 27 7E 4A 6C 09 0A  4B C9 47 CD 4F 4D 0B 20
```

D.3.147 Konrad Jankowski <versus@FreeBSD.org>

```
pub   1024D/A01C218A 2008-10-28
      Key fingerprint = A805 21DC 859F E941 D2EA  9986 2264 8E5D A01C 218A
uid                  Konrad Jankowski <versus@freebsd.org>
sub   2048g/56AE1959 2008-10-28
```

D.3.148 Weongyo Jeong <weongyo@FreeBSD.org>

```
pub   1024D/22354D7A 2007-12-28
      Key fingerprint = 138E 7115 A86F AA40 B509  5883 B387 DCE9 2235 4D7A
uid                  Weongyo Jeong <weongyo.jeong@gmail.com>
uid                  Weongyo Jeong <weongyo@freebsd.org>
sub   2048g/9AE6DAEE 2007-12-28
```

D.3.149 Peter Jeremy <peterj@FreeBSD.org>

```
pub   1024D/F00FB887 2005-10-20
      Key fingerprint = 0BF7 7A72 5894 EBE6 4F4D  7EEE FE8A 47BF F00F B887
uid                  Peter Jeremy <peterjeremy@acm.org>
uid                  [jpeg image of size 4413]
uid                  Peter Jeremy <peter.jeremy@auug.org.au>
uid                  Peter Jeremy <peterjeremy@optusnet.com.au>
uid                  Peter Jeremy (preferred) <peter@rulingia.com>
uid                  Peter Jeremy <peterj@freebsd.org>
sub   2048g/7E0B423B 2005-10-20
```

D.3.150 Tatuya JINMEI <jinmei@FreeBSD.org>

```
pub   1024D/ABA82228 2002-08-15
      Key fingerprint = BB70 3050 EE39 BE00 48BB  A5F3 5892 F203 ABA8 2228
uid                  JINMEI Tatuya <jinmei@FreeBSD.org>
uid                  JINMEI Tatuya <jinmei@jinmei.org>
uid                  JINMEI Tatuya (the KAME project) <jinmei@isl.rdc.toshiba.co.jp>
sub   1024g/8B43CF66 2002-08-15
```

D.3.151 Michael Johnson <ahze@FreeBSD.org>

```
pub  1024D/3C046FD6 2004-10-29 Michael Johnson (FreeBSD key) <ahze@FreeBSD.org>
     Key fingerprint = 363C 6ABA ED24 C23B 5F0C  3AB4 9F8B AA7D 3C04 6FD6
uid                           Michael Johnson (pgp key) <ahze@ahze.net>
sub  2048g/FA334AE3 2004-10-29
```

D.3.152 Mark Johnston <markj@FreeBSD.org>

```
pub   2048R/80A62628 2012-12-19
      Key fingerprint = AFEF AD33 1C4E FFE5 141E  0157 05A4 DA8B 80A6 2628
uid                  Mark Johnston <markj@freebsd.org>
sub   2048R/47C7D3C2 2012-12-19
```

D.3.153 Trevor Johnson <trevor@FreeBSD.org>

```
pub  1024D/3A3EA137 2000-04-20 Trevor Johnson <trevor@jpj.net>
     Key fingerprint = 7ED1 5A92 76C1 FFCB E5E3  A998 F037 5A0B 3A3E A137
sub  1024g/46C24F1E 2000-04-20
```

D.3.154 Tom Judge <tj@FreeBSD.org>

```
pub   2048R/81E22216 2012-05-27 [expires: 2017-05-26]
      Key fingerprint = 8EF8 36C8 44A6 9576 6ADB  EB0E 4252 33DC 81E2 2216
uid                  Tom Judge <tom@tomjudge.com>
uid                  Tom Judge <tjudge@sourcefire.com>
uid                  Tom Judge <tj@freebsd.org>
sub   2048R/2CA4AA0D 2012-05-27 [expires: 2017-05-26]
```

D.3.155 Alexander Kabaev <kan@FreeBSD.org>

```
pub   1024D/C9BE5D96 2002-07-01
      Key fingerprint = 7474 A847 DBF5 50A5 FC3E  F223 43AC F58C C9BE 5D96
uid                  Alexander Kabaev <kabaev@gmail.com>
uid                  Alexander Kabaev (FreeBSD committer account ID) <kan@FreeBSD.ORG>
sub   1024g/534D9E06 2002-07-01
```

D.3.156 Benjamin Kaduk <bjk@FreeBSD.org>

```
pub   4096R/8302FE9F 2011-08-20 [expires: 2013-07-21]
      Key fingerprint = 9FD9 F966 D914 5101 BE59  FE13 2D29 EEED 8302 FE9F
uid                  Benjamin Kaduk <bjk@FreeBSD.org>
sub   4096R/28698ABE 2011-08-20 [expires: 2013-08-19]
```

D.3.157 Poul-Henning Kamp <phk@FreeBSD.org>

```
pub  1024R/0358FCBD 1995-08-01 Poul-Henning Kamp <phk@FreeBSD.org>
     Key fingerprint = A3 F3 88 28 2F 9B 99 A2  49 F4 E2 FA 5A 78 8B 3E
```

D.3.158 Sergey Kandaurov <pluknet@FreeBSD.org>

```
pub   2048R/10607419 2010-10-04
      Key fingerprint = 020B EC25 7E1F 8BC5 C42C  513B 3F4E 97BA 1060 7419
uid                  Sergey Kandaurov (freebsd) <pluknet@freebsd.org>
uid                  Sergey Kandaurov <pluknet@gmail.com>
sub   2048R/5711F73B 2010-10-04
```

D.3.159 Coleman Kane <cokane@FreeBSD.org>

```
pub   1024D/C5DAB797 2007-07-22
      Key fingerprint = FC09 F326 4318 E714 DE45  6CB0 70C4 B141 C5DA B797
uid                  Coleman Kane (Personal PGP Key) <cokane@cokane.org>
uid                  Coleman Kane (Personal PGP Key) <cokane@FreeBSD.org>
sub   2048g/5C680129 2007-07-22
```

D.3.160 Takenori KATO <kato@FreeBSD.org>

```
pub   4096R/3CF9ACE7 2012-10-02
      Key fingerprint = 5B72 AEF9 B2F9 069D 54FE  CF60 444F 91C8 3CF9 ACE7
uid                  KATO Takenori <kato@FreeBSD.org>
uid                  KATO Takenori <kato@nendai.nagoya-u.ac.jp>
sub   4096R/1C593356 2012-10-02
```

D.3.161 Josef Karthauser <joe@FreeBSD.org>

```
pub  1024D/E6B15016 2000-10-19 Josef Karthauser <joe@FreeBSD.org>
     Key fingerprint = 7266 8EAF 82C2 D439 5642  AC26 5D52 1C8C E6B1 5016
uid                            Josef Karthauser <joe@tao.org.uk>
uid                            Josef Karthauser <joe@uk.FreeBSD.org>
uid                            [revoked] Josef Karthauser <josef@bsdi.com>
uid                            [revoked] Josef Karthauser <joe@pavilion.net>
sub  2048g/1178B692 2000-10-19
```

D.3.162 Vinod Kashyap <vkashyap@FreeBSD.org>

```
pub  1024R/04FCCDD3 2004-02-19 Vinod Kashyap (gnupg key) <vkashyap@freebsd.org>
     Key fingerprint = 9B83 0B55 604F E491 B7D2  759D DF92 DAA0 04FC CDD3
```

D.3.163 Kris Kennaway <kris@FreeBSD.org>

```
pub  1024D/68E840A5 2000-01-14 Kris Kennaway <kris@citusc.usc.edu>
     Key fingerprint = E65D 0E7D 7E16 B212 1BD6  39EE 5ABC B405 68E8 40A5
uid                            Kris Kennaway <kris@FreeBSD.org>
```

```
uid                           Kris Kennaway <kris@obsecurity.org>
sub  2048g/03A41C45 2000-01-14 [expires: 2006-01-14]
```

D.3.164 Giorgos Keramidas <keramida@FreeBSD.org>

```
pub   1024D/318603B6 2001-09-21
      Key fingerprint = C1EB 0653 DB8B A557 3829  00F9 D60F 941A 3186 03B6
uid                  Giorgos Keramidas <keramida@FreeBSD.org>
uid                  Giorgos Keramidas <keramida@ceid.upatras.gr>
uid                  Giorgos Keramidas <keramida@hellug.gr>
uid                  Giorgos Keramidas <keramida@linux.gr>
uid                  Giorgos Keramidas <gkeramidas@gmail.com>
sub   1024g/50FDBAD1 2001-09-21
```

D.3.165 Max Khon <fjoe@FreeBSD.org>

```
pub   1024D/6B87E212 2009-02-17
      Key fingerprint = 124D EC6C 6365 D41A 497A  9C3E FCF3 8708 6B87 E212
uid                  Max Khon <fjoe@FreeBSD.org>
uid                  Max Khon <fjoe@samodelkin.net>
sub   2048g/CB71491D 2009-02-17
```

D.3.166 Manolis Kiagias <manolis@FreeBSD.org>

```
pub   1024D/6E0FB494 2006-08-22
      Key fingerprint = F820 5AAF 7112 2CDD 23D8  3BDF 67F3 311A 6E0F B494
uid                  Manolis Kiagias <manolis@FreeBSD.org>
uid                  Manolis Kiagias <sonicy@otenet.gr>
uid                  Manolis Kiagias (A.K.A. sonic, sonicy, sonic2000gr) <sonic@diktia.dyndns.org
sub   2048g/EB94B411 2006-08-22
```

D.3.167 Jung-uk Kim <jkim@FreeBSD.org>

```
pub   2048R/D932A1CE 2012-11-19
      Key fingerprint = 2202 B5FB 78B7 A303 4919  B7C7 25E9 69B1 D932 A1CE
uid                  Jung-uk Kim <jkim@FreeBSD.org>
sub   2048R/41858FC6 2012-11-19
```

D.3.168 Zack Kirsch <zack@FreeBSD.org>

```
pub  1024D/1A725562 2010-11-05 Zack Kirsch <zack@freebsd.org>
     Key fingerprint = A8CC AA5E FB47 A386 E757  A2B8 BDD2 0684 1A72 5562
sub  1024g/6BFE2C06 2010-11-05
```

D.3.169 Jakub Klama <jceel@FreeBSD.org>

```
pub   2048R/2AAEA67D 2011-09-27
      Key fingerprint = 40D6 097A 174F 511B 80EB  F3A3 0946 4193 2AAE A67D
uid                  Jakub Klama <jceel@FreeBSD.org>
sub   2048R/5291BC4D 2011-09-27
```

D.3.170 Andreas Klemm <andreas@FreeBSD.org>

```
pub  1024D/6C6F6CBA 2001-01-06 Andreas Klemm <andreas.klemm@eu.didata.com>
     Key fingerprint = F028 D51A 0D42 DD67 4109  19A3 777A 3E94 6C6F 6CBA
uid                            Andreas Klemm <andreas@klemm.gtn.com>
uid                            Andreas Klemm <andreas@FreeBSD.org>
uid                            Andreas Klemm <andreas@apsfilter.org>
sub  2048g/FE23F866 2001-01-06
```

D.3.171 Johann Kois <jkois@FreeBSD.org>

```
pub  1024D/DD61C2D8 2004-06-27 Johann Kois <J.Kois@web.de>
     Key fingerprint = 8B70 03DB 3C45 E71D 0ED4  4825 FEB0 EBEF DD61 C2D8
uid                            Johann Kois <jkois@freebsd.org>
sub  1024g/568307CB 2004-06-27
```

D.3.172 Sergei Kolobov <sergei@FreeBSD.org>

```
pub  1024D/3BA53401 2003-10-10 Sergei Kolobov <sergei@FreeBSD.org>
     Key fingerprint = A2F4 5F34 0586 CC9C 493A  347C 14EC 6E69 3BA5 3401
uid                            Sergei Kolobov <sergei@kolobov.com>
sub  2048g/F8243671 2003-10-10
```

D.3.173 Maxim Konovalov <maxim@FreeBSD.org>

```
pub  1024D/2C172083 2002-05-21 Maxim Konovalov <maxim@FreeBSD.org>
     Key fingerprint = 6550 6C02 EFC2 50F1 B7A3  D694 ECF0 E90B 2C17 2083
uid                            Maxim Konovalov <maxim@macomnet.ru>
sub  1024g/F305DDCA 2002-05-21
```

D.3.174 Taras Korenko <taras@FreeBSD.org>

```
pub   1024D/8ACCC68B 2010-03-30
      Key fingerprint = 5128 2A8B 9BC1 A664 21E0  1E61 D838 54D3 8ACC C68B
uid                   Taras Korenko <taras@freebsd.org>
uid                   Taras Korenko <ds@ukrhub.net>
uid                   Taras Korenko <tarasishche@gmail.com>
```

```
sub   2048g/8D7CC0FA 2010-03-30 [expires: 2015-03-29]
```

D.3.175 Joseph Koshy <jkoshy@FreeBSD.org>

```
pub  1024D/D93798B6 2001-12-21 Joseph Koshy (FreeBSD) <jkoshy@freebsd.org>
     Key fingerprint = 0DE3 62F3 EF24 939F 62AA  2E3D ABB8 6ED3 D937 98B6
sub  1024g/43FD68E9 2001-12-21
```

D.3.176 Wojciech A. Koszek <wkoszek@FreeBSD.org>

```
pub   1024D/C9F25145 2006-02-15
      Key fingerprint = 6E56 C571 9D33 D23E 9A61  8E50 623C AD62 C9F2 5145
uid                  Wojciech A. Koszek <dunstan@FreeBSD.czest.pl>
uid                  Wojciech A. Koszek <wkoszek@FreeBSD.org>
sub   4096g/3BBD20A5 2006-02-15
```

D.3.177 Alex Kozlov <ak@FreeBSD.org>

```
pub   2048R/0D1D29A0 2012-03-01 [expires: 2024-02-27]
      Key fingerprint = 7774 4FCF 6AC9 126B BD0E  DBF3 5EBF 4968 0D1D 29A0
uid                  Alex Kozlov <ak@freebsd.org>
sub   2048R/2DD82C65 2012-03-01 [expires: 2024-02-27]
```

D.3.178 Steven Kreuzer <skreuzer@FreeBSD.org>

