FreeBSD Handbook

Release 2023.09.08

FreeBSD 中文社区翻译

Sep 11, 2023

目录

译者说明

说明

本手册内容由FreeBSD 中文社区[1]翻译，官方仓库为：https://github.com/FreeBSD-Ask/handbook

为提升阅读体验，使用d2lbook[2]进行编译成HTML和PDF、EPUB文档。

当前同步中文仓库commit:2023.09.01[3]

d2lbook生成静态页面：FreeBSD_Handbook_d2l[4]

最新PDF[5]

最新EPUB[6]

历史编译参见Actions[7]

FreeBSD 中文社区翻译项目

获取 **PDF** 文档

点击 https://freebsd.gitbook.io/freebsd-handbook/，选择右上角的“导出为 PDF”，如不成功你可多试几次。

手册版本说明

当前文档版本同步至官方文档 2023-7-18 commit a9c93bee1de624ad6e55f2f081bce23da8df57a6[8]。如需更新请提交 issue 或 pull request。

[1] https://handbook.bsdcn.org

[2] https://book.d2l.ai/

[3] https://github.com/FreeBSD-Ask/Handbook/commit/56703f4f

[4] https://fiercex.github.io/FreeBSD_Handbook_d2l/

[5] https://fiercex.github.io/FreeBSD_Handbook_d2l/files/FreeBSD_Handbook.pdf

[6] https://fiercex.github.io/FreeBSD_Handbook_d2l/files/FreeBSD_Handbook.epub

[7] https://github.com/fierceX/FreeBSD_Handbook_d2l/actions

[8] https://github.com/freebsd/freebsd-doc/commit/a9c93bee1de624ad6e55f2f081bce23da8df57a6

译者相关

机器翻译相关

章节	Deepl 机器翻译	格式调整
前言	ykla	魔王酱
第 1 章：简介	ykla	魔王酱
第 2 章：安装 FreeBSD	ykla	ulianchn38、鶡野鴿
第 3 章：FreeBSD 基础【3.1-3.2】	ykla	ykla
第 3 章：FreeBSD 基础【3.3-EOL】	亲爱的翻译官	ykla、鶡野鴿
第 4 章：安装应用程序: 软件包和 Ports	亲爱的翻译官	ykla、skadomsky
第 5 章：X Window 系统【5.1-5.2】	ykla	ulianchn38、冰
第 5 章：X Window 系统【5.3-5.4.6】	冰	ykla、ulianchn38
第 5 章：X Window 系统【5.4.6-EOL】	Lin	ykla、ulianchn38、冰
第 6 章：FreeBSD 中的 Wayland	ykla	ykla
第 7 章：桌面环境	胞嘧啶	ykla
第 8 章：多媒体	无目先生	ykla
第 9 章：配置 FreeBSD 内核	Jasonlecson	ykla、冰
第 10 章：打印【9.1-9.4】	潇潇雨竹	ykla
第 10 章：打印【9.5-EOL】	ulianchn38	ykla
第 11 章：Linux 二进制兼容层【10.1】	Altair	ykla
第 11 章：Linux 二进制兼容层【10.2-EOL】	亲爱的翻译官	ykla
第 12 章：wine	Jasonlecson	ykla、冰
第 13 章：配置与优化	胞嘧啶	ykla、冰
第 14 章：FreeBSD 的引导过程	徐艺扬	ykla、冰、鶡野鴿
第 15 章：安全	陈诚	ykla、冰
第 16 章：Jail	陈诚	ykla、冰
第 17 章：强制访问控制	陈诚	ykla、冰
第 18 章：安全事件审计	冰	ykla、鶡野鴿
第 19 章：存储	Jasonlecson	ykla、冰
第 20 章：GEOM: 模块化磁盘转换框架	Jasonlecson	ykla、冰
第 21 章：Z 文件系统（ZFS）	徐艺扬、冰	ykla、ulianchn38
第 22 章：其他文件系统	Jasonlecson	ykla
第 23 章：虚拟化	鶡野鴿	ykla、冰
第 24 章：本地化——i18n/L10n 的使用和设置	郴	ykla、冰
第 25 章：FreeBSD 更新与升级	亲爱的翻译官	ykla
第 26 章：DTrace	鶡野鴿	ykla、冰
第 27 章：USB 设备模式/USB OTG	Jasonlecson	ykla
第 28 章：串行通信	胞嘧啶	ykla、冰
第 29 章：PPP	ykla	ykla

continues on next page

表 1 – continued from previous page

章节	Deepl 机器翻译	格式调整
第 30 章：电子邮件	ykla	ykla
第 31 章：网络服务器	陈诚	ykla、冰
第 32 章：防火墙	陈诚	ykla、冰
第 33 章：高级网络	陈诚	ykla、冰
附录 A：获取 FreeBSD	亲爱的翻译官	ykla
附录 B：书目	ykla	ykla
附录 C：网络资源	亲爱的翻译官	ykla
附录 D：OpenPGP 密钥	ykla	ykla
术语表	亲爱的翻译官	ykla

名单排序以提交的先后顺序为准。未标明部分由 **ykla** 完成，下同。

校对相关（标记用户即为完成，下同）

章节	第一轮校对	第二轮校对	第三轮校对
前言	ykla	罗胜(superluosheng)	
第 1 章：简介	ykla	罗胜(superluosheng)	
第 2 章：安装 FreeBSD	ykla/Shengyun	罗胜(superluosheng)	
第 3 章：FreeBSD 基础	ykla/Shengyun	罗胜(superluosheng)	
第 4 章：安装应用程序: 软件包和 Ports	ykla/Shengyun	罗胜(superluosheng)	
第 5 章：X Window 系统	ykla/Shengyun	罗胜(superluosheng)	
第 6 章：FreeBSD 中的 Wayland	ykla/Shengyun		
第 7 章：桌面环境	ykla/Shengyun	罗胜(superluosheng)	
第 8 章：多媒体	ykla	罗胜(superluosheng)	
第 9 章：配置 FreeBSD 内核	ykla	罗胜(superluosheng)	
第 10 章：打印	ykla		
第 11 章：Linux 二进制兼容层	ykla	罗胜(superluosheng)	
第 12 章：wine	ykla		
第 13 章：配置与优化	ykla	罗胜(superluosheng)	
第 14 章：FreeBSD 的引导过程	ykla		
第 15 章：安全	ykla	罗胜(superluosheng)	
第 16 章：Jail	ykla	罗胜(superluosheng)	
第 17 章：强制访问控制	ykla/KCommit	罗胜(superluosheng)	
第 18 章：安全事件审计	ykla		
第 19 章：存储	ykla		
第 20 章：GEOM: 模块化磁盘转换框架	ykla		
第 21 章：Z 文件系统 (ZFS)	ykla	罗胜(superluosheng)	
第 22 章：其他文件系统	ykla	罗胜(superluosheng)	

continues on next page

表 2 – continued from previous page

章节	第一轮校对	第二轮校对	第三轮校对
第 23 章：虚拟化	ykla	罗胜(superluosheng)	
第 24 章：本地化——i18n/L10n 的使用和设置	ykla		
第 25 章：FreeBSD 更新与升级	ykla	罗胜(superluosheng)	
第 26 章：DTrace	ykla	罗胜(superluosheng)	
第 27 章：USB 设备模式/USB OTG	ykla		
第 28 章：串行通信	ykla	罗胜(superluosheng)	
第 29 章：PPP	ykla	罗胜(superluosheng)	
第 30 章：电子邮件	ykla	罗胜(superluosheng)	
第 31 章：网络服务器	ykla	罗胜(superluosheng)	
第 32 章：防火墙	ykla	罗胜(superluosheng)	
第 33 章：高级网络	ykla	罗胜(superluosheng)	
附录 A：获取 FreeBSD	ykla	罗胜(superluosheng)	
附录 B：书目	ykla		
附录 C：网络资源	ykla		
附录 D：OpenPGP 密钥	ykla		
术语表	ykla	罗胜(superluosheng)	

章节之间的未标明部分也经过了校对。当前已校对章节仍然可能存在一些问题需要解决，见 issue[9]。

网站部署与维护

- Shengyun
- ykla

Q & A

- Q：对 FreeBSD Handbook 进行简体中文翻译的必要性?
  - A：

俗话说的好，要想富先修路。要想推广和宣传 FreeBSD，也必须先翻译 Handbook。

本手册的翻译是为了活跃 FreeBSD 中文社区、助力 FreeBSD 系统生态发展。我们争取做到人人都可以写教程、参与开源协作。没有人强迫您阅读本手册，所以也请您不要诋毁我们。FreeBSD 中国境内的社区这十几年一本教程都没有，这不是我们想看到的。请不做事的人不要去嘲讽正在做事的。感谢您的体谅!

有些人认为有英语不需要翻译，看不懂的人就不配学习 FreeBSD。此言差矣，Handbook 不专门针对开发者这一个群体，而是针对所有人（https://docs.freebsd.org/en/books/porters-handbook/porting-why/ 专为开发者编写），类似于 Linux 的 wiki，但是对比下手册更有体系结构。FreeBSD 也并非专门为了某些精英而创设，不懂

[9] https://github.com/FreeBSD-Ask/Handbook/issues

英语是一件很正常的事情，不同的语言有不同的世界观，只有经过翻译才能让更多普通人走进 FreeBSD，发展 FreeBSD，结缘 FreeBSD。你懂英语，别人不一定懂，不能用你的标准去衡量所有人。FreeBSD 社区以及基金会从未说过不懂英语就无法使用 FreeBSD 这种话。

还有些人认为与其翻译 Handbook 不如我们自己去写原创性的文章，其实翻译handbook，恰恰是为了更好地撰写原创性的文章，我们的另一个项目——《FreeBSD 从入门到跑路》[10]目标就是包括所有 Handbook 有与无的内容。我们认为，翻译文档和贡献代码是同样重要的事情；如果你有原创教程，那我们有教程征集计划——https://docs.qq.com/doc/DSUJsUFBHTnVWQmtS 最重要的是，我们的进程已经完成等待校对。现在还在纠结这个问题这无异于在中国哲学史课程快要上完的时候还在讨论中国到底有没有哲学。

- Q：本项目为什么不能向 FreeBSD 上游进行合并？
  - A：

很简单，我们无法向上游贡献。https://wiki.freebsd.org/Doc/Translation 中的指南无法具体落实，其翻译进度一直是（99%）,实际上是 0。我们多次与 FreeBSD 简体中文翻译负责人（该负责人从 18 年就进行 FreeBSD handbook 的翻译，但始终未能进行）进行联络，始终无法得到及时回应。但本着 FreeBSD 是一个开源的项目，人人都可以 fork 的态度。并不会影响我们的工作进度。也不会导致本项目产生任何问题。我们认为，踏出第一步永远比空谈任何东西都更为重要！

现阶段需要先完成校对并且我们现在需要人手来进行校对并向上游进行合并。要求能够对照英文找到中文的翻译。直接翻译（照抄）ykla 给定的 po 文件。另外需要一名或多名提交者进行配合。目前 FreeBSD 上游的 doc 文档工具链存在 bug，编译出来的中文字体会乱码（疑似没有加载中文字体，只有日文字体），也许要等其修复。

我也非常疑惑。

- Q：怎样维护？
  - A：

翻译完毕后约 1 个月整体维护一次。对比官方 Handbook 进行更新。

- Q：怎样联系你？
  - A：ykla AT bsdcn DOT org.

翻译指南

翻译人员应该加入 **QQ 群 512905950**。**注意，此群仅讨论翻译相关事宜，其他事项者将被移出**。

https://docs.qq.com/doc/DSUtxYmVwU29EdGVn

[10] https://github.com/FreeBSD-Ask/FreeBSD-Ask

关于

简介

来到这里你可能什么也得不到。也将不会失去任何东西。FreeBSD 中文用户社区！恪守古老的法则，追寻真正的自由。BSD 方为真正 UNIX 哲学继承者。加入我们，共同推进 FreeBSD 中国化与世界化。

资源	链接
Telegram 群	https://t.me/freebsdba
QQ 群	787969044
Handbook 最新翻译	https://handbook.bsdcn.org
FreeBSD 入门书籍	https://book.bsdcn.org
FreeBSD 开发者手册最新翻译	https://porters-handbook.bsdcn.org
微信公众号	freebsdzh
BiliBili【B站】	https://space.bilibili.com/2120246893

扫码关注微信公众号：

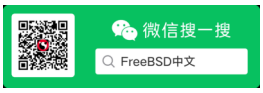


图1: 扫码关注微信公众号

FreeBSD 中文社区寄言：

每一个人的不经意付出和教程完善的分享，都是极具意义的，理论上会长久存在，你做的每一点一滴的努力与付出都会成为历史的印记和未来某些科技的驱动力的先驱。手册是死的，我们对手册的贡献是一直存活的，存活在那些使用 Handbook 的人的心中。

“**The best time to plant a tree is 20 years ago. The second-best time is now.**”（种一棵树最好的时间是十年前，其次是现在）——**Dambisa Moyo, dead aid**

我希望有更多的人参与进来，一起协作这个开源项目。英语水平和计算机水平固然重要，但是翻译凭借的不是傲慢与偏见，不是苦难哲学，也不是泥潭般的社区，更不是所谓的巨匠，而是我们的心，翻译是用心来完成的！用心，每个人都可以参与，无论是错别字的修正还是大章节的翻译，都是有意义的。

想参加的朋友可以看看 https://docs.qq.com/doc/DSUtxYmVwU29EdGVn

FreeBSD 中文社区的愿景

我们成立于 2018年3月17日，由贴吧——FreeBSD 吧发展到了 QQ 群（主群 787969044），Telegram 群，乃至于微信群。

我们的成员具有非常大的广泛性和普遍性，能够代表绝大多数 FreeBSD 用户的平均水平：可能根本没有听说过何为 FreeBSD，但这并不影响我们的交流与沟通。也许有人觉得这是浪费时间，但是没有新生力量的培养，何来 FreeBSD 的明天呢？谁不知道新人可能有很多坏习惯呢。

同鲁迅先生说的那样，但愿每个人都是一束光，照亮 FreeBSD 在中国大陆地区前进的光荣的荆棘路。也希望，你可以加入我们，共同组成漫天星光亦或者是星星之火。

无穷的远方，无数的人们，都和我有关。

我曾无数次眺望远山，想要找到一汪清泉，天总是不随人愿，还是没有找到。

我是谁，我们是谁？这个问题永远也不会有结果。

我们选择 FreeBSD，是因为想选择一个清晰、明了、可靠、稳固的一个操作系统在工作上给我们带来收益以及在生活中给我们带来乐趣。当然 FreeBSD 还存在很多问题，有待大家积极发现、探讨、完善，社会在进步，技术在进步，热情丝毫不减在持续，未来越来越美好。

黑名单与社区失信名单

见 http://chinafreebsd.org/。

FreeBSD 手册

Adaptec 是 Adaptec, Inc. 的注册商标。

Android 是 Google Inc. 的商标。

Heidelberg，Helvetica，Palatino 和 Times Roman 是 Heidelberger Druckmaschinen AG 在美国和其他国家的注册商标或商标。

Intuit 和 Quicken 是 Intuit Inc. 或其子公司在美国和其他国家的注册商标和/或注册服务商标。

LSI Logic，AcceleRAID，eXtremeRAID，MegaRAID 和 Mylex 是 LSI Logic Corp. 的商标或注册商标。

MATLAB 是 The MathWorks, Inc. 的注册商标。

SpeedTouch 是 Thomson 的商标。

VMware 是 VMware, Inc. 的商标。

Mathematica 是 Wolfram Research, Inc. 的注册商标。

Ogg Vorbis 和 Xiph.Org 是 Xiph.Org 的商标。

XFree86 是 The XFree86 Project, Inc. 的商标。

许多制造商和销售商用来区分其产品的名称被声明为商标。在本文档中出现这些名称时，如果 FreeBSD 项目意识到了商标声明，这些名称后面会跟随 ™ 或 ® 符号。

概述

欢迎来到 FreeBSD! 本手册涵盖了 *FreeBSD 13.2-RELEASE* 和 *FreeBSD 12.4-RELEASE* 的安装和日常使用。这本书是许多人不断工作的结果。可能某些部分已经过时。有兴趣帮助更新和扩展这篇文档的人应该向 FreeBSD 文档项目邮件列表[11] 发送电子邮件。

本书的最新版本可以从 FreeBSD 网站[12]上获得。旧版本可以从 https://docs.FreeBSD.org/doc/ 获得。这本书可以从 FreeBSD 的下载服务器[13]或众多的镜像站[14]中以多种格式或压缩方式进行下载。也可以在搜索页面[15]上对手册和其他文件进行搜索。

[11] https://lists.freebsd.org/subscription/freebsd-doc

[12] https://www.freebsd.org/

[13] https://download.freebsd.org/doc/

[14] https://docs.freebsd.org/en/books/handbook/mirrors#mirrors

[15] https://www.freebsd.org/search/

前言

致读者

FreeBSD 新手会发现，本书的第一部分引导用户完成了 FreeBSD 的安装过程，并渐进地介绍了 UNIX® 的概念和惯例。学习这部分只需要你有探索的欲望和接受新概念的能力。

在你完成了这一部分后，手册的第二部分，也就是更加繁杂的部分，为 FreeBSD 系统管理员提供了各种感兴趣的主题的全面参考。其中一些章节可能会建议你事先进行一些预习，这将在每章开头的概述中注明。

有关其他信息来源的列表，请阅读参考文献[16]。

[16] https://docs.freebsd.org/en/books/handbook/bibliography/index.html#bibliography

第四版

当前版本的手册代表了工作小组的不懈努力，该小组一直在审查和更新所有手册的内容。这些是自手册第四版以来的主要更新。

- 手册已经从 Docbook 转换为 Hugo 和 AsciiDoctor。
- 建立了 FreeBSD 文档门户网站。
- 加入了关于在 FreeBSD 下安装和配置 Wayland 的信息。
- 对书目进行了广泛的更新。

第三版

当前的在线手册凝结了数百名贡献者在过去十年的不懈努力。以下是自 2004 年出版的两卷第三版以来的一些重要变化：

- 加入了如何在 FreeBSD 上用 WINE[17] 运行 Windows® 应用程序的有关信息。
- DTrace[18] 增加了有关强大的性能分析工具的信息。
- 其他文件系统[19] 增加了 FreeBSD 中的非原生文件系统的有关信息，例如来自 Sun™ 的 ZFS。
- 安全事件审计[20]增加了关于 FreeBSD 中新的审计功能的信息，并解释了其用途。
- 虚拟化[21]增加了在虚拟化软件上安装 FreeBSD 的有关信息。
- 安装 FreeBSD[22] 增加了关于使用新的安装工具 bsdinstall 安装 FreeBSD 的章节。

[17] https://docs.freebsd.org/en/books/handbook/wine/index.html#wine

[18] https://docs.freebsd.org/en/books/handbook/dtrace/index.html#dtrace

[19] https://docs.freebsd.org/en/books/handbook/filesystems/index.html#filesystems

[20] https://docs.freebsd.org/en/books/handbook/audit/index.html#audit

[21] https://docs.freebsd.org/en/books/handbook/virtualization/index.html#virtualization

[22] https://docs.freebsd.org/en/books/handbook/bsdinstall/index.html#bsdinstall

第二版（2004）

第三版是 FreeBSD 文档项目成员两年多来工作的结晶。纸质出版物的内容越来越多，以至于有必要以两卷的形式出版。以下是新版中的主要变化：

- 配置与优化[23]增加了关于 ACPI 电源和资源管理、`cron` 系统工具和更多内核参数优化的新信息。
- 安全[24]增加了有关虚拟专用网络（VPN）、文件系统访问控制列表（ACL）和安全公告的新信息。
- 强制访问控制[25]是此版本的一个新章节。它解释了什么是 MAC 以及如何使用这一机制来保护 FreeBSD 系统。
- 存储[26]增加了关于 USB 存储设备、文件系统快照、文件系统配额、文件系统和网络文件系统，以及加密磁盘分区的新信息。
- 在 PPP[27] 中加入了一个故障排除的部分。
- 电子邮件[28]增加了如何使用其他邮件传输代理、SMTP 认证、UUCP、fetchmail、procmail 和其他高级主题的有关信息。
- 网络服务器[29]是此版的全新内容。该章包括有关设置 Apache HTTP 服务器、ftpd 和为 Microsoft® Windows® 客户设置 Samba 服务器的信息。高级网络中的一些章节被移到这里，以改善表述。
- 在高级网络[30]中增加了关于使用 FreeBSD 的 Bluetooth® 设备、建立无线网络和异步传输模式（ATM）网络的新信息。
- 增加了一份术语表，用以说明整本书中出现的术语。
- 对全书中的和表格进行了一些美化。

[23] https://docs.freebsd.org/en/books/handbook/config/index.html#config-tuning
[24] https://docs.freebsd.org/en/books/handbook/security/index.html#security
[25] https://docs.freebsd.org/en/books/handbook/mac/index.html#mac
[26] https://docs.freebsd.org/en/books/handbook/disks/index.html#disks
[27] https://docs.freebsd.org/en/books/handbook/ppp-and-slip/index.html#ppp-and-slip
[28] https://docs.freebsd.org/en/books/handbook/mail/index.html#mail
[29] https://docs.freebsd.org/en/books/handbook/network-servers/index.html#network-servers
[30] https://docs.freebsd.org/en/books/handbook/advanced-networking/index.html#advanced-networking

第一版（2001）

第二版是 FreeBSD 文档项目忠实的成员们两年多来工作的结晶。以下是本版中的主要变化：

- 增加了一个完整的索引。
- 所有的 ASCII 图表都被图片所取代。
- 每章都增加了一个标准的概述，以快速总结该章所包含的信息，以及希望读者了解的内容。
- 内容被有次序地重新组织为三个部分：“入门”，“系统管理”，以及“附录”。
- 对 FreeBSD 基础[31]进行了扩展，包含了关于进程、守护进程和信号的额外信息。
- 对安装应用程序：软件包和 Ports[32] 的内容进行了扩充，包含了关于二进制软件包管理器的额外信息。
- X Window 系统[33]已经被完全重写，重点是在 XFree86™ 4.X 上使用现代桌面技术，如 KDE 和 GNOME。
- FreeBSD 的引导过程[34]得到了扩展。
- 存储[35]部分由过去的“磁盘”和“备份”两章合并而成。我们认为将这些主题作为一个章节来介绍会更容易理解。还增加了一个关于 RAID（包括硬件和软件）的章节。
- 串行通信[36]已被完全重新组织，并针对 FreeBSD 4.X/5.X 进行了更新。
- 对 PPP[37] 进行了大幅更新。
- 高级网络[38]中加入了许多新的章节。
- 电子邮件[39]增加了关于配置 sendmail 的更多信息。

[31] https://docs.freebsd.org/en/books/handbook/basics/index.html#basics
[32] https://docs.freebsd.org/en/books/handbook/ports/index.html#ports
[33] https://docs.freebsd.org/en/books/handbook/x11/index.html#x11
[34] https://docs.freebsd.org/en/books/handbook/boot/index.html#boot
[35] https://docs.freebsd.org/en/books/handbook/disks/index.html#disks
[36] https://docs.freebsd.org/en/books/handbook/serialcomms/index.html#serialcomms
[37] https://docs.freebsd.org/en/books/handbook/ppp-and-slip/index.html#ppp-and-slip
[38] https://docs.freebsd.org/en/books/handbook/advanced-networking/index.html#advanced-networking
[39] https://docs.freebsd.org/en/books/handbook/mail/index.html#mail

- Linux® 二进制兼容层[40]已经扩展到包括关于安装 Oracle® 和 SAP® R/3® 的信息。
- 本书第二版包括了以下新主题：
  - 配置与优化[41]。
  - 多媒体[42]。

[40] https://docs.freebsd.org/en/books/handbook/linuxemu/index.html#linuxemu

[41] https://docs.freebsd.org/en/books/handbook/config/index.html#config-tuning

[42] https://docs.freebsd.org/en/books/handbook/multimedia/index.html#multimedia

本书的组织结构

本书在逻辑上被分为五个不同的部分：第一部分，入门，包括 FreeBSD 的安装和基本使用。希望读者能按顺序阅读这些章节，但可以跳过你熟悉的章节。第二部分，常见任务，涵盖了 FreeBSD 的一些常用功能。这一节以及后面的所有章节都可以不按顺序阅读。每一章的开头都有一个简洁的概述，说明了这一章的内容和读者应该已知的内容。这样做的目的是让普通读者能够跳过这些已知章节，找到感兴趣的章节。第三部分，系统管理，包括管理主题。第四部分，网络通信，包括网络和服务器主题。第五部分包含了参考信息的附录。

简介[43]

向新用户介绍 FreeBSD。并简要说明了 FreeBSD 项目的历史，目标和开发模式。

安装 FreeBSD[44]

引导用户使用 bsdinstall 完成 FreeBSD 9.x 及以后版本的完整安装过程。

FreeBSD 基础[45]

介绍了 FreeBSD 操作系统的基本命令和功能。如果你熟悉 Linux® 或其他 UNIX®，那么你可以跳过这一章。

安装应用程序: 软件包和 Ports[46]

包含了如何使用 FreeBSD 创新的“ports”和标准二进制软件包来安装第三方软件。

X Window[47]

介绍了常见的 X Window 系统，特别是在 FreeBSD 上使用的 X11。还介绍了常见的桌面环境，如 KDE 和 GNOME。

Wayland[48]

[43] https://docs.freebsd.org/en/books/handbook/introduction/index.html#introduction

[44] https://docs.freebsd.org/en/books/handbook/bsdinstall/index.html#bsdinstall

[45] https://docs.freebsd.org/en/books/handbook/bsdinstall/index.html#bsdinstall

[46] https://docs.freebsd.org/en/books/handbook/ports/index.html#ports

[47] https://docs.freebsd.org/en/books/handbook/x11/index.html#x11

[48] https://docs.freebsd.org/en/books/handbook/book/#wayland

介绍了 Wayland 显示服务器的一般情况，特别是在 FreeBSD 上使用的 Wayland。还介绍了常见的合成器，如 Wayfire、Hikari 和 Sway。

桌面应用程序[49]

列出了一些常见的桌面应用程序，如网络浏览器和生产力工具，并介绍了如何在 FreeBSD 上安装它们。

多媒体[50]

演示了如何为你的系统设置声音和视频播放支持。还列出了一些音频和视频应用。

配置 FreeBSD 内核[51]

介绍了为什么你可能需要配置一个新的内核，并提供了配置、编译和安装定制内核的详细说明。

打印[52]

介绍了在 FreeBSD 上管理打印机的情况，包括关于横幅页面、打印审计和初始设置的信息。

Linux® 二进制兼容层[53]

WINE[54]

介绍了 WINE 并提供了详细的安装说明。还说明了 WINE 的操作方式，如何安装 GUI 助手，如何在 FreeBSD 上运行 Windows® 应用程序，并提供了其他提示和解决方案。

介绍了 FreeBSD 的 Linux® 兼容层。还提供了许多流行的 Linux® 应用程序（如 Oracle® 和 Mathematica®）的详细安装说明。

配置与优化[55]

介绍了系统管理员可以用来优化 FreeBSD 系统的参数，以获得最佳性能。还介绍了 FreeBSD 中使用的各种配置文件以及在何处可以找到它们。

FreeBSD 的引导过程[56]

介绍了 FreeBSD 的引导过程，并解释了如何通过配置选项来控制这个过程。

安全[57]

列举了许多不同的工具，包括 Kerberos、IPsec 和 OpenSSH，以帮助确保你的 FreeBSD 系统安全。

Jail[58]

介绍了 jail 框架，以及 jail 相对于 FreeBSD 传统的 chroot 支持所做的改进。

[49] https://docs.freebsd.org/en/books/handbook/desktop/index.html#desktop
[50] https://docs.freebsd.org/en/books/handbook/multimedia/index.html#multimedia
[51] https://docs.freebsd.org/en/books/handbook/kernelconfig/index.html#kernelconfig
[52] https://docs.freebsd.org/en/books/handbook/printing/index.html#printing
[53] https://docs.freebsd.org/en/books/handbook/linuxemu/index.html#linuxemu
[54] https://docs.freebsd.org/en/books/handbook/book/#wine
[55] https://docs.freebsd.org/en/books/handbook/config/index.html#config-tuning
[56] https://docs.freebsd.org/en/books/handbook/boot/index.html#boot
[57] https://docs.freebsd.org/en/books/handbook/security/index.html#security
[58] https://docs.freebsd.org/en/books/handbook/jails/index.html#jails

强制访问控制[59]

解释了什么是强制访问控制（MAC），以及如何利用这一机制来保护 FreeBSD 系统。

安全事件审计[60]

介绍了什么是 FreeBSD 事件审计，如何安装，配置，以及如何审计和监控审计跟踪。

存储[61]

介绍了如何用 FreeBSD 管理存储设备和文件系统。这包括物理磁盘、RAID 阵列、光学介质和磁带、内存支持的磁盘以及网络文件系统。

GEOM：模块化磁盘转换框架[62]

介绍了什么是 FreeBSD 中的 GEOM 框架，以及如何配置各种支持的 RAID 级别。

OpenZFS 存储平台[63]

介绍了 OpenZFS 存储平台，并提供了快速入门指南和有关在 FreeBSD 下运行 OpenZFS 的高级主题的信息。

其他文件系统[64]

检查对 FreeBSD 下的非原生文件系统的支持，如 ext2、ext3 和 ext4。

虚拟化[65]

介绍了虚拟化系统提供的功能，以及如何在 FreeBSD 中使用它们。

本地化——i18n/L10n 的使用和设置[66]

介绍了如何用英语以外的语言使用 FreeBSD。涵盖了系统和应用层面的本地化。

更新与升级 FreeBSD[67]

阐述了 FreeBSD-STABLE、FreeBSD-CURRENT 和 FreeBSD-RELEASE 之间的区别。说明了哪些用户会从跟踪开发系统中受益，并概述了这个过程。说明了用户可以采取的方法来更新他们的系统到最新的安全版本。

DTrace[68]

介绍了如何在 FreeBSD 中配置和使用 Sun™ 的 DTrace 工具。动态跟踪可以通过进行实时的系统分析来帮助定位性能问题。

[59] https://docs.freebsd.org/en/books/handbook/mac/index.html#mac
[60] https://docs.freebsd.org/en/books/handbook/audit/index.html#audit
[61] https://docs.freebsd.org/en/books/handbook/disks/index.html#disks
[62] https://docs.freebsd.org/en/books/handbook/geom/index.html#geom
[63] https://docs.freebsd.org/en/books/handbook/book/#zfs
[64] https://docs.freebsd.org/en/books/handbook/filesystems/index.html#filesystems
[65] https://docs.freebsd.org/en/books/handbook/virtualization/index.html#virtualization
[66] https://docs.freebsd.org/en/books/handbook/l10n/index.html#l10n
[67] https://docs.freebsd.org/en/books/handbook/cutting-edge/index.html#updating-upgrading
[68] https://docs.freebsd.org/en/books/handbook/dtrace/index.html#dtrace