```
pub   1024D/E0D6F907 2009-03-16 [expires: 2013-04-25]
      Key fingerprint = 8D8F 14D6 ED9F 6BD0 7756  7A46 66BA B4B6 E0D6 F907
uid                  Steven Kreuzer <skreuzer@exit2shell.com>
uid                  Steven Kreuzer <skreuzer@freebsd.org>
```

D.3.179 Gábor Kövesdán <gabor@FreeBSD.org>

```
pub   1024D/2373A6B1 2006-12-05
      Key fingerprint = A42A 10D6 834B BEC0 26F0  29B1 902D D04F 2373 A6B1
uid                  Gabor Kovesdan <gabor@FreeBSD.org>
sub   2048g/92B0A104 2006-12-05
```

D.3.180 Ana Kukec <anchie@FreeBSD.org>

```
pub   2048R/510D23BB 2010-04-18
      Key fingerprint = 0A9B 0ABB 0E1C B5A4 3408  398F 778A C3B4 510D 23BB
uid                  Ana Kukec <anchie@FreeBSD.org>
```

```
sub   2048R/699E4DDA 2010-04-18
```

D.3.181 Roman Kurakin <rik@FreeBSD.org>

```
pub   1024D/C8550F4C 2005-12-16 [expires: 2008-12-15]
      Key fingerprint = 25BB 789A 6E07 E654 8E59  0FA9 42B1 937C C855 0F4C
uid                  Roman Kurakin <rik@FreeBSD.org>
sub   2048g/D15F2AB6 2005-12-16 [expires: 2008-12-15]
```

D.3.182 Hideyuki KURASHINA <rushani@FreeBSD.org>

```
pub  1024D/439ADC57 2002-03-22 Hideyuki KURASHINA <rushani@bl.mmtr.or.jp>
     Key fingerprint = A052 6F98 6146 6FE3 91E2  DA6B F2FA 2088 439A DC57
uid                            Hideyuki KURASHINA <rushani@FreeBSD.org>
uid                            Hideyuki KURASHINA <rushani@jp.FreeBSD.org>
sub  1024g/64764D16 2002-03-22
```

D.3.183 Jun Kuriyama <kuriyama@FreeBSD.org>

```
pub  1024D/FE3B59CD 1998-11-23 Jun Kuriyama <kuriyama@imgsrc.co.jp>
     Key fingerprint = 5219 55CE AC84 C296 3A3B  B076 EE3C 4DBB FE3B 59CD
uid                            Jun Kuriyama <kuriyama@FreeBSD.org>
uid                            Jun Kuriyama <kuriyama@jp.FreeBSD.org>
sub  2048g/1CF20D27 1998-11-23
```

D.3.184 René Ladan <rene@FreeBSD.org>

```
pub   4096R/0A3789B7 2012-11-18
      Key fingerprint = 101A 716B 162B 00E5 5BED  EA05 ADBB F861 0A37 89B7
uid                  René Ladan <rene@freebsd.org>
sub   4096R/B67184C6 2012-11-18
```

D.3.185 Julien Laffaye <jlaffaye@FreeBSD.org>

```
pub   2048R/6AEBE420 2011-06-06
      Key fingerprint = 031A B449 B383 5C3B B618  E2F4 BAD0 0F0E 6AEB E420
uid                  Julien Laffaye <jlaffaye@FreeBSD.org>
sub   2048R/538B8D5B 2011-06-06
```

D.3.186 Clement Laforet <clement@FreeBSD.org>

```
pub  1024D/0723BA1D 2003-12-13 Clement Laforet (FreeBSD committer address) <clement@FreeBSD.org>
     Key fingerprint = 3638 4B14 8463 A67B DC7E  641C B118 5F8F 0723 BA1D
uid                           Clement Laforet <sheepkiller@cultdeadsheep.org>
uid                           Clement Laforet <clement.laforet@cotds.org>
sub  2048g/23D57658 2003-12-13
```

D.3.187 Max Laier <mlaier@FreeBSD.org>

```
pub   1024D/3EB6046D 2004-02-09
      Key fingerprint = 917E 7F25 E90F 77A4 F746  2E8D 5F2C 84A1 3EB6 046D
uid                  Max Laier <max@love2party.net>
uid                  Max Laier <max.laier@ira.uka.de>
uid                  Max Laier <mlaier@freebsd.org>
uid                  Max Laier <max.laier@tm.uka.de>
sub   4096g/EDD08B9B 2005-06-28
```

D.3.188 Erwin Lansing <erwin@FreeBSD.org>

```
pub   1024D/15256990 1998-07-03
      Key fingerprint = FB58 9797 299A F18E 2D3E  73D6 AB2F 5A5B 1525 6990
uid                  Erwin Lansing <erwin@lansing.dk>
uid                  Erwin Lansing <erwin@FreeBSD.org>
uid                  Erwin Lansing <erwin@droso.dk>
uid                  Erwin Lansing <erwin@droso.org>
uid                  Erwin Lansing <erwin@aauug.dk>
sub   2048g/7C64013D 1998-07-03
```

D.3.189 Ganael Laplanche <martymac@FreeBSD.org>

```
pub   1024D/10B87391 2006-01-13
      Key fingerprint = D59D 984D 8988 7BB9 DA37  BA77 757E D5F0 10B8 7391
uid                  Ganael LAPLANCHE <ganael.laplanche@martymac.org>
uid                  Ganael LAPLANCHE <martymac@martymac.com>
uid                  Ganael LAPLANCHE <ganael.laplanche@martymac.com>
uid                  Ganael LAPLANCHE <martymac@martymac.org>
uid                  Ganael LAPLANCHE <martymac@pasteur.fr>
uid                  Ganael LAPLANCHE <ganael.laplanche@pasteur.fr>
uid                  Ganael LAPLANCHE <martymac@FreeBSD.org>
sub   2048g/D65069D5 2006-01-13
```

D.3.190 Greg Larkin <glarkin@FreeBSD.org>

```
pub   1024D/1C940290 2003-10-09
      Key fingerprint = 8A4A 80AA F26C 8C2C D01B  94C6 D2C4 68B8 1C94 0290
uid                  Greg Larkin (The FreeBSD Project) <glarkin@FreeBSD.org>
uid                  Gregory C. Larkin (SourceHosting.Net, LLC) <glarkin@sourcehosting.net>
uid                  [jpeg image of size 6695]
sub   2048g/47674316 2003-10-09
```

D.3.191 Frank J. Laszlo <laszlof@FreeBSD.org>

```
pub   4096R/012360EC 2006-11-06 [expires: 2011-11-05]
      Key fingerprint = 3D93 21DB B5CC 1339 E4B4  1BC4 AD50 C17C 0123 60EC
uid                  Frank J. Laszlo <laszlof@FreeBSD.org>
```

D.3.192 Dru Lavigne <dru@FreeBSD.org>

```
pub   1024D/C6AA2E94 2013-01-22
      Key fingerprint = 6CC4 2180 F27C 29B6 5A9C  EC0D A454 DC05 C6AA 2E94
uid                  Dru Lavigne <dru@freebsd.org>
sub   1024g/7FAC82EA 2013-01-22
```

D.3.193 Sam Lawrance <lawrance@FreeBSD.org>

```
pub   1024D/32708C59 2003-08-14
      Key fingerprint = 1056 2A02 5247 64D4 538D  6975 8851 7134 3270 8C59
uid                  Sam Lawrance <lawrance@FreeBSD.org>
uid                  Sam Lawrance <boris@brooknet.com.au>
sub   2048g/0F9CCF92 2003-08-14
```

D.3.194 Nate Lawson <njl@FreeBSD.org>

```
pub   1024D/60E5AC11 2007-02-07
      Key fingerprint = 18E2 7E5A FD6A 199B B08B  E9FB 73C8 DB67 60E5 AC11
uid                  Nate Lawson <nate@root.org>
sub   2048g/CDBC7E1B 2007-02-07
```

D.3.195 Jeremie Le Hen <jlh@FreeBSD.org>

```
pub   2048D/8BF6CF92 2012-04-18
      Key fingerprint = 66C9 B361 16CA BFF6 5C07  DA0A 28DE 3702 8BF6 CF92
uid                  Jeremie Le Hen <jeremie@le-hen.org>
uid                  Jeremie Le Hen <jeremie@lehen.org>
uid                  Jeremie Le Hen <ttz@chchile.org>
```

```
uid                  Jeremie Le Hen <jlh@FreeBSD.org>
sub   2048g/045479A3 2012-04-18
```

D.3.196 Yen-Ming Lee <leeym@FreeBSD.org>

```
pub   1024D/93FA8BD6 2007-05-21
      Key fingerprint = DEC4 6E7F 69C0 4AC3 21ED  EE65 6C0E 9257 93FA 8BD6
uid                  Yen-Ming Lee <leeym@leeym.com>
sub   2048g/899A3931 2007-05-21
```

D.3.197 Sam Leffler <sam@FreeBSD.org>

```
pub   1024D/BD147743 2005-03-28
      Key fingerprint = F618 F2FC 176B D201 D91C  67C6 2E33 A957 BD14 7743
uid                  Samuel J. Leffler <sam@freebsd.org>
sub   2048g/8BA91D05 2005-03-28
```

D.3.198 Jean-Yves Lefort <jylefort@FreeBSD.org>

```
pub   1024D/A3B8006A 2002-09-07
      Key fingerprint = CC99 D1B0 8E44 293D 32F7  D92E CB30 FB51 A3B8 006A
uid                  Jean-Yves Lefort <jylefort@FreeBSD.org>
uid                  Jean-Yves Lefort <jylefort@brutele.be>
sub   4096g/C9271AFC 2002-09-07
```

D.3.199 Alexander Leidinger <netchild@FreeBSD.org>

```
pub   1024D/72077137 2002-01-31
      Key fingerprint = AA3A 8F69 B214 6BBD 5E73  C9A0 C604 3C56 7207 7137
uid                  Alexander Leidinger <netchild@FreeBSD.org>
uid                  [jpeg image of size 19667]
sub   2048g/8C9828D3 2002-01-31
```

D.3.200 Andrey V. Elsukov <ae@FreeBSD.org>

```
pub   2048R/10C8A17A 2010-05-29
      Key fingerprint = E659 1E1B 41DA 1516 F0C9  BC00 01C5 EA04 10C8 A17A
uid                  Andrey V. Elsukov <ae@freebsd.org>
uid                  Andrey V. Elsukov <bu7cher@yandex.ru>
sub   2048R/0F6D64C5 2010-05-29
```

D.3.201 Dejan Lesjak <lesi@FreeBSD.org>

```
pub  1024D/96C5221F 2004-08-18 Dejan Lesjak <lesi@FreeBSD.org>
     Key fingerprint = 2C5C 02EA 1060 1D6D 9982  38C0 1DA7 DBC4 96C5 221F
uid                            Dejan Lesjak <dejan.lesjak@ijs.si>
sub  1024g/E0A69278 2004-08-18
```

D.3.202 Achim Leubner <achim@FreeBSD.org>

```
pub   2048R/2E15B3C1 2013-01-22
      Key fingerprint = 2A48 0317 D477 2A07 2AD9  CF1C 7C1D 832E 2E15 B3C1
uid                  Achim Leubner <achim@freebsd.org>
sub   2048R/E275EF01 2013-01-22
```

D.3.203 Chuck Lever <cel@FreeBSD.org>

```
pub   1024D/8FFC2B87 2006-02-13
      Key fingerprint = 6872 923F 5012 F88B 394C  2F69 37B4 8171 8FFC 2B87
uid                  Charles E. Lever <cel@freebsd.org>
sub   2048g/9BCE0459 2006-02-13
```

D.3.204 Greg Lewis <glewis@FreeBSD.org>

```
pub  1024D/1BB6D9E0 2002-03-05 Greg Lewis (FreeBSD) <glewis@FreeBSD.org>
     Key fingerprint = 2410 DA6D 5A3C D801 65FE  C8DB DEEA 9923 1BB6 D9E0
uid                            Greg Lewis <glewis@eyesbeyond.com>
sub  2048g/45E67D60 2002-03-05
```

D.3.205 Qing Li <qingli@FreeBSD.org>

```
pub   2048R/A3CA4C13 2013-06-12 [expires: 2017-06-12]
      Key fingerprint = E37B CB18 35D1 F01B 7D7B  1000 0EAF 4BEA A3CA 4C13
uid                  Qing Li <qingli@freebsd.org>
sub   2048R/EF3A9370 2013-06-12 [expires: 2017-06-12]
```

D.3.206 Xin Li <delphij@FreeBSD.org>

```
pub   1024D/CAEEB8C0 2004-01-28
      Key fingerprint = 43B8 B703 B8DD 0231 B333  DC28 39FB 93A0 CAEE B8C0
uid                  Xin LI <delphij@FreeBSD.org>
uid                  Xin LI <delphij@frontfree.net>
uid                  Xin LI <delphij@delphij.net>
uid                  Xin LI <delphij@geekcn.org>
```

```
pub   1024D/42EA8A4B 2006-01-27 [expired: 2008-01-01]
      Key fingerprint = F19C 2616 FA97 9C13 2581  C6F3 85C5 1CCE 42EA 8A4B
uid                  Xin LI <delphij@geekcn.org>
uid                  Xin LI <delphij@FreeBSD.org>
uid                  Xin LI <delphij@delphij.net>

pub   1024D/18EDEBA0 2008-01-02 [expired: 2010-01-02]
      Key fingerprint = 79A6 CF42 F917 DDCA F1C2  C926 8BEB DB04 18ED EBA0
uid                  Xin LI <delphij@geekcn.org>
uid                  Xin LI <delphij@FreeBSD.org>
uid                  Xin LI <delphij@delphij.net>

pub   2048R/3FCA37C1 2010-01-10 [expired: 2012-01-10]
      Key fingerprint = 27EA 5D6C 9398 BA7F B205  8F70 04CE F812 3FCA 37C1
uid                  Xin LI <delphij@delphij.net>
uid                  Xin LI <delphij@gmail.com>
uid                  Xin LI <delphij@geekcn.org>
uid                  Xin LI <delphij@FreeBSD.org>

pub   4096R/2E54AB2C 2011-12-05
      Key fingerprint = D95C D3C3 8FA8 25C2 C62B  9FEA 0887 6D93 2E54 AB2C
uid                  Xin Li <delphij@geekcn.org>
uid                  Xin Li <delphij@delphij.net>
uid                  Xin Li <delphij@FreeBSD.org>
sub   4096R/7832B740 2011-12-05
sub   2048R/BC50FBB3 2011-12-05 [expires: 2013-12-05]
sub   2048R/C894647D 2011-12-05 [expires: 2013-12-05]
```

D.3.207 Tai-hwa Liang <avatar@FreeBSD.org>

```
pub  1024R/F4013AB1 1998-05-13 Tai-hwa Liang <avatar@FreeBSD.org>
     Key fingerprint = 5B 05 1D 37 7F 35 31 4E  5D 38 BD 07 10 32 B9 D0
uid                           Tai-hwa Liang <avatar@mmlab.cse.yzu.edu.tw>
```

D.3.208 Ying-Chieh Liao <ijliao@FreeBSD.org>

```
pub  1024D/11C02382 2001-01-09 Ying-Chieh Liao <ijliao@CCCA.NCTU.edu.tw>
     Key fingerprint = 4E98 55CC 2866 7A90 EFD7  9DA5 ACC6 0165 11C0 2382
uid                            Ying-Chieh Liao <ijliao@FreeBSD.org>
uid                            Ying-Chieh Liao <ijliao@csie.nctu.edu.tw>
uid                            Ying-Chieh Liao <ijliao@dragon2.net>
uid                            Ying-Chieh Liao <ijliao@tw.FreeBSD.org>
sub  4096g/C1E16E89 2001-01-09
```

D.3.209 Ulf Lilleengen <lulf@FreeBSD.org>

```
pub   1024D/ADE1B837 2009-08-19 [expires: 2014-08-18]
      Key fingerprint = 3822 B4E6 6D1C 6F71 4AA8  7A27 ADDF C400 ADE1 B837
uid                  Ulf Lilleengen <ulf.lilleengen@gmail.com>
uid                  Ulf Lilleengen <lulf@pvv.ntnu.no>
uid                  Ulf Lilleengen <lulf@stud.ntnu.no>
uid                  Ulf Lilleengen <lulf@FreeBSD.org>
uid                  Ulf Lilleengen <lulf@idi.ntnu.no>
sub   2048g/B5409122 2009-08-19 [expires: 2014-08-18]
```

D.3.210 Clive Lin <clive@FreeBSD.org>

```
pub  1024D/A008C03E 2001-07-30 Clive Lin <clive@tongi.org>
     Key fingerprint = FA3F 20B6 A77A 6CEC 1856  09B0 7455 2805 A008 C03E
uid                           Clive Lin <clive@CirX.ORG>
uid                           Clive Lin <clive@FreeBSD.org>
sub  1024g/03C2DC87 2001-07-30 [expires: 2005-08-25]
```

D.3.211 Po-Chien Lin <pclin@FreeBSD.org>

```
pub   4096R/865C427F 2013-02-05
      Key fingerprint = CF3B AB13 4C94 6388 B047  B599 8B28 1692 865C 427F
uid                  Po-Chien Lin <pclin@FreeBSD.org>
uid                  Po-Chien Lin <linpc@cs.nctu.edu.tw>
sub   4096R/F31280BA 2013-02-05
```

D.3.212 Yi-Jheng Lin <yzlin@FreeBSD.org>

```
pub   2048R/A34C6A8A 2009-07-20
      Key fingerprint = 7E3A E981 BB7C 5D73 9534  ED39 0222 04D3 A34C 6A8A
uid                  Yi-Jheng Lin (FreeBSD) <yzlin@FreeBSD.org>
sub   2048R/B4D776FE 2009-07-20
```

D.3.213 Mark Linimon <linimon@FreeBSD.org>

```
pub   1024D/84C83473 2003-10-09
      Key fingerprint = 8D43 1B55 D127 0BFC 842E  1C96 803C 5A34 84C8 3473
uid                  Mark Linimon <linimon@FreeBSD.org>
uid                  Mark Linimon <linimon@lonesome.com>
sub   1024g/24BFF840 2003-10-09
```

D.3.214 Tilman Keskinöz <arved@FreeBSD.org>

```
pub   1024D/807AC53A 2002-06-03 [expires: 2013-09-07]
      Key fingerprint = A92F 344F 31A8 B8DE DDFA  7FB4 7C22 C39F 807A C53A
uid                  Tilman Keskinöz <arved@arved.at>
uid                  Tilman Keskinöz <arved@FreeBSD.org>
sub   1024g/FA351986 2002-06-03 [expires: 2013-09-07]
```

D.3.215 Dryice Liu <dryice@FreeBSD.org>

```
pub   1024D/77B67874 2005-01-28
      Key fingerprint = 8D7C F82D D28D 07E5 EF7F  CD25 6B5B 78A8 77B6 7874
uid                  Dryice Dong Liu (Dryice) <dryice@FreeBSD.org>
uid                  Dryice Dong Liu (Dryice) <dryice@liu.com.cn>
uid                  Dryice Dong Liu (Dryice) <dryice@hotpop.com>
uid                  Dryice Dong Liu (Dryice) <dryiceliu@gmail.com>
uid                  Dryice Dong Liu (Dryice) <dryice@dryice.name>
sub   2048g/ECFA49E4 2005-01-28
```

D.3.216 Tong Liu <nemoliu@FreeBSD.org>

```
pub   1024D/ECC7C907 2007-07-10
      Key fingerprint = B62E 3109 896B B283 E2FA  60FE A1BA F92E ECC7 C907
uid                  Tong LIU <nemoliu@FreeBSD.org>
sub   4096g/B6D7B15D 2007-07-10
```

D.3.217 Zachary Loafman <zml@FreeBSD.org>

```
pub   1024D/4D65492D 2009-05-26
      Key fingerprint = E513 4AE9 5D6D 8BF9 1CD3  4389 4860 D79B 4D65 492D
uid                  Zachary Loafman <zml@FreeBSD.org>
sub   2048g/1AD659F0 2009-05-26
```

D.3.218 Juergen Lock <nox@FreeBSD.org>

```
pub   1024D/1B6BFBFD 2006-12-22
      Key fingerprint = 33A7 7FAE 51AF 00BC F0D3  ECCE FAFD 34C1 1B6B FBFD
uid                  Juergen Lock <nox@FreeBSD.org>
sub   2048g/251229D1 2006-12-22
```

D.3.219 Remko Lodder <remko@FreeBSD.org>

```
pub   4096R/3F774079 2012-11-11 [expires: 2016-11-11]
      Key fingerprint = 7EE4 C4AF DCA3 E0B4 479B  A344 7135 8ED6 3F77 4079
uid                  Remko Lodder <remko@FreeBSD.org>
sub   4096R/59F38CB0 2012-11-11 [expires: 2016-11-11]
```

D.3.220 Alexander Logvinov <avl@FreeBSD.org>

```
pub   1024D/1C47D5C0 2009-05-28
      Key fingerprint = 8B5F 880A 382B 075E E707  9DB2 E135 4176 1C47 D5C0
uid                  Alexander Logvinov <alexander@logvinov.com>
uid                  Alexander Logvinov (FreeBSD Ports Committer) <avl@FreeBSD.org>
uid                  Alexander Logvinov <ports@logvinov.com>
uid                  Alexander Logvinov <logvinov@gmail.com>
uid                  Alexander Logvinov <logvinov@yandex.ru>
sub   2048g/60BDD4BB 2009-05-28
```

D.3.221 Isabell Long <issyl0@FreeBSD.org>

```
pub   4096R/EB83C2BD 2009-09-26
      Key fingerprint = D55A 42E7 0974 EFD9 3939  56B9 6E6B E425 EB83 C2BD
uid                  Isabell Long <isabell@issyl0.co.uk>
uid                  Isabell Long <me@issyl0.co.uk>
uid                  Isabell Long <isabell121@gmail.com>
uid                  Isabell Long (BitFolk Ltd.) <isabell@bitfolk.com>
uid                  Isabell Long (College) <IL18685@woking.ac.uk>
uid                  Isabell Long (The Open University) <il948@my.open.ac.uk>
uid                  Isabell Long (Mailing lists address.) <lists@issyl0.co.uk>
uid                  Isabell Long (YRS) <isabell@youngrewiredstate.org>
uid                  Isabell Long (FreeBSD) <issyl0@FreeBSD.org>
```

D.3.222 Scott Long <scottl@FreeBSD.org>

```
pub  1024D/017C5EBF 2003-01-18 Scott A. Long (This is my official FreeBSD key) <scottl@freebsd.org
     Key fingerprint = 34EA BD06 44F7 F8C3 22BC  B52C 1D3A F6D1 017C 5EBF
sub  1024g/F61C8F91 2003-01-18
```

D.3.223 Rick Macklem <rmacklem@FreeBSD.org>

```
pub   1024D/7FB9C5F1 2009-04-05
      Key fingerprint = B9EA 767A F6F3 3786 E0C7  434A 05C6 70D6 7FB9 C5F1
uid                  Rick Macklem <rmacklem@freebsd.org>
sub   1024g/D0B20E8A 2009-04-05
```

D.3.224 Bruce A. Mah <bmah@FreeBSD.org>

```
pub   1024D/5BA052C3 1997-12-08
      Key fingerprint = F829 B805 207D 14C7 7197  7832 D8CA 3171 5BA0 52C3
uid                  Bruce A. Mah <bmah@acm.org>
uid                  Bruce A. Mah <bmah@ca.sandia.gov>
uid                  Bruce A. Mah <bmah@ieee.org>
uid                  Bruce A. Mah <bmah@cisco.com>
uid                  Bruce A. Mah <bmah@employees.org>
uid                  Bruce A. Mah <bmah@freebsd.org>
uid                  Bruce A. Mah <bmah@packetdesign.com>
uid                  Bruce A. Mah <bmah@kitchenlab.org>
sub   2048g/B4E60EA1 1997-12-08
```

D.3.225 Ruslan Makhmatkhanov <rm@FreeBSD.org>

```
pub   2048R/F60D756F 2011-11-10
      Key fingerprint = 9D18 8A88 304C B78B 8003  0379 4574 0BAF F60D 756F
uid                  Ruslan Makhmatkhanov <rm@FreeBSD.org>
sub   2048R/B658C269 2011-11-10
```

D.3.226 Mike Makonnen <mtm@FreeBSD.org>

```
pub  1024D/7CD41F55 2004-02-06 Michael Telahun Makonnen <mtm@FreeBSD.Org>
     Key fingerprint = AC7B 5672 2D11 F4D0 EBF8  5279 5359 2B82 7CD4 1F55
uid                           Michael Telahun Makonnen <mtm@tmsa-inc.com>
uid                           Mike Makonnen <mtm@identd.net>
uid                           Michael Telahun Makonnen <mtm@acs-et.com>
sub  2048g/E7DC936B 2004-02-06
```

D.3.227 David Malone <dwmalone@FreeBSD.org>

```
pub   512/40378991 1994/04/21 David Malone <dwmalone@maths.tcd.ie>
            Key fingerprint = 86 A7 F4 86 39 2C 47 2C  C1 C2 35 78 8E 2F B8 F5
```

D.3.228 Dmitry Marakasov <amdmi3@FreeBSD.org>

```
pub   1024D/F9D2F77D 2008-06-15 [expires: 2010-06-15]
      Key fingerprint = 55B5 0596 FF1E 8D84 5F56  9510 D35A 80DD F9D2 F77D
uid                  Dmitry Marakasov <amdmi3@amdmi3.ru>
uid                  Dmitry Marakasov <amdmi3@FreeBSD.org>
sub   2048g/2042CDD8 2008-06-15
```

D.3.229 Koop Mast <kwm@FreeBSD.org>

```
pub  1024D/F95426DA 2004-09-10 Koop Mast <kwm@rainbow-runner.nl>
     Key fingerprint = C66F 1835 0548 3440 8576  0FFE 6879 B7CD F954 26DA
uid                            Koop Mast <kwm@FreeBSD.org>
sub  1024g/A782EEDD 2004-09-10
```

D.3.230 Ed Maste <emaste@FreeBSD.org>

```
pub   2048R/50A17BF4 2012-12-18
      Key fingerprint = 0C08 ECC9 3A0A 8500 AB95  B553 49C4 7851 50A1 7BF4
uid                  Ed Maste <emaste@freebsd.org>
sub   2048R/08FA5F72 2012-12-18
```

D.3.231 Cherry G. Mathew <cherry@FreeBSD.org>

```
pub   2048R/2D066FE1 2007-05-22
      Key fingerprint = FBF1 89FF 81BB E1C7 6C1B  378D 3438 20E9 2D06 6FE1
uid                  Cherry G. Mathew (FreeBSD email) <cherry@FreeBSD.org>
uid                  "Cherry G. Mathew" (NetBSD email) <cherry@NetBSD.org>
sub   2048R/7B2C4166 2007-05-22
```

D.3.232 Makoto Matsushita <matusita@FreeBSD.org>

```
pub   1024D/20544576 1999-04-18
      Key fingerprint = 71B6 13BF B262 2DD8 2B7C  6CD0 EB2D 4147 2054 4576
uid                  Makoto Matsushita <matusita@matatabi.or.jp>
uid                  Makoto Matsushita <matusita@FreeBSD.org>
uid                  Makoto Matsushita <matusita@jp.FreeBSD.ORG>
uid                  Makoto Matsushita <matusita@ist.osaka-u.ac.jp>
sub   1024g/F1F3C94D 1999-04-18
```

D.3.233 Martin Matuska <mm@FreeBSD.org>

```
pub   1024D/4261B0D1 2007-02-05
      Key fingerprint = 17C4 3F32 B3DE 3ED7 E84E  5592 A76B 8B03 4261 B0D1
uid                  Martin Matuska <martin@matuska.org>
uid                  Martin Matuska <mm@FreeBSD.org>
uid                  Martin Matuska <martin.matuska@wu-wien.ac.at>
sub   2048g/3AC9A5A6 2007-02-05
```

D.3.234 Sergey Matveychuk <sem@FreeBSD.org>

```
pub   1024D/B71F605D 1999-10-13
      Key fingerprint = 4704 F374 DB28 BEC6 51C8  1322 4DC9 4BD8 B71F 605D
uid                  Sergey Matveychuk <sem@FreeBSD.org>
uid                  Sergey Matveychuk <sem@ciam.ru>
uid                  Sergey Matveychuk <sem@core.inec.ru>
sub   2048g/DEAF9D91 1999-10-13
```

D.3.235 Tom McLaughlin <tmclaugh@FreeBSD.org>