USB Device Mode / USB OTG[69]

解释 FreeBSD 上 USB Device Mode 和 USB On The Go (USB OTG) 的使用。

PPP[70]

介绍了如何在 FreeBSD 使用 PPP 来连接远程系统。

电子邮件[71]

解释了电子邮件服务器的不同组成部分，并深入探讨了最流行的邮件服务器软件的简单配置主题：sendmail。

网络服务器[72]

提供了详细的说明和配置文件的例子，以便将你的 FreeBSD 机器设置为网络文件系统服务器、域名服务器、网络信息系统服务器或时间同步服务器。

防火墙[73]

解释了基于软件的防火墙背后的理念，并提供了关于配置不同的 FreeBSD 防火墙的详细说明。

高级网络[74]

介绍了许多网络主题，包括与局域网中的其他计算机共享互联网连接、高级路由主题、无线网络、Bluetooth®、ATM、IPv6，以及更多。

获取 FreeBSD[75]

列举了获得 FreeBSD CDROM 或 DVD 的不同方式，以及在互联网上可让你下载 FreeBSD 的网站。

书目[76]

本书涉及到许多不同的主题，可能会让你渴望得到更详细的解释。书目中列出了许多在文中被引用的优秀书籍。

网络资源[77]

介绍了许多可供 FreeBSD 用户发布问题和进行 FreeBSD 技术交流的论坛。

OpenPGP 密钥[78]

列出了一些 FreeBSD 开发者的 PGP 公钥。

[69] https://docs.freebsd.org/en/books/handbook/book/#usb-device-mode

[70] https://docs.freebsd.org/en/books/handbook/ppp-and-slip/index.html#ppp-and-slip

[71] https://docs.freebsd.org/en/books/handbook/mail/index.html#mail

[72] https://docs.freebsd.org/en/books/handbook/network-servers/index.html#network-servers

[73] https://docs.freebsd.org/en/books/handbook/firewalls/index.html#firewalls

[74] https://docs.freebsd.org/en/books/handbook/advanced-networking/index.html#advanced-networking

[75] https://docs.freebsd.org/en/books/handbook/mirrors/index.html#mirrors

[76] https://docs.freebsd.org/en/books/handbook/bibliography/index.html#bibliography

[77] https://docs.freebsd.org/en/books/handbook/eresources/index.html#eresources

[78] https://docs.freebsd.org/en/books/handbook/pgpkeys/index.html#pgpkeys

本书中使用的一些约定

为了提供一致的、易于阅读的文本，全书遵循了几个惯例：

排版规则

斜体

首次使用文件名、网址、强调文字和技术术语时都使用 *斜体*。

等宽

错误信息、命令、环境变量、ports 名称、主机名、用户名、组名、设备名称、变量和代码片段使用等宽字体。

粗体

粗体字用于应用程序、命令和按键。

用户输入

按键用**粗体**显示，以便从其他文本中脱颖而出。要同时输入的组合键在键之间用“+”表示，例如：

`Ctrl+Alt+Del`

意味着用户应该同时输入`Ctrl`、`Alt`和`Del`键。

要依次输入的键会用逗号隔开，例如：`Ctrl+X`，`Ctrl+S`：

`Ctrl+X, Ctrl+S`

意味着用户应该同时输入`Ctrl`和`X`键，然后再同时输入`Ctrl`和`S`键。

示例

以 **C:>** 开头的例子表示 MS-DOS® 命令。除非另有说明，这些命令可以在现代 Microsoft® Windows® 环境中的“命令提示符”窗口中执行。

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

以 # 开头的例子表示在 FreeBSD 中必须以超级用户身份调用的命令。你可以以 root 身份登录来输入命令，或者以你的正常账户登录并使用 su(1)[79] 来获得超级用户的权限。

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

以% 开头的例子表示应该从普通用户账户调用的命令。除非另有说明，用于设置环境变量和其他 shell 命令使用 C shell 语法。

```
% top
```

[79] https://www.freebsd.org/cgi/man.cgi?query=su&sektion=1&format=html

致谢

你手中的这本书凝结了全世界数百人的努力。无论他们带来了错别字的修正，还是提交了完整的章节，所有的贡献都是有用的。

有几家公司通过向作者支付全职工作费用、支付出版费用等方式支持本文档的开发。特别是 BSDi（后来被风河系统公司[80]收购）支付给 FreeBSD 文档项目的成员，让他们在 2000 年 3 月第一版书籍出版之前，全职从事改进本书的工作（ISBN 1-57176-241-8）。之后风河系统公司又为几位作者支付了费用，对打印输出的基础设施进行了一些改进，并为文本增加了一些章节。这项工作最终达成了 2001 年 11 月第二版书籍的出版（ISBN 1-57176-303-1）。在 2003-2004 年，Freebsd Mall, Inc[81] 为几个贡献者支付了报酬以改进手册，为第三版书籍做准备。第三版书籍被分成了两卷。这两卷都已出版，分别是 The FreeBSD Handbook 3rd Edition Volume 1: User Guide (ISBN 1-57176-327-9) 和 The FreeBSD Handbook 3rd Edition Volume 2: Administrators Guide (ISBN 1-57176-328-7)。

[80] http://www.windriver.com/

[81] http://www.freebsdmall.com/

第一部分：快速开始

本手册的这部分适用于对 FreeBSD 新手和管理员。这些章节：

- 介绍 FreeBSD。
- 引导读者完成安装过程。
- 教授 UNIX® 的基础知识。
- 展示如何安装大量适用于 FreeBSD 的第三方应用程序。
- 介绍 X，即 UNIX® 窗口系统，并详细说明如何配置能让用户更加高效的桌面环境。
- 介绍 Wayland，一个新的 UNIX® 显示服务器。

文本中的正向引用数量被尽量减少，以便这一部分可以从头到尾地阅读，最大程度地减少翻页。

第一章简介

1.1.概述

感谢您对 FreeBSD 的兴趣！以下章节涉及了 FreeBSD 项目的各个方面，如其历史、目标、开发模型等等。

阅读完本章后，您将会了解：

- 其他计算机操作系统的关系如何。
- FreeBSD 项目的历史。
- FreeBSD 项目的目标。
- FreeBSD 开源开发模型的基础知识。
- 当然，还有：“FreeBSD”这个名称的由来。

1.2.欢迎来到 FreeBSD！

1.2. 欢迎来到 FreeBSD！

FreeBSD 是一个开源的、符合标准的类 Unix 操作系统，适用于 x86（32 位和 64 位）、ARM®、AArch64、RISC-V®、MIPS®、POWER®、PowerPC® 和 Sun UltraSPARC® 计算机。它提供了现在被视为理所当然的所有功能，如抢占式多任务处理、内存保护、虚拟内存、多用户功能、SMP 支持，以及各种适用于不同语言和框架的开源开发工具，以及围绕 X Window System、KDE 或 GNOME 的桌面功能。它的独特优势包括：

- 自由的开源许可证，赋予你自由修改和扩展其源代码，并将其并入开源项目和封闭产品中，而不会施加与强制复制许可证类似的限制，同时避免潜在的许可证不兼容问题。
- 强大的 *TCP/IP* 网络——FreeBSD 实现了行业标准协议，具有日益增长的性能和可伸缩性。这使得它在服务器、路由/防火墙角色中都能很好地匹配——实际上许多公司和供应商正是出于这个目的使用它。

- 完全集成的 *OpenZFS* 支持，包括基于 ZFS 的根文件系统、ZFS 启动环境、故障管理、管理员委派、jail 支持、FreeBSD 特定文档以及系统安装程序支持。
- 广泛的安全功能，从强制访问控制框架到 Capsicum 能力和沙箱机制。
- 为所有支持的架构提供超过 *3* 万个预构建软件包，以及使你轻松构建自己的定制软件包的 Ports Collection。
- 文档——除了手册和不同作者编写的涵盖从系统管理到内核内部的主题的书籍外，还有 man(1)[82] 页面，不仅适用于用户空间的守护进程、实用程序和配置文件，还适用于内核驱动程序 API（第 9 节）和个别驱动程序（第 4 节）。
- 简单一致的存储库结构和构建系统——FreeBSD 使用单个存储库来存放所有组件，包括内核和用户空间。这加上一个统一且易于定制的构建系统以及周密的开发流程，使得将 FreeBSD 集成到自己的产品构建基础设施中变得轻而易举。
- 忠于 *Unix* 哲学，倾向于组合性而不是硬编码行为的单一“一体化”守护进程。
- 与 *Linux* 二进制兼容，使得可以无需虚拟化即可运行许多 Linux 二进制文件。

FreeBSD 基于加利福尼亚大学伯克利分校（University of California at Berkeley）的计算机系统研究组（Computer Systems Research Group，CSRG）的 4.4BSD-Lite 发行版，并延续了 BSD 系统开发的卓越传统。除了计算机系统研究组提供的优秀工作外，FreeBSD 项目还投入了大量人时，以扩展功能并对系统进行微调，以在实际负载情况下实现最大性能和可靠性。FreeBSD 在性能和可靠性方面与其他开源和商业产品相当，同时还提供了其他地方无法获得的前沿功能。

1.2.1. FreeBSD 可以做什么？

FreeBSD 可以用于的应用领域实际上仅受限于你自己的想象力。从软件开发到工厂自动化，从库存控制到远程卫星天线方位校正；如果商业 UNIX® 产品可以完成，那么很可能你也可以使用 FreeBSD 完成！FreeBSD 还受益于全球研究中心和大学开发的成千上万个高质量应用程序，这些应用程序通常可以以极低或零成本获得。

因为 FreeBSD 自身的源代码是自由可用的，所以系统也可以根据特定应用或项目进行几乎前所未有的程度的定制，这在大多数主要商业供应商的操作系统中通常是不可能的。以下只是一些当前人们正在使用 FreeBSD 的应用示例：

- 互联网服务： FreeBSD 内置的强大 TCP/IP 网络使其成为各种互联网服务的理想平台，例如：
  - Web 服务器
  - IPv4 和 IPv6 路由
  - 防火墙和 NAT（“IP 伪装”）网关
  - FTP 服务器
  - 邮件服务器

[82] https://man.freebsd.org/cgi/man.cgi?query=man&sektion=1&format=html

  - 以及更多…

- 教育：你是计算机科学或相关工程领域的学生吗？了解操作系统、计算机体系结构和网络的最佳方法莫过于通过 FreeBSD 提供的实际操作和深入了解。还有许多免费提供的 CAD、数学和图形设计软件包，使那些主要将计算机用于完成其他工作的人受益匪浅！
- 研究：由于整个系统的源代码都可用，FreeBSD 是研究操作系统以及其他计算机科学领域的优秀平台。FreeBSD 自由可用的性质还使得远程团队能够在不必担心特殊许可协议或开放论坛中讨论内容受限的情况下合作开发或分享创意。
- 网络：需要新的路由器？需要一个名称服务器（DNS）？需要一个防火墙来保护你的内部网络？FreeBSD 可以将闲置在角落的未使用 PC 转变为具有复杂数据包过滤功能的高级路由器。
- 嵌入式系统：FreeBSD 是构建嵌入式系统的绝佳平台。通过对 ARM®、MIPS® 和 PowerPC® 平台的支持，加上强大的网络堆栈、尖端功能以及宽松的 BSD 许可证[83]，FreeBSD 是构建嵌入式路由器、防火墙和其他设备的优秀基础。
- 桌面：使用免费提供的 X11 服务器和 Wayland 显示服务器，FreeBSD 是低成本桌面解决方案的绝佳选择。FreeBSD 提供许多开源桌面环境的选择，包括标准的 GNOME 和 KDE 图形用户界面。FreeBSD 甚至可以从中央服务器“无盘”启动，使个别工作站的成本更低，管理更加便捷。
- 软件开发：基本的 FreeBSD 系统配备了一套完整的开发工具，包括完整的 C/C++ 编译器和调试器套件。通过 ports 和 packages，还提供了许多其他语言的支持。

FreeBSD 可以免费下载，也可以通过 CD-ROM 或 DVD 获得。有关获取 FreeBSD 的更多信息，请参阅获取 FreeBSD[84]。

1.2.2. 谁在使用 FreeBSD？

FreeBSD 以其 Web 服务能力而闻名。一份基于 FreeBSD 构建其产品和服务的公司的推荐列表[85]可以在 FreeBSD 基金会的网站上找到。维基百科还维护了一个基于 FreeBSD 的产品列表[86]。

1.3.关于 FreeBSD 项目

1.3. 关于 FreeBSD 项目

以下部分提供了有关该项目的一些背景信息，包括简要历史、项目目标以及项目的开发模型[87]。

[83] https://docs.freebsd.org/en/books/faq/#bsd-license-restrictions

[84] https://docs.freebsd.org/en/books/handbook/mirrors/#mirrors

[85] https://freebsdfoundation.org/about-us/testimonials/

[86] https://en.wikipedia.org/wiki/List_of_products_based_on_FreeBSD

[87] https://docs.freebsd.org/en/books/dev-model/

1.3.1. FreeBSD 简要历史

FreeBSD 项目始于 1993 年初，部分是由非官方的 386BSDPatchkit 的最后三位协调员：Nate Williams、Rod Grimes 和 Jordan Hubbard 的构想。

最初的目标是生成 386BSD 的中间快照，以解决一些补丁机制无法解决的问题。项目的早期工作名称是 386BSD 0.5 或 386BSD Interim，以引用这一事实。

386BSD 是 Bill Jolitz 开发的操作系统，到那个时候已经受到了将近一年的严重忽视。随着每一天过去，补丁机制变得让人越来越不舒服，他们决定通过提供这个临时的“清理”快照来协助 Bill。然而，在没有明确表明将采取什么行动的情况下，Bill Jolitz 突然决定退出该项目，计划被粗鲁地打断。

即使在没有 Bill 的支持的情况下，三人认为目标仍然值得追求，因此他们采用了 David Greenman 提出的名称“FreeBSD”。在咨询了当前系统的用户后，制定了初始目标，在清楚地认识到该项目甚至有可能成为现实后，Jordan 联系了 Walnut Creek CDROM，以改善 FreeBSD 的发行渠道，以满足那些没有便捷访问互联网的人的需求。Walnut Creek CDROM 不仅支持在 CD 上分发 FreeBSD，还提供了一台用于开发工作和快速互联网连接的机器。没有 Walnut Creek CDROM 对当时一个完全未知项目的几乎空前的信心，很难想象 FreeBSD 会取得如今如此迅猛的发展。

第一个 CD-ROM（以及全网）发行版本是 FreeBSD 1.0，于 1993 年 12 月发布。它基于加州大学伯克利分校的 4.3BSD-Lite（“Net/2”）磁带，其中许多组件还由 386BSD 和自由软件基金会提供。对于首次发布来说，它是一个相当不错的成功，并在 1994 年 5 月发布了极其成功的 FreeBSD 1.1。

在这个时候，一些意外的风暴云在地平线上形成，当时 Novell 和加州大学伯克利分校解决了关于 Berkeley Net/2 磁带的法律地位的长期诉讼。解决方案的一个条件是加州大学伯克利分校承认 Net/2 的三个文件是“有负担的”代码，必须予以删除，因为它们是 Novell 的财产，而 Novell 则曾经从 AT&T 那里收购过这些文件。作为交换，加州大学伯克利分校获得了 Novell 的“祝福”，即 4.4BSD-Lite 发布时，它将被宣布为不受限制的，所有现有的 Net/2 用户都将被强烈鼓励切换到 4.4BSD-Lite。这也包括了 FreeBSD，项目被赋予了截至 1994 年 7 月底停止运送其自己基于 Net/2 的产品的权利。根据协议的条款，在截止日期之前，项目被允许发布最后一个版本，即 FreeBSD 1.1.5.1。

随后，FreeBSD 开始从全新且相当不完整的 4.4BSD-Lite 位重新发明自己。尽管只有与 System V 共享内存和信号量相关的三个文件被删除，但对 BSD 分发进行了许多其他更改和错误修复，因此将所有 FreeBSD 开发合并到 4.4BSD-Lite 是一项巨大的任务。项目花费了将近到 1994 年 11 月才完成这个过程，并在 1994 年 12 月发布了 FreeBSD 2.0。尽管它在很多方面仍然不太完善，但这次发布是一个重要的成功，并在 1995 年 6 月发布了更稳定且易于安装的 FreeBSD 2.0.5 版本。

从那时起，FreeBSD 每次都进行了一系列的发布，不断改进之前版本的稳定性、速度和功能集。

目前，长期的开发项目仍在 14.0-CURRENT（main）分支中进行，14.0 的快照版本将根据工作进展的情况持续提供。

1.3.2. FreeBSD 项目目标

FreeBSD 项目的目标是提供可以用于任何目的且无附加条件的软件。我们中的许多人在代码（和项目）中投入了相当大的资源，偶尔会期望一些财务补偿，但我们绝对不会坚持这一点。我们认为我们的首要任务是向任何人提供代码，无论出于何种目的，以便代码得到尽可能广泛的使用并提供尽可能广泛的益处。我相信这也是自由软件的最基本目标之一，我们全力支持这一点。

在我们的源代码中，属于 GNU 通用公共许可证（GPL）或库通用公共许可证（LGPL）的代码可能会附加稍多一些的条件，尽管只是强制要求开放源代码而不是别的。由于在商业使用 GPL 软件中可能出现的额外复杂情形，因此在合适的情况下，我们更倾向于用更宽松的 BSD 许可证发布软件。

1.3.3. FreeBSD 开发模式

FreeBSD 的开发过程非常开放、灵活[88]，它实际上由世界各地成千上万的人贡献代码构建而成的，这可以从我们的贡献者列表[89]中看出。FreeBSD 的开发基础设施允许这些成千上万的贡献者通过互联网进行协作。我们始终在寻找新的志愿者，有兴趣更紧密地参与的人应该查阅有关《为 FreeBSD 贡献[90]》的文章。

无论是独立工作还是紧密合作，了解有关 FreeBSD 项目及其开发过程的有用信息：

Git 存储库

多年来，FreeBSD 的中央源代码由 CVS[91]（Concurrent Versions System，一种免费的源代码控制工具）维护。在 2008 年 6 月，项目切换为使用 SVN[92]（Subversion）。由于 CVS 引入的技术限制在源代码树的迅速扩展和已存储的历史量方面变得明显，因此认为必须进行切换。文档项目和 Ports Collection 仓库也分别在 2012 年 5 月和 2012 年 7 月从 CVS 切换到 SVN。然后，在 2020 年 12 月，项目将源代码和文档存储库迁移到 Git[93]，而 Ports 则于 2021 年 4 月追随迁移[94]。有关获取 FreeBSD src/ 存储库的更多信息，请参阅《获取源代码[95]》部分，有关获取 FreeBSD Ports Collection 的详细信息，请参阅《使用 Ports[96]》。

提交者列表

提交者是具有推送访问权限的人，有权对 FreeBSD 源代码进行修改（“提交者”这个术语来自 `commit`，这是用于将新更改引入存储库的源代码控制命令）。任何人都可以向 Bug 数据库[97]提交错误报告。在提交错误报告之前，可以使用 FreeBSD 邮件列表、IRC 渠道或论坛来帮助验证问题是否确实是错误。

FreeBSD 核心团队

如果 FreeBSD 项目是一家公司，那么 *FreeBSD 核心团队*就相当于董事会。核心团队的主要任务是确保整个项目处于良好状态并朝着正确的方向前进。邀请专注和负责任的开发者加入我们的提交者组是核心团队的

[88] https://docs.freebsd.org/en/books/dev-model/

[89] https://docs.freebsd.org/en/articles/contributors/

[90] https://docs.freebsd.org/en/articles/contributing/

[91] https://www.nongnu.org/cvs/

[92] https://subversion.apache.org/

[93] https://www.freebsd.org/status/report-2020-10-2020-12.html#Git-Migration-Working-Group

[94] https://www.freebsd.org/status/report-2021-04-2021-06/#_git_migration_working_group

[95] https://docs.freebsd.org/en/books/handbook/cutting-edge/#synching

[96] https://docs.freebsd.org/en/books/handbook/ports/#ports-using

[97] https://bugs.freebsd.org/submit/

职能之一，招募新的核心团队成员则是在其他人离开时的职责之一。当前的核心团队是从提交者候选人中选举出来的，在 2022 年 5 月进行了选举。选举每 2 年举行一次。

注意

与大多数开发人员一样，当涉及到 FreeBSD 开发时，大多数核心团队成员也是志愿者，并且并不从项目中获得财务收益，因此“承诺”也不应被误解为“有保障的支持”。上述的“董事会”类比并不是很准确，可能更适合说这些人是在不明智的判断下为了 FreeBSD 而放弃了他们的生活！

FreeBSD 基金会

FreeBSD 基金会[98]是一个 501(c)(3)、总部位于美国的非营利组织，致力于支持和推广全球范围内的 FreeBSD 项目和社区。基金会通过项目拨款资助软件开发，并让员工立即回应紧急问题并实施新功能和功能。基金会购买硬件以改进和维护 FreeBSD 基础设施，并资助员工以提高测试覆盖率、持续集成和自动化。基金会通过在世界各地的技术会议和活动中宣传 FreeBSD 来推动 FreeBSD。基金会还提供研讨会、教育材料和演示，以招募更多的 FreeBSD 用户和贡献者。基金会还代表 FreeBSD 项目执行合同、许可协议和其他需要承认的法律事物。

外部贡献者

最后，但绝不是最不重要的，最大的开发者群体是用户自己，他们几乎在不断地向我们提供反馈和错误修复。保持与 FreeBSD 基础系统开发的联系的主要方法是订阅 FreeBSD 技术讨论邮件列表[99]，其中讨论此类事项。对于移植第三方应用程序，可以使用 FreeBSD ports 邮件列表[100]。关于文档——FreeBSD 文档项目邮件列表[101]。有关各种 FreeBSD 邮件列表的更多信息，请参阅互联网上的资源[102]。

FreeBSD 贡献者名单[103]很长且在不断增长，那么为什么不通过向 FreeBSD 贡献一些东西[104]来加入呢？提供代码并不是唯一的方式！

总之，我们的开发模式组织成为一组松散的同心圆。中心化模式旨在方便 FreeBSD 用户跟踪一个中央代码库，而不是阻止潜在的贡献者加入！我们的愿望是提供一个稳定的操作系统，带有一套大量协调的应用程序[105]，用户可以轻松安装和使用——这种模式在实现这一目标方面非常成功。

对于那些希望成为 FreeBSD 开发人员的人，我们只要求有些与当前人员对其持续成功一样的奉献精神！

[98] https://freebsdfoundation.org/

[99] https://lists.freebsd.org/subscription/freebsd-hackers

[100] https://lists.freebsd.org/subscription/freebsd-ports

[101] https://lists.freebsd.org/subscription/freebsd-doc

[102] https://docs.freebsd.org/en/books/handbook/eresources/#eresources

[103] https://docs.freebsd.org/en/articles/contributors/

[104] https://docs.freebsd.org/en/articles/contributing/

[105] https://docs.freebsd.org/en/books/handbook/ports/#ports

1.3.4. 第三方程序

除了基础发行版外，FreeBSD 还提供了一个移植的软件集，其中包含了成千上万个常见的常用程序。这些移植的程序范围从 HTTP 服务器到游戏、编程语言、编辑器，几乎覆盖了各种各样的应用。大约有 36000 个 Port；整个 Ports 大约需要 3 GB 的空间。要编译一个 Port，您只需切换到要安装的程序的目录，输入 `make install` 命令，然后让系统执行其余的操作。几乎每个 Port 都提供了预编译的“软件包”，那些不希望从源代码编译自己的 Port 的人可以使用简单的命令（`pkg install`）来安装。有关软件包和 Port 的更多信息，请参阅《安装应用程序：软件包和 Port[106]》。

1.3.5. 其他文档

所有受支持的 FreeBSD 版本在安装程序中提供了在初始系统设置期间安装其他文档的选项，位于 **/usr/local/share/doc/freebsd** 目录下。也可以稍后使用软件包来安装文档：

```
# pkg install en-freebsd-doc
```

对于本地化版本，请将“en”替换为所选语言的语言前缀。请注意，某些本地化版本可能已过时，可能包含不再正确或相关的信息。您可以使用网络浏览器查看本地安装的手册，使用以下链接：

FreeBSD 手册

/usr/local/share/doc/freebsd/en/books/handbook/handbook_en.pdf

FreeBSD FAQ

/usr/local/share/doc/freebsd/en/books/faq/faq_en.pdf

您可以随时在《文档门户[107]》上找到最新的文档。

[106] https://docs.freebsd.org/en/books/handbook/ports/#ports

[107] https://docs.freebsd.org/

第二章安装 FreeBSD

2.1. 概述

FreeBSD 支持不同的架构，包括 amd64, ARM®, RISC-V®, 和 PowerPC®。根据不同的架构和平台，可以下载[108]不同的镜像来安装或直接运行 FreeBSD。

有以下类型镜像：

- 虚拟机磁盘镜像，如 `qcow2`、`vmdk`、`vhd` 和 raw 设备镜像。这些不是安装镜像，而是已经预装了 FreeBSD 并准备好进行安装后任务的镜像。虚拟机镜像也常用于云环境中。
- SD 卡镜像，用于嵌入式系统，如树莓派（Raspberry Pi）。这些文件必须被解压缩并以 raw 镜像的形式写入 SD 卡，主板将从中启动。
- 安装镜像，从 ISO 或 USB 设备启动，将 FreeBSD 安装在磁盘上，用于一般的台式机、笔记本电脑或服务器设备。

本章的其余部分说明了第三种情况，解释了如何使用基于文本的安装程序 bsdinstall 来安装 FreeBSD。安装程序和这里显示的内容可能会有一些细微的差别，因此，请将本章作为一个通用的指南，而不是一套文字说明。

读完本章后，你将知道：

- 如何获得 FreeBSD 镜像并创建 FreeBSD 安装介质。
- 如何启动 bsdinstall。
- bsdinstall 会问的问题，它们意味着什么，以及如何回答它们。
- 如何解决安装失败的问题。
- 如何在确认安装之前访问 FreeBSD 的 live 版本。

[108] https://www.freebsd.org/where/

2.2.最低硬件要求

安装 FreeBSD 所需的硬件要求因架构和版本而异。FreeBSD RELEASE 所支持的硬件架构和设备会在 FreeBSD 发行信息[109]页面上列出。在 FreeBSD 下载页面[110]上也有如何针对不同架构选择正确镜像的建议。

2.3.安装前的准备工作

在确定设备符合安装 FreeBSD 的最低硬件要求之后，就可以下载安装文件并准备好安装盘。在这之前，请检查设备是否已经准备好进行安装，可以通过核对下面这个检查表中的任务来进行：

1.备份重要数据

在安装任何操作系统之前，总是需要先备份所有的重要数据。不要将备份存储在将要安装的系统上。而应将数据保存在可移动磁盘，如 USB 设备、网络上的另一个系统，或在线备份服务上。在开始安装前检查备份，以确保它包含所有需要的文件。一旦安装程序格式化了系统的磁盘，存储在该磁盘上的所有数据就会丢失。

2.决定在哪里安装 **FreeBSD**

如果 FreeBSD 是唯一要安装的操作系统，就可以跳过这一步。但如果 FreeBSD 将与其他操作系统共享磁盘，请决定哪个在磁盘或分区上用于安装 FreeBSD。

在 i386 和 amd64 架构中，有两种分区方案可以将磁盘划分为多个分区。传统的主引导记录（MBR）拥有一个最多可定义四个主分区的分区表。由于历史原因，FreeBSD 将这些主分区称为 *slice*。这些主分区中的某一个可以被拓展为包含多个 逻辑分区的扩展分区。*GUID* 分区表（GPT，全局唯一标识分区表）是一种较新、较简单的磁盘分区方法。常见的 GPT 实现允许每个磁盘有多达 128 个主分区，消除了对逻辑分区的需求。

FreeBSD 引导加载器需要一个主分区或 GPT 分区。如果所有的主分区或 GPT 分区都已经被使用，则必须为 FreeBSD 腾出一个主分区。要在不删除现有数据的情况下创建一个分区，可以使用分区调整工具来缩小现有的分区，并利用释放的空间创建一个新的分区。

各种免费的和商业的分区调整工具列在 http://en.wikipedia.org/wiki/List_of_disk_partitioning_software 上。GParted Live（https://gparted.org/livecd.php）是预装了 GParted 分区编辑器的免费的 Live CD。GParted 也被预装在许多其他的 Linux Live CD 发行版中。

警告

如果使用得当，磁盘缩减工具可以安全地创造空间来创建一个新的分区。由于存在选择错误分区的可能性，因此在修改磁盘分区之前，一定要备份所有重要数据，并验证备份的完整性。

包含不同操作系统的磁盘分区使得在一台计算机上安装多个操作系统成为可能。另一种方法是使用虚拟化（虚拟化技术[111]），它允许多个操作系统同时运行，而无需修改任何磁盘分区。

3.收集网络信息

[109] https://www.freebsd.org/releases/

[110] https://www.freebsd.org/where/

[111] https://docs.freebsd.org/en/books/handbook/virtualization/index.html#virtualization

某些安装 FreeBSD 的方法需要网络连接来下载安装文件。在之后的安装中，安装程序会要求对系统的网络接口进行设置。

如果网络中有 DHCP 服务器，可以用它来提供自动的网络配置。如果 DHCP 不可用，则必须从本地网络管理员或互联网服务提供商那里获得系统的下列网络信息：

所需的网络信息

a. IP 地址

b. 子网掩码

c. 默认网关的 IP 地址

d. 网络域名

e. 网络的 DNS 服务器的 IP 地址

4.检查 **FreeBSD** 勘误表

尽管 FreeBSD 项目努力确保每个版本的 FreeBSD 都尽可能的稳定，但偶尔还是会有一些 bug 悄悄产生。在极罕见的情形下，这些 bug 会影响安装过程。随着这些问题的发现和修正，它们会在每个版本的 FreeBSD 勘误表页面上注明。在安装前请检查勘误表，以确保没有可能影响安装的问题。

所有发行版本的信息和勘误表都可以在 FreeBSD 网站上的发行信息部分找到（https://www.freebsd.org/releases/）。

2.3.1.准备好安装设备

FreeBSD 安装程序不是一个可以在其他操作系统中运行的应用程序。而是需要下载一个 FreeBSD 安装文件，将其刻录到与其文件类型和大小相关的设备上（CD、DVD 或 USB），然后从插入的设备启动系统进行安装。