```
pub   1024D/E2F7B3D8 2005-05-24
      Key fingerprint = 7692 B222 8D23 CF94 1993  0138 E339 E225 E2F7 B3D8
uid                  Tom McLaughlin (Personal email address) <tmclaugh@sdf.lonestar.org>
uid                  Tom McLaughlin (Work email address) <tmclaughlin@meditech.com>
uid                  Tom McLaughlin (FreeBSD email address) <tmclaugh@FreeBSD.org>
sub   2048g/16838F62 2005-05-24
```

D.3.236 Jean Milanez Melo <jmelo@FreeBSD.org>

```
pub   1024D/AA5114BF 2006-03-03
      Key fingerprint = 826D C2AA 6CF2 E29A EBE7  4776 D38A AB83 AA51 14BF
uid                  Jean Milanez Melo <jmelo@FreeBSD.org>
uid                  Jean Milanez Melo <jmelo@freebsdbrasil.com.br>
sub   4096g/E9E1CBD9 2006-03-03
```

D.3.237 Kenneth D. Merry <ken@FreeBSD.org>

```
pub  1024D/54C745B5 2000-05-15 Kenneth D. Merry <ken@FreeBSD.org>
     Key fingerprint = D25E EBC5 F17A 9E52 84B4  BF14 9248 F0DA 54C7 45B5
uid                           Kenneth D. Merry <ken@kdm.org>
sub  2048g/89D0F797 2000-05-15

pub  1024R/2FA0A505 1995-10-30 Kenneth D. Merry <ken@plutotech.com>
     Key fingerprint = FD FA 85 85 95 C4 8E E8  98 1A CA 18 56 F0 00 1F
```

D.3.238 Dirk Meyer <dinoex@FreeBSD.org>

```
pub  1024R/331CDA5D 1995-06-04 Dirk Meyer <dinoex@FreeBSD.org>
     Key fingerprint = 44 16 EC 0A D3 3A 4F 28  8A 8A 47 93 F1 CF 2F 12
uid                           Dirk Meyer <dirk.meyer@dinoex.sub.org>
uid                           Dirk Meyer <dirk.meyer@guug.de>
```

D.3.239 Yoshiro Sanpei MIHIRA <sanpei@FreeBSD.org>

```
pub  1024R/391C5D69 1996-11-21 sanpei@SEAPLE.ICC.NE.JP
     Key fingerprint = EC 04 30 24 B0 6C 1E 63  5F 5D 25 59 3E 83 64 51
uid                            MIHIRA Yoshiro <sanpei@sanpei.org>
uid                            Yoshiro MIHIRA <sanpei@FreeBSD.org>
uid                            MIHIRA Yoshiro <sanpei@yy.cs.keio.ac.jp>
uid                            MIHIRA Yoshiro <sanpei@cc.keio.ac.jp>
uid                            MIHIRA Yoshiro <sanpei@educ.cc.keio.ac.jp>
uid                            MIHIRA Yoshiro <sanpei@st.keio.ac.jp>
```

D.3.240 Robert Millan <rmh@FreeBSD.org>

```
pub   4096R/DEA2C38E 2009-08-14
      Key fingerprint = A537 F029 AAAE 0E9C 39A7  C22C BB9D 98D9 DEA2 C38E
uid                  Robert Millan <rmh@debian.org>
uid                  Robert Millan <rmh@freebsd.org>
uid                  Robert Millan <rmh@gnu.org>
sub   4096R/65A0A9CE 2009-08-14
sub   4096R/41F37946 2009-08-14
```

D.3.241 Stephen Montgomery-Smith <stephen@FreeBSD.org>

```
pub   2048R/9A92D807 2011-06-14
      Key fingerprint = 2B61 D82E 168E F08B 6E08  712E 2DF1 2BD1 9A92 D807
uid                  Stephen Montgomery-Smith <stephen@freebsd.org>
sub   2048R/A4BA6560 2011-06-14
```

D.3.242 Marcel Moolenaar <marcel@FreeBSD.org>

```
pub  1024D/61EE89F6 2002-02-09 Marcel Moolenaar <marcel@xcllnt.net>
     Key fingerprint = 68BB E2B7 49AA FF69 CA3A  DF71 A605 A52D 61EE 89F6
sub  1024g/6EAAB456 2002-02-09
```

D.3.243 Kris Moore <kmoore@FreeBSD.org>

```
pub   1024D/6294612C 2009-05-26
      Key fingerprint = 8B70 9876 346F 1F97 5687  6950 4C92 D789 6294 612C
uid                  Kris Moore <kmoore@freebsd.org>
sub   2048g/A7FFE8FB 2009-05-26
```

D.3.244 Dmitry Morozovsky <marck@FreeBSD.org>

```
pub   1024D/6B691B03 2001-07-20
      Key fingerprint = 39AC E336 F03D C0F8 5305  B725 85D4 5045 6B69 1B03
uid                  Dmitry Morozovsky <marck@rinet.ru>
uid                  Dmitry Morozovsky <marck@FreeBSD.org>
sub   2048g/44D656F8 2001-07-20
```

D.3.245 Alexander Motin <mav@FreeBSD.org>

```
pub   1024D/0577BACA 2007-04-20 [expires: 2012-04-18]
      Key fingerprint = 0E84 B263 E97D 3E48 161B  98A2 D240 A09E 0577 BACA
uid                  Alexander Motin <mav@freebsd.org>
uid                  Alexander Motin <mav@mavhome.dp.ua>
uid                  Alexander Motin <mav@alkar.net>
sub   2048g/4D59D1C2 2007-04-20 [expires: 2012-04-18]
```

D.3.246 Felippe de Meirelles Motta <lippe@FreeBSD.org>

```
pub   1024D/F2CF7DAE 2008-09-02 [expires: 2010-09-02]
      Key fingerprint = 0532 A900 286D DAFD 099D  394D 231B AF20 F2CF 7DAE
uid                  Felippe de Meirelles Motta (FreeBSD Ports Committer) <lippe@FreeBSD.org>
sub   2048g/38E8EEF3 2008-09-02 [expires: 2010-09-02]
```

D.3.247 Rich Murphey <rich@FreeBSD.org>

```
pub  1024R/583443A9 1995-03-31 Rich Murphey <rich@lamprey.utmb.edu>
     Key fingerprint = AF A0 60 C4 84 D6 0C 73  D1 EF C0 E9 9D 21 DB E4
```

D.3.248 Akinori MUSHA <knu@FreeBSD.org>

```
pub  1024D/9FD9E1EE 2000-03-21 Akinori MUSHA <knu@and.or.jp>
     Key fingerprint = 081D 099C 1705 861D 4B70  B04A 920B EFC7 9FD9 E1EE
uid                            Akinori MUSHA <knu@FreeBSD.org>
uid                            Akinori MUSHA <knu@idaemons.org>
uid                            Akinori MUSHA <knu@ruby-lang.org>
sub  1024g/71BA9D45 2000-03-21
```

D.3.249 Thomas Möstl <tmm@FreeBSD.org>

```
pub  1024D/419C776C 2000-11-28 Thomas Moestl <tmm@FreeBSD.org>
     Key fingerprint = 1C97 A604 2BD0 E492 51D0  9C0F 1FE6 4F1D 419C 776C
uid                            Thomas Moestl <tmoestl@gmx.net>
uid                            Thomas Moestl <t.moestl@tu-bs.de>
```

```
sub  2048g/ECE63CE6 2000-11-28
```

D.3.250 Masafumi NAKANE <max@FreeBSD.org>

```
pub  1024D/CE356B59 2000-02-19 Masafumi NAKANE <max@wide.ad.jp>
     Key fingerprint = EB40 BCAB 4CE5 0764 9942  378C 9596 159E CE35 6B59
uid                            Masafumi NAKANE <max@FreeBSD.org>
uid                            Masafumi NAKANE <max@accessibility.org>
uid                            Masafumi NAKANE <kd5pdi@qsl.net>
sub  1024g/FA9BD48B 2000-02-19
```

D.3.251 Maho Nakata <maho@FreeBSD.org>

```
pub   1024D/F28B4069 2009-02-09
      Key fingerprint = 3FE4 99A9 6F41 8161 4F5F  240C 8615 A60C F28B 4069
uid                  Maho NAKATA (NAKATA's FreeBSD.org alias) <maho@FreeBSD.org>
sub   2048g/6B49098E 2009-02-09
```

D.3.252 Yoichi NAKAYAMA <yoichi@FreeBSD.org>

```
pub  1024D/E0788E46 2000-12-28 Yoichi NAKAYAMA <yoichi@assist.media.nagoya-u.ac.jp>
     Key fingerprint = 1550 2662 46B3 096C 0460  BC03 800D 0C8A E078 8E46
uid                            Yoichi NAKAYAMA <yoichi@eken.phys.nagoya-u.ac.jp>
uid                            Yoichi NAKAYAMA <yoichi@FreeBSD.org>
sub  1024g/B987A394 2000-12-28
```

D.3.253 Edward Tomasz Napierala <trasz@FreeBSD.org>

```
pub   1024D/8E53F00E 2007-04-13
      Key fingerprint = DD8F 91B0 12D9 6237 42D9  DBE1 AFC8 CDE9 8E53 F00E
uid                  Edward Tomasz Napierala <trasz@FreeBSD.org>
sub   2048g/7C1F5D67 2007-04-13
```

D.3.254 David Naylor <dbn@FreeBSD.org>

```
pub   1024D/FF6916B2 2008-04-09
      Key fingerprint = 6540 B47C 54AA 3EBA B23B  58AC 51A6 8580 FF69 16B2
uid                  David Naylor <dbn@freebsd.org>
uid                  David Naylor <naylor.b.david@gmail.com>
sub   4096g/77FA885C 2008-04-09
```

D.3.255 Alexander Nedotsukov <bland@FreeBSD.org>

```
pub  1024D/D004116C 2003-08-14 Alexander Nedotsukov <bland@FreeBSD.org>
     Key fingerprint = 35E2 5020 55FC 2071 4ADD  1A4A 86B6 8A5D D004 116C
sub  1024g/1CCA8D46 2003-08-14
```

D.3.256 George V. Neville-Neil <gnn@FreeBSD.org>

```
pub   1024D/440A33D2 2002-09-17
      Key fingerprint = AF66 410F CC8D 1FC9 17DB  6225 61D8 76C1 440A 33D2
uid                  George V. Neville-Neil <gnn@freebsd.org>
uid                  George V. Neville-Neil <gnn@neville-neil.com>
sub   2048g/95A74F6E 2002-09-17
```

D.3.257 Simon L. Nielsen <simon@FreeBSD.org>

```
pub   1024D/FF7490AB 2007-01-14
      Key fingerprint = 4E92 BA8D E45E 85E2 0380  B264 049C 7480 FF74 90AB
uid                  Simon L. Nielsen <simon@FreeBSD.org>
uid                  Simon L. Nielsen <simon@nitro.dk>
sub   2048g/E3F5A76E 2007-01-14
```

D.3.258 Robert Noland <rnoland@FreeBSD.org>

```
pub   1024D/8A9F44E3 2007-07-24
      Key fingerprint = 107A 0C87 E9D0 E581 677B  2A28 3384 EB43 8A9F 44E3
uid                  Robert C. Noland III <rnoland@FreeBSD.org>
uid                  Robert C. Noland III (Personal Key) <rnoland@2hip.net>
sub   2048g/76C3CF00 2007-07-24
```

D.3.259 Anders Nordby <anders@FreeBSD.org>

```
pub  1024D/00835956 2000-08-13 Anders Nordby <anders@fix.no>
     Key fingerprint = 1E0F C53C D8DF 6A8F EAAD  19C5 D12A BC9F 0083 5956
uid                            Anders Nordby <anders@FreeBSD.org>
sub  2048g/4B160901 2000-08-13
```

D.3.260 Michael Nottebrock <lofi@FreeBSD.org>

```
pub  1024D/6B2974B0 2002-06-06 Michael Nottebrock <michaelnottebrock@gmx.net>
     Key fingerprint = 1079 3C72 0726 F300 B8EC  60F9 5E17 3AF1 6B29 74B0
uid                            Michael Nottebrock <lofi@freebsd.org>
uid                            Michael Nottebrock <lofi@tigress.com>
uid                            Michael Nottebrock <lofi@lofi.dyndns.org>
```

```
uid                          Michael Nottebrock <michaelnottebrock@web.de>
uid                          Michael Nottebrock <michaelnottebrock@meitner.wh.uni-dortmund.de>
sub  1024g/EF652E04 2002-06-06 [expires: 2004-06-15]
```

D.3.261 David O'Brien <obrien@FreeBSD.org>

```
pub  1024R/34F9F9D5 1995-04-23 David E. O'Brien <defunct - obrien@Sea.Legent.com>
     Key fingerprint = B7 4D 3E E9 11 39 5F A3  90 76 5D 69 58 D9 98 7A
uid                          David E. O'Brien <obrien@NUXI.com>
uid                          deobrien@ucdavis.edu
uid                          David E. O'Brien <whois Do38>
uid                          David E. O'Brien <obrien@FreeBSD.org>
uid                          David E. O'Brien <dobrien@seas.gwu.edu>
uid                          David E. O'Brien <obrien@cs.ucdavis.edu>
uid                          David E. O'Brien <defunct - obrien@media.sra.com>
uid                          David E. O'Brien <obrien@elsewhere.roanoke.va.us>
uid                          David E. O'Brien <obrien@Nuxi.com>

pub  1024D/7F9A9BA2 1998-06-10 "David E. O'Brien" <obrien@cs.ucdavis.edu>
     Key fingerprint = 02FD 495F D03C 9AF2 5DB7  F496 6FC8 DABD 7F9A 9BA2
uid                          "David E. O'Brien" <obrien@NUXI.com>
uid                          "David E. O'Brien" <obrien@FreeBSD.org>
sub  3072g/BA32C20D 1998-06-10
```

D.3.262 Jimmy Olgeni <olgeni@FreeBSD.org>

```
pub   2048R/6450AE47 2012-11-01
      Key fingerprint = 7133 AB4D DFC8 0A0D F891  B0D2 90B7 A98E 6450 AE47
uid                  Giacomo Olgeni <olgeni@olgeni.com>
uid                  Jimmy Olgeni <olgeni@FreeBSD.org>
uid                  Giacomo Olgeni <olgeni@moviereading.com>
uid                  Giacomo Olgeni <olgeni@unimaccess.com>
uid                  Giacomo Olgeni <olgeni@colby.it>
uid                  Giacomo Olgeni <olgeni@colby.eu>
uid                  Giacomo Olgeni <olgeni@colby.tv>
sub   2048R/1988BB4B 2012-11-01
```

D.3.263 Philip Paeps <philip@FreeBSD.org>

```
pub   4096R/C5D34D05 2006-10-22
      Key fingerprint = 356B AE02 4763 F739 2FA2  E438 2649 E628 C5D3 4D05
uid                  Philip Paeps <philip@paeps.cx>
uid                  Philip Paeps <philip@nixsys.be>
uid                  Philip Paeps <philip@fosdem.org>
uid                  Philip Paeps <philip@freebsd.org>
uid                  Philip Paeps <philip@pub.telenet.be>
sub   1024D/035EFC58 2006-10-22
sub   2048g/6E5FD7D6 2006-10-22
```

D.3.264 Josh Paetzel <jpaetzel@FreeBSD.org>

```
pub   2048D/F6F63F01 2012-09-21
      Key fingerprint = 1D8D 506E B58C BD10 DC8C  97E1 D6AD 8621 F6F6 3F01
uid                  Josh Paetzel <josh@tcbug.org>
uid                  Josh Paetzel <josh@ixsystems.com>
uid                  Josh Paetzel <jpaetzel@FreeBSD.org>
sub   2048R/F32EF801 2012-09-21
sub   2048R/51F1335D 2012-09-21
sub   2048g/9BC280CD 2012-09-21
sub   2048g/CC793500 2012-09-21
```

D.3.265 Gábor Páli <pgj@FreeBSD.org>

```
pub   4096R/6D7E445C 2013-06-14 [expires: 2018-06-13]
      Key fingerprint = 7AD5 76BA AF2D 14B9 6D45  440B C013 309D 6D7E 445C
uid                  Páli Gábor János (Primary identity) <pali.gabor@gmail.com>
uid                  Páli Gábor János (Eötvös Loránd University) <pgj@inf.elte.hu>
uid                  Gabor Pali (FreeBSD committer) <pgj@FreeBSD.org>
uid                  Páli Gábor János (Magyar BSD Egyesület) <pgj@bsd.hu>
uid                  Páli Gábor János (Eötvös Loránd University) <pgj@elte.hu>
sub   4096R/A57B06AB 2013-06-14 [expires: 2018-06-13]
```

D.3.266 Hiren Panchasara <hiren@FreeBSD.org>

```
pub   4096R/61913185 2013-04-13 [expires: 2014-04-13]
      Key fingerprint = 3336 8104 8D15 B238 2465  136B 4A61 462F 6191 3185
uid                  hiren panchasara <hiren@freebsd.org>
```

D.3.267 Hiten Pandya <hmp@FreeBSD.org>

```
pub  1024D/938CACA8 2004-02-13 Hiten Pandya (FreeBSD) <hmp@FreeBSD.org>
     Key fingerprint = 84EB C75E C75A 50ED 304E  E446 D974 7842 938C ACA8
uid                           Hiten Pandya <hmp@backplane.com>
sub  2048g/783874B5 2004-02-13
```

D.3.268 Dima Panov <fluffy@FreeBSD.org>

```
pub   1024D/93E3B018 2006-11-08
      Key fingerprint = C73E 2B72 1FFD 61BD E206  1234 A626 76ED 93E3 B018
uid                  Dima Panov (FreeBSD.ORG Committer) <fluffy@FreeBSD.ORG>
uid                  Dima Panov (at home) <Fluffy@Fluffy.Khv.RU>
uid                  Dima Panov (at home) <fluffy.khv@gmail.com>
sub   2048g/89047419 2006-11-08

pub   4096R/D5398F29 2009-08-09
```

```
      Key fingerprint = 2D30 2CCB 9984 130C 6F87  BAFC FB8B A09D D539 8F29
uid                  Dima Panov (FreeBSD.ORG Committer) <fluffy@FreeBSD.ORG>
uid                  Dima Panov (at Home) <fluffy@Fluffy.Khv.RU>
uid                  Dima Panov (at GMail) <fluffy.khv@gmail.com>
sub   4096R/915A7785 2009-08-09
```

D.3.269 Andrew Pantyukhin <sat@FreeBSD.org>

```
pub   1024D/6F38A569 2006-05-06
      Key fingerprint = 4E94 994A C2EF CB86 C144  3B04 3381 67C0 6F38 A569
uid                  Andrew Pantyukhin <infofarmer@gubkin.ru>
uid                  Andrew Pantyukhin <sat@FreeBSD.org>
uid                  Andrew Pantyukhin <infofarmer@gmail.com>
uid                  Andrew Pantyukhin <infofarmer@mail.ru>
sub   2048g/5BD4D469 2006-05-06
```

D.3.270 Navdeep Parhar <np@FreeBSD.org>

```
pub   1024D/ACAB8812 2009-06-08
      Key fingerprint = C897 7AFB AFC0 4DA9 7B76  D991 CAB2 2B93 ACAB 8812
uid                  Navdeep Parhar <np@FreeBSD.org>
sub   2048g/AB61D2DC 2009-06-08
```

D.3.271 Rui Paulo <rpaulo@FreeBSD.org>

```
pub   4096R/39CB4153 2010-02-03
      Key fingerprint = ABE8 8465 DE8F F04D E9C8  3FF6 AF89 B2E6 39CB 4153
uid                  Rui Paulo <rpaulo@FreeBSD.org>
uid                  Rui Paulo <rpaulo@gmail.com>
sub   4096R/F87D2F34 2010-02-03
```

D.3.272 Mark Peek <mp@FreeBSD.org>

```
pub  1024D/330D4D01 2002-01-27 Mark Peek <mp@FreeBSD.org>
     Key fingerprint = 510C 96EE B4FB 1B0A 2CF8  A0AF 74B0 0B0E 330D 4D01
sub  1024g/9C6CAC09 2002-01-27
```

D.3.273 Peter Pentchev <roam@FreeBSD.org>

```
pub   1024D/16194553 2002-02-01
      Key fingerprint = FDBA FD79 C26F 3C51 C95E  DF9E ED18 B68D 1619 4553
uid                  Peter Pentchev <roam@ringlet.net>
uid                  Peter Pentchev <roam@cnsys.bg>
uid                  Peter Pentchev <roam@sbnd.net>
```

```
uid                  Peter Pentchev <roam@online.bg>
uid                  Peter Pentchev <roam@orbitel.bg>
uid                  Peter Pentchev <roam@FreeBSD.org>
uid                  Peter Pentchev <roam@techlab.office1.bg>
uid                  Peter Pentchev <roam@hoster.bg>
uid                  Peter Pentchev <roam@space.bg>
sub   1024g/7074473C 2002-02-01

pub   4096R/2527DF13 2009-10-16
      Key fingerprint = 2EE7 A7A5 17FC 124C F115  C354 651E EFB0 2527 DF13
uid                  Peter Pentchev <roam@ringlet.net>
uid                  Peter Pentchev <roamer@users.sourceforge.net>
uid                  Peter Pentchev <roam@cpan.org>
uid                  Peter Pentchev <roam@cnsys.bg>
uid                  Peter Pentchev <roam@sbnd.net>
uid                  Peter Pentchev <roam@online.bg>
uid                  Peter Pentchev <roam@orbitel.bg>
uid                  Peter Pentchev <roam@FreeBSD.org>
uid                  Peter Pentchev <roam@techlab.office1.bg>
uid                  Peter Pentchev <roam@hoster.bg>
uid                  Peter Pentchev <roam@space.bg>
uid                  Peter Pentchev <roam-guest@alioth.debian.org>
uid                  Peter Pentchev <ppentchev@alumni.princeton.edu>
sub   4096R/D0B337AA 2009-10-16
```

D.3.274 Denis Peplin <den@FreeBSD.org>

```
pub  1024D/485DDDF5 2003-09-11 Denis Peplin <den@FreeBSD.org>
     Key fingerprint = 495D 158C 8EC9 C2C1 80F5  EA96 6F72 7C1C 485D DDF5
sub  1024g/E70BA158 2003-09-11
```

D.3.275 Christian S.J. Peron <csjp@FreeBSD.org>

```
pub   1024D/033FA33C 2009-05-16
      Key fingerprint = 74AA 6040 89A7 936E D970  DDC0 CC71 6954 033F A33C
uid                  Christian S.J. Peron <csjp@FreeBSD.ORG>
sub   2048g/856B194A 2009-05-16
```

D.3.276 Gerald Pfeifer <gerald@FreeBSD.org>

```
pub  1024D/745C015A 1999-11-09 Gerald Pfeifer <gerald@pfeifer.com>
     Key fingerprint = B215 C163 3BCA 0477 615F  1B35 A5B3 A004 745C 015A
uid                           Gerald Pfeifer <Gerald.Pfeifer@vibe.at>
uid                           Gerald Pfeifer <pfeifer@dbai.tuwien.ac.at>
uid                           Gerald Pfeifer <gerald@pfeifer.at>
uid                           Gerald Pfeifer <gerald@FreeBSD.org>
sub  1536g/F0156927 1999-11-09
```

D.3.277 Giuseppe Pilichi <jacula@FreeBSD.org>

```
pub   4096R/8B9F4B8B 2006-03-08
      Key fingerprint = 31AD 73AE 0EC0 16E5 4108  8391 D942 5F20 8B9F 4B8B
uid                  Giuseppe Pilichi (Jacula Modyun) <jacula@FreeBSD.org>
uid                  Giuseppe Pilichi (Jacula Modyun) <jaculamodyun@gmail.com>
uid                  Giuseppe Pilichi (Jacula Modyun) <gpilch@gmail.com>
uid                  Giuseppe Pilichi (Jacula Modyun) <jacula@gmail.com>
sub   4096R/FB4D05A3 2006-03-08
```

D.3.278 John Polstra <jdp@FreeBSD.org>

```
pub  1024R/BFBCF449 1997-02-14 John D. Polstra <jdp@polstra.com>
     Key fingerprint = 54 3A 90 59 6B A4 9D 61  BF 1D 03 09 35 8D F6 0D
```

D.3.279 Kirill Ponomarew <krion@FreeBSD.org>

```
pub   1024D/AEB426E5 2002-04-07
      Key fingerprint = 58E7 B953 57A2 D9DD 4960  2A2D 402D 46E9 AEB4 26E5
uid                  Kirill Ponomarew <krion@voodoo.bawue.com>
uid                  Kirill Ponomarew <krion@guug.de>
uid                  Kirill Ponomarew <krion@FreeBSD.org>
sub   1024D/05AC7CA0 2006-01-30 [expires: 2008-01-30]
sub   2048g/C3EE5537 2006-01-30 [expires: 2008-01-30]
```

D.3.280 Stephane E. Potvin <sepotvin@FreeBSD.org>

```
pub   1024D/3097FE7B 2002-08-06
      Key fingerprint = 6B56 62FA ADE1 6F46 BB62  8B1C 99D3 97B5 3097 FE7B
uid                  Stephane E. Potvin <sepotvin@videotron.ca>
uid                  Stephane E. Potvin <stephane.potvin@telcobridges.com>
uid                  Stephane E. Potvin <stephane_potvin@telcobridges.com>
uid                  Stephane E. Potvin <sepotvin@FreeBSD.org>
sub   2048g/0C427BC9 2002-08-06
```

D.3.281 Mark Pulford <markp@FreeBSD.org>

```
pub  1024D/182C368F 2000-05-10 Mark Pulford <markp@FreeBSD.org>
     Key fingerprint = 58C9 C9BF C758 D8D4 7022  8EF5 559F 7F7B 182C 368F
uid                           Mark Pulford <mark@kyne.com.au>
sub  2048g/380573E8 2000-05-10
```

D.3.282 Alejandro Pulver <alepulver@FreeBSD.org>

```
pub   1024D/945C3F61 2005-11-13
      Key fingerprint = 085F E8A2 4896 4B19 42A4  4179 895D 3912 945C 3F61
uid                  Alejandro Pulver (Ale's GPG key pair) <alepulver@FreeBSD.org>
uid                  Alejandro Pulver (Ale's GPG key pair) <alejandro@varnet.biz>
sub   2048g/6890C6CA 2005-11-13
```

D.3.283 Thomas Quinot <thomas@FreeBSD.org>

```
pub  1024D/393D2469 1999-09-23 Thomas Quinot <thomas@cuivre.fr.eu.org>
 Empreinte de la clé = 4737 A0AD E596 6D30 4356  29B8 004D 54B8 393D 2469
uid                            Thomas Quinot <thomas@debian.org>
uid                            Thomas Quinot <thomas@FreeBSD.org>
sub  1024g/8DE13BB2 1999-09-23
```

D.3.284 Herve Quiroz <hq@FreeBSD.org>

```
pub  1024D/85AC8A80 2004-07-22 Herve Quiroz <hq@FreeBSD.org>
     Key fingerprint = 14F5 BC56 D736 102D 41AF  A07B 1D97 CE6C 85AC 8A80
uid                            Herve Quiroz <herve.quiroz@esil.univ-mrs.fr>
sub  1024g/8ECCAFED 2004-07-22
```

D.3.285 Doug Rabson <dfr@FreeBSD.org>

```
pub   1024D/59F57821 2004-02-07
      Key fingerprint = 9451 C4FE 1A7E 117B B95F  1F8F B123 456E 59F5 7821
uid                  Doug Rabson <dfr@nlsystems.com>
sub   1024g/6207AA32 2004-02-07
```

D.3.286 Lars Balker Rasmussen <lbr@FreeBSD.org>

```
pub   1024D/9EF6F27F 2006-04-30
      Key fingerprint = F251 28B7 897C 293E 04F8  71EE 4697 F477 9EF6 F27F
uid                  Lars Balker Rasmussen <lbr@FreeBSD.org>
sub   2048g/A8C1CFD4 2006-04-30
```

D.3.287 Chris Rees <crees@FreeBSD.org>

```
pub   2048R/1E12E96A 2012-08-26
      Key fingerprint = 8C57 BE3B D320 5FFC C4C3  C0B0 900F 45A6 1E12 E96A
uid                  Chris Rees <crees@FreeBSD.org>
sub   2048R/C10740CD 2012-08-26 [expires: 2013-08-26]
```

D.3.288 Jim Rees <rees@FreeBSD.org>

```
pub   512/B623C791 1995/02/21 Jim Rees <rees@umich.edu>
          Key fingerprint =  02 5F 1B 15 B4 6E F1 3E  F1 C5 E0 1D EA CC 17 88
```

D.3.289 Benedict Reuschling <bcr@FreeBSD.org>