可以在 FreeBSD 下载页面[112] 找到 FreeBSD 的安装文件。每个安装文件的名称都包括 FreeBSD 的发行版本、架构和文件类型。

安装文件有几种格式，有使用 xz(1)[113] 压缩的和未经压缩的。这些格式因计算机架构和设备类型而异。

安装文件类型：

- `-bootonly.iso`：这是最精简的安装文件，因为它只包含安装程序。在安装过程中需要一个正常的互联网连接，因为安装程序会下载 FreeBSD 安装所需的文件。应该用刻录软件把这个文件刻录到光盘上。
- `-disc1.iso`：这个文件包含了安装 FreeBSD 所需的所有文件，包含 FreeBSD 的源代码，以及 ports。该文件应该被用于刻录光盘。
- `-dvd1.iso`：这个文件包含了各种安装 FreeBSD 所需的所有文件，以及其源代码和 ports。它还包含了一套用于安装窗口管理器和一些应用程序的常用二进制包，这样就可以用安装设备安装一个完整的系统，而不需要连接到互联网。该文件应该被用于刻录光盘。
- `-memstick.img`：这个文件包含了安装 FreeBSD 所需的所有文件，以及其源代码和 ports。应该按照下面的说明把它刻录到 U 盘上。

[112] https://www.freebsd.org/where/

[113] https://www.freebsd.org/cgi/man.cgi?query=xz&sektion=1&format=html

- `-mini-memstick.img`：和 `-bootonly.iso` 一样，不包括安装所需文件，但会根据需要下载它们。在安装过程中需要一个正常的互联网连接。应该如将镜像文件写入 U 盘[114]中所示将此文件写入 U 盘。

在下载镜像文件后，在相同目录中至少下载其中一个 *checksum*，校验和文件。共有两个 校验和文件，均以版本号和架构名称命名，例如 **CHECKSUM.SHA256-FreeBSD-13.1-RELEASE-amd64** 和 **CHECKSUM.SHA512-FreeBSD-13.1-RELEASE-amd64**。

在下载其中一个（或两个）文件后，计算镜像文件的 校验和，并与 校验和文件中显示的内容进行比对。注意，你需要将计算出的校验和与正确的文件进行比对，因为它们分别对应于两种不同的算法——SHA256 和 SHA512。FreeBSD 提供了 sha256(1)[115] 和 sha512(1)[116] 可以用来计算 校验和。其他操作系统也有类似的程序。

可以通过执行 sha256sum(1)[117]（或 sha512sum(1)[118]）自动完成在 FreeBSD 中验证 校验和的过程：

```
% sha256sum -c CHECKSUM.SHA256-FreeBSD-13.1-RELEASE-amd64 FreeBSD-13.1-RELEASE-amd64-
↪dvd1.iso
FreeBSD-13.1-RELEASE-amd64-dvd1.iso: OK
```

这些校验和必须完全匹配。如果校验和不匹配，就说明镜像文件已损坏，必须重新下载。

2.3.1.1.将镜像文件写入 USB 设备

***.img** 文件是对存储设备完整内容的 镜像。它 不能被当做文件复制到目标设备上。有几种不同的应用程序可以将 ***.img** 文件写入 U 盘。本节将介绍其中的两个。

重要

在继续进行之前，请备份 U 盘上一切重要的数据。这个程序将删除 U 盘上的现有数据。

使用 **dd** 刻录镜像的步骤

警告

这个例子使用 **/dev/da0** 作为目标设备，镜像将被写入该设备。要 非常谨慎地使用正确的设备，因为这个命令会破坏指定的目标设备上的现有数据。

这个命令行工具在 BSD、Linux® 和 Mac OS® 系统上可用。要使用 `dd` 刻录镜像，插入 U 盘并确定其设备名称。然后，指定下载的安装文件的名称和 U 盘的设备名称。这个例子将 amd64 安装镜像刻录到现有 FreeBSD 系统的第一个 USB 设备上。

```
# dd if=FreeBSD-13.1-RELEASE-amd64-memstick.img of=/dev/da0 bs=1M conv=sync
```

如果这个命令执行失败，请检查有无挂载 U 盘，且设备名称应是磁盘的名称，而不是一个分

[114] https://docs.freebsd.org/en/books/handbook/bsdinstall/#bsdinstall-usb

[115] https://www.freebsd.org/cgi/man.cgi?query=sha256&sektion=1&format=html

[116] https://www.freebsd.org/cgi/man.cgi?query=sha512&sektion=1&format=html

[117] https://www.freebsd.org/cgi/man.cgi?query=sha256sum&sektion=1&format=html

[118] https://www.freebsd.org/cgi/man.cgi?query=sha512sum&sektion=1&format=html

区。一些操作系统可能要求必须用 sudo(8)[119] 运行这个命令。dd(1)[120] 的语法在不同的平台上略有不同；例如，Mac OS® 需要小写的 `bs=1m`。像 Linux® 这样的系统可能会对写入先进行缓冲。要保证所有的写入操作完成，请使用 sync(8)[121] 命令。

使用 **Windows®** 刻录镜像的步骤

警告

请确保选定正确的盘符，因为指定磁盘上的现有数据将被覆盖和销毁。

1. 获得用于 **Windows®** 的 **Image Writer**

Image Writer for Windows® 是一个免费软件，可以无误地将镜像文件写入存储设备。可从 win32diskimager 主页[122] 下载，并将其解压到一个文件夹中。

2、用 **Image Writer** 写入镜像

双击 Win32DiskImager 图标以启动该程序。检查 `Device` 下显示的盘符字母对应插入的存储设备。点击文件夹图标，选择要写入存储设备的镜像文件。点击 **Save** 确认镜像文件名。检查是否一切正确，并且存储设备中的文件夹没有在其他窗口中打开。一切准备就绪后，点击 **Write**，将镜像文件写入存储设备。

2.4. 开始安装

重要

默认情况下，在出现以下信息之前，安装程序不会对磁盘进行任何修改：

```
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?
```

在此之前均可安全退出。如果担心有什么地方配置不正确，只要在这之前关闭计算机，设备的磁盘就不会有任何变化。

本节介绍了如何从安装设备启动系统，安装设备是按照准备安装介质[123]指导准备好的。当使用可启动系统的 U 盘时，在打开计算机之前先插入 U 盘。当从 CD 或 DVD 光盘启动时，请在启动计算机后立即放入光盘。在不同架构的设备下设置系统从插入的设备启动的方式会有所不同。

[119] https://www.freebsd.org/cgi/man.cgi?query=sudo&sektion=8&format=html

[120] https://www.freebsd.org/cgi/man.cgi?query=dd&sektion=1&format=html

[121] https://www.freebsd.org/cgi/man.cgi?query=sync&sektion=8&format=html

[122] https://sourceforge.net/projects/win32diskimager/

[123] https://docs.freebsd.org/en/books/handbook/bsdinstall/#bsdinstall-installation-media

2.4.1.FreeBSD 启动菜单

系统从安装设备启动以后，将显示一个类似于下图的菜单：

图2: FreeBSD 启动菜单

在默认情况下，在启动到 FreeBSD 安装程序之前，该菜单会等待 10 秒钟让用户进行输入；如果已安装 FreeBSD，则会出现在引导 FreeBSD 之前。按空格键来暂停启动计时器以查看选择。要选择某个选项，按菜单条目对应的高亮数字、字符或按键。以下是可选菜单条目：

- `Boot Multi User`：（多用户模式）这将继续 FreeBSD 的引导过程。如果启动计时器已经暂停，按 `1`，大写或小写的 `B`，或回车键继续。
- `Boot Single User`：（单用户模式）此模式可以用来修复现有的 FreeBSD 安装，如 “单用户模式”[124] 中所述。按 `2` 或大写或小写的 `S` 来进入这个模式。
- `Escape to loader prompt`：（切换至引导提示符）这将引导系统进入修复提示界面，该界面包含有限数量的底层命令。这个提示界面将在 “第三阶段”[125] 中进行说明。按 `3` 或 `Esc` 键启动进入这个提示。
- `Reboot`：（重启系统）重启计算机。
- `Cons`: 允许通过 显卡, 串口, 同时（串口为主）或 同时（显卡为主）进行安装。
- `Kernel`：（内核）加载一个不同的内核。

[124] https://docs.freebsd.org/en/books/handbook/boot/index.html#boot-singleuser

[125] https://docs.freebsd.org/en/books/handbook/boot/index.html#boot-loader

- `Configure Boot Options:`（配置启动参数）打开 FreeBSD 开机选项菜单[126]中介绍的菜单。

图3: FreeBSD 启动引导菜单

开机选项菜单分为两个部分。第一部分可以用来返回到主启动菜单或将任何已调整的选项重置为默认值。

下一个部分是用来调整可用的选项，通过按下高亮数字或字符来使选项 `On` 或 `Off`。系统将始终使用这些参数启动，直到它们被修改。有几个选项可以通过这个菜单进行调整：

- `ACPI Support:`（ACPI 支持）如果系统在启动过程中挂起，请尝试将该选项切换为 `off`。
- `Safe Mode:`（安全模式）如果即使将 `ACPI Support` 设置为 `Off`，系统在启动时仍然挂起，请尝试将这个选项设置为 `On`。
- `Single User:`（单用户模式）将这个选项切换到`On`来修复现有的 FreeBSD 安装，如单用户模式[127]中所述。如果问题得到解决了，再将其设置为 `Off` 即可。
- `Verbose:`（详情模式）把这个选项切换到`On`，可以在启动过程中看到更详细的信息。这在排除某个硬件故障时可能很有用。

在做出必要的选择后，按 `1` 或退格键键返回主启动菜单，然后按回车键继续启动 FreeBSD。当 FreeBSD 进行硬件设备探测并加载安装程序时，会出现一系列的启动信息。启动完成以后，将显示欢迎菜单[128]。

[126] https://docs.freebsd.org/en/books/handbook/book/#bsdinstall-boot-options-menu

[127] https://docs.freebsd.org/en/books/handbook/boot/index.html#boot-singleuser

[128] https://docs.freebsd.org/en/books/handbook/book/#bsdinstall-choose-mode

图4: FreeBSD 开机选项菜单

按回车键，选择默认的 **Install**，进入安装程序。本章的剩余部分将对这个安装程序的使用方法进行介绍。另外，可使用左右箭头或键入彩色字母来选择所需的菜单项。**Shell** 可以用来访问一个 FreeBSD shell，以便在安装前使用命令行工具来对磁盘进行分区。**Live CD** 选项可以用来在安装前先试用 FreeBSD。Live 版本在使用 Live CD[129] 中有说明。

> 技巧
>
> 要查看开机信息，包括硬件设备探测，按大写或小写的 S，然后按回车键可进入 shell。在 shell 提示符下，输入 `more /var/run/dmesg.boot`，并使用空格键滚动浏览信息。完成后，键入 `exit` 来返回欢迎菜单。

[129] https://docs.freebsd.org/en/books/handbook/book/#using-live-cd

2.5.使用 bsdinstall

本节展示了 bsdinstall 菜单的顺序以及在安装系统前将被询问的信息类型。使用方向键选中一个菜单选项，然后用空格键选择或取消选择该菜单项。完成后，按回车键保存选择并进入下一界面。

2.5.1.选择键盘布局

在开始这个过程之前，bsdinstall 将加载键盘布局[130]，如图所示。

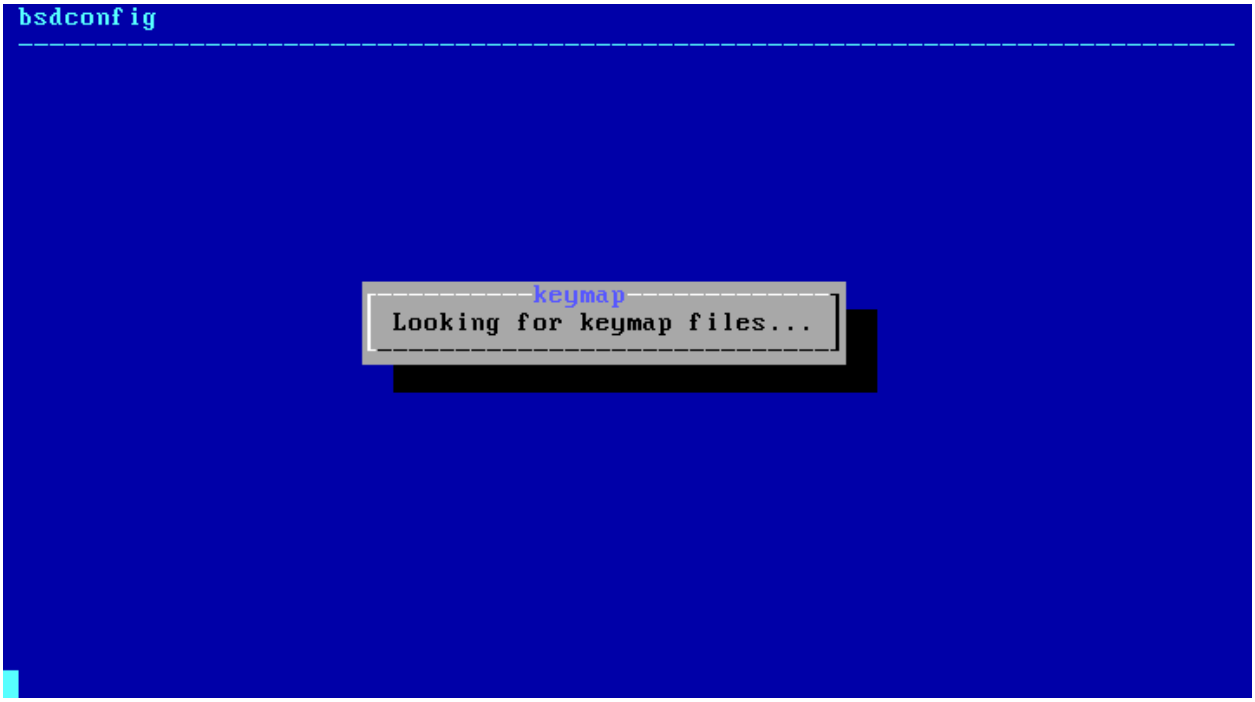


图5: 键盘布局

键盘布局文件加载完毕后，bsdinstall 会显示键盘布局选择菜单[131]中所示的菜单。使用键盘上的上下键来选择最能代表你键盘布局的选项，并按回车键来保存选择。

[130] https://docs.freebsd.org/en/books/handbook/bsdinstall/#bsdinstall-keymap-loading

[131] https://docs.freebsd.org/en/books/handbook/bsdinstall/#bsdinstall-keymap-10

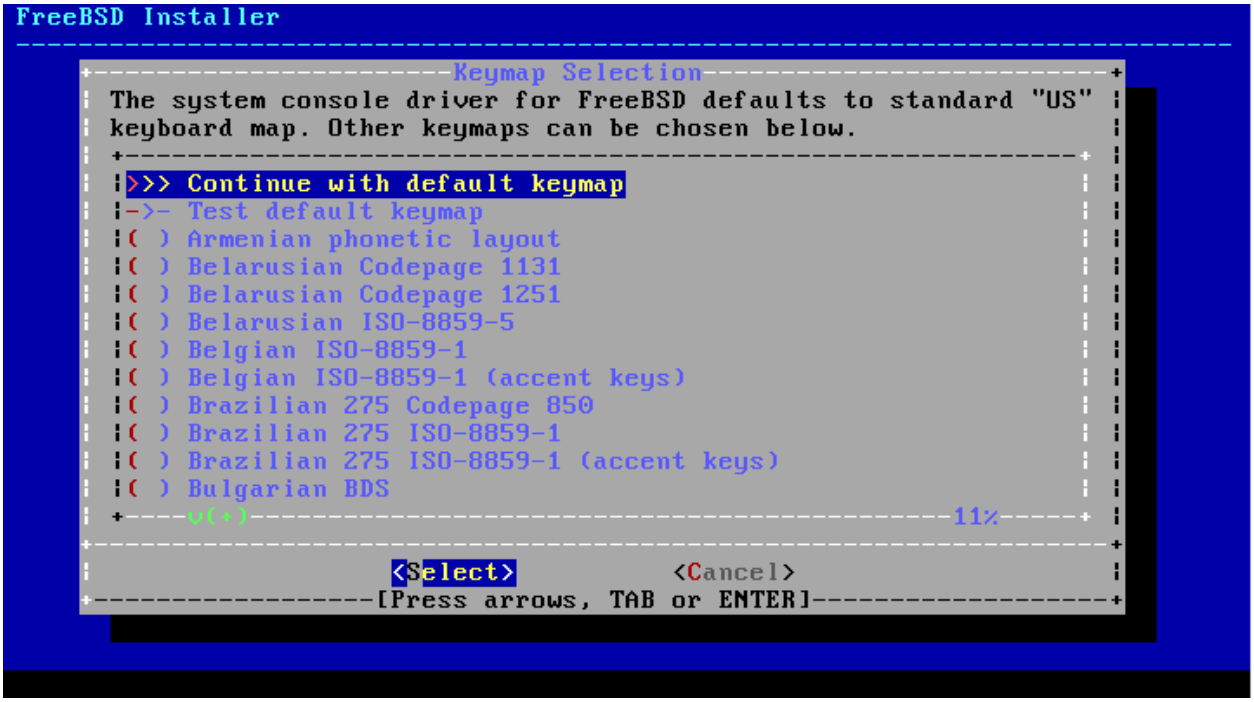


图6: 显示所有支持的键盘的键盘图选择菜单

注意

按 Esc 键将退出这个菜单，并使用默认的键盘布局。如果不确定该选择哪一项，United States of America ISO-8859-1 也是一个稳妥的选择。

此外，当选择其他键盘布局时，用户可以对其进行测试，并确保它是正确的，然后再继续，如键盘布局测试菜单[132]所示。

[132] https://docs.freebsd.org/en/books/handbook/bsdinstall/#bsdinstall-keymap-testing

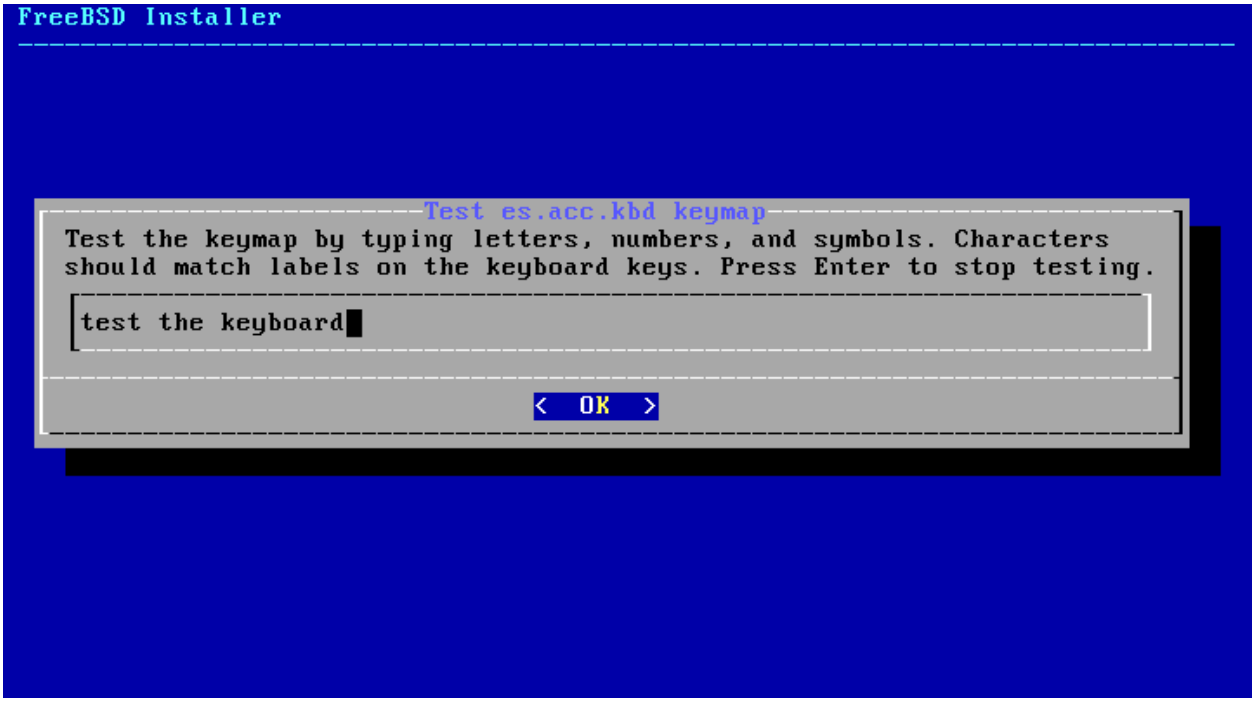


图7: 键盘布局测试菜单

2.5.2.设置主机名

下一个 bsdinstall 菜单是用来设置新安装系统的主机名。

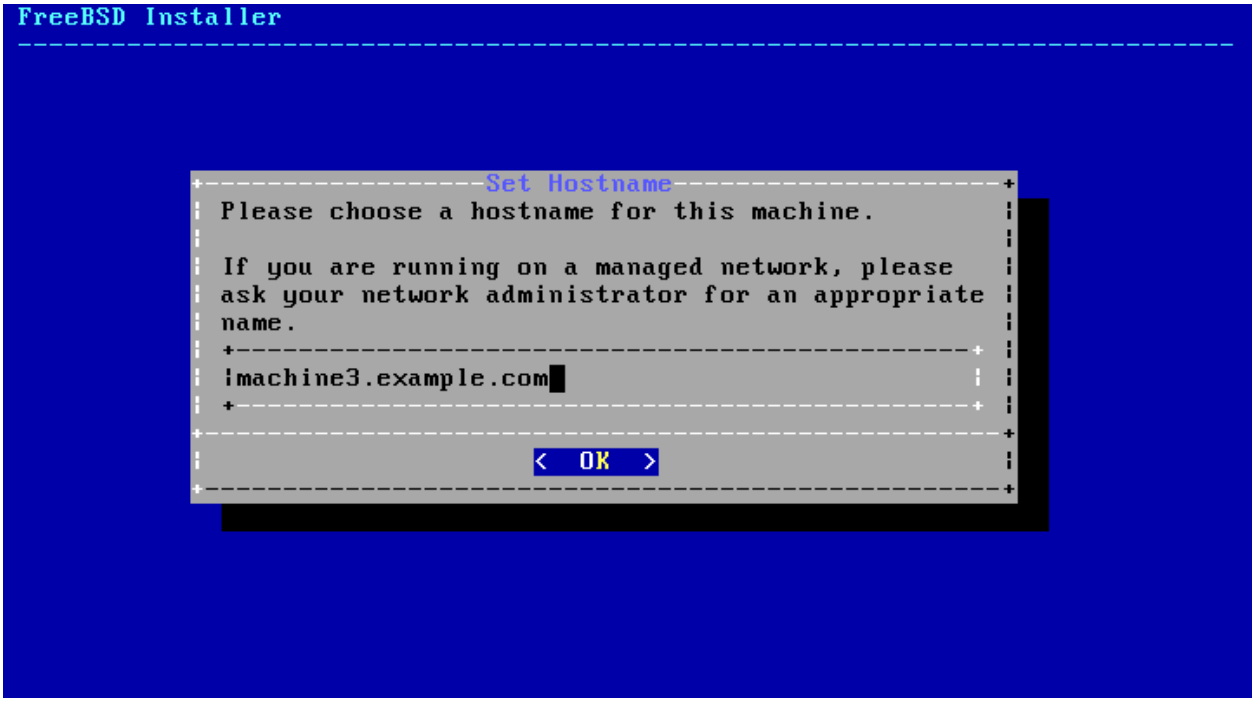


图8: 设置主机名

键入一个在网络上唯一的主机名。它应该是一个完全限定的主机名，如 `machine3.example.com`。

2.5.3.选择要安装的组件

接下来 bsdinstall 会提示选择要安装的组件。

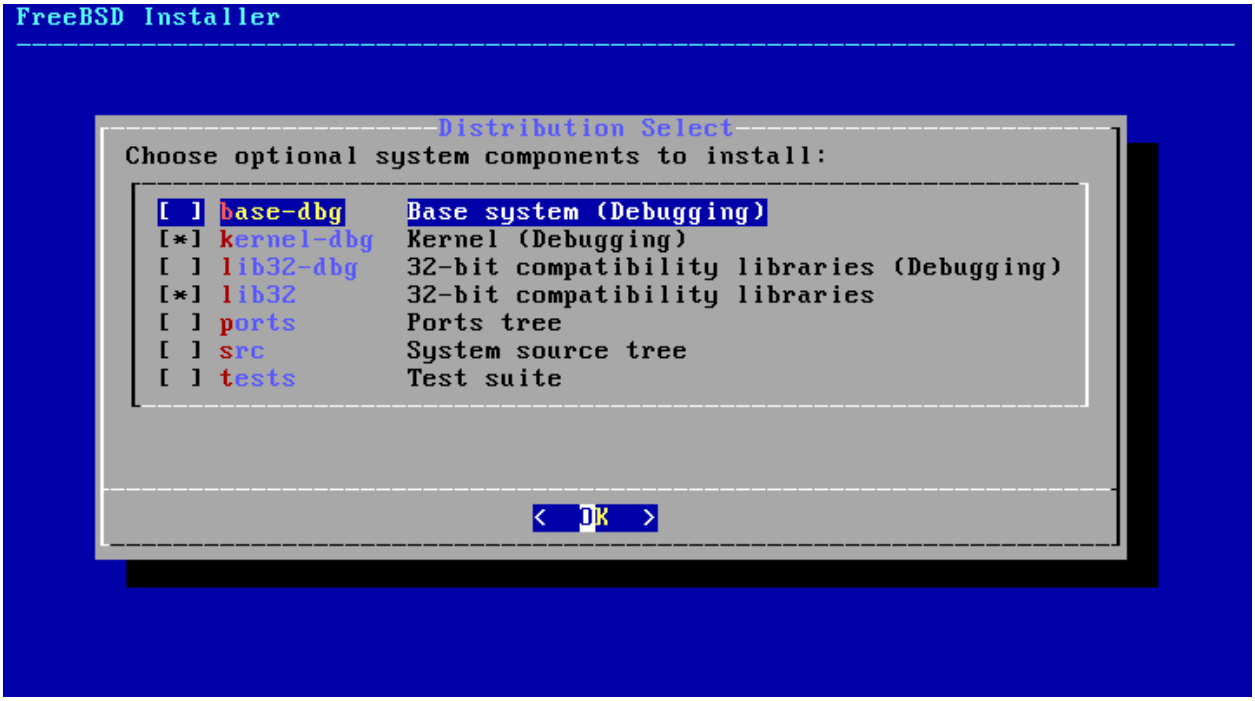


图9: 可以安装的不同组件。例如：base-dbg, lib32, ports, 等等。

决定安装哪些组件主要取决于系统的预期用途和可用的磁盘空间大小。FreeBSD 内核和用户空间，统称为 基本系统，为默认安装。根据架构的不同，其中一些组件可能不会出现：

- `base-dbg`——基本工具，如 `cat`、`ls` 以及许多其他已激活调试符号的工具。
- `kernel-dbg`——激活了调试符号的内核和模块。
- `lib32-dbg`——用于在 64 位版本的 FreeBSD 上运行 32 位应用程序的兼容库并激活调试符号。
- `lib32`——用于在 64 位版本的 FreeBSD 上运行 32 位应用程序的兼容性库。
- `ports`——FreeBSD ports 是一个文件夹，可以自动下载、编译和安装第三方软件包。安装应用程序：软件包和 Ports[133] 讨论了如何使用 ports。

  警告

  安装程序不检查是否有足够的磁盘空间。请只在有足够的硬盘空间时才选择这个选项。FreeBSD ports 会占用大约 3GB 的磁盘空间。

- `src`——包含内核和用户空间的完整的 FreeBSD 源代码。尽管大多数应用程序都不需要它，但在构建设备驱动程序、内核模块，或某些来自 ports 的应用程序时，可能需要它。它也被用于开发 FreeBSD 本身。完整的源代码需要 1GB 的磁盘空间，重新编译整个 FreeBSD 系统需要额外的 5GB 空间。
- `tests`——FreeBSD 测试套件。

[133] https://docs.freebsd.org/en/books/handbook/ports/index.html#ports

2.5.4.通过网络安装

通过网络安装[134]中显示的菜单只在通过 **-bootonly.iso** 或 **-mini-memstick.img** 安装时出现，因为这种安装镜像不包含所需的安装文件。由于安装文件必须通过网络连接来获取，这个菜单提示必须先配置网络接口。如果在这个过程的任何步骤中显示这个菜单，请记住按照配置网络接口[135]的说明进行操作。

图10: 表示没有找到某些组件，将使用网络下载。

2.6.分配磁盘空间

接下来的菜单可用来选择分配磁盘空间的方案。

[134] https://docs.freebsd.org/en/books/handbook/bsdinstall/#bsdinstall-netinstall-notify

[135] https://docs.freebsd.org/en/books/handbook/bsdinstall/#bsdinstall-config-network-dev

图11: 显示不同的分区选项。例如手册，shell 等。

bsdinstall 为用户提供了四种分配磁盘空间的方案：

- `Auto (ZFS)` 分区将创建一个使用 ZFS 作为根分区的系统，并为 引导环境提供可选的 GELI 加密支持。
- `Auto (UFS)` 分区将使用 `UFS` 文件系统自动设置磁盘分区。
- `Manual` 分区允许专业用户通过菜单选项创建自定义分区。
- `Shell` 将打开 shell 提示符，专业用户可以使用 gpart(8)[136]、fdisk(8)[137] 和 bsdlabel(8)[138] 等命令行工具创建自定义分区。

本节列出了在分配磁盘分区时需要考虑的问题，并演示了不同分区的使用方法。

2.6.1.设计分区布局

文件系统的默认分区布局是整个系统使用一个文件系统。当使用 `UFS` 时，如果你有足够的磁盘空间或多个磁盘，可能值得考虑使用多个文件系统。在布局文件系统时，请记住，硬盘的外部轨道较内部轨道传输数据的速度快。因此，零碎和经常读写的文件系统应该靠近硬盘的外部，而较大的分区，如 **/usr** 应该放在磁盘的内部。推荐以类似的顺序创建分区：**/**、**swap**、**/var** 然后是 **/usr**。

/var 分区的大小反映了机器的用途。这个分区用来存放邮件、日志文件和打印后台处理程序。邮件和日志文件可以增长到意想不到的大小，这取决于用户数量和日志文件的保存时间。一般来说，大多数用户的

[136] https://www.freebsd.org/cgi/man.cgi?query=gpart&sektion=8&format=html

[137] https://www.freebsd.org/cgi/man.cgi?query=fdisk&sektion=8&format=html

[138] https://www.freebsd.org/cgi/man.cgi?query=bsdlabel&sektion=8&format=html

/var 很少会占用超过 1GB 的可用磁盘空间。

> 注意
>
> 有时，**/var/tmp** 需要大量的磁盘空间。当安装新软件时，打包工具会在 **/var/tmp** 下提取软件包的临时存档。如果 **/var/tmp** 下没有足够的磁盘空间，大型软件包（如 Firefox 或 LibreOffice）的安装可能会很麻烦。

/usr 分区存放着许多支持系统的文件，包括 FreeBSD ports 和系统源代码。这个分区建议至少要有 2GB 的空间。另外，请注意，用户的主目录默认放在 `/usr/home` 中，但也可以放在其他分区上。在默认情况下，`/home` 是指向 `/usr/home` 的符号链接。

在选择分区大小的时候，要牢记空间需求。一个分区的空间用完了，而另一个分区却几乎用不完，这可能是个麻烦事。

根据经验，交换空间应该是物理内存（RAM）的两倍（(对于较大的内存配置来说，可以不到两倍)。拥有较小内存的系统可能会因为有更多的交换空间而表现得更好。配置过小的交换空间会导致虚拟内存页面管理效率低下，如果以后添加更多的内存，可能会产生问题。

在具有多个 SCSI 磁盘或在不同控制器上运行的多个 IDE 磁盘的大型设备上，建议在每个磁盘上都配置交换空间，最多可为四个磁盘配置。交换分区的大小应该大致相同。虽然内核可以处理任意大小的交换空间，但是当内部数据结构扩展到最大交换分区的 4 倍时，保持交换分区接近相同的大小将可让内核以最佳方式在不同的磁盘上串联交换空间。大的交换空间可能会引起内核对总配置的交换空间数量的警告信息。按照警告信息的指示，可通过增加允许用于跟踪交换分配的内存量来提高限制。在被迫重启之前，可能更容易从失控的程序中恢复。

通过对系统进行适当的分区，在零碎文件写入量大的分区中引入的碎片将不会干扰频繁读取的分区。让写入量大的分区更接近磁盘的边缘，可提高频繁写入分区的 I/O 性能。虽然可能较大的分区也需要提升 I/O 性能，但将它们移向磁盘边缘不会比将 **/var** 移向磁盘边缘带来的性能改善那么明显。

2.6.2.使用 UFS 进行向导式分区

当选择这种方法时，菜单将显示可用的磁盘。如果使用了多个磁盘，请选择要将 FreeBSD 安装到哪个磁盘。

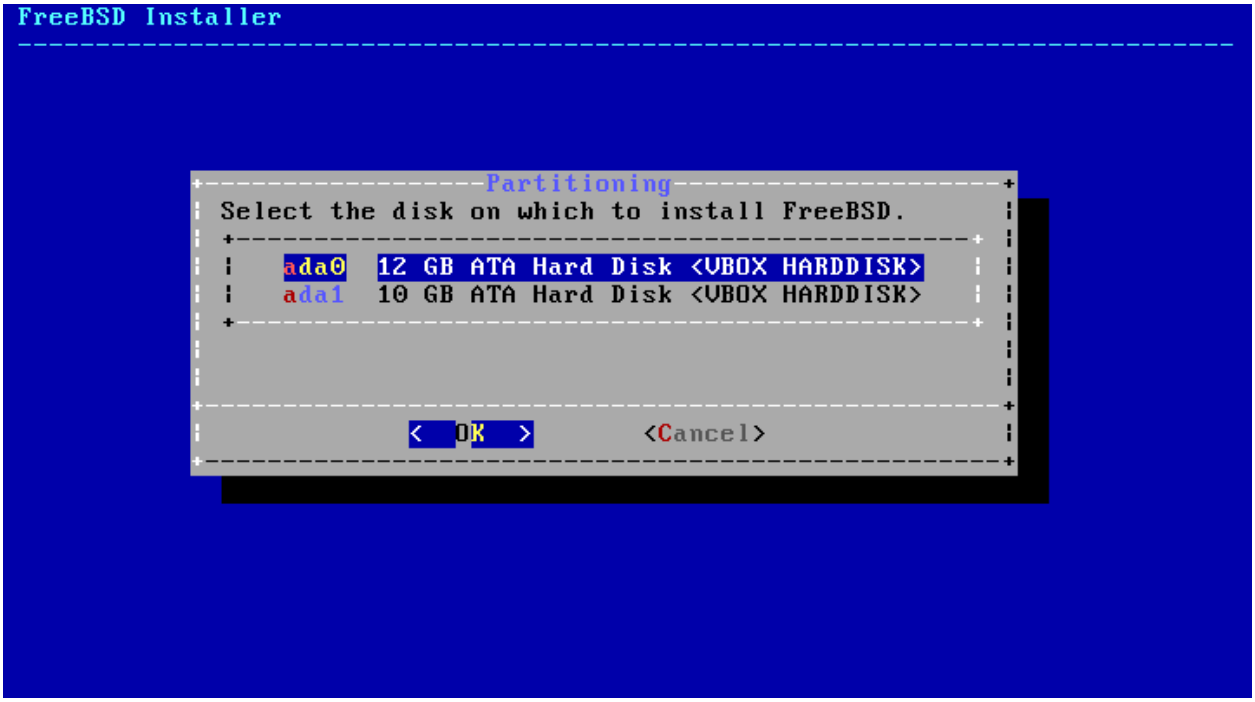


图12: 显示可以安装FreeBSD的磁盘列表

选择了磁盘以后，接下来的菜单就会提示安装到整块磁盘或利用可用空间创建一个分区。如果选择 **Entire Disk**，将自动创建一个使用整个磁盘的通用分区布局。选择 **Partition** 则用磁盘上未使用的空间创建一个分区布局。

图13: 菜单询问用户是否要使用磁盘上的所有可用空间，或者是否要建立一个分区

在选择了 **Entire Disk** 之后，bsdinstall 会显示一个对话框，表明磁盘所有内容将被删除。

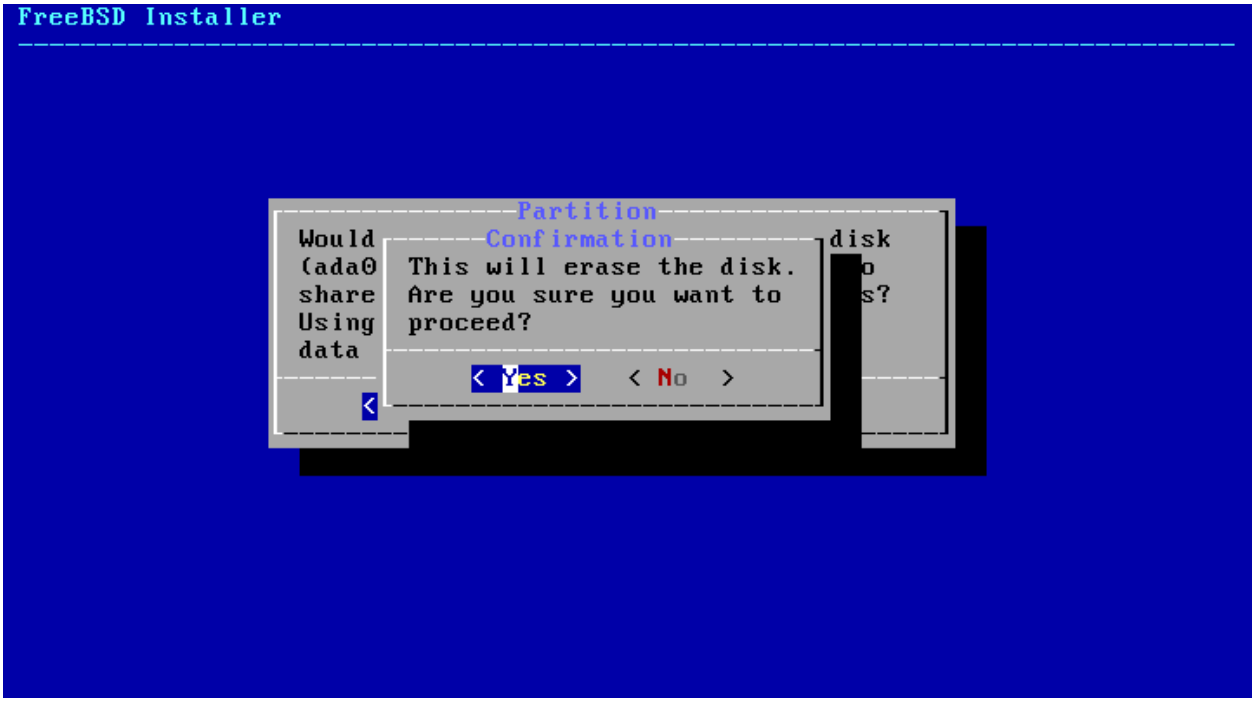


图14: 指示用户磁盘上的所有数据将被删除并要求确认的菜单

接下来的菜单显示了可用分区表的列表。GPT 通常是最合适 amd64 计算机的选择。与 GPT 不兼容的老式计算机应该使用 MBR。其他分区表一般用于不常见的或较老的计算机。更多信息可在分区表[139]中找到。

图15: 向用户显示不同类型的分区并请求其中一个分区的菜单。

在创建了分区布局之后，请检查它以确保其符合安装的需要。选择 **Revert** 可以将分区重置为初始值，按 **Auto** 可以重新自动创建 FreeBSD 分区。也可以手动创建、修改或删除分区。当确认分区后，选择 **Finish** 来继续安装。

[139] https://docs.freebsd.org/en/books/handbook/bsdinstall/#partition-schemes

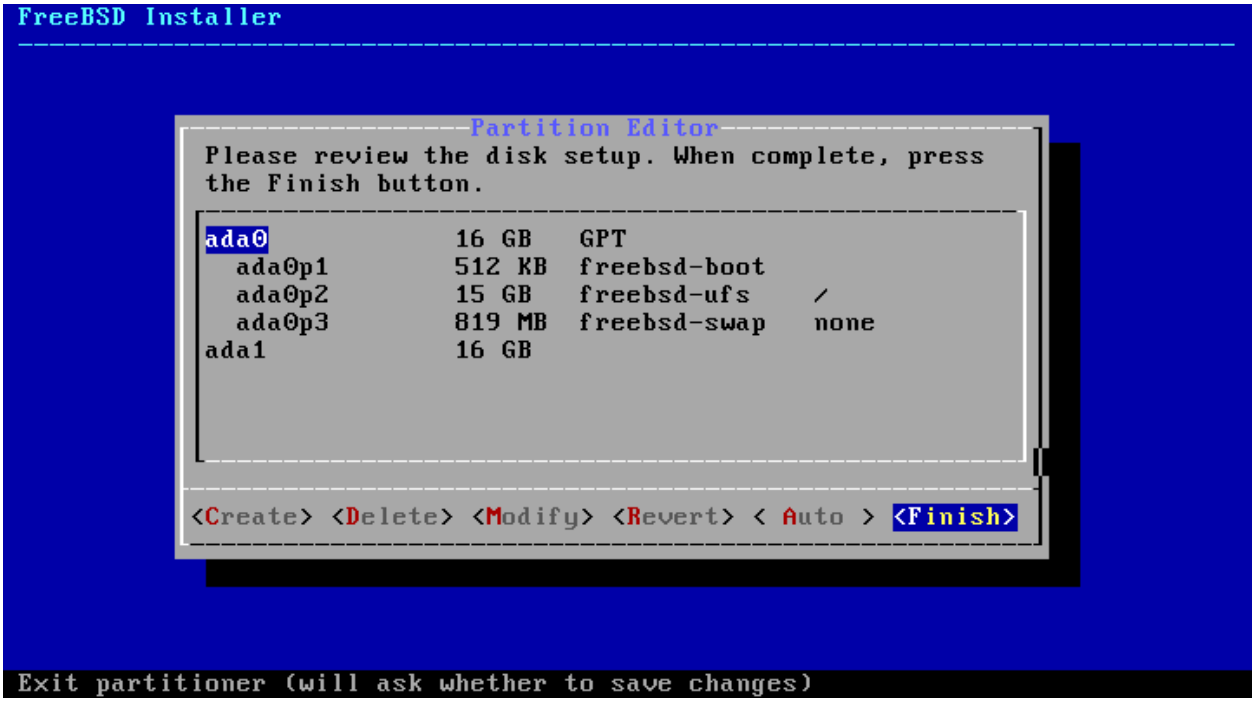


图16: 显示创建的分区的菜单

磁盘被配置好之后，接下来的菜单提供了一个在选定的磁盘被格式化之前进行最后修改的机会。。如果需要进行修改，请选择 **Back**，返回到主分区菜单。**Revert & Exit** 将退出安装程序，不对磁盘做任何改变。此外，选择 **Commit**，开始进行安装过程。

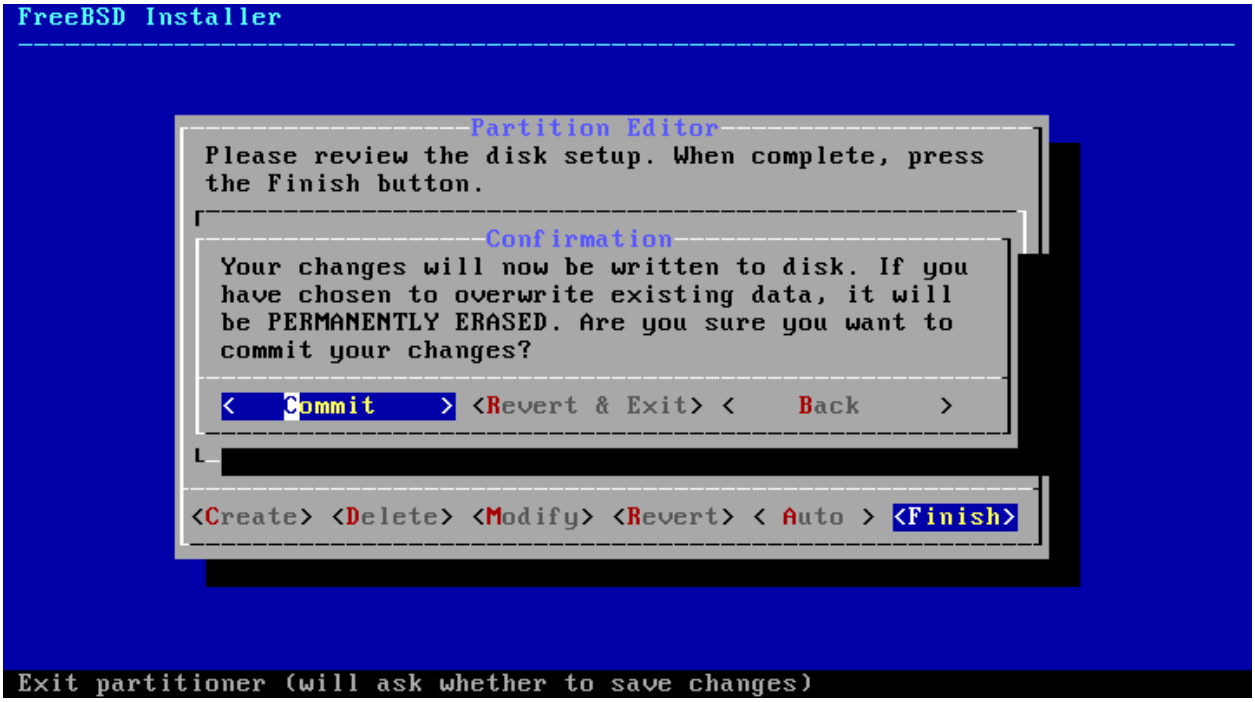


图17: 菜单显示给用户，所有的改变将被写入磁盘，并告知如果他决定继续，现有数据将被永久删除。

要继续安装过程，请进入获取安装文件[140]。

2.6.3.手动分区

手动分区将直接使用分区编辑器进行操作：

[140] https://docs.freebsd.org/en/books/handbook/bsdinstall/#bsdinstall-fetching-distribution

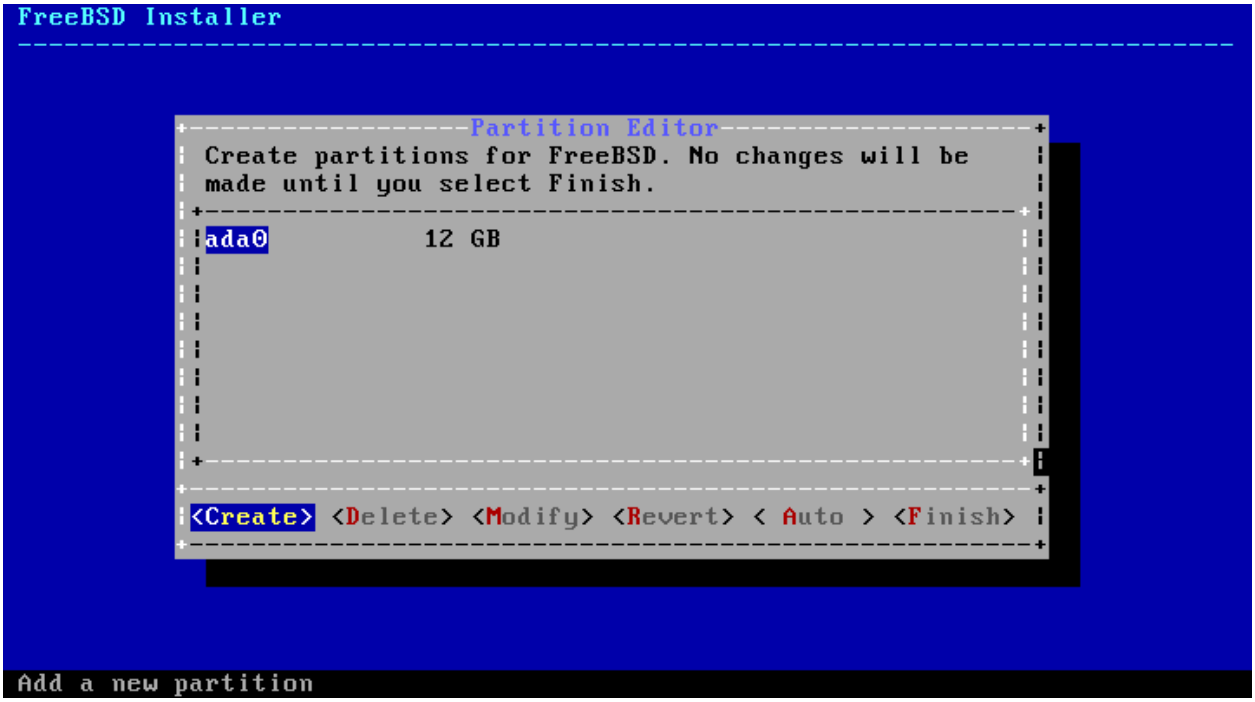


图18: 显示分区编辑器的菜单。

突出显示安装磁盘（本例中为 **ada0**），并选择 **Create** 以显示可用分区表的菜单。

图19: 显示不同类型分区方案的菜单

GPT 通常是 amd64 电脑的最合适选择。与 GPT 不兼容的老式计算机应该使用 MBR。其他分区表一般用于不常见的或较老的计算机。

表格 1.分区表

缩写	说明
APM	PowerPC® 使用的 Apple 分区表
BSD	无 MBR 的 BSD 标签，有时也称作 *危险专用模式*，因为非 BSD 磁盘工具可能无法识别它
GPT	(GUID 分区表[141])
MBR	(主引导记录[142])

选择并创建了分区表后，再次选择 **Create** 来创建分区。`Tab` 键用于在各选项之间移动光标。

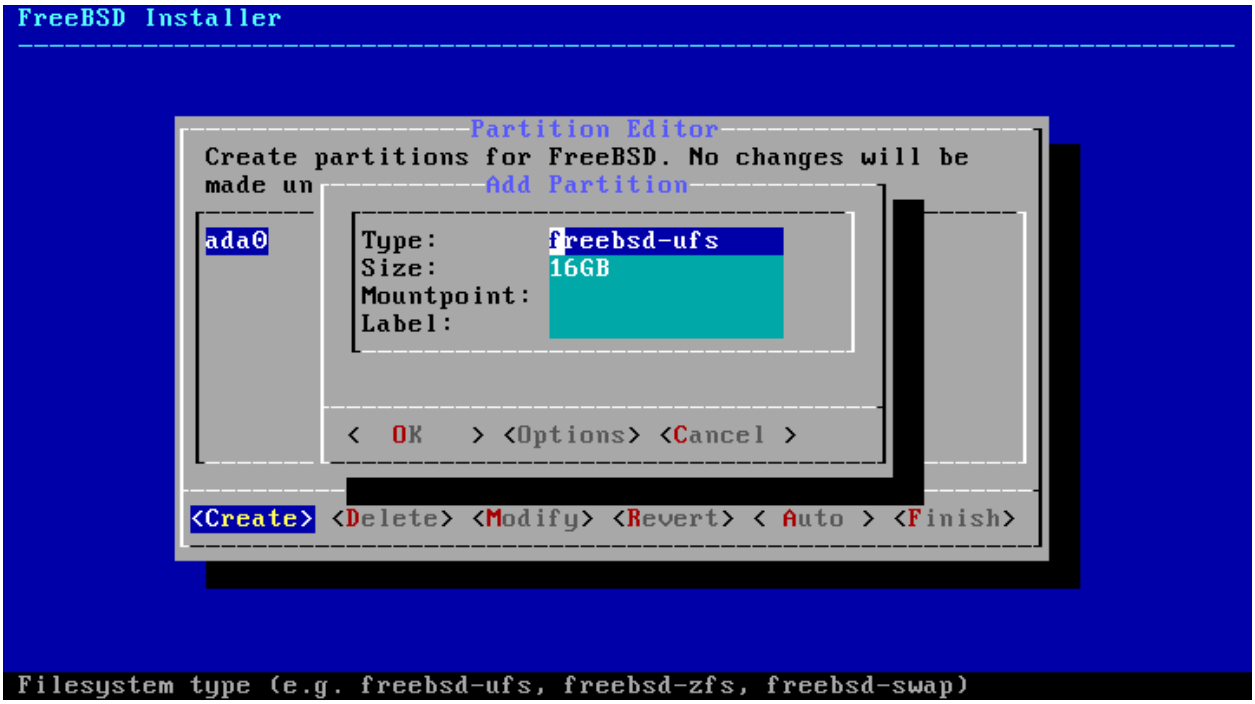


图20: 菜单要求为新分区提供类型、大小、挂载点和标签。

无论使用 ufs 还是 zfs，标准的 FreeBSD GPT 安装都至少要使用三个分区：

- `freebsd-boot` 或 `efi`——存放 FreeBSD 的引导代码。
- `freebsd-ufs`——FreeBSD UFS 文件系统。
- `freebsd-zfs`——FreeBSD ZFS 文件系统。关于 ZFS 的更多信息可以在 Z 文件系统（ZFS）[143]中找到。
- `freebsd-swap`——FreeBSD 交换空间。

[141] http://en.wikipedia.org/wiki/GUID_Partition_Table

[142] http://en.wikipedia.org/wiki/Master_boot_record

[143] https://docs.freebsd.org/en/books/handbook/zfs/index.html#zfs

关于可用的 GPT 分区类型的说明，请参考 gpart(8)[144]。

可创建多个文件系统分区，有些人可能偏好采用传统的布局，为 **/**、**/var**、**/tmp** 和 **/usr** 分开分区。请看创建传统分割文件系统分区的例子[145]。

技巧

注意，在有足够内存的系统上，`/tmp` 可以在以后作为一个基于内存的文件系统（tmpfs(5)[146]）而添加。

可以用常见的缩写来输入 `Size`：*K* 代表 KB，*M* 代表 MB，*G* 代表 GB。

技巧

正确的扇区对齐才能提供最好的性能，使分区大小为 4K 字节的偶数倍有助于确保在具有 512 字节或 4K 字节扇区的设备上对齐。一般来说，使用 1M 或 1G 的偶数倍的分区大小是确保每个分区从 4K 的偶数倍开始的最简单方法。有一个例外：由于目前引导代码的限制，*freebsd-boot* 分区不应大于 512K。

如果该分区将包含文件系统，则需要一个`Mountpoint`（挂载点）。如果只创建了一个 UFS 分区，挂载点应该是 **/**。

`Label`（标签）是一个分区的名称，人们将通过它来了解该分区。如果硬盘被连接到不同的控制器或端口，磁盘的名称或编号可能有所不同，但分区的标签不会改变。如在 **/etc/fstab** 等文件中使用标签而不是磁盘名称和分区号，可使系统在硬件变化时有更大的兼容性。GPT 标签在连接磁盘时会显示在 **/dev/gpt/**。其他分区表的标签的功能各不相同，它们的标签显示在 **/dev/** 的各自目录中。

技巧

每个分区上的标签应是独一无二的，以避免相同的标签产生冲突。可以将计算机名称、用途或位置中的几个字母添加到标签中。例如，在名为 `lab` 的计算机上使用 `labroot` 或 `rootfslab` 作为 UFS root 分区的标签。

例 **1.**创建传统的分离式文件系统分区

对于传统的分区布局而言，即 **/**、**/var**、**/tmp** 和 **/usr** 目录在各自的分区上是一个独立的文件系统。对于一块 20G 大小磁盘而言，首先创建一个 GPT 分区表，然后按图所示创建分区。如果目标磁盘上有更多的空间，则增加交换分区和 **/var** 分区的大小比较有用。这里显示的标签以 `ex` 为前缀，表示“example”（示例），读者应该结合上文使用其他独一无二的标签。

默认情况下，FreeBSD 的 **gptboot** 认为第一个 UFS 分区是 **/** 分区。

[144] https://www.freebsd.org/cgi/man.cgi?query=gpart&sektion=8&format=html

[145] https://docs.freebsd.org/en/books/handbook/bsdinstall/#bsdinstall-part-manual-splitfs

[146] https://www.freebsd.org/cgi/man.cgi?query=tmpfs&sektion=5&format=html

分区类型	大小	挂载点	标签
freebsd-boot	512K		
freebsd-ufs	2G	**/**	exrootfs
freebsd-swap	4G		exswap
freebsd-ufs	2G	**/var**	exvarfs
freebsd-ufs	1G	**/tmp**	extmpfs
freebsd-ufs	采取默认值（使用磁盘的剩余部分）	**/usr**	exusrfs

自定义分区创建完毕后，选择 **Finish** 继续安装，并进入获取安装文件[147]。

2.6.4.使用 Root-on-ZFS 进行向导式分区

这种分区模式只能使用整个磁盘，并且会擦除整个磁盘的内容。ZFS 的主配置菜单提供了许多选项来控制池的创建。

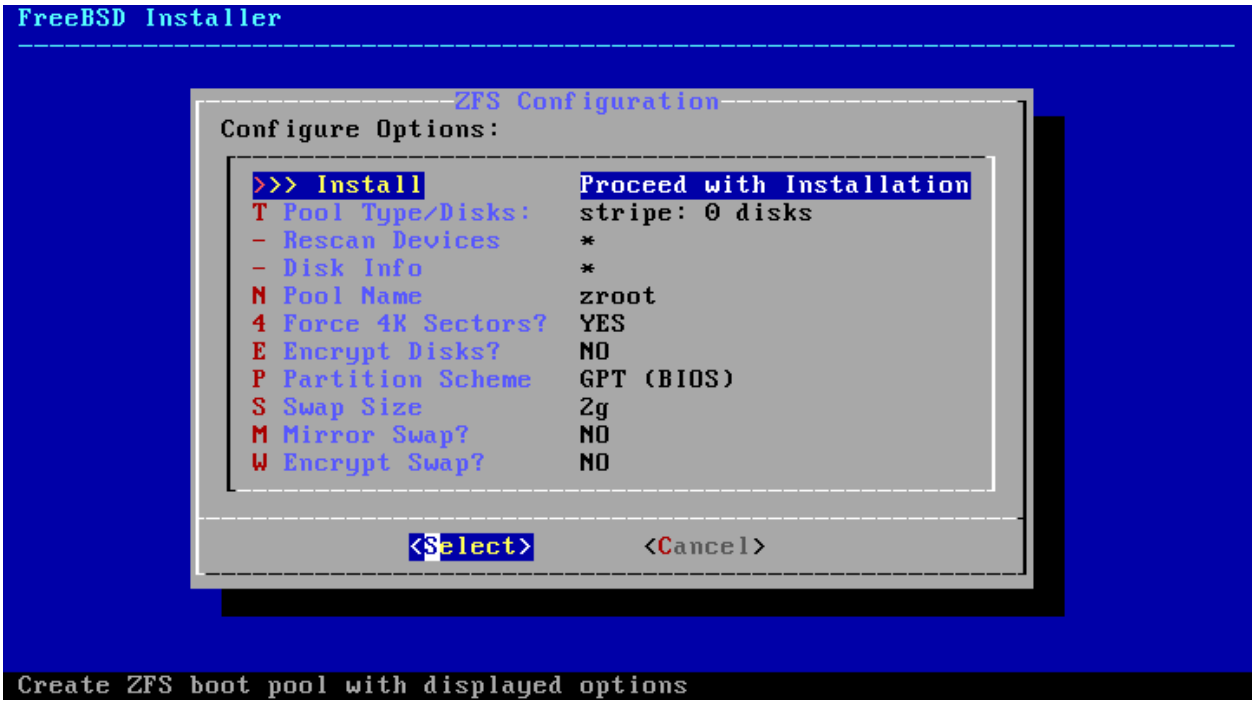


图21: 显示配置ZFS池的不同选项的菜单

下面是这个菜单中选项的摘要介绍：

- `Install`——使用选定的选项继续安装。

[147] https://docs.freebsd.org/en/books/handbook/bsdinstall/#bsdinstall-fetching-distribution

- `Pool Type/Disks`——配置 `Pool Type` 和构成池的磁盘。，除了在条带模式下，ZFS 自动安装程序目前只支持创建单一顶层的`vdev`。要创建更复杂的池，请通过 Shell 分区模式[148]中的说明来创建池。
- `Rescan Devices`——重新加载可用磁盘的列表。
- `Disk Info`——该菜单可以用来检查每个磁盘，包括它的分区表和其他各种信息，如设备型号和序列号（如果有）。
- `Pool Name`——新建池的名称。默认名称是 *zroot*。
- `Force 4K Sectors?`——强制使用 4K 扇区。默认情况下，安装程序会自动创建与 4K 边界对齐的分区并强制 ZFS 使用 4K 扇区。这在 512 字节扇区的磁盘上也是安全的，而且还有一个额外的好处，那就是确保在 512 字节磁盘上创建的池将来可以添加 4K 扇区的磁盘，以作为额外的存储空间或替换故障磁盘。按回车键以选择是否激活它。
- `Encrypt Disks?`——加密磁盘允许用户使用 GELI 对磁盘进行加密。更多关于磁盘加密的信息可以在使用 geli 进行磁盘加密[149]中找到。按回车键，选择是否激活它。
- `Partition Scheme`——选择分区表。在大多数情况下推荐 GPT。按回车键在不同的选项中进行选择。
- `Swap Size`——建立交换空间的大小。
- `Mirror Swap?`——是否在磁盘之间做镜像交换。请注意，启用镜像交换将破坏崩溃转储。按回车键来激活或不激活该选项。
- `Encrypt Swap?`——是否对交换分区进行加密。这将在每次系统启动时用一个临时密钥对交换分区进行加密，并在重启时丢弃。按回车键可以选择是否激活它。在加密 Swap[150] 中有更多关于加密交换分区的信息。

选择 `T` 来配置 `Pool Type` 和构成池的磁盘。

[148] https://docs.freebsd.org/en/books/handbook/bsdinstall/#bsdinstall-part-shell