```
pub   1024D/4A819348 2009-05-24
      Key fingerprint = 2D8C BDF9 30FA 75A5 A0DF  D724 4D26 502E 4A81 9348
uid                  Benedict Reuschling <bcr@FreeBSD.org>
sub   2048g/8DA16EDD 2009-05-24
```

D.3.290 Tom Rhodes <trhodes@FreeBSD.org>

```
pub   1024D/FB7D88E1 2008-05-07
      Key fingerprint = 8279 3100 2DF2 F00E 7FDD  AC2C 5776 23AB FB7D 88E1
uid                  Tom Rhodes (trhodes) <trhodes@FreeBSD.org>
sub   4096g/7B0CD79F 2008-05-07
```

D.3.291 Benno Rice <benno@FreeBSD.org>

```
pub   4096R/C5F10BED 2013-05-21 [expires: 2017-05-21]
      Key fingerprint = 77EB 5A9E 97C7 2D2D 6D0A  1B6C C619 4C61 C5F1 0BED
uid                  Benno Rice <benno@FreeBSD.org>
uid                  Benno Rice <benno@jeamland.net>
sub   4096R/408068BC 2013-05-21 [expires: 2017-05-21]
```

D.3.292 Beech Rintoul <beech@FreeBSD.org>

```
pub   2048D/68DFAE1F 2013-02-26
      Key fingerprint = D58B 3E9D B0E3 E081 EC6F  69D9 CDA3 51DD 68DF AE1F
uid                  Beech Rintoul <beech@freebsd.org>
sub   2048g/960F45D9 2013-02-26
```

D.3.293 Matteo Riondato <matteo@FreeBSD.org>

```
pub   1024D/1EC56BEC 2003-01-05 [expires: 2009-09-07]
      Key fingerprint = F0F3 1B43 035D 65B1 08E9  4D66 D8CA 78A5 1EC5 6BEC
uid                  Matteo Riondato (Rionda) <matteo@FreeBSD.ORG>
uid                  Matteo Riondato (Rionda) <rionda@riondabsd.net>
uid                  Matteo Riondato (Rionda) <rionda@gufi.org>
uid                  Matteo Riondato (Rionda) <matteo@riondato.com>
uid                  Matteo Riondato (Rionda) <rionda@riondato.com>
uid                  Matteo Riondato (Rionda) <rionda@FreeSBIE.ORG>
```

```
uid                  Matteo Riondato (Rionda) <rionda@autistici.org>
sub   2048g/87C44A55 2008-09-23 [expires: 2009-09-23]
```

D.3.294 Ollivier Robert <roberto@FreeBSD.org>

```
pub   1024D/7DCAE9D3 1997-08-21
      Key fingerprint = 2945 61E7 D4E5 1D32 C100  DBEC A04F FB1B 7DCA E9D3
uid                  Ollivier Robert <roberto@keltia.freenix.fr>
uid                  Ollivier Robert <roberto@FreeBSD.org>
sub   2048g/C267084D 1997-08-21
```

D.3.295 Craig Rodrigues <rodrigc@FreeBSD.org>

```
pub   1024D/3998479D 2005-05-20
      Key fingerprint = F01F EBE6 F5C8 6DC2 954F  098F D20A 8A2A 3998 479D
uid                  Craig Rodrigues <rodrigc@freebsd.org>
uid                  Craig Rodrigues <rodrigc@crodrigues.org>
sub   2048g/AA77E09B 2005-05-20
```

D.3.296 Guido van Rooij <guido@FreeBSD.org>

```
pub  1024R/599F323D 1996-05-18 Guido van Rooij <guido@gvr.org>
     Key fingerprint = 16 79 09 F3 C0 E4 28 A7  32 62 FA F6 60 31 C0 ED
uid                            Guido van Rooij <guido@gvr.win.tue.nl>

pub  1024D/A95102C1 2000-10-25 Guido van Rooij <guido@madison-gurkha.nl>
     Key fingerprint = 5B3E 51B7 0E7A D170 0574  1E51 2471 117F A951 02C1
uid                            Guido van Rooij <guido@madison-gurkha.com>
sub  1024g/A5F20553 2000-10-25
```

D.3.297 Eygene Ryabinkin <rea@FreeBSD.org>

```
pub   3072D/8152ECFB 2010-10-27
      Key fingerprint = 82FE 06BC D497 C0DE 49EC  4FF0 16AF 9EAE 8152 ECFB
uid                  Eygene Ryabinkin <rea-fbsd@codelabs.ru>
uid                  Eygene Ryabinkin <rea@freebsd.org>
uid                  Eygene Ryabinkin <rea@codelabs.ru>
sub   3072g/5FC03749 2010-10-27
```

D.3.298 Aleksandr Rybalko <ray@FreeBSD.org>

```
pub   2048R/4B7B7A4E 2011-05-24
      Key fingerprint = BB9F D01D 7327 0B33 B2F5  6C72 EC49 E6ED 4B7B 7A4E
uid                  Aleksandr Rybalko (Aleksandr Rybalko FreeBSD project identification) <ray@fr
```

```
sub   2048R/99F9F9EF 2011-05-24
```

D.3.299 Niklas Saers <niklas@FreeBSD.org>

```
pub  1024D/C822A476 2004-03-09 Niklas Saers <niklas@saers.com>
     Key fingerprint = C41E F734 AF0E 3D21 7499  9EB1 9A31 2E7E C822 A476
sub  1024g/81E2FF36 2004-03-09
```

D.3.300 Boris Samorodov <bsam@FreeBSD.org>

```
pub   1024D/ADFD5C9A 2006-06-21
      Key fingerprint = 81AA FED0 6050 208C 0303  4007 6C03 7263 ADFD 5C9A
uid                  Boris Samorodov (FreeBSD) <bsam@freebsd.org>
sub   2048g/7753A3F1 2006-06-21
```

D.3.301 Mark Santcroos <marks@FreeBSD.org>

```
pub   1024D/DBE7EB8E 2005-03-08
      Key fingerprint = C0F0 44F3 3F15 520F 6E32  186B BE0A BA42 DBE7 EB8E
uid                  Mark Santcroos <marks@ripe.net>
uid                  Mark Santcroos <mark@santcroos.net>
uid                  Mark Santcroos <marks@freebsd.org>
sub   2048g/FFF80F85 2005-03-08
```

D.3.302 Bernhard Schmidt <bschmidt@FreeBSD.org>

```
pub   1024D/5F754FBC 2009-06-15
      Key fingerprint = 6B87 C8A9 6BA5 6B18 11CF  8C38 A1B7 0731 5F75 4FBC
uid                  Bernhard Schmidt <bschmidt@FreeBSD.org>
uid                  Bernhard Schmidt <bschmidt@techwires.net>
sub   1024g/1945DC1D 2009-06-15
```

D.3.303 Wolfram Schneider <wosch@FreeBSD.org>

```
Type Bits/KeyID    Date       User ID
pub  1024/2B7181AD 1997/08/09 Wolfram Schneider <wosch@FreeBSD.org>
            Key fingerprint = CA 16 91 D9 75 33 F1 07  1B F0 B4 9F 3E 95 B6 09
```

D.3.304 Ed Schouten <ed@FreeBSD.org>

```
pub   4096R/3491A2BB 2011-03-12 [expires: 2016-03-10]
      Key fingerprint = A110 5982 A887 74A2 F4B1  D70A 6E5E D8FE 3491 A2BB
uid                  Ed Schouten (The FreeBSD Project) <ed@FreeBSD.org>
uid                  Ed Schouten <ed@80386.nl>
sub   4096R/81BB41E6 2011-03-12 [expires: 2016-03-10]
```

D.3.305 David Schultz <das@FreeBSD.org>

```
pub  1024D/BE848B57 2001-07-19 David Schultz <das@FreeBSD.ORG>
     Key fingerprint = 0C12 797B A9CB 19D9 FDAF  2A39 2D76 A2DB BE84 8B57
uid  David Schultz <dschultz@uclink.Berkeley.EDU>
uid  David Schultz <das@FreeBSD.ORG>
sub  2048g/69206E8E 2001-07-19
```

D.3.306 Michael Scheidell <scheidell@FreeBSD.org>

```
pub   2048R/34622C1D 2011-11-16
      Key fingerprint = 0A0C 9ECA 18EC 47AC C715  2187 91B9 F9FE 3462 2C1D
uid                  Michael Scheidell <scheidell@freebsd.org>
sub   2048R/8F241971 2011-11-16
```

D.3.307 Jens Schweikhardt <schweikh@FreeBSD.org>

```
pub  1024D/0FF231FD 2002-01-27 Jens Schweikhardt <schweikh@FreeBSD.org>
     Key fingerprint = 3F35 E705 F02F 35A1 A23E  330E 16FE EA33 0FF2 31FD
uid                            Jens Schweikhardt <schweikh@schweikhardt.net>
sub  1024g/6E93CACC 2002-01-27 [expires: 2005-01-26]
```

D.3.308 Matthew Seaman <matthew@FreeBSD.org>

```
pub   1024D/60AE908C 2005-12-17 [expires: 2012-03-21]
      Key fingerprint = B555 2A96 274E D248 5734  0EB4 F0C8 E4E7 60AE 908C
uid                  Matthew Seaman <m.seaman@infracaninophile.co.uk>
uid                  Matthew Seaman <m.seaman@black-earth.co.uk>
uid                  Matthew Seaman <matthew@freebsd.org>
sub   2048g/58BFDA29 2005-12-17 [expires: 2012-03-21]
sub   1024D/9B19F956 2006-12-18 [expires: 2012-03-21]
```

D.3.309 Thomas-Martin Seck <tmseck@FreeBSD.org>

```
pub   1024D/DF46EE05 2000-11-22
      Key fingerprint = A38F AE66 6B11 6EB9 5D1A  B67D 2444 2FE1 DF46 EE05
uid                  Thomas-Martin Seck (Privat 2) <tmseck@netcologne.de>
uid                  Thomas-Martin Seck (Privat) <tmseck@web.de>
uid                  Thomas-Martin Seck (FreeBSD) <tmseck@FreeBSD.org>
sub   2048g/3DC33B0F 2000-11-22
```

D.3.310 Stanislav Sedov <stas@FreeBSD.org>

```
pub   4096R/092FD9F0 2009-05-23
      Key fingerprint = B83A B15D 929A 364A D8BC  B3F9 BF25 A231 092F D9F0
uid                  Stanislav Sedov <stas@FreeBSD.org>
uid                  Stanislav Sedov <stas@SpringDaemons.com>
uid                  Stanislav Sedov (Corporate email) <stas@deglitch.com>
uid                  Stanislav Sedov (Corporate email) <stas@ht-systems.ru>
uid                  Stanislav Sedov (Corporate email) <ssedov@3playnet.com>
uid                  Stanislav Sedov <ssedov@mbsd.msk.ru>
uid                  Stanislav Sedov (Corporate email) <ssedov@swifttest.com>
sub   4096R/6FD2025F 2009-05-23
```

D.3.311 Johan van Selst <johans@FreeBSD.org>

```
pub   4096R/D3AE8D3A 2009-09-01
      Key fingerprint = 31C8 D089 DDB6 96C6 F3C1  29C0 A9C8 6C8D D3AE 8D3A
uid                  Johan van Selst
uid                  Johan van Selst <johans@gletsjer.net>
uid                  Johan van Selst <johans@stack.nl>
uid                  Johan van Selst <johans@FreeBSD.org>
uid                  Johan van Selst (GSWoT:NL50) <johans@gswot.org>
sub   2048R/B002E38C 2009-09-01
sub   2048R/1EBCAECB 2009-09-01
sub   2048R/639A1446 2009-09-01
sub   3072D/6F2708F4 2009-09-01
sub   4096g/D6F89E83 2009-09-01
```

D.3.312 Bakul Shah <bakul@FreeBSD.org>

```
pub   1024D/86AEE4CB 2006-04-20
      Key fingerprint = 0389 26E8 381C 6980 AEC0  10A5 E540 A157 86AE E4CB
uid                  Bakul Shah <bakul@freebsd.org>
sub   2048g/5C3DCC24 2006-04-20
```

D.3.313 Gregory Neil Shapiro <gshapiro@FreeBSD.org>

```
pub  1024R/4FBE2ADD 2000-10-13 Gregory Neil Shapiro <gshapiro@gshapiro.net>
     Key fingerprint = 56 D5 FF A7 A6 54 A6 B5  59 10 00 B9 5F 5F 20 09
uid                            Gregory Neil Shapiro <gshapiro@FreeBSD.org>

pub  1024D/F76A9BF5 2001-11-14 Gregory Neil Shapiro <gshapiro@FreeBSD.org>
     Key fingerprint = 3B5E DAF1 4B04 97BA EE20  F841 21F9 C5BC F76A 9BF5
uid                            Gregory Neil Shapiro <gshapiro@gshapiro.net>
sub  2048g/935657DC 2001-11-14

pub  1024D/FCE56561 2000-10-14 Gregory Neil Shapiro <gshapiro@FreeBSD.org>
     Key fingerprint = 42C4 A87A FD85 C34F E77F  5EA1 88E1 7B1D FCE5 6561
uid                            Gregory Neil Shapiro <gshapiro@gshapiro.net>
sub  1024g/285DC8A0 2000-10-14 [expires: 2001-10-14]
```

D.3.314 Arun Sharma <arun@FreeBSD.org>

```
pub  1024D/7D112181 2003-03-06 Arun Sharma <arun@sharma-home.net>
     Key fingerprint = A074 41D6 8537 C7D5 070E  0F78 0247 1AE2 7D11 2181
uid                            Arun Sharma <arun@freebsd.org>
uid                            Arun Sharma <arun.sharma@intel.com>
sub  1024g/ACAD98DA 2003-03-06 [expires: 2005-03-05]
```

D.3.315 Wesley Shields <wxs@FreeBSD.org>

```
pub   1024D/17F0AA37 2007-12-27
      Key fingerprint = 96D1 2E6B F61C 2F3D 83EF  8F0B BE54 310C 17F0 AA37
uid                  Wesley Shields <wxs@FreeBSD.org>
uid                  Wesley Shields <wxs@atarininja.org>
sub   2048g/2EDA1BB8 2007-12-27
```

D.3.316 Norikatsu Shigemura <nork@FreeBSD.org>