[149] https://docs.freebsd.org/en/books/handbook/disks/index.html#disks-encrypting-geli

[150] https://docs.freebsd.org/en/books/handbook/disks/index.html#swap-encrypting

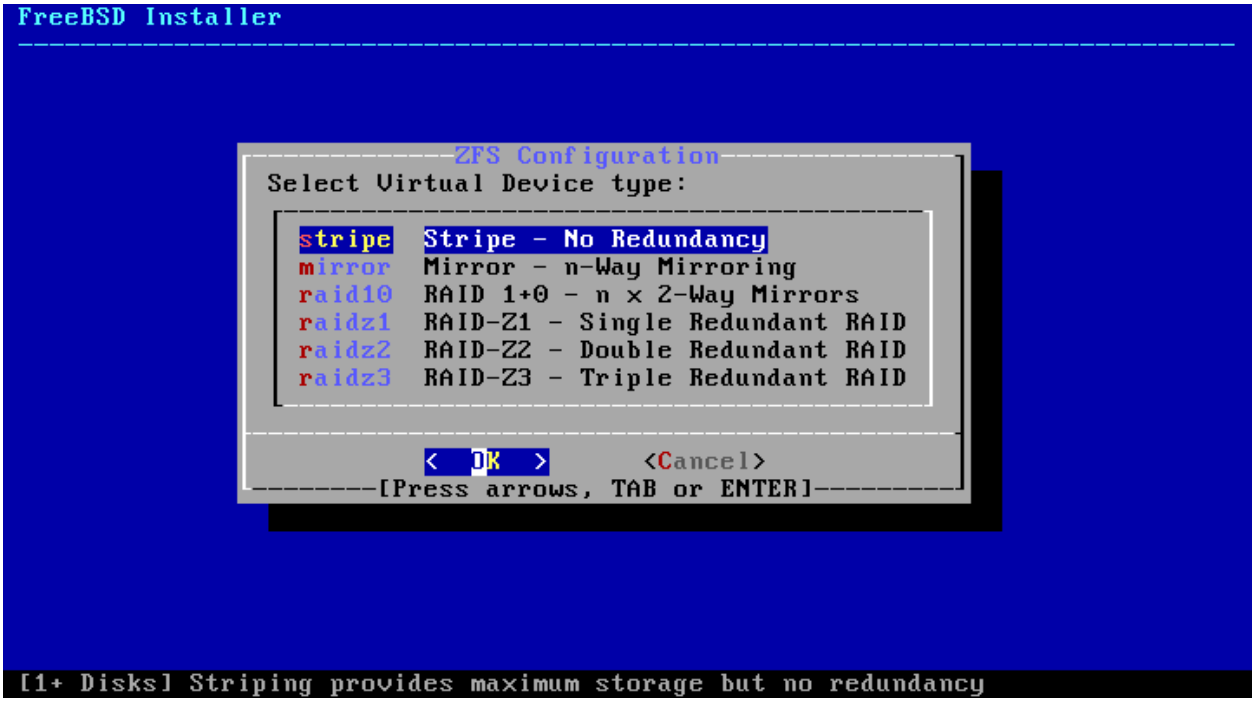


图22: 请求虚拟设备类型的菜单。例如：Stripe, mirror, raidz1

下面是在这个菜单中可以选择的 `Pool Type` 的摘要：

- `stripe`——条带提供所有连接设备相加的最大存储量，但没有冗余。如果任何一个磁盘发生故障，池中的数据将不可逆地丢失。
- `mirror`——镜像在每个磁盘上存储所有数据的一个完整备份。镜像提供了良好的读取性能，因为数据是从所有磁盘上平行读取的。写入性能较慢，因为数据必须写到池中的所有磁盘上。可允许除一个磁盘外的所有磁盘发生故障。这个选项至少需要两个磁盘。
- `raid10`——镜像阵列条带。提供最好的性能，但存储量最少。这个选项需要偶数个硬盘，且至少需要四个硬盘。
- `raidz1`——单一冗余 RAID。可允许一个磁盘同时发生故障。该选项至少需要 3 个磁盘。
- `raidz2`——双重冗余 RAID。可允许两个磁盘同时发生故障。该选项至少需要 4 个磁盘。
- `raidz3`——三重冗余 RAID。可允许三个磁盘同时发生故障。该选项至少需要 5 个磁盘。

选择了 `Pool Type` 之后，就会显示可用的磁盘列表，并提示用户选择一个或多个磁盘来组成池。然后，配置会被验证，以确保有足够的磁盘被选中。如果验证失败，选择 **<Change Selection>** 回到磁盘列表，或者 **<Back>** 改变 `Pool Type`。

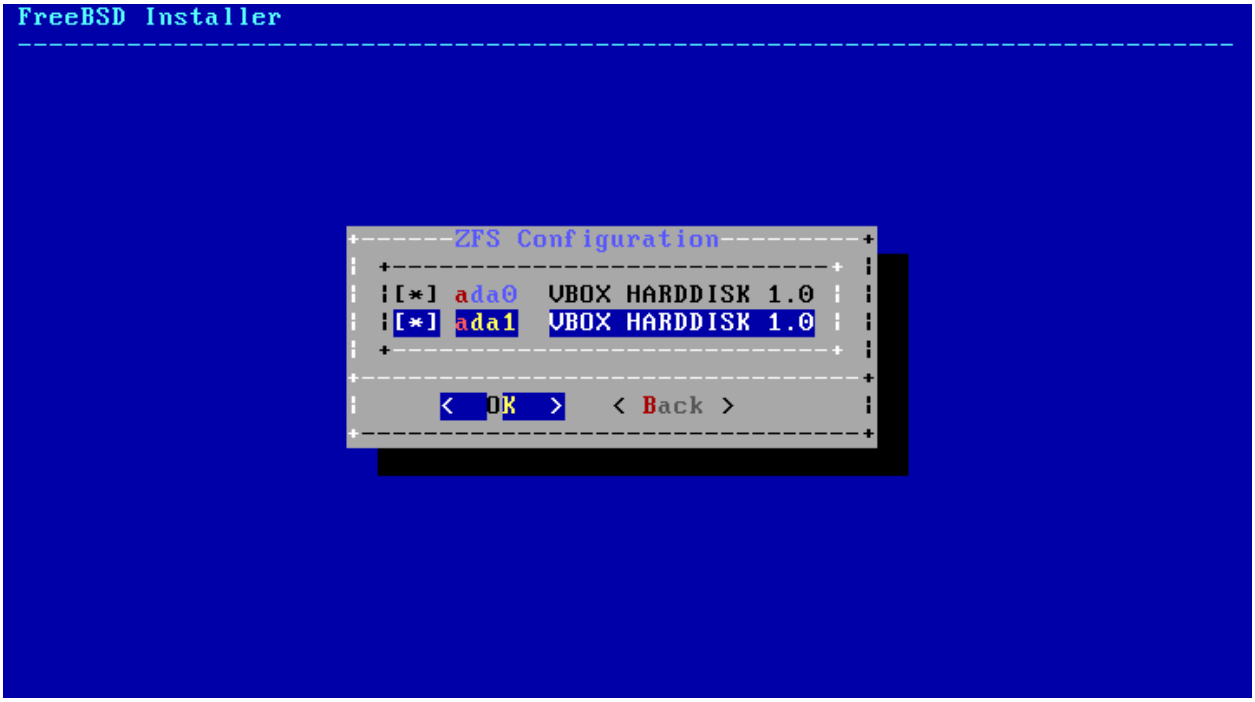


图23: 要求将多少个磁盘添加到池中的菜单

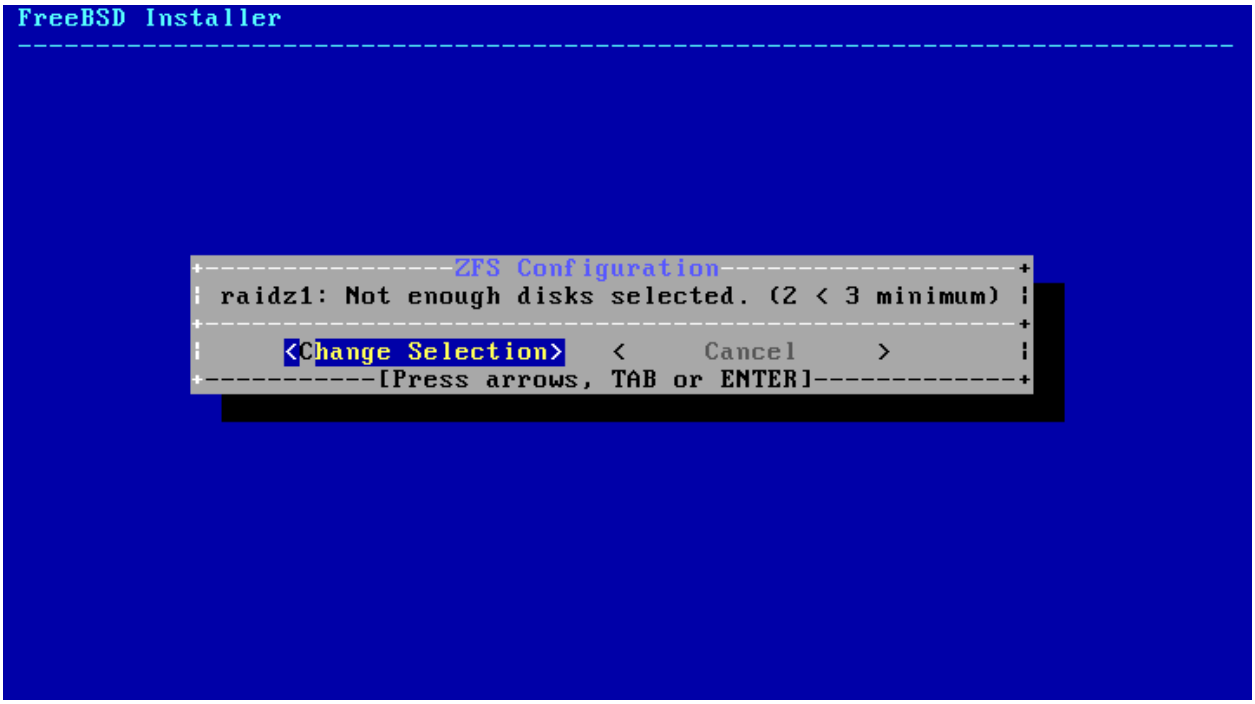
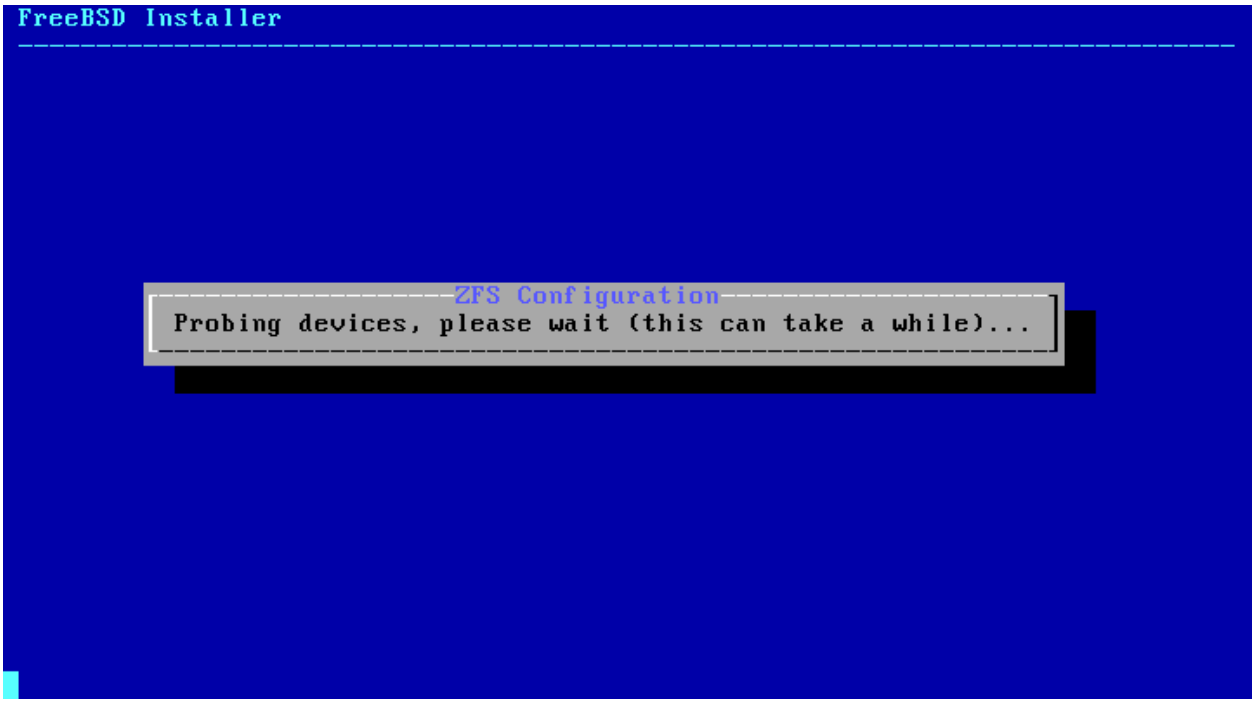


图24: 表示没有选择足够的磁盘的菜单。

如果列表中缺少一个或多个磁盘，或者磁盘是在安装程序启动后连接的，请选择 **- Rescan Devices** 设备来重新加载可用磁盘的列表。



为了避免意外地擦除错误的磁盘，**- Disk Info** 菜单可以用来检查每个磁盘，包括其分区表和其他各种信息，如设备型号和序列号（如果有的话）。

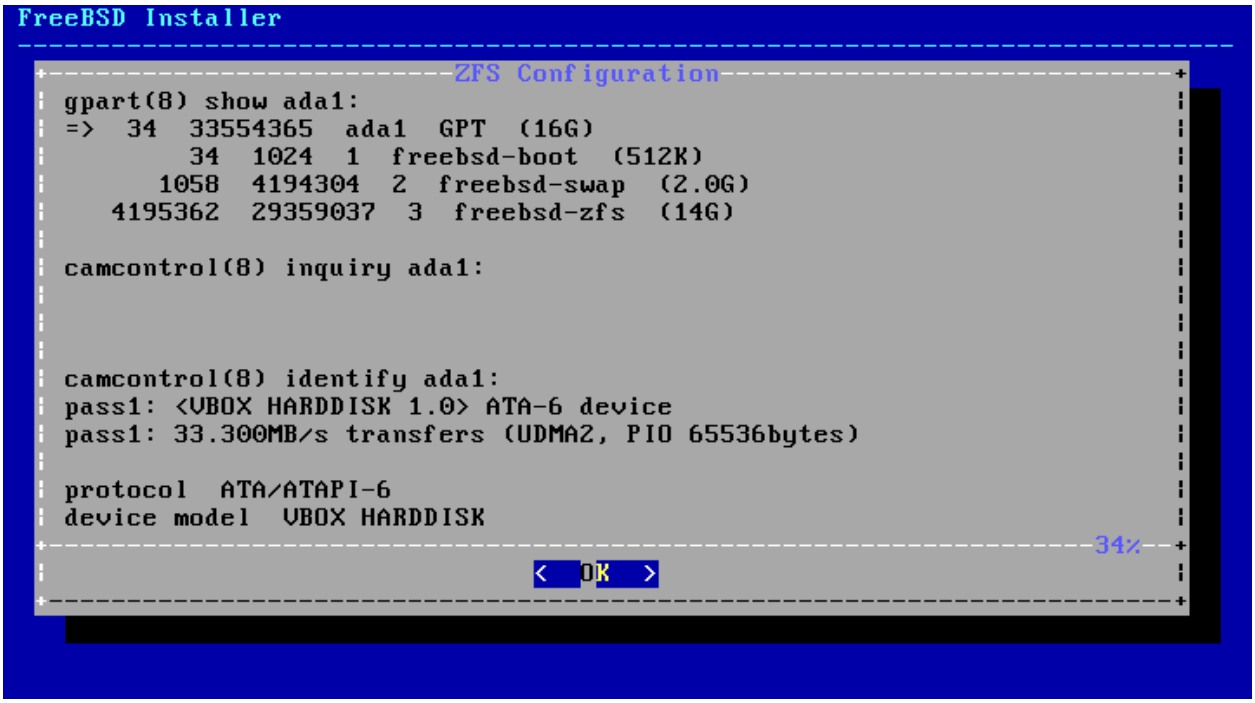


图25: 重新扫描设备

选择 N 可配置 Pool Name。输入所需的名称，然后选择 **<OK>** 创建它，或 **<Cancel>** 返回主菜单并保

留默认名称。

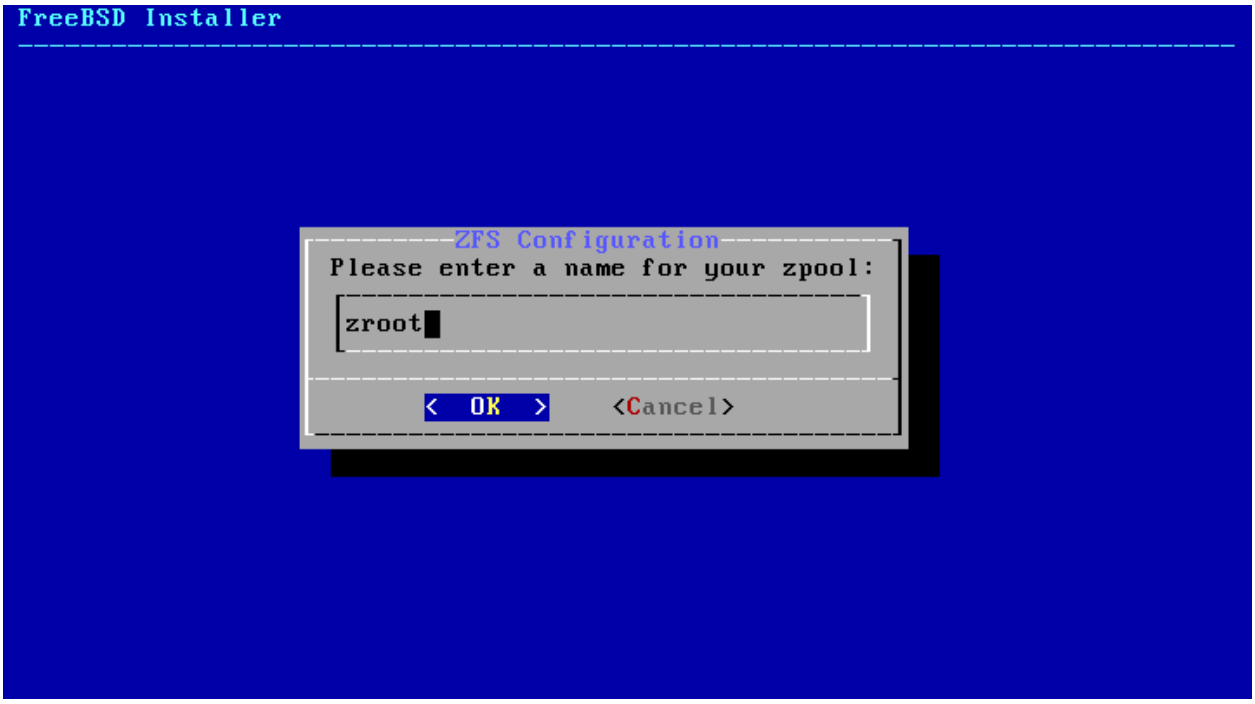


图26: 请求池的名称的菜单

选择 S 来设置交换分区大小。输入所需的交换分区容量，然后选择 **<OK>** 建立，或选择 **<Cancel>** 返回主菜单，让其使用默认值。

图27: 请求交换内存数量的菜单

所有的选项都被设置为所需的值之后，选择菜单顶部的 **>>> Install**。然后安装程序提供了一个最后确认的机会，在所选磁盘的内容被销毁以创建 ZFS 池之前，可以取消。

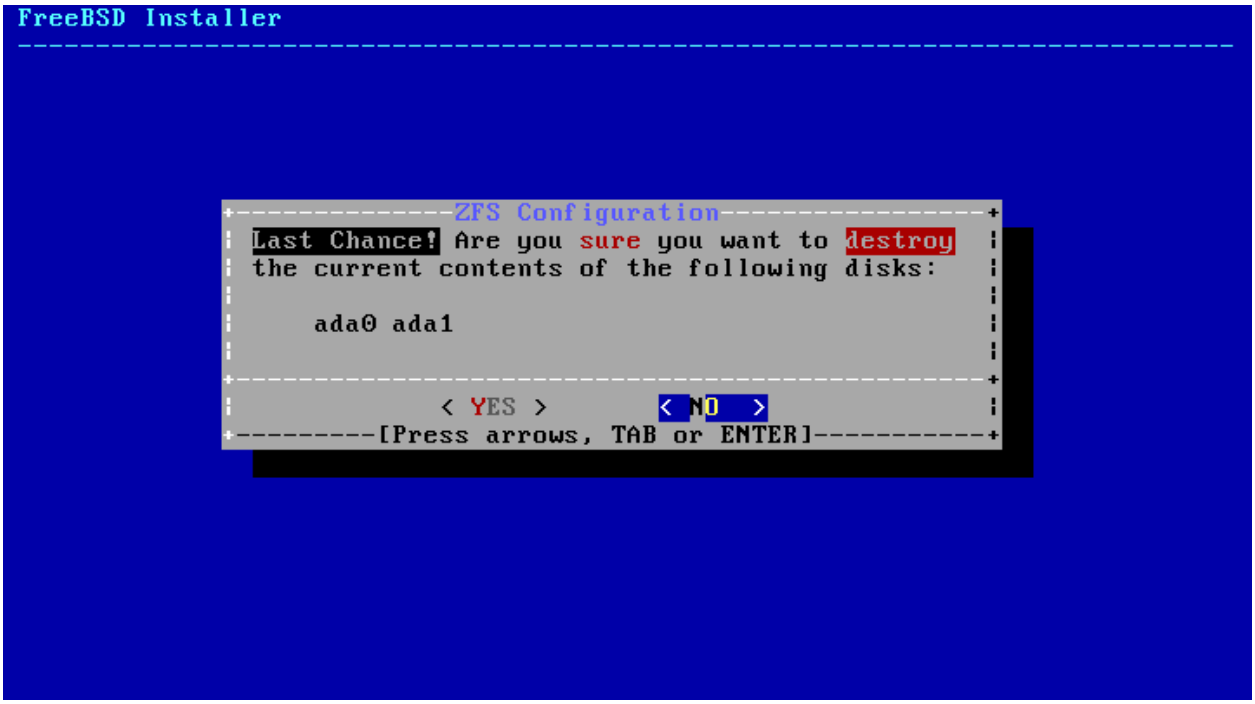


图28: 向用户表明数据将被丢失的菜单

如果启用了 GELI 磁盘加密，安装程序将提示输入用于加密磁盘的口令两次。而后开始进行加密的初始化。

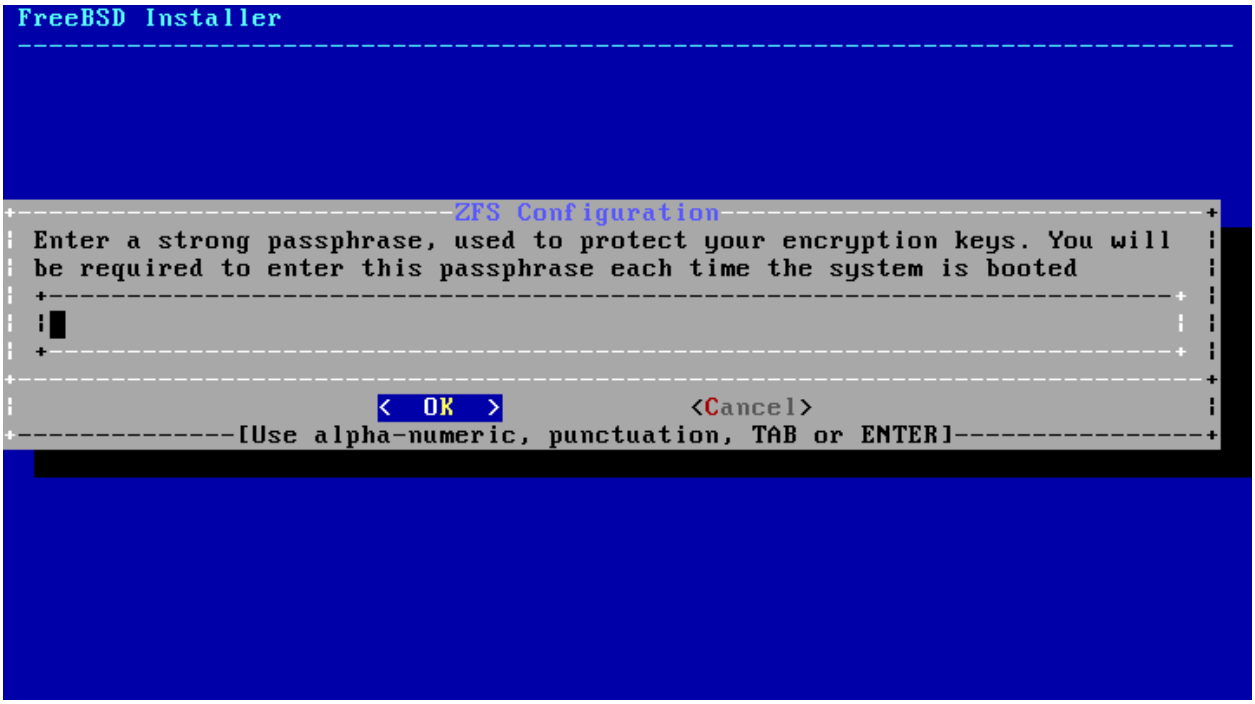


图29: 要求提供密码的菜单，对设备进行加密。

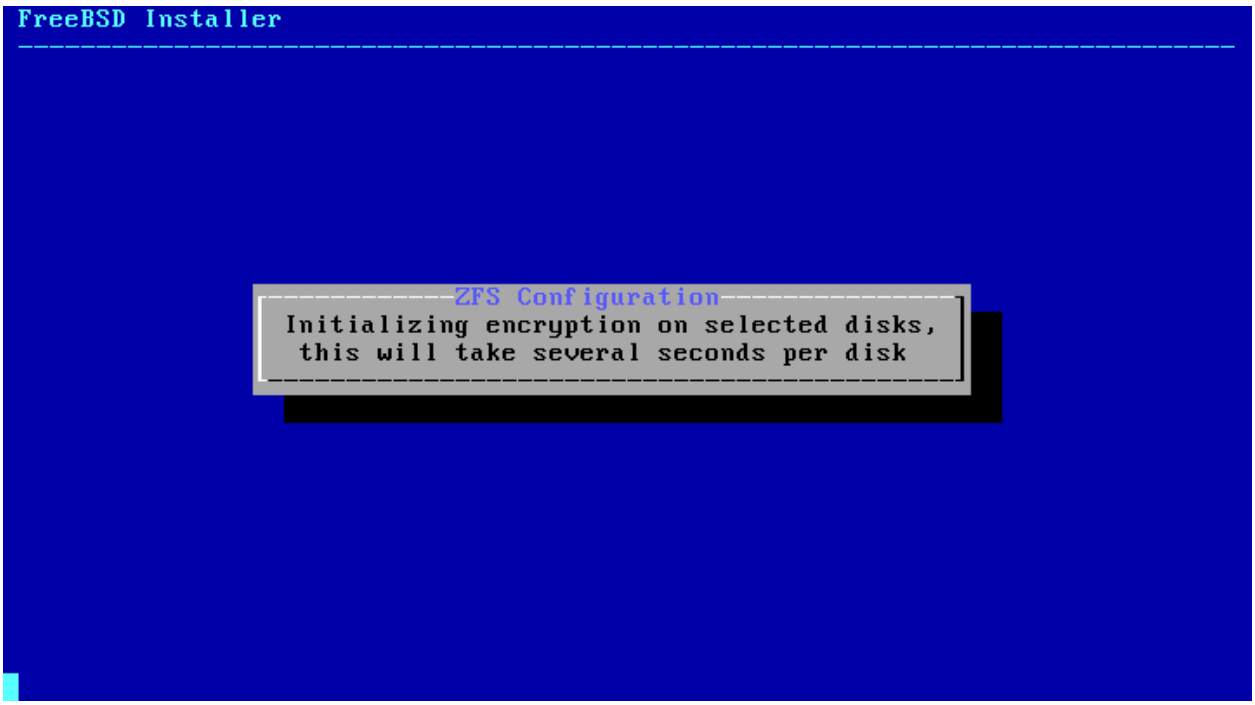


图30: 显示加密正在初始化的菜单。

然后安装会正常进行。要继续安装，请进入获取安装文件[151]。

2.6.5.SHELL 分区模式

在创建高级安装时，bsdinstall 的分区菜单可能无法提供所需的灵活度。专业用户可以从分区菜单中选择 **Shell** 选项，以便手动对磁盘进行分区，创建文件系统，填写 **/tmp/bsdinstall_etc/fstab**，并在 **/mnt** 下加载文件系统。完成后，键入`exit`，返回 bsdinstall，继续安装。

2.7. 获取安装文件

安装时间取决于你所选择的组件、安装设备和计算机的速度。接下来会有一系列的信息来显示当前进度。

首先，安装程序会格式化所选磁盘并初始化分区。接下来，若使用的是 `bootonly media` 或 `mini memstick`，它会下载所选的组件：

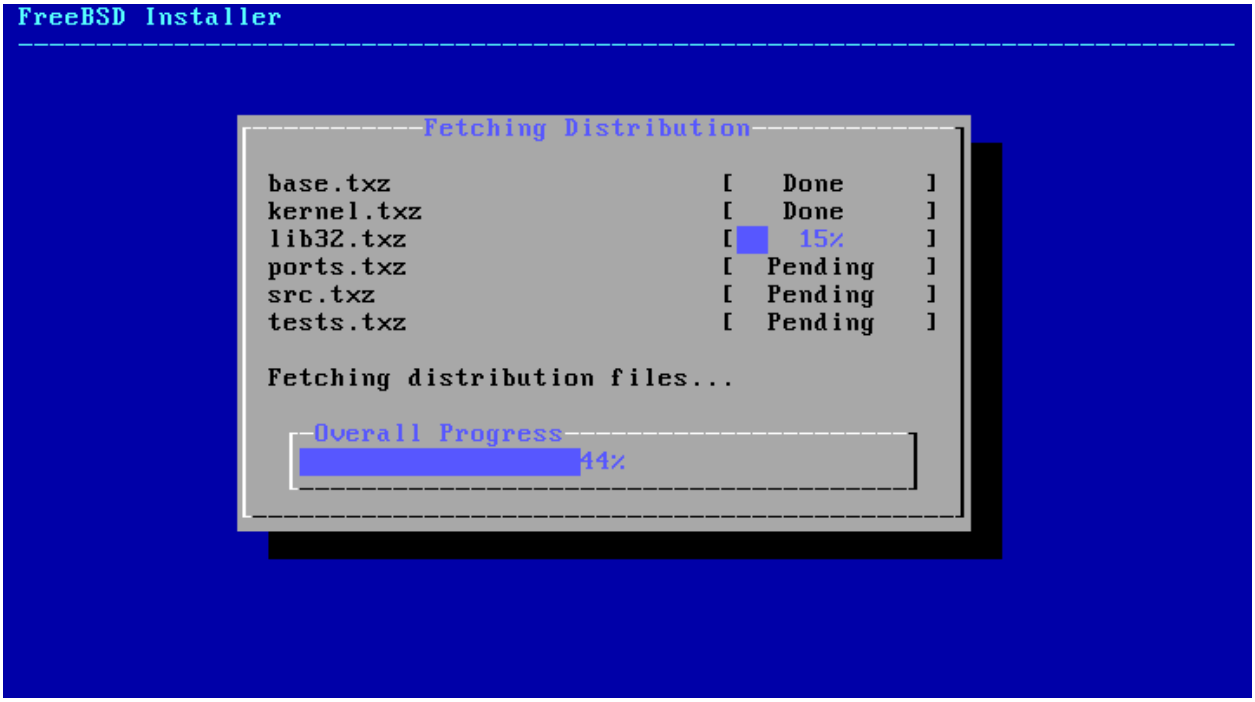


图31: 获取安装文件

接下来，将校验安装文件，以确保它们在下载过程中没有被破坏或从安装介质中误读：

[151] https://docs.freebsd.org/en/books/handbook/bsdinstall/#bsdinstall-fetching-distribution

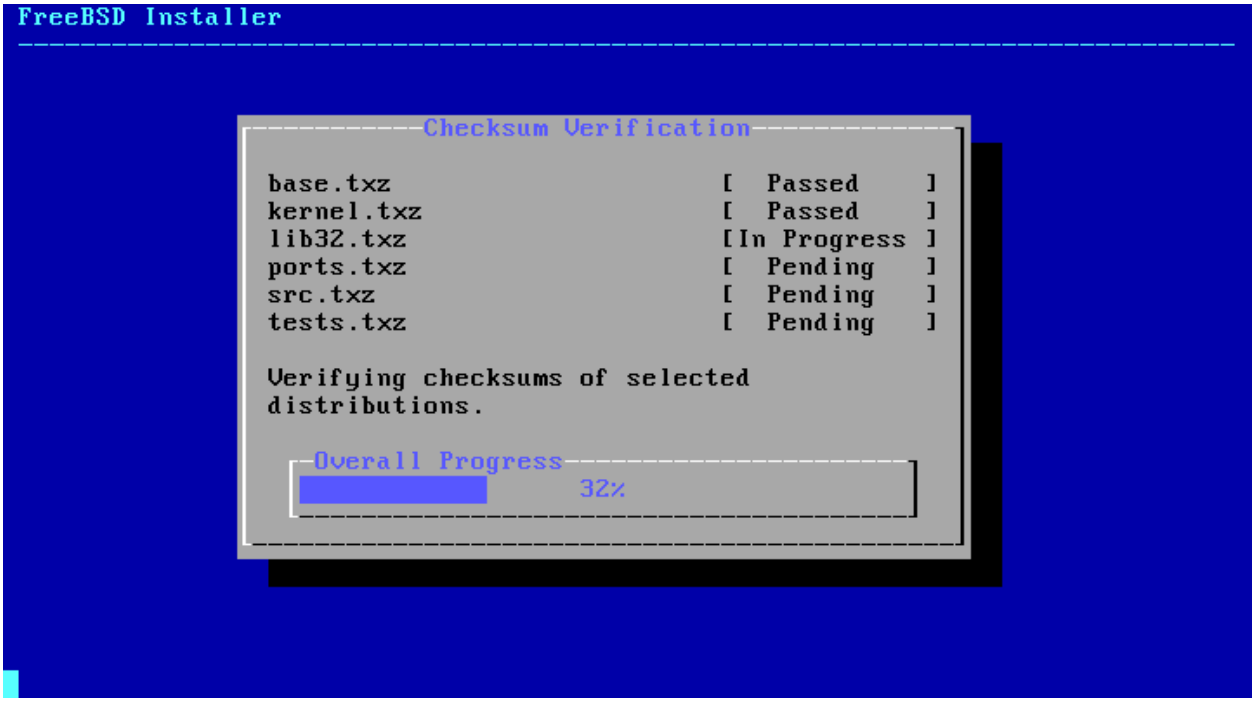


图32: 校验安装文件

最后，经过校验的安装文件被解压到磁盘：

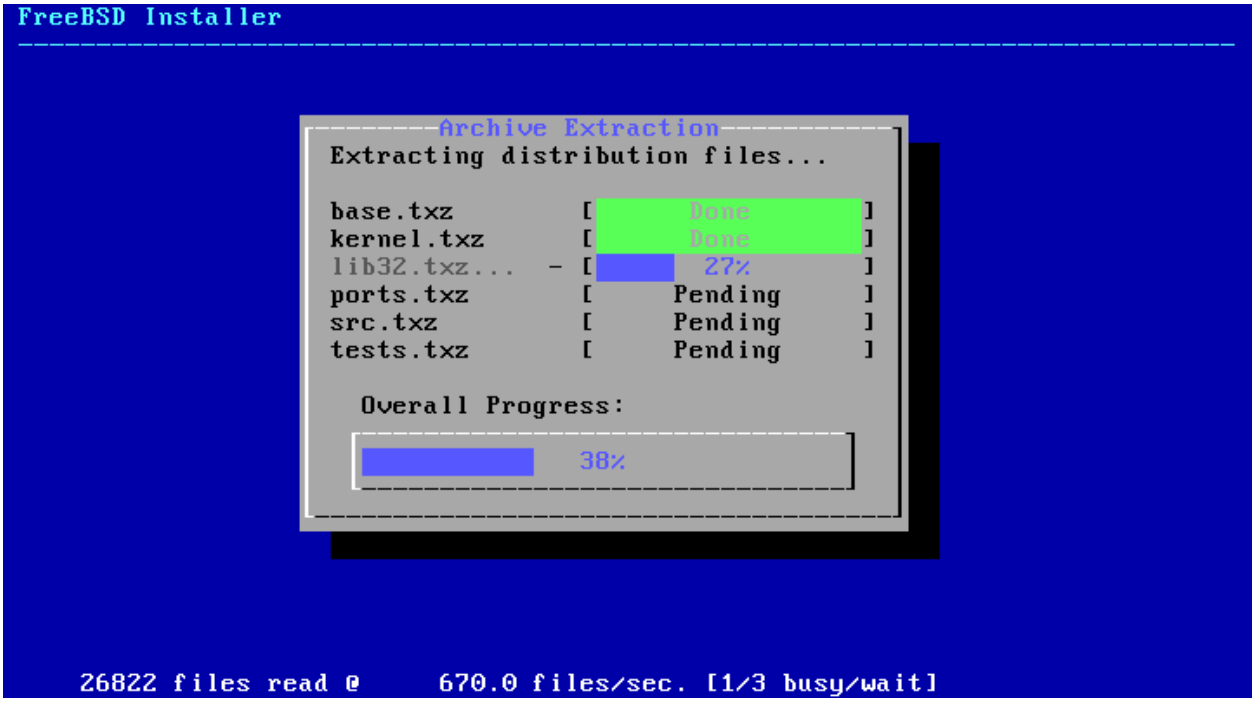


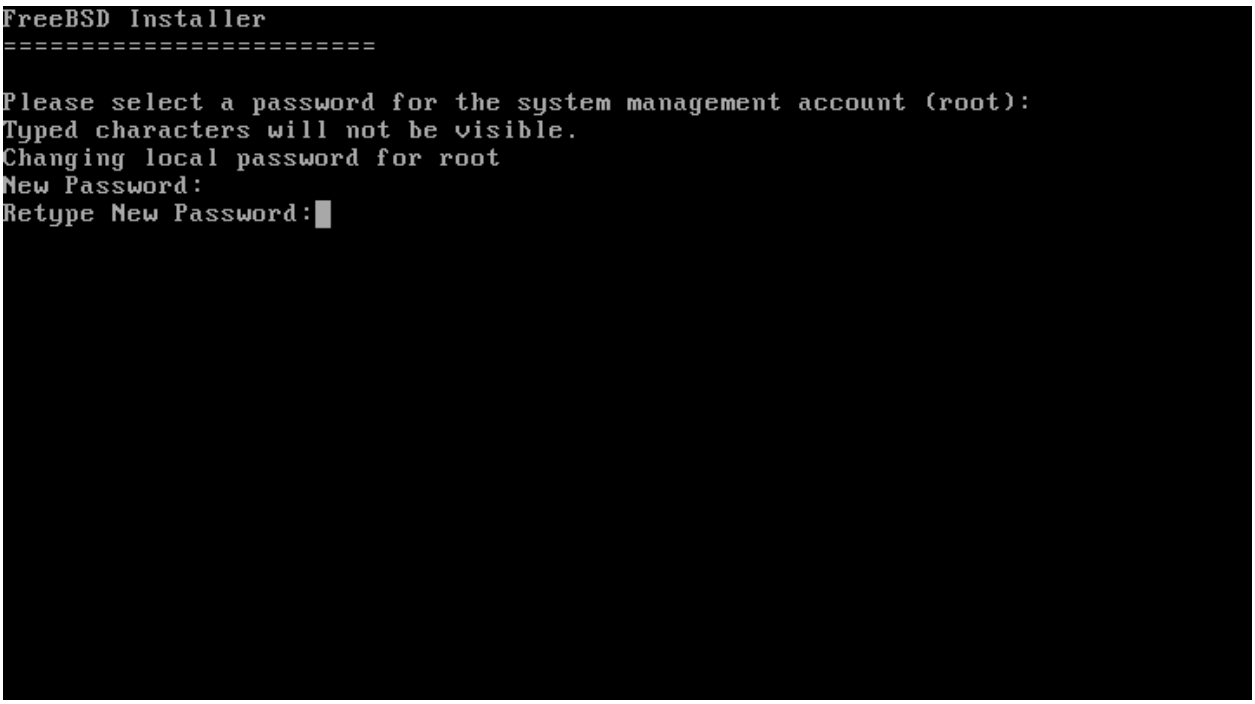
图33: 解压安装文件

所有所需的安装文件都被解压缩后，bsdinstall 会显示一个安装后的配置界面。可用的安装后配置选项将在下一节进行说明。

2.8.网络、账户、时区、服务和安全

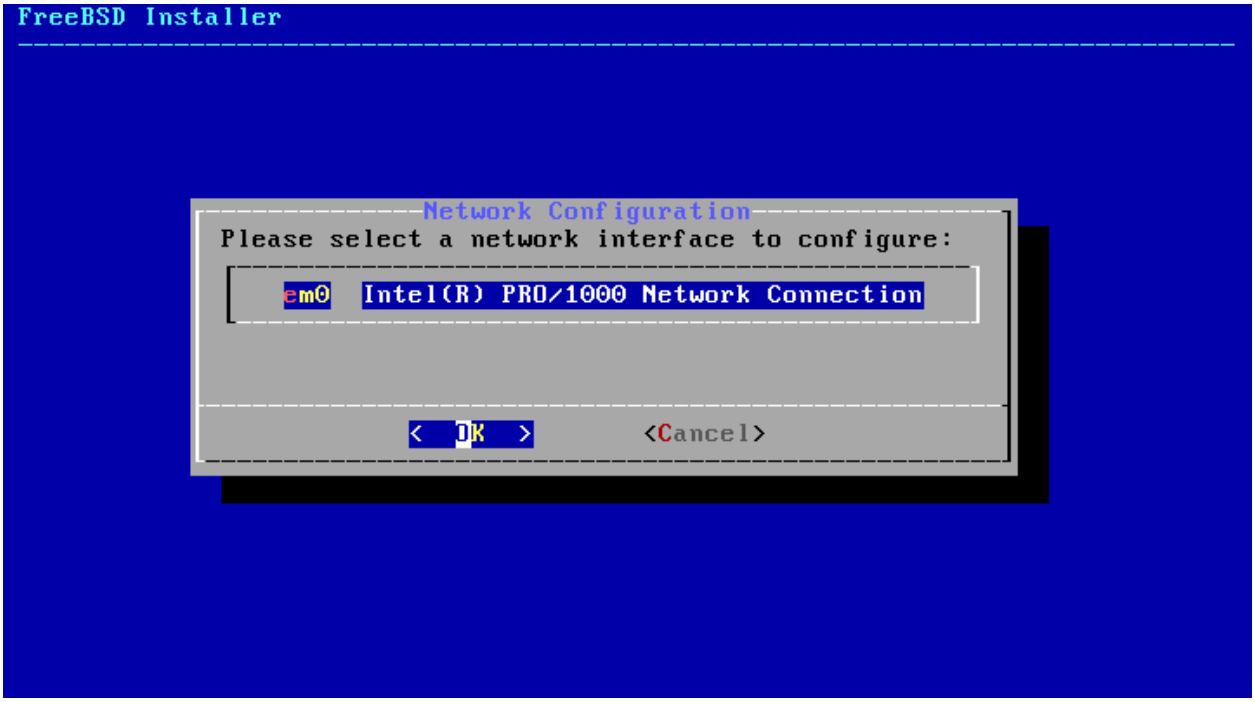
2.8.1.设置 root 密码

首先，必须设置 root 密码。在输入密码时，正在输入的字符并不会显示在屏幕上。密码必须输入两次，以防止输入错误。



2.8.2.网络设置

接下来，会显示计算机上检测到的网卡列表。选择要配置的网卡。



如果选择了以太网卡，安装程序将跳到选择 IPv4 网络中显示的菜单。如果选择了无线网络接口，系统将转而扫描无线接入点。

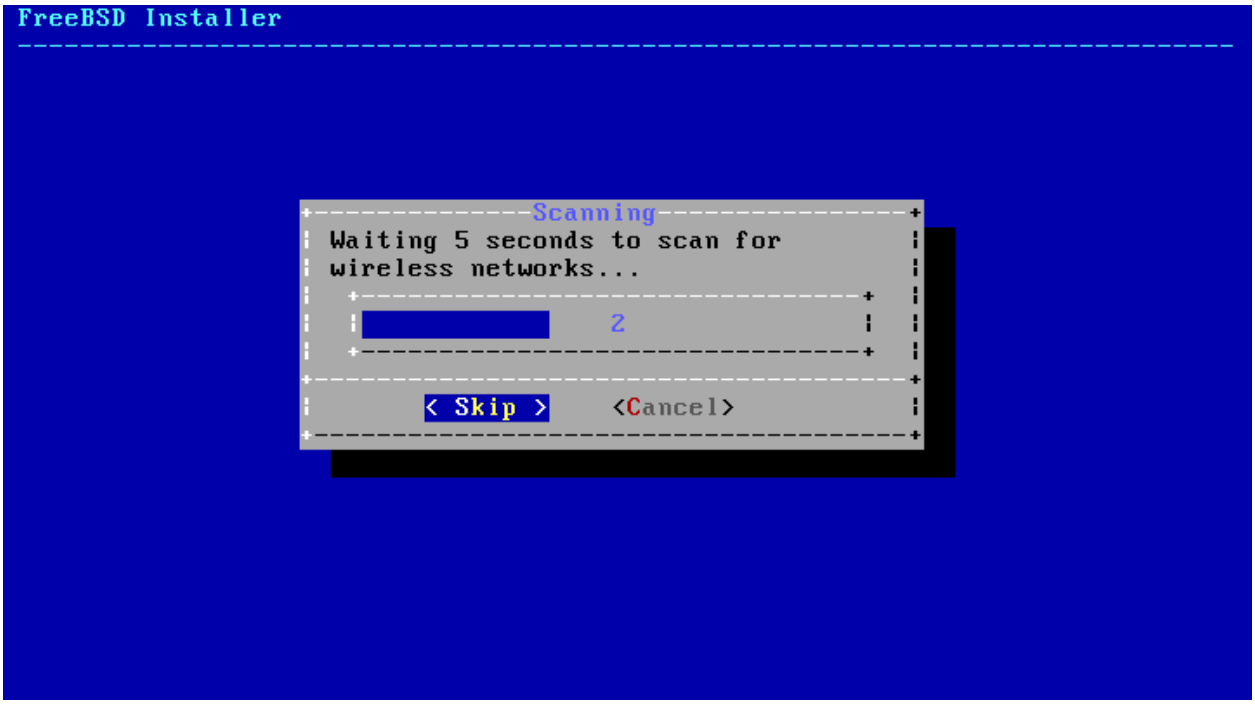


图34: 扫描无线接入点

无线网络是由服务集标识（SSID）来识别的；这是给予每个网络的简短、独特的名称。在扫描过程中发现的 SSID 会被列出，然后是对该网络可用的加密类型的描述。如果所需的 SSID 没有出现在列表中，选择

Rescan 以再次扫描。如果想要的网络仍然没有出现，请检查天线连接是否有问题，或者尝试将电脑移到离接入点更近的地方。每次改变后都要 Rescan。

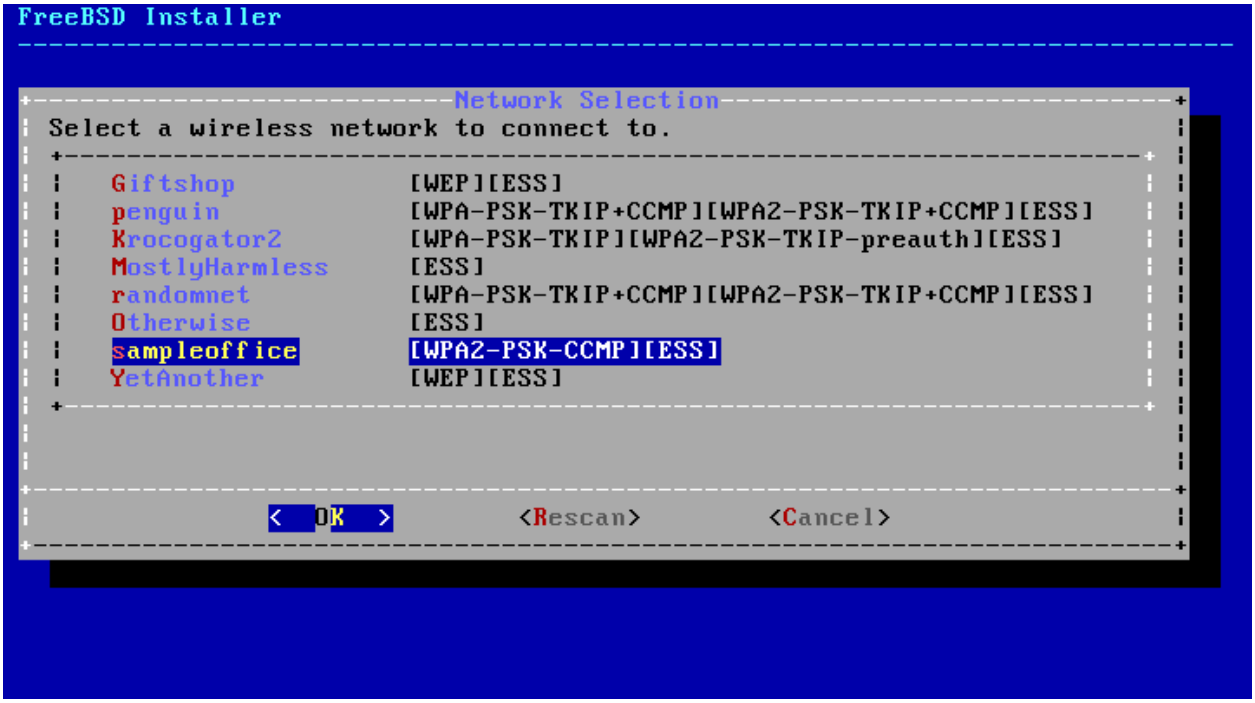


图35: 选择无线网络

接下来，输入连接到所选无线网络的加密信息。强烈建议使用 WPA2 加密，而不是旧的加密类型（如 WEP），因为后者提供的安全性不高。如果网络使用 WPA2，请输入密码，也被称为预共享密钥（PSK）。出于安全考虑，输入框中的字符会显示为星号。

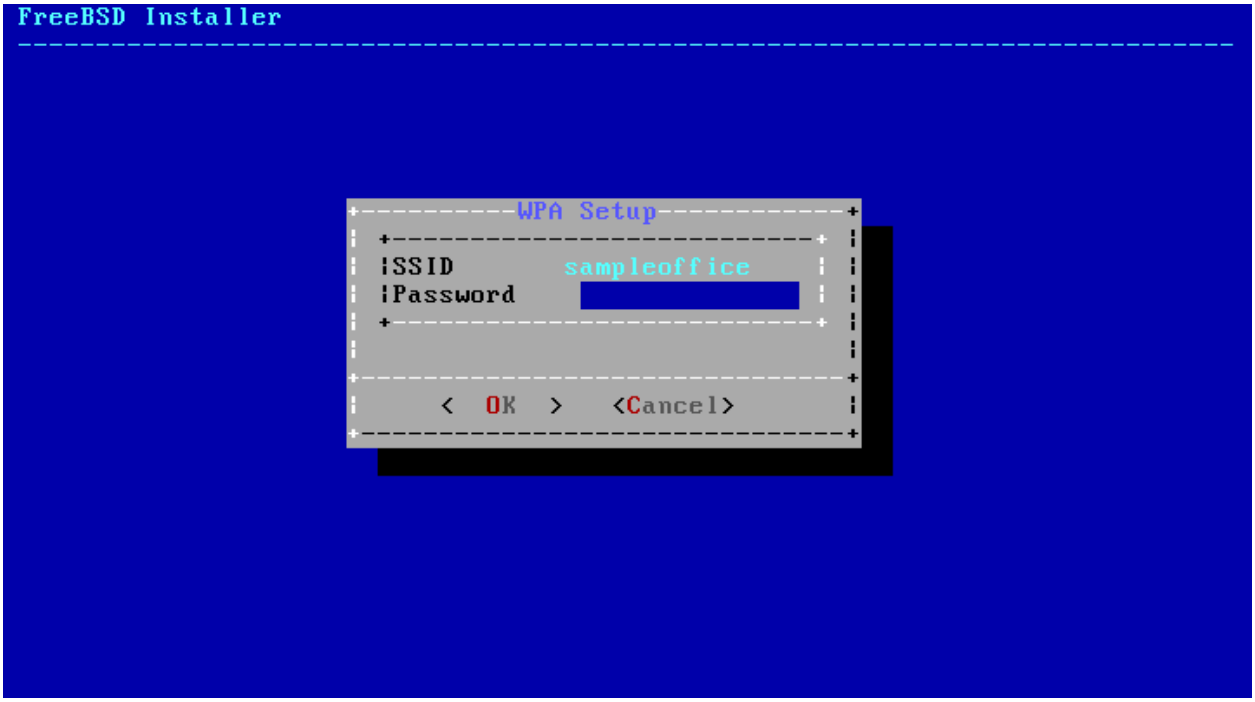


图36: WPA2 设置

接下来，选择是否要在以太网卡或无线卡上配置 IPv4 地址：

图37: 选择 IPv4 网络

有两种方法可以配置 IPv4。DHCP 会自动正确配置网络，如果网络中提供了 DHCP 服务器，就应该使用这种方法。否则，需要以静态配置的方式手动输入寻址信息。

注意

不要随意输入网络信息，因为这没用。如果没有 DHCP 服务器，请从网络管理员或互联网服务提供商那里获得所需网络信息[152]中列出的信息。

如果有 DHCP 服务器，在下一个菜单中选择 **Yes** 来自动配置网络。当安装程序找到 DHCP 服务器并获得系统的寻址信息时，会出现一分钟左右的停顿。

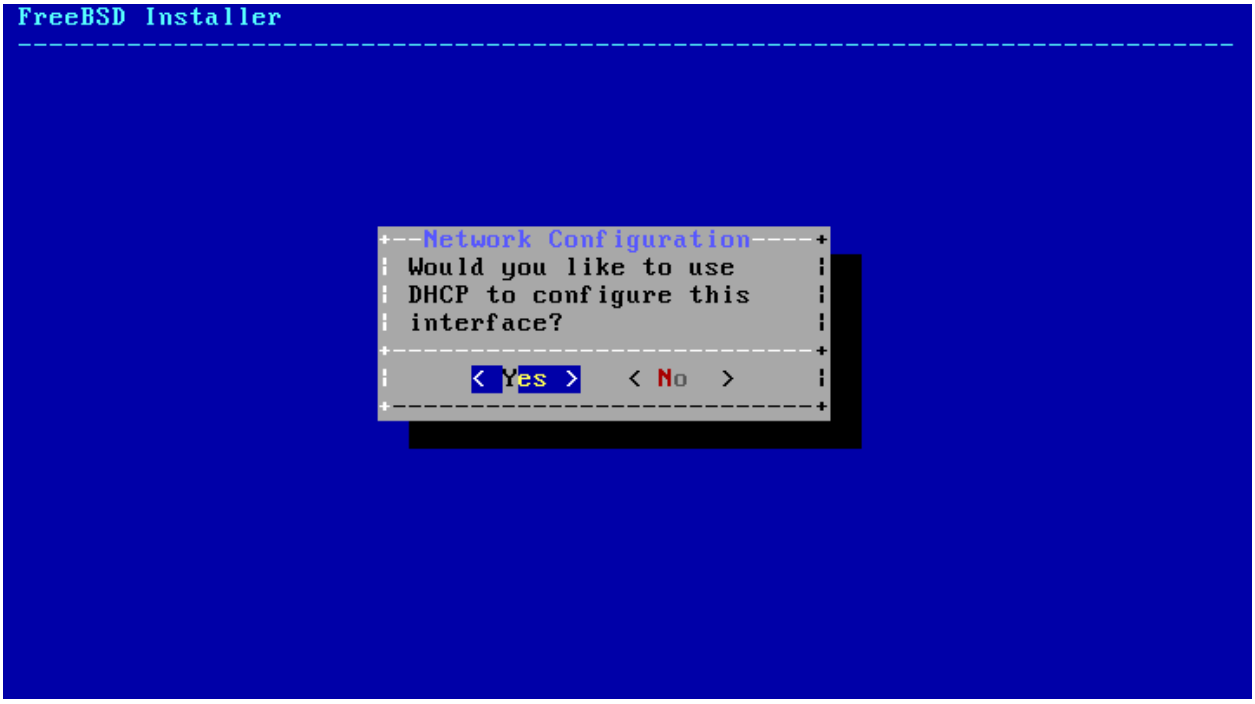


图38: 选择IPv4 DHCP配置

如果没有 DHCP 服务器，请选择 **No**，并在此菜单中输入以下寻址信息：

[152] https://docs.freebsd.org/en/books/handbook/book/#bsdinstall-collect-network-information

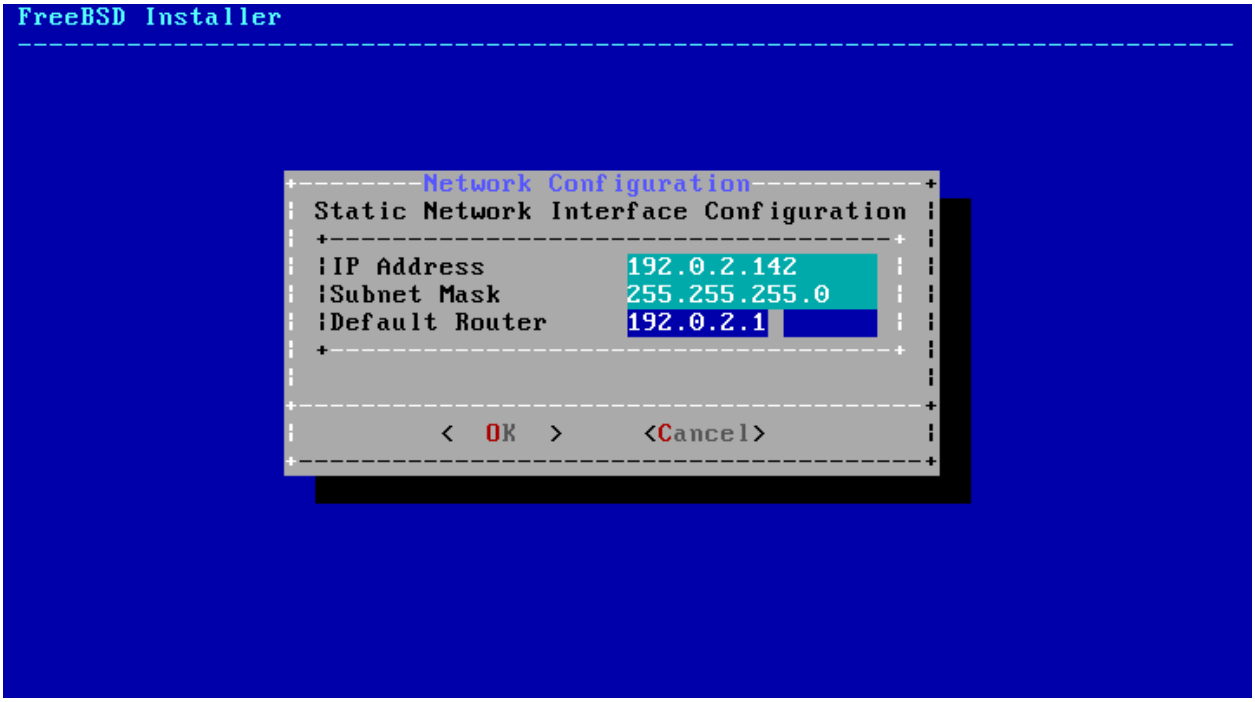


图39: 静态 IPv4 配置

- `IP Address` - 分配给该计算机的 IPv4 地址。该地址必须是唯一的，并且没有被本地网络中的其他设备使用。
- `Subnet Mask` - 网络的子网掩码。
- `Default Router` - 网络的默认网关的 IP 地址。