```
pub   1024D/7104EA4E 2005-02-14
      Key fingerprint = 9580 60A3 B58A 0864 79CB  779A 6FAE 229B 7104 EA4E
uid                  Norikatsu Shigemura <nork@cityfujisawa.ne.jp>
uid                  Norikatsu Shigemura <nork@ninth-nine.com>
uid                  Norikatsu Shigemura <nork@FreeBSD.org>
sub   4096g/EF56997E 2005-02-14
```

D.3.317 Shteryana Shopova <syrinx@FreeBSD.org>

```
pub   1024D/1C139BC5 2006-10-07
      Key fingerprint = B83D 2451 27AB B767 504F  CB85 4FB1 C88B 1C13 9BC5
uid                  Shteryana Shopova (syrinx) <shteryana@FreeBSD.org>
```

```
sub   2048g/6D2E9C98 2006-10-07
```

D.3.318 Vanilla I. Shu <vanilla@FreeBSD.org>

```
pub  1024D/ACE75853 2001-11-20 Vanilla I. Shu <vanilla@FreeBSD.org>
     Key fingerprint = 290F 9DB8 42A3 6257 5D9A  5585 B25A 909E ACE7 5853
sub  1024g/CE695D0E 2001-11-20
```

D.3.319 Ashish SHUKLA <ashish@FreeBSD.org>

```
pub   4096R/E74FA4B0 2010-04-13
      Key fingerprint = F682 CDCC 39DC 0FEA E116  20B6 C746 CFA9 E74F A4B0
uid                  Ashish SHUKLA <wahjava@gmail.com>
uid                  Ashish SHUKLA <wahjava@googlemail.com>
uid                  Ashish SHUKLA <wahjava.ml@gmail.com>
uid                  Ashish SHUKLA <wahjava@members.fsf.org>
uid                  Ashish SHUKLA <wahjava@perl.org.in>
uid                  Ashish SHUKLA <wahjava@users.sourceforge.net>
uid                  Ashish SHUKLA <wah.java@yahoo.com>
uid                  Ashish SHUKLA <wah_java@hotmail.com>
uid                  Ashish SHUKLA <ashish.shukla@airtelmail.in>
uid                  Ashish SHUKLA <wahjava@member.fsf.org>
uid                  [jpeg image of size 4655]
uid                  Ashish SHUKLA (FreeBSD Committer Address) <ashish@FreeBSD.ORG>
sub   4096R/F20D202D 2010-04-13
```

D.3.320 Bruce M. Simpson <bms@FreeBSD.org>

```
pub  1024D/860DB53B 2003-08-06 Bruce M Simpson <bms@freebsd.org>
     Key fingerprint = 0D5F 1571 44DF 51B7 8B12  041E B9E5 2901 860D B53B
sub  2048g/A2A32D8B 2003-08-06 [expires: 2006-08-05]
```

D.3.321 Dmitry Sivachenko <demon@FreeBSD.org>

```
pub  1024D/13D5DF80 2002-03-18 Dmitry Sivachenko <mitya@cavia.pp.ru>
     Key fingerprint = 72A9 12C9 BB02 46D4 4B13  E5FE 1194 9963 13D5 DF80
uid                           Dmitry S. Sivachenko <demon@FreeBSD.org>
sub  1024g/060F6DBD 2002-03-18
```

D.3.322 Jesper Skriver <jesper@FreeBSD.org>

```
pub  1024D/F9561C31 2001-03-09 Jesper Skriver <jesper@FreeBSD.org>
     Key fingerprint = 6B88 9CE8 66E9 E631 C9C5  5EB4 22AB F0EC F956 1C31
uid                            Jesper Skriver <jesper@skriver.dk>
```

```
uid                           Jesper Skriver <jesper@wheel.dk>
sub  1024g/777C378C 2001-03-09
```

D.3.323 Ville Skyttä <scop@FreeBSD.org>

```
pub  1024D/BCD241CB 2002-04-07 Ville Skyttä <ville.skytta@iki.fi>
     Key fingerprint = 4E0D EBAB 3106 F1FA 3FA9  B875 D98C D635 BCD2 41CB
uid                           Ville Skyttä <ville.skytta@xemacs.org>
uid                           Ville Skyttä <scop@FreeBSD.org>
sub  2048g/9426F4D1 2002-04-07
```

D.3.324 Andrey Slusar <anray@FreeBSD.org>

```
pub   1024D/AE7B5418 2005-12-12
      Key fingerprint = DE70 C24B 55A0 4A06 68A1  D425 3C59 9A9B AE7B 5418
uid                  Andrey Slusar <anray@ext.by>
uid                  Andrey Slusar <anrays@gmail.com>
uid                  Andrey Slusar <anray@FreeBSD.org>
sub   2048g/7D0EB77D 2005-12-12
```

D.3.325 Florian Smeets <flo@FreeBSD.org>

```
pub   1024D/C942BF09 2008-10-24
      Key fingerprint = 54BB 157B 8DB2 9E46 4A3C  69AB 6A9A 3C3F C942 BF09
uid                  Florian Smeets <flo@smeets.im>
uid                  Florian Smeets <flo@kasimir.com>
uid                  Florian Smeets <flo@FreeBSD.org>
sub   2048g/4AAF040E 2008-10-24
```

D.3.326 Gleb Smirnoff <glebius@FreeBSD.org>

```
pub   2048D/6C7E5E82 2013-01-30 [expires: 2023-08-25]
      Key fingerprint = 6E06 7260 B83D CF2C A93C  566F 5185 0968 6C7E 5E82
uid                  Gleb Smirnoff <glebius@FreeBSD.org>
sub   2048g/11E89DCE 2013-01-30 [expires: 2023-08-25]
```

D.3.327 Ken Smith <kensmith@FreeBSD.org>

```
pub  1024D/29AEA7F6 2003-12-02 Ken Smith <kensmith@cse.buffalo.edu>
     Key fingerprint = 4AB7 D302 0753 8215 31E7  F1AD FC6D 7855 29AE A7F6
uid                           Ken Smith <kensmith@freebsd.org>
sub  1024g/0D509C6C 2003-12-02
```

D.3.328 Ben Smithurst <ben@FreeBSD.org>

```
pub  1024D/2CEF442C 2001-07-11 Ben Smithurst <ben@LSRfm.com>
     Key fingerprint = 355D 0FFF B83A 90A9 D648  E409 6CFC C9FB 2CEF 442C
uid                           Ben Smithurst <ben@vinosystems.com>
uid                           Ben Smithurst <ben@smithurst.org>
uid                           Ben Smithurst <ben@FreeBSD.org>
uid                           Ben Smithurst <csxbcs@comp.leeds.ac.uk>
uid                           Ben Smithurst <ben@scientia.demon.co.uk>
sub  1024g/347071FF 2001-07-11
```

D.3.329 Dag-Erling Smørgrav <des@FreeBSD.org>

```
pub   4096R/F94E87B2 2013-02-15 [expires: 2015-01-01]
      Key fingerprint = 578A 3F4F 9E04 9FCF 3576  BF82 BB9B 471B F94E 87B2
uid                  Dag-Erling Smørgrav <des@usit.uio.no>
uid                  Dag-Erling Smørgrav <des@des.no>
uid                  Dag-Erling Smørgrav <des@freebsd.org>
uid                  [jpeg image of size 4779]
sub   4096R/F4DE87F5 2013-02-15 [expires: 2015-01-01]
```

D.3.330 Maxim Sobolev <sobomax@FreeBSD.org>

```
pub  1024D/888205AF 2001-11-21 Maxim Sobolev <sobomax@FreeBSD.org>
     Key fingerprint = 85C9 DCB0 6828 087C C977  3034 A0DB B9B7 8882 05AF
uid                           Maxim Sobolev <sobomax@mail.ru>
uid                           Maxim Sobolev <sobomax@altavista.net>
uid                           Maxim Sobolev <vegacap@i.com.ua>

pub  1024D/468EE6D8 2003-03-21 Maxim Sobolev <sobomax@portaone.com>
     Key fingerprint = 711B D315 3360 A58F 9A0E  89DB 6D40 2558 468E E6D8
uid                           Maxim Sobolev <sobomax@FreeBSD.org>
uid                           Maxim Sobolev <sobomax@mail.ru>
uid                           Maxim Sobolev <vegacap@i.com.ua>

pub  1024D/6BEC980A 2004-02-13 Maxim Sobolev <sobomax@portaone.com>
     Key fingerprint = 09D5 47B4 8D23 626F B643  76EB DFEE 3794 6BEC 980A
uid                           Maxim Sobolev <sobomax@FreeBSD.org>
uid                           Maksym Sobolyev (It’s how they call me in official documents. Pret
uid                           Maksym Sobolyev (It’s how they call me in official documents. Pret
sub  2048g/16D049AB 2004-02-13 [expires: 2005-02-12]
```

D.3.331 Alan Somers <asomers@FreeBSD.org>

```
pub   4096R/DA05FCE8 2013-04-25 [expires: 2018-04-24]
      Key fingerprint = 9CD4 C982 738F 8B90 25E8  E6B3 5F74 63BC DA05 FCE8
uid                  Alan Somers <asomers@freebsd.org>
uid                  Alan Somers <asomers@gmail.com>
```

```
sub   4096R/4E121B3E 2013-04-25 [expires: 2018-04-24]
```

D.3.332 Brian Somers <brian@FreeBSD.org>

```
pub  1024R/666A7421 1997-04-30 Brian Somers <brian@freebsd-services.com>
     Key fingerprint = 2D 91 BD C2 94 2C 46 8F  8F 09 C4 FC AD 12 3B 21
uid                            Brian Somers <brian@awfulhak.org>
uid                            Brian Somers <brian@FreeBSD.org>
uid                            Brian Somers <brian@OpenBSD.org>
uid                            Brian Somers <brian@uk.FreeBSD.org>
uid                            Brian Somers <brian@uk.OpenBSD.org>
```

D.3.333 Stacey Son <sson@FreeBSD.org>

```
pub   1024D/CE8319F3 2008-07-08
      Key fingerprint = 64C7 8D92 C1DF B940 1171  5ED3 186A 758A CE83 19F3
uid                  Stacey Son <sson@FreeBSD.org>
uid                  Stacey Son <stacey@son.org>
uid                  Stacey Son <sson@byu.net>
uid                  Stacey Son <sson@secure.net>
uid                  Stacey Son <sson@dev-random.com>
sub   2048g/0F724E52 2008-07-08
```

D.3.334 Nicolas Souchu <nsouch@FreeBSD.org>

```
pub  1024D/C744F18B 2002-02-13 Nicholas Souchu <nsouch@freebsd.org>
     Key fingerprint = 992A 144F AC0F 40BA 55AE  DE6D 752D 0A6C C744 F18B
sub  1024g/90BD3231 2002-02-13
```

D.3.335 Suleiman Souhlal <ssouhlal@FreeBSD.org>

```
pub  1024D/2EA50469 2004-07-24 Suleiman Souhlal <ssouhlal@FreeBSD.org>
     Key fingerprint = DACF 89DB 54C7 DA1D 37AF  9A94 EB55 E272 2EA5 0469
sub  2048g/0CDCC535 2004-07-24
```

D.3.336 Ulrich Spörlein <uqs@FreeBSD.org>

```
pub   2048R/4AAF82CE 2010-01-27 [expires: 2015-01-26]
      Key fingerprint = 08DF A6A0 B1EB 98A5 EDDA  9005 A3A6 9864 4AAF 82CE
uid                   Ulrich Spörlein <uqs@spoerlein.net>
uid                   Ulrich Spoerlein <uspoerlein@gmail.com>
uid                   Ulrich Spörlein (The FreeBSD Project) <uqs@FreeBSD.org>
uid                   Ulrich Spörlein <ulrich.spoerlein@web.de>
sub   2048R/162E8BD2 2010-01-27 [expires: 2015-01-26]
```

D.3.337 Rink Springer <rink@FreeBSD.org>

```
pub   1024D/ECEDBFFF 2003-09-19
      Key fingerprint = A8BE 9C82 9B81 4289 A905  418D 6F73 BAD2 ECED BFFF
uid                  Rink Springer <rink@il.fontys.nl>
uid                  Rink Springer (FreeBSD Project) <rink@FreeBSD.org>
uid                  Rink Springer <rink@stack.nl>
sub   2048g/3BC3E67E 2003-09-19
```

D.3.338 Vsevolod Stakhov <vsevolod@FreeBSD.org>

```
pub   4096R/90081437 2012-05-16 [expires: 2017-05-15]
      Key fingerprint = DD9A 126C E675 1EA5 2A97  04A3 0764 7B67 9008 1437
uid                  Vsevolod Stakhov <vsevolod@FreeBSD.org>
sub   4096R/4A5A0B54 2012-05-16 [expires: 2017-05-15]
```

D.3.339 Ryan Steinmetz <zi@FreeBSD.org>

```
pub   1024D/7AD7FAF2 2004-01-21
      Key fingerprint = EF36 D45A 5CA9 28B1 A550  18CD A43C D111 7AD7 FAF2
uid                  Ryan Steinmetz <zi@FreeBSD.org>
uid                  Ryan Steinmetz <rpsfa@rit.edu>
uid                  Ryan Steinmetz <zi@zi0r.com>
sub   1024g/058BC057 2004-01-21
sub   4096g/0EB108D2 2006-02-27
sub   1024D/FEF36DD7 2006-02-27
```

D.3.340 Randall R. Stewart <rrs@FreeBSD.org>

```
pub   1024D/0373B8B2 2006-09-01
      Key fingerprint = 74A6 810E 6DEA D69B 6496  5FA9 8AEF 4166 0373 B8B2
uid                  Randall R Stewart <randall@lakerest.net>
uid                  Randall R Stewart <rrs@cisco.com>
uid                  Randall R Stewart <rrs@FreeBSD.org>
sub   2048g/88027C0B 2006-09-01
```

D.3.341 Murray Stokely <murray@FreeBSD.org>

```
pub  1024D/0E451F7D 2001-02-12 Murray Stokely <murray@freebsd.org>
     Key fingerprint = E2CA 411D DD44 53FD BB4B  3CB5 B4D7 10A2 0E45 1F7D
sub  1024g/965A770C 2001-02-12
```

D.3.342 Volker Stolz <vs@FreeBSD.org>

```
pub  1024R/3FD1B6B5 1998-06-16 Volker Stolz <vs@freebsd.org>
     Key fingerprint = 69 6F BD A0 2E FE 19 66  CF B9 68 6E 41 7D F9 B9
uid                           Volker Stolz <stolz@i2.informatik.rwth-aachen.de> (LSK)
uid                           Volker Stolz <vs@foldr.org>
```

D.3.343 Ryan Stone <rstone@FreeBSD.org>