下一屏将询问该接口是否应配置为 IPv6。如果 IPv6 可用并且需要，选择 **Yes** 来选择它。

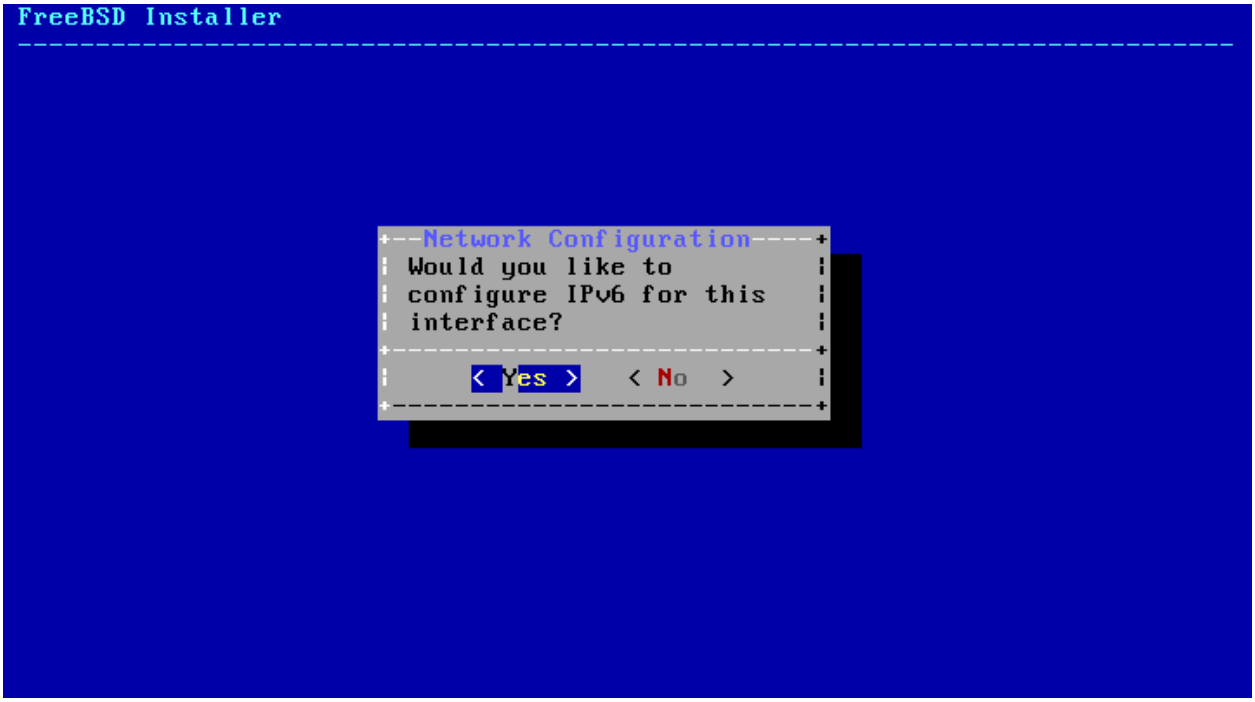


图40: 选择 IPv6 网络

IPv6 也有两种配置方法。无状态地址自动配置（SLAAC）将自动向本地路由器请求正确的配置信息。更多信息请参考 rfc4862[153]。静态配置需要手动输入网络信息。

如果有 IPv6 路由器，请在下一个菜单中选择 **Yes**，以自动配置网络接口。安装程序在找到路由器并获得系统的寻址信息时，会出现一分钟左右的停顿。

[153] http://tools.ietf.org/html/rfc4862

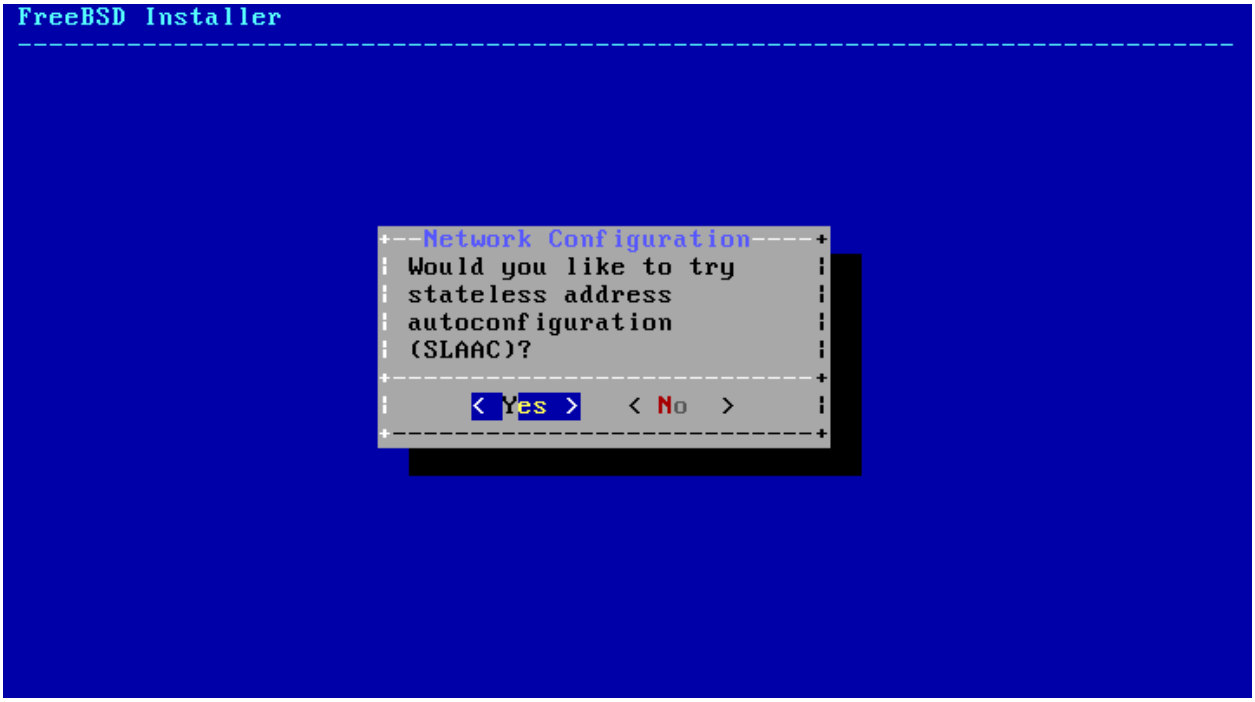


图41: 选择 IPv6 SLAAC 配置

如果没有 IPv6 路由器，请选择 **No** 并在该菜单中输入以下寻址信息。

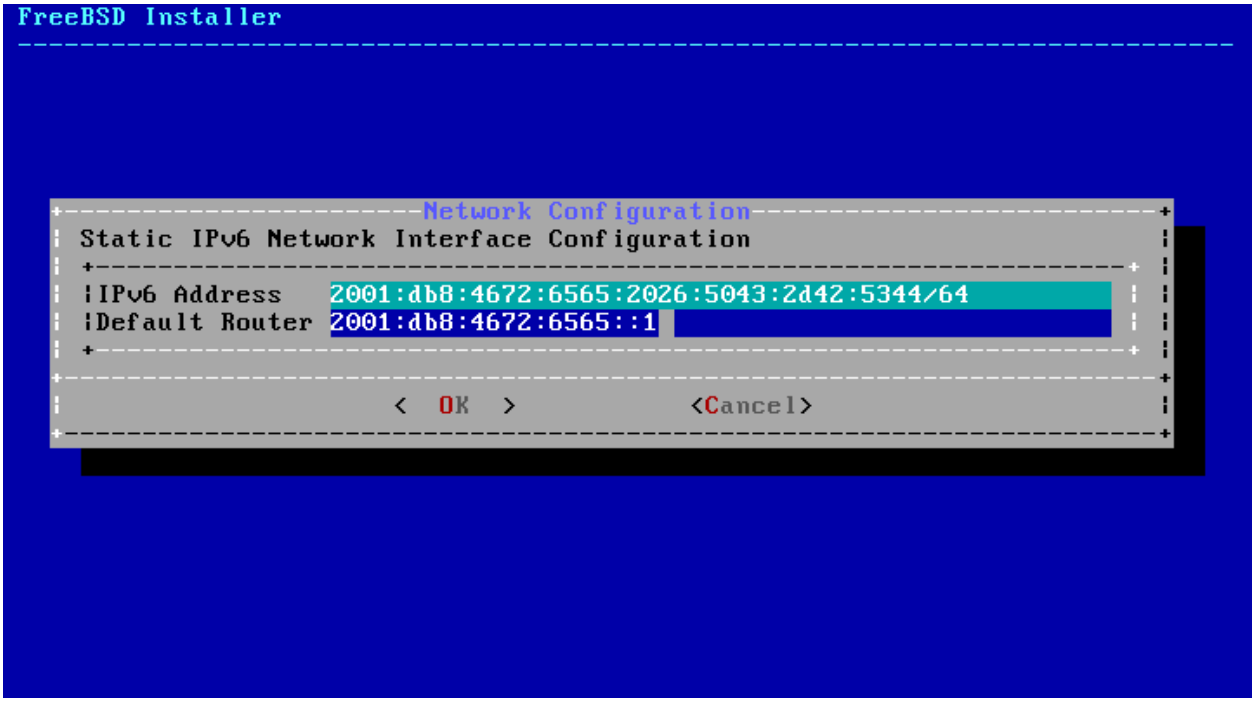


图42: 静态 IPv6 配置

- `IPv6 Address` - 分配给该计算机的 IPv6 地址。该地址必须是唯一的，并且没有被本地网络中的其他

设备使用。

- `Default Router` - 网络的默认网关的 IPv6 地址。

最后一个网络配置菜单用于配置域名系统（DNS）解析器，它将主机名转换为网络地址。如果使用了 DHCP 或 SLAAC 来自动配置网络，那么 `Resolver Configuration` 的值可能已经被填入。若没有，在 `Search` 中输入本地网络的域名。`DNS #1` 和 `DNS #2` 是 DNS 服务器的 IPv4 / IPv6 地址。至少需要一个 DNS 服务器。

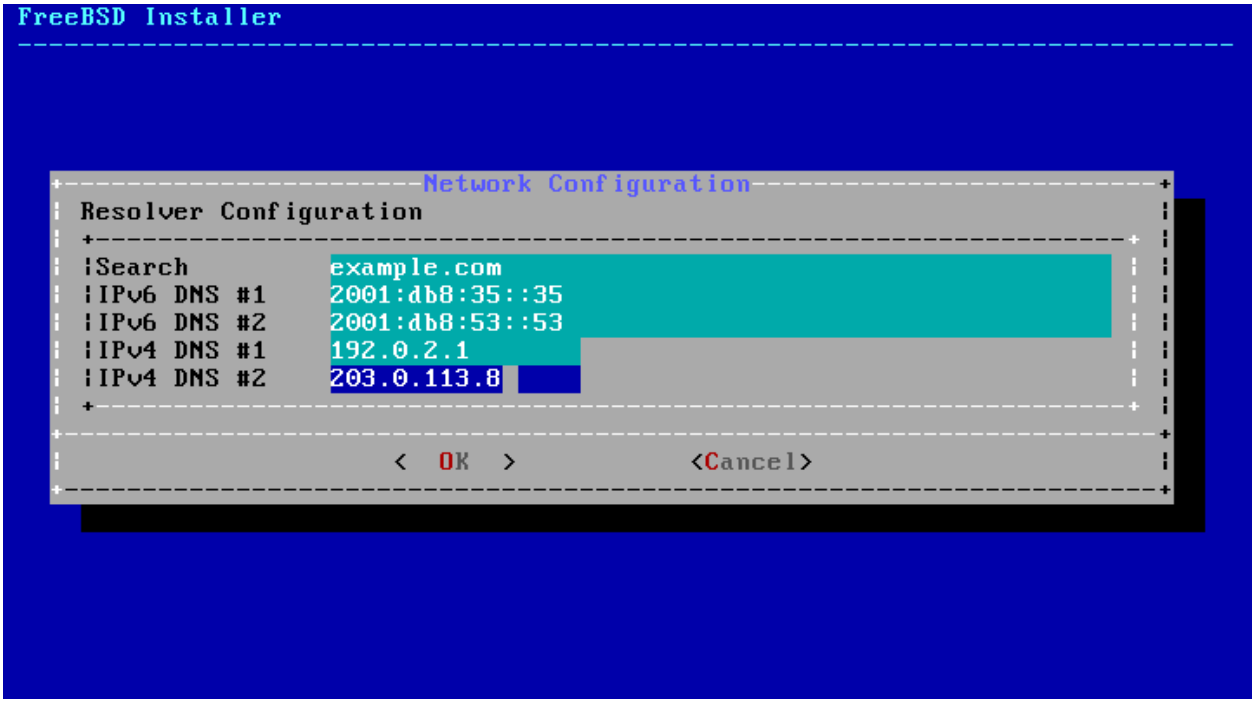


图43: DNS配置

配置好界面后，选择一个与安装 FreeBSD 的计算机位于同一地区的镜像站点。当镜像站点靠近目标计算机时，可以更快地检索到文件，从而减少安装时间。

图44: 选择镜像站

2.8.3.设置时区

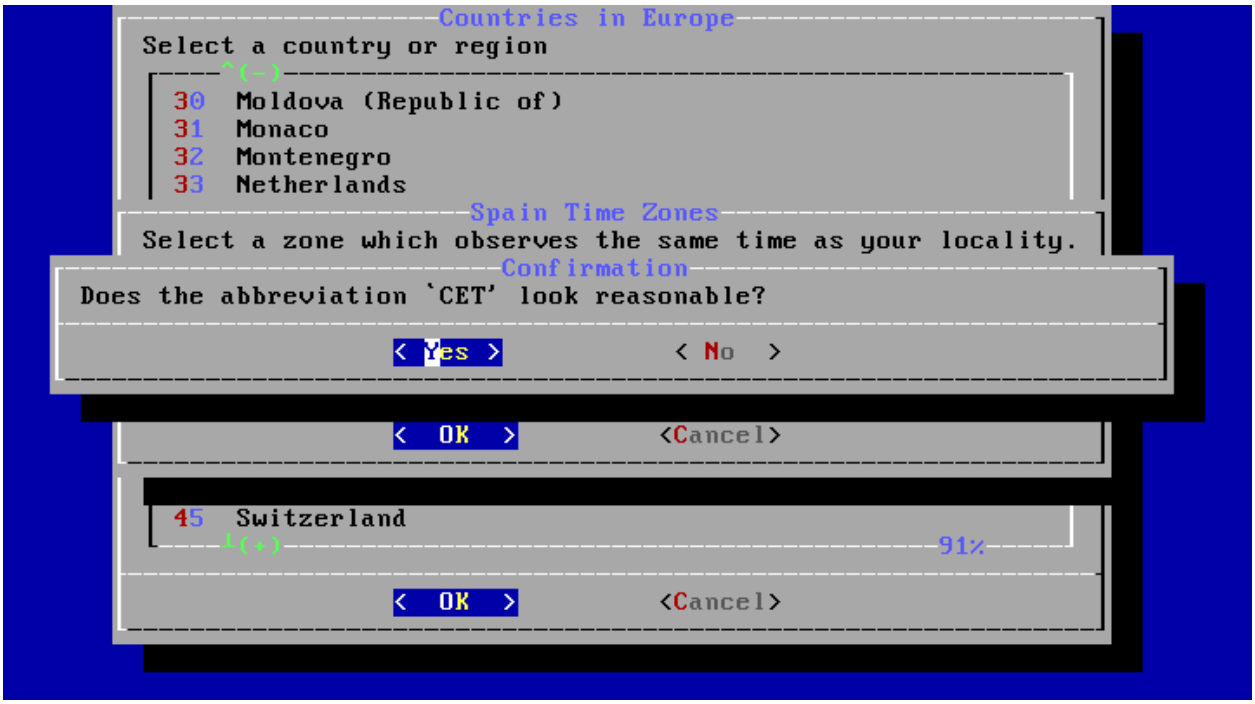
接下来的一系列菜单用于通过选择地理区域、国家和时区来确定正确的本地时间。设置时区可以使系统自动纠正地区时间变化，如夏令时，并正确执行其他与时区有关的功能。

这里显示的例子是针对位于欧洲西班牙大陆时区的机器。根据地理位置的不同，选择也会有所不同。

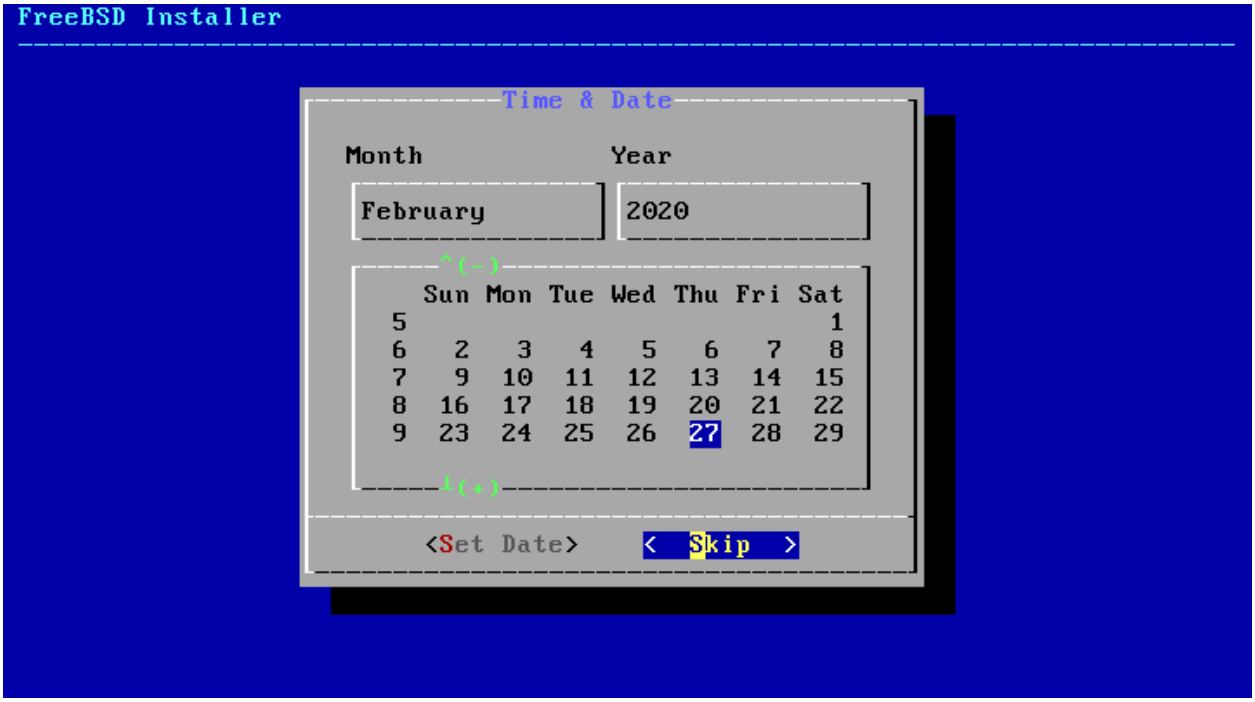
使用方向键选择适当的区域，然后按回车键。

用方向键选择适当的国家，然后按回车键。

使用箭头键选择适当的时区并按下回车键。



确认时区的缩写是正确的。

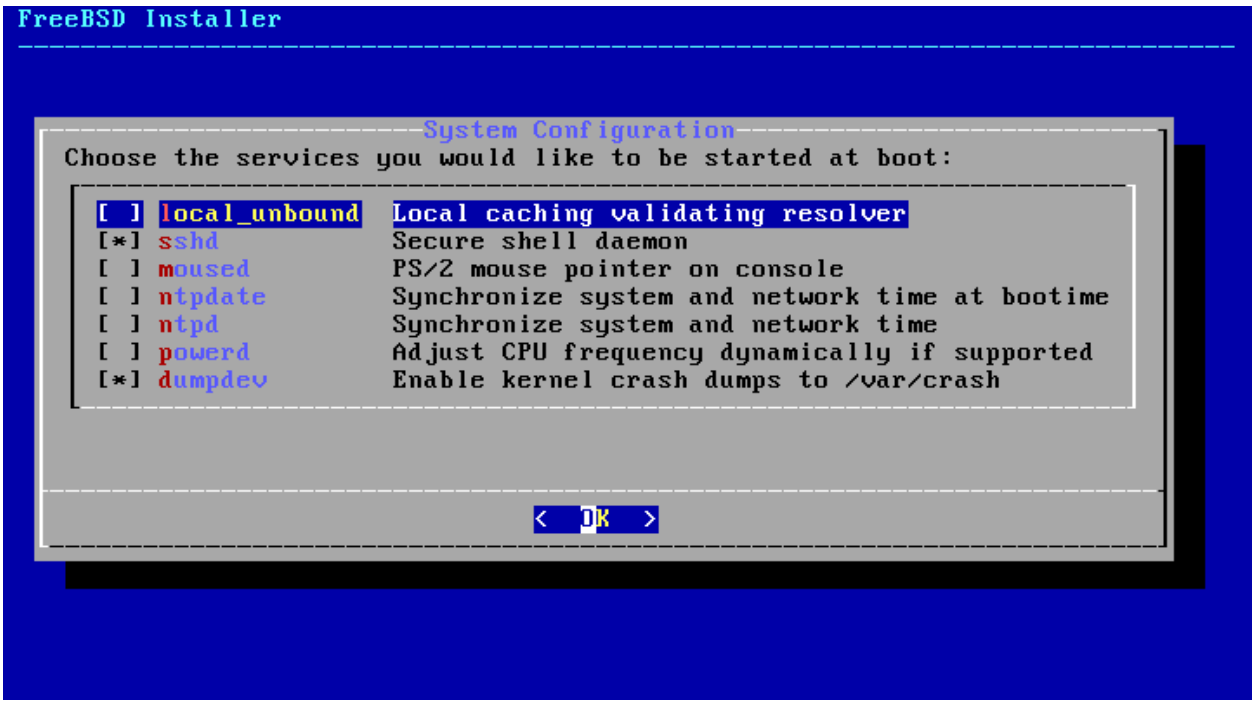


使用方向键选择适当的日期，然后按 **Set Date**。否则，可以通过按 **Skip** 来跳过日期的选择。

使用方向键选择适当的时间，然后按 **Set Time**。否则，可以通过按 **Skip** 来跳过时间的选择。

2.8.4.开启服务

接下来的菜单用于配置哪些服务会开机自启。所有这些服务都是可选的。只需启动系统运行所需的服务。



这里是在这个菜单中可以启用的服务摘要：

- `local_unbound`——启用本地 DNS 的 unbound。请牢记，这是一个仅用于作为本地缓存转发解析器的配置。如果目标是为整个网络建立解析器，请安装 dns/unbound[154]。
- `sshd`——Secure Shell（SSH）守护程序用于经过加密的连接远程访问系统。只有在系统允许远程登录时才启用这个服务。
- `moused`——如果要在命令行系统控制台使用鼠标，则启用该服务。
- `ntpdate`——启用开机时的自动时钟同步功能。注意这个程序的功能现在在守护进程 ntpd(8)[155] 中可用，ntpdate(8)[156] 工具将很快退役。
- `ntpd`——用于自动时钟同步的网络时间协议（NTP）的守护进程。如果你希望将你的系统时钟与远程时间服务器或池同步，请启用这项服务。
- `powerd`——系统的电源控制工具，用于电源控制和节能。
- `dumpdev`——崩溃转储在调试系统故障时很有用，所以推荐用户启用它。

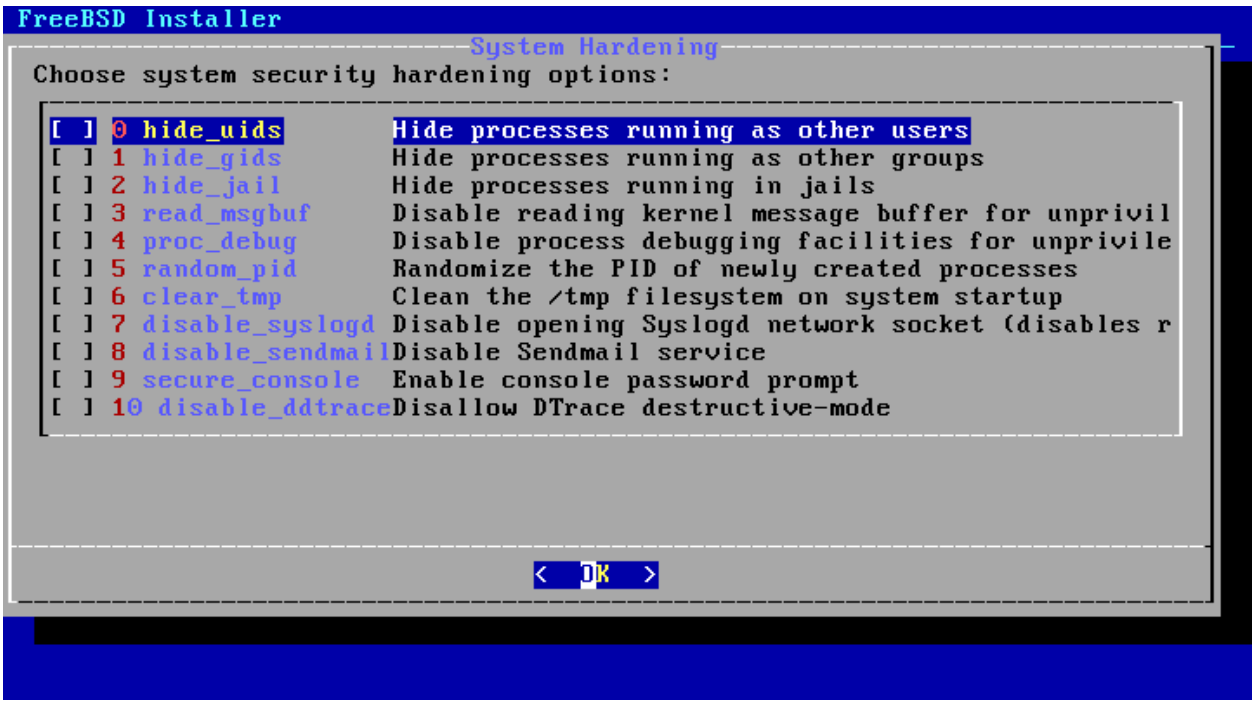
[154] https://cgit.freebsd.org/ports/tree/dns/unbound/pkg-descr

[155] https://www.freebsd.org/cgi/man.cgi?query=ntpd&sektion=8&format=html

[156] https://www.freebsd.org/cgi/man.cgi?query=ntpdate&sektion=8&format=html

2.8.4.启用安全加固选项

接下来的菜单是用来配置启用安全选项。所有这些选项都是可选的。但我们推荐开启它们。



下面是这个菜单中可以启用的选项的摘要：

- `hide_uids`——隐藏以其他用户身份运行的进程（UID）。这可以防止没有特权的用户看到其他用户的运行进程。
- `hide_gids`——隐藏以其他组的名义运行的进程（GID）。这可以防止没有特权的用户看到其他组的运行进程。
- `hide_jail`——隐藏在 jail 中运行的进程。这可以防止没有特权的用户看到在 jail 内运行的进程。
- `read_msgbuf`——禁止非特权用户使用 dmesg(8)[157] 查看内核日志缓冲区的信息，以防止读取内核缓冲区中的信息。
- `proc_debug`——禁用非特权用户的进程调试功能，禁用各种非特权的进程间调试服务，包括一些 procfs 功能、`trace()` 和 `ktrace()`。请注意，这也会妨碍调试工具，例如 lldb(1)[158]、truss(1)[159]、procstat(1)[160]，以及某些脚本语言（如 PHP）中的一些内置调试功能。
- `random_pid`——进程 PID 随机化。
- `clear_tmp`——在系统启动时清理 **/tmp**。

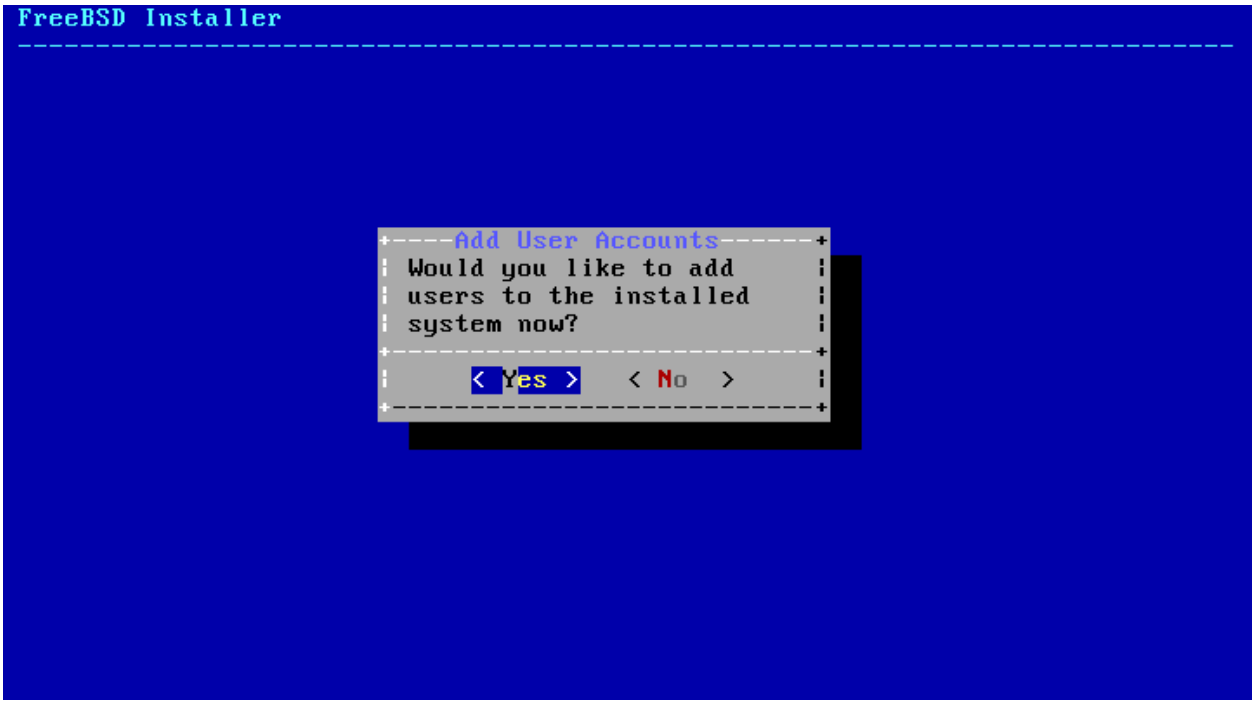
[157] https://www.freebsd.org/cgi/man.cgi?query=dmesg&sektion=8&format=html
[158] https://www.freebsd.org/cgi/man.cgi?query=lldb&sektion=1&format=html
[159] https://www.freebsd.org/cgi/man.cgi?query=truss&sektion=1&format=html
[160] https://www.freebsd.org/cgi/man.cgi?query=procstat&sektion=1&format=html

- `disable_syslogd`——禁止打开 syslogd 网络套接字。在默认情况下，FreeBSD 以安全的方式使用 `-s` 参数运行 syslogd。这可以防止守护进程在 514 端口监听传入的 UDP 请求。启用该选项后，syslogd 将以标志 `-ss` 运行，它可以阻止 syslogd 打开任何端口。要获得更多信息，请参考 syslogd(8)[161]。
- `disable_sendmail`——禁用 sendmail 邮件传输代理。
- `secure_console`——在进入单用户模式时，命令提示符会要求输入 `root` 密码。
- `disable_ddtrace`——Dtrace 在某些运行模式下可能会对内核的正常运行造成影响。除非用户明确启用，否则不得使用该破坏性操作。要在使用 DTrace 时启用该选项，请使用参数 `-w`。如需更多信息，请查阅 dtrace(1)[162]。
- `enable_aslr` -启用地址空间布局随机化。关于地址空间布局随机化的更多信息，可以参考维基百科的文章[163]。