```
pub   1024D/3141B73A 2010-04-13
      Key fingerprint = 4A6D DC04 DDC5 0822 2687  A086 FD3F 16CB 3141 B73A
uid                  Ryan Stone (FreeBSD) <rstone@freebsd.org>
sub   2048g/A8500B5F 2010-04-13
```

D.3.344 Søren Straarup <xride@FreeBSD.org>

```
pub   1024D/E683AD40 2006-09-28
      Key fingerprint = 8A0E 7E57 144B BC25 24A9  EC1A 0DBC 3408 E683 AD40
uid                  Soeren Straarup <xride@xride.dk>
uid                  Soeren Straarup <xride@FreeBSD.org>
uid                  Soeren Straarup <xride@x12.dk>
sub   2048g/2B18B3B8 2006-09-28
```

D.3.345 Marius Strobl <marius@FreeBSD.org>

```
pub   1024D/E0AC6F8D 2004-04-16
      Key fingerprint = 3A6C 4FB1 8BB9 4F2E BDDC  4AB6 D035 799C E0AC 6F8D
uid                  Marius Strobl <marius@FreeBSD.org>
uid                  Marius Strobl <marius@alchemy.franken.de>
sub   1024g/08BBD875 2004-04-16
```

D.3.346 Carlo Strub <cs@FreeBSD.org>

```
pub   3072R/D06F0BD7 2012-11-25 [expires: 2017-11-24]
      Key fingerprint = 61A4 F2B8 2A6C B81E 5557  0798 78E7 DE70 D06F 0BD7
uid                  Carlo Strub <cs@carlostrub.ch>
uid                  Carlo Strub <cs@FreeBSD.org>
sub   3072R/71C75997 2012-11-25 [expires: 2017-11-24]
sub   3072R/318AEB16 2012-11-25 [expires: 2017-11-24]
```

D.3.347 Cheng-Lung Sung <clsung@FreeBSD.org>

```
pub  1024D/956E8BC1 2003-09-12 Cheng-Lung Sung <clsung@FreeBSD.org>
     Key fingerprint = E0BC 57F9 F44B 46C6 DB53  8462 F807 89F3 956E 8BC1
uid                            Cheng-Lung Sung (Software Engineer) <clsung@dragon2.net>
uid                            Cheng-Lung Sung (Alumnus of CSIE, NCTU, Taiwan) <clsung@sungsung.c
uid                            Cheng-Lung Sung (AlanSung) <clsung@tiger2.net>
uid                            Cheng-Lung Sung (FreeBSD@Taiwan) <clsung@freebsd.csie.nctu.edu.tw>
uid                            Cheng-Lung Sung (Ph.D. Student of NTU.EECS) <d92921016@ntu.edu.tw>
uid                            Cheng-Lung Sung (FreeBSD Freshman) <clsung@tw.freebsd.org>
uid                            Cheng-Lung Sung (ports committer) <clsung@FreeBSD.org>
sub  1024g/1FB800C2 2003-09-12
```

D.3.348 Gregory Sutter <gsutter@FreeBSD.org>

```
pub  1024D/845DFEDD 2000-10-10 Gregory S. Sutter <gsutter@zer0.org>
     Key fingerprint = D161 E4EA 4BFA 2427 F3F9  5B1F 2015 31D5 845D FEDD
uid                            Gregory S. Sutter <gsutter@freebsd.org>
uid                            Gregory S. Sutter <gsutter@daemonnews.org>
uid                            Gregory S. Sutter <gsutter@pobox.com>
sub  2048g/0A37BBCE 2000-10-10
```

D.3.349 Koichi Suzuki <metal@FreeBSD.org>

```
pub  1024D/AE562682 2004-05-23 SUZUKI Koichi <metal@FreeBSD.org>
     Key fingerprint = 92B9 A202 B5AB 8CB6 89FC  6DD1 5737 C702 AE56 2682
sub  4096g/730E604B 2004-05-23
```

D.3.350 Ryusuke SUZUKI <ryusuke@FreeBSD.org>

```
pub   1024D/63D29724 2009-12-18
      Key fingerprint = B108 7109 2E62 BECB 0F78  FE65 1B9A D1BE 63D2 9724
uid                   Ryusuke SUZUKI <ryusuke@FreeBSD.org>
uid                   Ryusuke SUZUKI <ryusuke@jp.FreeBSD.org>
sub   1024g/5E4DD044 2009-12-18
```

D.3.351 Gary W. Swearingen <garys@FreeBSD.org>

```
pub   1024D/FAA48AD5 2005-08-22 [expires: 2007-08-22]
      Key fingerprint = 8292 CC3E 81B5 E54F E3DD  F987 FA52 E643 FAA4 8AD5
uid                   Gary W. Swearingen <garys@freebsd.org>
sub   2048g/E34C3CA0 2005-08-22 [expires: 2007-08-22]
```

D.3.352 Yoshihiro Takahashi <nyan@FreeBSD.org>

```
pub   4096R/6624859E 2012-11-18
      Key fingerprint = 1CA5 445E 7ABD BC21 AEC0  7B89 47D7 4EFF 6624 859E
uid                  Yoshihiro TAKAHASHI <nyan@furiru.org>
uid                  Yoshihiro TAKAHASHI <nyan@FreeBSD.org>
uid                  Yoshihiro TAKAHASHI <nyan@jp.FreeBSD.org>
sub   4096R/362726EA 2012-11-18
```

D.3.353 Sahil Tandon <sahil@FreeBSD.org>

```
pub   2048R/C016D977 2010-04-08
      Key fingerprint = 6AD2 BA99 8E3A 8DA6 DFC1  53CF DBD0 6001 C016 D977
uid                  Sahil Tandon <sahil@tandon.net>
uid                  Sahil Tandon <sahil@FreeBSD.org>
sub   2048R/F7776FBC 2010-04-08
```

D.3.354 TAKATSU Tomonari <tota@FreeBSD.org>

```
pub   1024D/67F58F29 2009-05-17
      Key fingerprint = 6940 B575 FC4A FA26 C094  279A 4B9B 6326 67F5 8F29
uid                  TAKATSU Tomonari <tota@FreeBSD.org>
sub   2048g/18B112CD 2009-05-17
```

D.3.355 Romain Tartière <romain@FreeBSD.org>

```
pub   3072R/5112336F 2010-04-09
      Key fingerprint = 8234 9A78 E7C0 B807 0B59  80FF BA4D 1D95 5112 336F
uid                  Romain Tartière <romain@blogreen.org>
uid                  Romain Tartière (FreeBSD) <romain@FreeBSD.org>
sub   3072R/C1B2B656 2010-04-09
sub   3072R/8F8125F4 2010-04-09
```

D.3.356 Sylvio Cesar Teixeira <sylvio@FreeBSD.org>

```
pub   2048R/AA7395A1 2009-10-28
      Key fingerprint = B319 6AAF 0016 4308 6D93  E652 3C5F 21A2 AA73 95A1
uid                  Sylvio Cesar Teixeira (My key) <sylvio@FreeBSD.org>
sub   2048R/F758F556 2009-10-28
```

D.3.357 Ion-Mihai Tetcu <itetcu@FreeBSD.org>

```
pub   4096R/29597D20 2013-05-02
      Key fingerprint = AB6F 39B6 605D E6B7 0D54  ED3D BCA2 129A 2959 7D20
uid                  Ion-Mihai Tetcu (FreeBSD Committer key) <itetcu@FreeBSD.org>
sub   4096R/EC9E17E3 2013-05-02
```

D.3.358 Mikhail Teterin <mi@FreeBSD.org>

```
pub  1024R/3FC71479 1995-09-08 Mikhail Teterin <mi@aldan.star89.galstar.com>
     Key fingerprint = 5F 15 EA 78 A5 40 6A 0F  14 D7 D9 EA 6E 2B DA A4
```

D.3.359 Gordon Tetlow <gordon@FreeBSD.org>

```
pub  1024D/357D65FB 2002-05-14 Gordon Tetlow <gordont@gnf.org>
     Key fingerprint = 34EF AD12 10AF 560E C3AE  CE55 46ED ADF4 357D 65FB
uid                            Gordon Tetlow <gordon@FreeBSD.org>
sub  1024g/243694AB 2002-05-14
```

D.3.360 Lars Thegler <lth@FreeBSD.org>

```
pub  1024D/56B0CA08 2004-05-31 Lars Thegler <lth@FreeBSD.org>
     Key fingerprint = ABAE F98C EA78 1C8D 6FDD  CB27 1CA9 5A63 56B0 CA08
uid                            Lars Thegler <lars@thegler.dk>
sub  1024g/E8C58EF3 2004-05-31
```

D.3.361 Jase Thew <jase@FreeBSD.org>

```
pub   3072R/3EEAF1EB 2012-05-30
      Key fingerprint = F5FB 959F CF1B 6550 054E  2819 A484 BCDB 3EEA F1EB
uid                  Jase Thew (FreeBSD) <jase@FreeBSD.org>
uid                  Jase Thew <freebsd@beardz.net>
```

D.3.362 David Thiel <lx@FreeBSD.org>

```
pub   1024D/A887A9B4 2006-11-30 [expires: 2011-11-29]
      Key fingerprint = F08F 6A12 738F C9DF 51AC  8C62 1E30 7CBE A887 A9B4
uid                  David Thiel <lx@FreeBSD.org>
sub   2048g/B9BD92C5 2006-11-30 [expires: 2011-11-29]
```

D.3.363 Fabien Thomas <fabient@FreeBSD.org>

```
pub   1024D/07745930 2009-03-16
      Key fingerprint = D8AC EFA2 2FBD 7788 9628  4E8D 3F35 3B88 0774 5930
uid                  Fabien Thomas <fabient@FreeBSD.org>
sub   2048g/BC173395 2009-03-16
```

D.3.364 Thierry Thomas <thierry@FreeBSD.org>

```
pub   1024D/C71405A2 1997-10-11
      Key fingerprint = 3BB8 F358 C2F1 776C 65C9  AE51 73DE 698C C714 05A2
uid                  Thierry Thomas <thierry@pompo.net>
uid                  Thierry Thomas <tthomas@mail.dotcom.fr>
uid                  Thierry Thomas (FreeBSD committer) <thierry@FreeBSD.org>
sub   1024R/C5529925 2003-11-26
sub   2048g/05CF3992 2008-02-05
```

D.3.365 Andrew Thompson <thompsa@FreeBSD.org>

```
pub   1024D/BC6B839B 2005-05-05
      Key fingerprint = DE74 3F49 B97C A170 C8F1  8423 CAB6 9D57 BC6B 839B
uid                  Andrew Thompson <thompsa@freebsd.org>
uid                  Andrew Thompson <andy@fud.org.nz>
sub   2048g/92E370FB 2005-05-05
```

D.3.366 Florent Thoumie <flz@FreeBSD.org>

```
pub   1024D/5147DCF4 2004-12-04
      Key fingerprint = D203 AF5F F31A 63E2 BFD5  742B 3311 246D 5147 DCF4
uid                  Florent Thoumie (FreeBSD committer address) <flz@FreeBSD.org>
uid                  Florent Thoumie (flz) <florent@thoumie.net>
uid                  Florent Thoumie (flz) <flz@xbsd.org>
uid                  [jpeg image of size 1796]
sub   2048g/15D930B9 2004-12-04
```

D.3.367 Jilles Tjoelker <jilles@FreeBSD.org>

```
pub   4096R/D5AE6220 2011-07-02
      Key fingerprint = 4AF5 F1CC BDD7 700B F005  79A4 A2C4 C4D4 D5AE 6220
uid                  Jilles Tjoelker <jilles@stack.nl>
uid                  Jilles Tjoelker <tjoelker@zonnet.nl>
uid                  Jilles Tjoelker (FreeBSD) <jilles@FreeBSD.org>
sub   4096R/14CB5775 2011-07-02
```

D.3.368 Ganbold Tsagaankhuu <ganbold@FreeBSD.org>

```
pub   1024D/78F6425E 2008-02-26 [expires: 2013-02-24]
      Key fingerprint = 9B8E DC41 D3F4 F7FC D8EA  417C D4F7 2AEF 78F6 425E
uid                  Ganbold <ganbold@freebsd.org>
sub   2048g/716FCBF9 2008-02-26 [expires: 2013-02-24]
```

D.3.369 Michael Tuexen <tuexen@FreeBSD.org>

```
pub   1024D/04EEDABE 2009-06-08
      Key fingerprint = 493A CCB8 60E6 5510 A01D  360E 8497 B854 04EE DABE
uid                  Michael Tuexen <tuexen@FreeBSD.org>
sub   2048g/F653AA03 2009-06-08
```

D.3.370 Andrew Turner <andrew@FreeBSD.org>

```
pub   2048R/31B31614 2010-07-01
      Key fingerprint = 08AC 2C57 F14F FDD1 2232  B5CD AA16 EFB8 31B3 1614
uid                  Andrew Turner <andrew@freebsd.org>
uid                  Andrew Turner <andrew@fubar.geek.nz>
sub   2048R/9ACBF138 2010-07-01
```

D.3.371 Hajimu UMEMOTO <ume@FreeBSD.org>

```
pub   1024D/BF9071FE 2005-03-17
      Key fingerprint = 1F00 0B9E 2164 70FC 6DC5  BF5F 04E9 F086 BF90 71FE
uid                  Hajimu UMEMOTO <ume@mahoroba.org>
uid                  Hajimu UMEMOTO <ume@FreeBSD.org>
uid                  Hajimu UMEMOTO <ume@jp.FreeBSD.org>
sub   2048g/748DB3B0 2005-03-17
```

D.3.372 Stephan Uphoff <ups@FreeBSD.org>

```
pub  2048R/D684B04A 2004-10-06 Stephan Uphoff <ups@freebsd.org>
     Key fingerprint = B5D2 04AE CA8F 7055 7474  3C85 F908 7F55 D684 B04A
uid                           Stephan Uphoff <ups@tree.com>
sub  2048R/A15F921B 2004-10-06
```

D.3.373 Bryan Venteicher <bryanv@FreeBSD.org>

```
pub   4096R/E97DB7DB 2012-11-05
      Key fingerprint = 0F8F 11EF F4D2 EDCA ECEA  CB16 744C BF25 E97D B7DB
uid                  Bryan Venteicher (DITC) <bryanv@daemonicthecloset.org>
uid                  Bryan Venteicher (FreeBSD) <bryanv@freebsd.org>
```