2.8.5.添加用户

接下来的菜单提示至少要创建一个用户账户。建议使用非 `root` 的用户账户身份登录系统。当以 `root` 身份登录时，基本上任何事都没有限制或保护。以普通用户身份登录更安全、更有保障。

选择 **Yes** 来添加新用户。



[161] https://www.freebsd.org/cgi/man.cgi?query=syslogd&sektion=8&format=html

[162] https://www.freebsd.org/cgi/man.cgi?query=dtrace&sektion=1&format=html

[163] https://en.wikipedia.org/wiki/Address_space_layout_randomization

按照提示，输入所要求的用户账户信息。输入用户信息[164]中显示的例子创建了一个名为 asample 的账户。

```
FreeBSD Installer
=======================
Add Users

Username: asample
Full name: Arthur Sample
Uid (Leave empty for default):
Login group [asample]:
Login group is asample. Invite asample into other groups? []: wheel
Login class [default]:
Shell (sh csh tcsh nologin) [sh]: csh
Home directory [/home/asample]:
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]:
```

此处是需要输入的信息的摘要：

- Username——用户登录时要输入的名字。常见的惯例是使用名字的第一个字母与姓氏相结合，只要每个用户名对系统来说是唯一的即可。用户名区分大小写且不应包含任何空格。
- Full name——用户的全名。作为用户账户的说明，可以包含空格，。
- Uid——用户 ID。通常留空，系统会自动分配一个值。
- Login group——组。通常留空，使用默认值。
- Invite user into other groups?——额外的组，用户将被添加为成员。如果用户需要管理权限，在这里输入 wheel。
- Login class——通常留空，使用默认值。
- Shell——键入列表中的一项来设置用户的交互式 shell。关于 shell 的更多信息，请参考 shells[165]。
- Home directory——用户主目录。通常使用默认值。
- Home directory permissions——用户主目录的权限。通常使用默认值。
- Use password-based authentication?——通常回答 yes，这样用户在登录时就会被提示输入他们的密码。

[164] https://docs.freebsd.org/en/books/handbook/book/#bsdinstall-add-user2

[165] https://docs.freebsd.org/en/books/handbook/basics/index.html#shells

- `Use an empty password?`——通常是 `no`，因为空白的密码是不安全的。
- `Use a random password?`——通常是 `no`，这样用户可以在下一个提示中设置自己的密码。
- `Enter password`——输入用户的密码。输入的字符不会被显示在屏幕上。
- `Enter password again`——必须再次输入密码进行验证。
- `Lock out the account after creation?`——通常是 `no`，这样用户就可以登录。

在输入所有详细信息后，会显示一个摘要供复核。如果输错了什么，输入 `no`，然后再试一次。如果一切正确，输入 `yes` 来完成新用户的创建。

```
Login group [asample]:
Login group is asample. Invite asample into other groups? []: wheel
Login class [default]:
Shell (sh csh tcsh nologin) [sh]: csh
Home directory [/home/asample]:
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   : asample
Password   : *****
Full Name  : Arthur Sample
Uid        : 1001
Class      :
Groups     : asample wheel
Home       : /home/asample
Home Mode  :
Shell      : /bin/csh
Locked     : no
OK? (yes/no): yes
adduser: INFO: Successfully added (asample) to the user database.
Add another user? (yes/no):
```

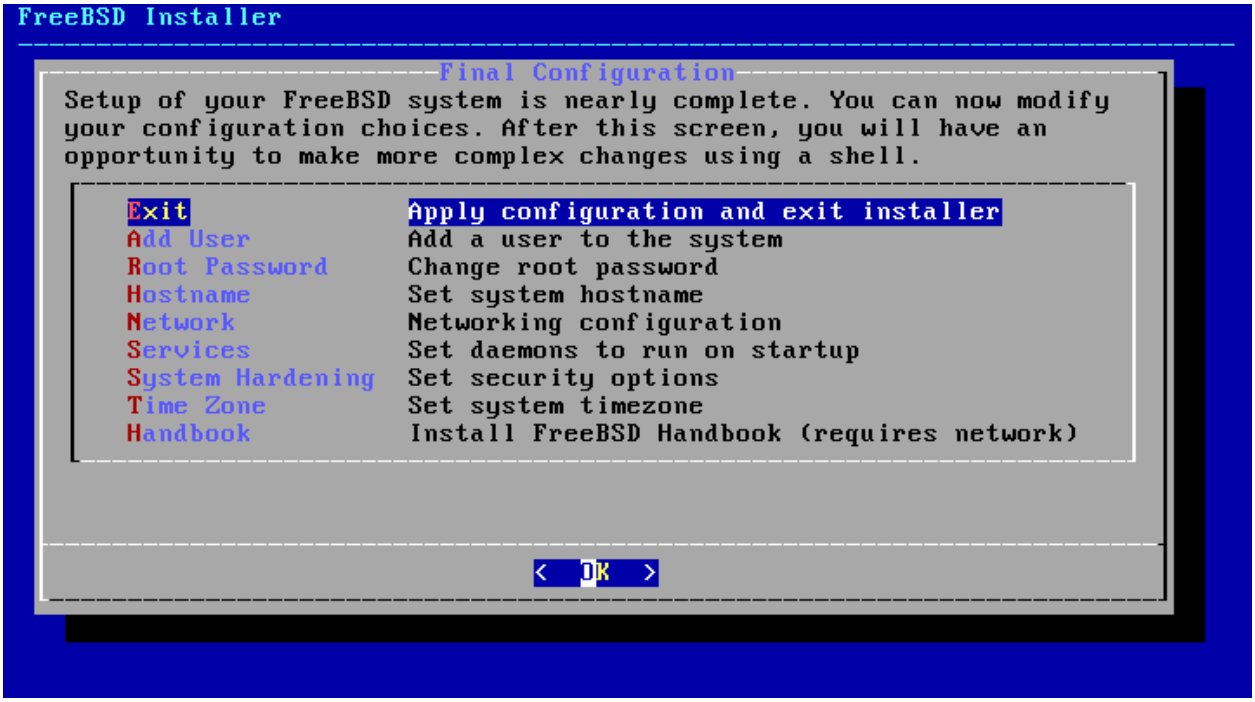
如果需要添加更多的用户，请在 `Add another user?` 这个问题上回答 `yes`。输入 `no` 可完成添加用户步骤并继续安装。

关于添加用户和用户管理的更多信息，请参阅用户和基本账户管理[166]。

2.8.7.最终配置

在所有东西都被安装和配置好之后，会提供最后一次修改设置的机会。

[166] https://docs.freebsd.org/en/books/handbook/basics/index.html#users-synopsis

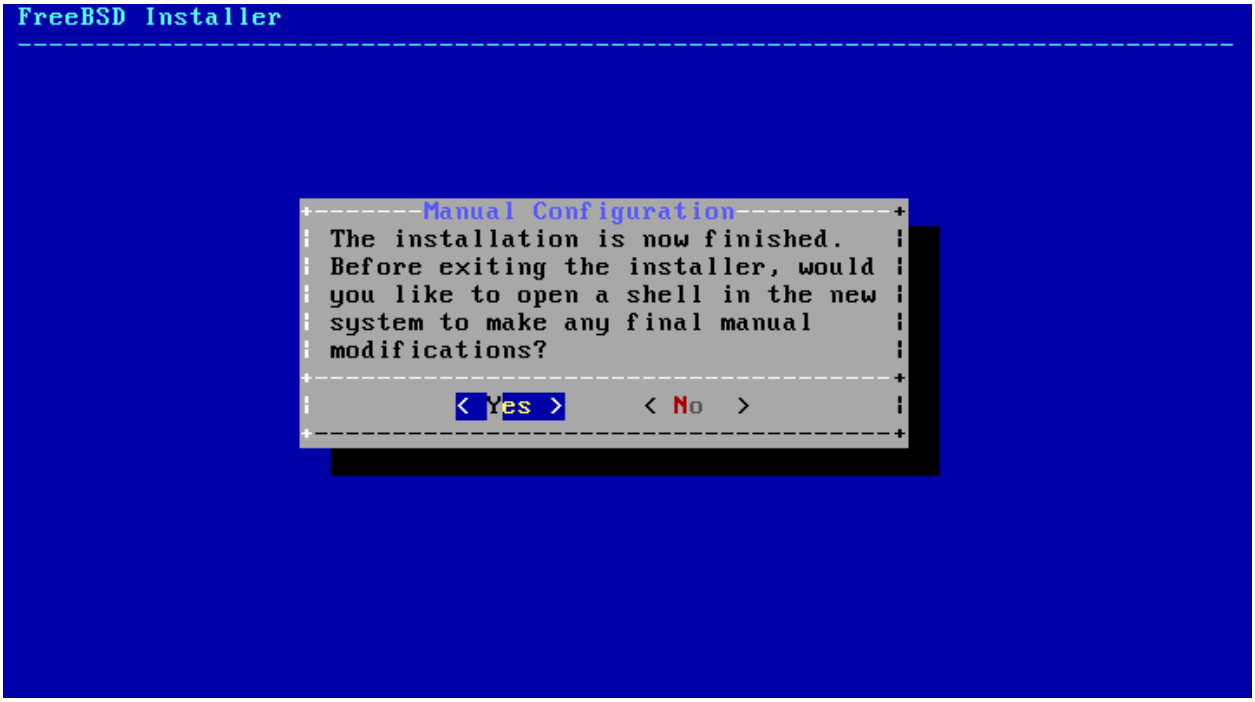


在完成安装之前，使用此菜单进行任何修改或做任何额外的配置。

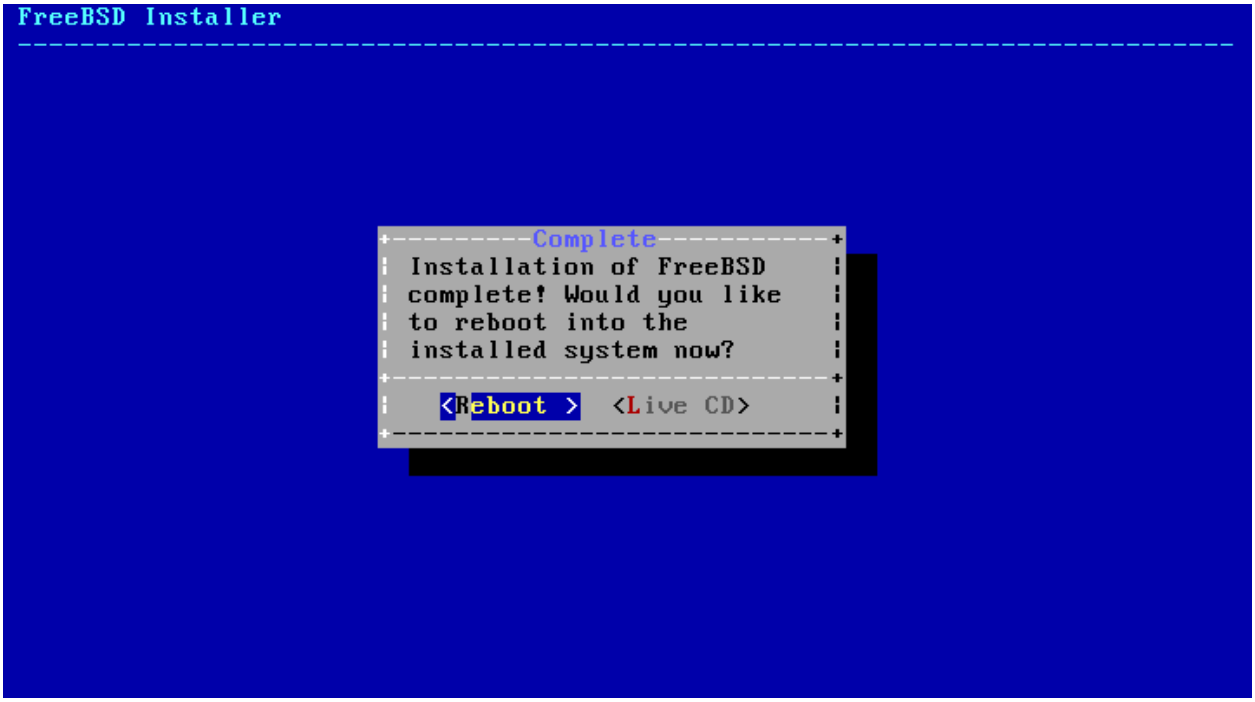
- `Add User`——在添加用户[167]中说明。
- `Root Password`——在设置 root 密码[168]章节中说明。
- `Hostname`——在设置主机名[169]章节中说明。
- `Network`——在配置网络接口[170]章节中说明。
- `Services`——在启用服务章节[171]中说明。
- `System Hardening`——在启用加固安全[172]选项中说明。
- `Time Zone`——在设置时区章节[173]中进行了说明。
- `Handbook`——下载并安装 FreeBSD 手册。

在完成配置后，选择 **Exit**。

[167] https://docs.freebsd.org/en/books/handbook/book/#bsdinstall-addusers
[168] https://docs.freebsd.org/en/books/handbook/book/#bsdinstall-post-root
[169] https://docs.freebsd.org/en/books/handbook/book/#bsdinstall-hostname
[170] https://docs.freebsd.org/en/books/handbook/book/#bsdinstall-config-network-dev
[171] https://docs.freebsd.org/en/books/handbook/book/#bsdinstall-sysconf
[172] https://docs.freebsd.org/en/books/handbook/book/#bsdinstall-hardening
[173] https://docs.freebsd.org/en/books/handbook/book/#bsdinstall-timezone



bsdinstall 会提示是否有任何额外的配置需要在重启到新系统之前完成。选择 **Yes** 会进入到新系统的 shell，或者选择 **No** 进入安装的最后步骤。



如果需要做进一步的配置或特殊的设置，选择 **Live CD** 来启动安装设备进入 Live CD 模式。

如果安装完成，选择 **Reboot** 来重新启动计算机并开始使用新的 FreeBSD 系统。不要忘记移除 FreeBSD 的安装介质，否则计算机可能会再次进入安装程序。

当 FreeBSD 启动时，会显示许多可供参考的信息。在系统完成启动后，会显示登录提示：在 `login:` 的提示下，输入安装时添加的用户名。避免以 `root` 身份登录。了解关于在需要管理权限时如何成为超级用户的说明，请参考超级用户[174]。

按 `Scroll-Lock` 键打开缓冲区，就可以查看启动时出现的信息。可以用 `PgUp`、`PgDn` 和方向键来回滚信息。完成后，再按一次 `Scroll-Lock` 键，解除画面锁定并返回到控制台。要在系统开机一段时间后查看这些信息，可以在命令提示符下输入 `less /var/run/dmesg.boot`。查看后按 `q` 键返回到命令行。

如果在选择要开启的其他服务[175]中启用了 `sshd`，第一次启动时可能会慢一些，因为系统会生成 SSH 主机密钥。后续的启动会恢复正常速度。然后会显示密钥的指纹，如下例所示：

```
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 .     |
```

(continues on next page)

[174] https://docs.freebsd.org/en/books/handbook/basics/index.html#users-superuser

[175] https://docs.freebsd.org/en/books/handbook/book/#bsdinstall-config-serv

(continued from previous page)

```
|      .o. .      |
+-----------------+
Starting sshd.
```

关于密钥指纹和 SSH 的更多信息请参考 OpenSSH[176]。

FreeBSD 不会预装图形界面。请参考 X Window 系统[177]以了解更多关于安装和配置图形化窗口管理器的信息。

正确地关闭 FreeBSD 计算机有助于保护数据和硬件免受损害。**在系统尚未正常关机之前，请不要切断电源！**若用户是 `wheel` 的成员，可在命令行输入 `su` 并输入 root 密码，成为超级用户。然后输入 `shutdown -p now`，系统就会干净利落地关机，如果硬件支持，就会自动切断电源。

2.9.故障排除

这一节涵盖了基本安装的故障排除，包含人们已经报告的常见问题。

检查 FreeBSD 版本的硬件说明 (https://www.freebsd.org/releases/) 文件，以确定是否支持该硬件。

> 注意
>
> 一些安装问题可以通过更新各种硬件组件上的固件来避免或缓解，其中最明显的是主板。主板固件通常被称为 BIOS。大多数主板和计算机制造商都有一个升级和升级信息的网站。
>
> 除非是必要的关键更新，否则制造商通常不建议升级主板 BIOS。更新过程可能出错，导致 BIOS 不完整以至于电脑无法使用。

如果系统在启动过程中探测硬件时挂起，或者在安装过程中表现得很奇怪，可能罪魁祸首是 ACPI。FreeBSD 在 i386 和 amd64 平台上广泛使用了系统的 ACPI 服务，如果在启动过程中检测到它，就可以帮助配置系统。不幸的是，在 ACPI 驱动程序和系统主板及 BIOS 固件中可能仍然存在一些问题。可以通过在第三阶段启动加载器中设置 `hint.acpi.0.disabled` 提示来禁用 ACPI：

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

这在每次系统启动时都会被重置，因此有必要在 **/boot/loader.conf** 这个文件中加入 `hint.acpi.0.disabled="1"`。更多关于启动引导的信息可以在概述[178]中找到。

[176] https://docs.freebsd.org/en/books/handbook/security/index.html#openssh

[177] https://docs.freebsd.org/en/books/handbook/x11/index.html#x11

[178] https://docs.freebsd.org/en/books/handbook/boot/index.html#boot-synopsis

2.10.使用 Live CD

bsdinstall 的欢迎菜单[179]提供了一个 **Live CD** 选项。这对于那些还在怀疑 FreeBSD 是否是适合他们的操作系统，并想在安装前测试一些功能的人来说是很有用的。

在使用 **Live CD** 之前，应该注意以下几点：

- 要进入系统，需要进行认证。用户名是 `root`，密码为空。
- 由于系统直接在安装设备上运行，运行效率将明显低于安装在硬盘上的系统。
- 该选项只提供了一个命令提示符，而非图形界面。

[179] https://docs.freebsd.org/en/books/handbook/book/#bsdinstall-choose-mode

第三章 FreeBSD 基础

3.1.概述

本章介绍了 FreeBSD 操作系统的基本命令和功能。其中大部分内容与其它的类 UNIX® 操作系统很相似。我们建议 FreeBSD 新用户仔细阅读本章。

读完本章后，你将会了解：

- 如何使用和配置虚拟控制台。
- 如何在 FreeBSD 上创建和管理用户与组。
- UNIX® 文件权限和 FreeBSD 文件标志的工作原理。
- FreeBSD 默认的文件系统布局。
- FreeBSD 的磁盘结构。
- 如何挂载和卸载文件系统。
- 什么是进程、守护进程以及信号。
- 什么是 shell，以及如何改变默认的登录环境。
- 如何使用基本的文本编辑器。
- 什么是设备及设备节点。
- 如何阅读手册以获得更多信息。

3.2.虚拟控制台和终端

如果没有把 FreeBSD 的图形界面配置为开机自启，那么系统将启动到命令行登录提示界面，如本例所示：

```
FreeBSD/amd64 (pc3.example.org) (ttyv0)

login:
```

第一行包含了一些有关系统的信息。`amd64` 表示本例中的系统运行的是 64 位版本的 FreeBSD。主机名是 `pc3.example.org`，`ttyv0` 表示这是“系统控制台”。第二行是登录提示。

由于 FreeBSD 是一个多用户系统，它需要一些方法来区分不同的用户。因此每个用户在获得对系统中的程序的访问权之前必须登录系统。每个用户都有一个独特的 `username` 和一个个人的 `password`。

要登录到系统控制台，输入在系统安装时配置的用户名，如添加用户[180]中所述，然后按回车键。然后输入与该用户名相关的密码，按回车键。出于安全考虑，密码不会被显示出来。

输入正确的密码后，将显示当日信息（MOTD），然后是命令提示符。根据创建用户时选择的 shell，这个提示符会是 `#`、`$` 或是 `%` 字符之一。该提示符代表用户现在已经登录到 FreeBSD 系统控制台，并准备尝试执行可用命令。

3.2.1.虚拟终端

虽然可以用系统控制台来与系统进行交互，但通过键盘用命令行工作的 FreeBSD 系统用户通常会转而登录到一个虚拟控制台。这是因为系统信息被默认配置为显示在系统控制台。这些信息会在用户正在处理的命令或文件时不断出现，使用户难以集中精力处理手头的工作。

FreeBSD 预置了几个虚拟控制台用于输入命令。每个虚拟控制台都有自己的登录提示和 shell，且很容易在不同的虚拟控制台之间切换。这实际上提供了相当于在图形环境中同时打开了多个窗口的命令行。

FreeBSD 保留了从 `Alt+F1` 到 `Alt+F8` 的组合键，用于在虚拟控制台之间切换。使用 `Alt+F1` 切换到系统控制台（`ttyv0`），`Alt+F2` 访问第一个虚拟控制台（`ttyv1`），`Alt+F3` 访问第二个虚拟控制台（`ttyv2`），以此类推。当使用 Xorg 作为图形控制台时，该组合变为 `Ctrl+Alt+F1`，以返回到基于文本的虚拟控制台。

当从一个控制台切换到下一个控制台时，FreeBSD 会切换屏幕输出。其结果是产生一种拥有多个虚拟屏幕和键盘的错觉，都可以用来输入命令让 FreeBSD 运行。当用户切换到另一个虚拟控制台时，在虚拟控制台中启动的程序不会停止运行。

请参考 kbdcontrol(1)[181], vidcontrol(1)[182], atkbd(4)[183], syscons(4)[184], 和 vt(4)[185] 以了解关于 FreeBSD 控制台及其键盘驱动程序的更多技术介绍。

[180] https://docs.freebsd.org/en/books/handbook/bsdinstall/index.html#bsdinstall-addusers
[181] https://www.freebsd.org/cgi/man.cgi?query=kbdcontrol&sektion=1&format=html
[182] https://www.freebsd.org/cgi/man.cgi?query=vidcontrol&sektion=1&format=html
[183] https://www.freebsd.org/cgi/man.cgi?query=atkbd&sektion=4&format=html
[184] https://www.freebsd.org/cgi/man.cgi?query=syscons&sektion=4&format=html
[185] https://www.freebsd.org/cgi/man.cgi?query=vt&sektion=4&format=html

在 FreeBSD 中，可用的虚拟控制台的数量是在 /etc/ttys 的这一部分进行配置的：

```
# name    getty                           type  status comments
#
ttyv0   "/usr/libexec/getty Pc"         xterm   on  secure
# Virtual terminals
ttyv1   "/usr/libexec/getty Pc"         xterm   on  secure
ttyv2   "/usr/libexec/getty Pc"         xterm   on  secure
ttyv3   "/usr/libexec/getty Pc"         xterm   on  secure
ttyv4   "/usr/libexec/getty Pc"         xterm   on  secure
ttyv5   "/usr/libexec/getty Pc"         xterm   on  secure
ttyv6   "/usr/libexec/getty Pc"         xterm   on  secure
ttyv7   "/usr/libexec/getty Pc"         xterm   on  secure
ttyv8   "/usr/X11R6/bin/xdm -nodaemon"  xterm   off secure
```

要禁用某个虚拟控制台，在代表该虚拟控制台的行的开头加一个注释符号（#）即可。例如，要把可用的虚拟控制台的数量从 8 个减少到 4 个，在代表虚拟控制台 ttyv5 到 ttyv8 的最后四行前面加一个 # 即可。不要注释系统控制台 ttyv0 所在行。注意，如果已经根据 X Window 系统[186]安装并配置了 Xorg，那么最后一个虚拟控制台（ttyv8）将用于访问图形界面。

关于这个文件中每一列的详细解释以及虚拟控制台的可用选项，请参考 ttys(5)[187]。

3.2.2.单用户模式的控制台

FreeBSD 的启动菜单提供了一个叫做“Boot Single User”的选项。如果选择了这个选项，系统将启动到一个被称为“单用户模式”的特殊模式。这种模式通常用于修复无法启动的系统，或者在不知道 root 密码的情况下将其重置。在单用户模式下，网络和其他虚拟控制台不可用。然而，对系统的完全 root 访问是可用的，而且在默认情况下，无需 root 密码。由于这些原因，需要有对键盘的物理访问权限才能进入该模式，确定谁对键盘有物理访问权是保护 FreeBSD 系统安全时需要考虑的问题。

可以在 **/etc/ttys** 的这一部分找到控制单用户模式的设置：

```
# 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
```

默认情况下，状态被设置为 secure。这假定谁有对键盘拥有物理访问权并不重要，或者它受到物理安全策略的控制。如果这个设置被改为 insecure，则假设环境本身是不安全的，因为任何人都可以访问键盘。当这一行被改为 insecure 时，FreeBSD 将在用户选择启动到单用户模式时提示输入 root 密码。

注意

[186] https://docs.freebsd.org/en/books/handbook/x11/index.html#x11

[187] https://www.freebsd.org/cgi/man.cgi?query=ttys&sektion=5&format=html

当把这个设置改为 `*insecure*`的时候，要小心了！如果忘记了 root 密码，虽然仍可启动至单用户模式，但对于不熟悉 FreeBSD 引导过程的人来说可能会很困难。

3.2.3.改变控制台的分辨率

FreeBSD 控制台的默认分辨率可以调整为 1024x768、1280x1024, 或任何其他由显卡和显示器支持的尺寸。要使用不同的分辨率，请加载 VESA 模块：

```
# kldload vesa
```

要确定硬件支持哪些分辨率，使用 vidcontrol(1)[188]。要获得支持的分辨率的列表，请执行以下命令：

```
# vidcontrol -i mode
```

该命令的输出列出了硬件所支持的分辨率。要选择新的分辨率，请以 root 用户身份使用 vidcontrol(1)[189] 指定该分辨率：

```
# vidcontrol MODE_279
```

如果新的分辨率符合预期，可以在启动时通过将其添加到 /etc/rc.conf 中进行永久设置：

```
allscreens_flags="MODE_279"
```

3.3.用户和基本账户管理

FreeBSD 允许多用户同时使用计算机。虽然在任何时候都只有一个用户可以坐在屏幕前使用键盘，但任意数量的用户都可以通过网络登录到系统。每个为了使用该系统的用户都应该有自己的用户账户。

本章介绍了：

- FreeBSD 系统中不同类型的用户账户。
- 如何添加、删除和修改用户账户。
- 如何设置限制来控制允许用户和组访问的资源。
- 如何创建组并将用户添加为组的成员。

[188] https://www.freebsd.org/cgi/man.cgi?query=vidcontrol&sektion=1&format=html
[189] https://www.freebsd.org/cgi/man.cgi?query=vidcontrol&sektion=1&format=html

3.3.1.账户类型

由于对 FreeBSD 系统的所有访问都是通过帐户实现的，而且所有进程都是由用户运行的，因此用户和帐户的管理非常重要。

主要有三种类型的账户：系统账户、用户账户和超级用户账户。

3.3.1.1.系统账户

系统账户用于运行服务，如 DNS、邮件和网络服务器。这样做的原因是安全问题；如果所有的服务都以超级用户的身份运行，它们就可以不受限制地行动。

系统账户例如 `daemon`、`operator`、`bind`、`news`、`www`。

`nobody` 是通用的无特权系统账户。然而，使用 `nobody` 的服务越多，该用户将与更多的文件和进程相关联，因此该用户的特权也就越大。

3.3.1.2.用户账户

用户账户被分配给真实的人，用来登录和使用系统。每个访问系统的人都应该有一个独特的用户账户。这使管理员能够发现谁在做什么，并防止用户篡改其他用户的设置。

每个用户都可以通过配置默认的 shell、编辑器、组合键和语言设置来设置自己的环境，以适应他们对系统的使用。

在 FreeBSD 系统上的每个用户账户都有一些相关的信息。

User name

用户名称是在 `login:` 提示符下输入的。每个用户都必须有一个唯一的用户名。在 passwd(5)[190] 中记录了一些创建有效用户名的规则。建议使用由八个或更少的小写字母组成的用户名，以保持与应用程序的向后兼容性。

Password

每个账户都有一个对应的密码。

User ID (UID)

用户 ID（UID）是一组数字，用于在 FreeBSD 系统中唯一地识别用户。当命令使用到了用户名时，会先将其转换为 UID。建议使用小于 65535 的 UID，因为过高的值可能会引起某些软件的兼容性问题。

Group ID (GID)

组 ID（GID）是一组数字，用于唯一地识别用户所属的主要组。组是根据用户的 GID 而不是 UID 来控制对资源的访问一种机制。这可以大大减少一些配置文件的大小，并允许用户成为多个组的成员。建议使用 65535 或更小的 GID，因为过高的 GID 可能会引起某些软件的兼容性问题。

[190] https://www.freebsd.org/cgi/man.cgi?query=passwd&sektion=5&format=html

Login class

登录分级是对组机制的扩展，在为不同用户定制系统时提供了额外的灵活性。登录分级在配置登录分级[191]中有进一步讨论。

Password change time

默认情况下，密码是不会过期的。然而，可以在每个用户的基础上启用密码过期，迫使部分或所有用户在一定时间后改变他们的密码。

Account expiration time

默认情况下，FreeBSD 不对账户进行过期保护。当创建需要有限寿命的账户时，例如学校的学生账户，可以使用 pw(8)[192] 指定账户的过期时间。在过期时间过后，就不能用这个账户来登录系统了，不过会保留这个账户的目录和文件。

User’s full name

用户名称对 FreeBSD 来说是账户的唯一标识，但不一定反映用户的真实姓名。与注释类似，这一信息可以包含空格、大写字母，并且长度可超过 8 个字符。

Home directory

主目录是系统中某个目录的完整路径。当用户登录时，这是用户的起始目录。一个常见的惯例是把所有用户的主目录放在 `/home/username` 或 `/usr/home/username` 下。每个用户都把他们的个人文件和子目录存放在自己的主目录下。

User shell

shell 为用户提供了与系统交互的默认环境。有许多不同种类的 shell，有经验的用户会有自己的偏好，这反映在他们的帐户设置中。

3.3.1.3.超级用户

超级用户，通常称为 `root`，用于管理系统，没有权限限制。由于这个原因，它不应该被用于日常工作，如发送和接收邮件，以及对系统的一般探索或编程。

超级用户与普通用户不同，可以不受限制地操作，滥用超级用户可能导致巨大的灾难。普通用户因为权限不足而无法因人为失误而破坏操作系统。因此建议以普通用户登录，只有在命令需要额外权限时才成为超级用户。

对于以超级用户身份发布的任何命令，一定要进行双重和三重检查，因为一个额外的空格或缺失的字符可能意味着不可修复的数据损失。

有几种方法可以获得超级用户的权限。虽然人们可以直接以 `root` 身份登录，但这是非常不可取的。

相反，使用 su(1)[193] 来成为超级用户。如果在运行这个命令时指定了 `-`，用户也将继承 root 用户的环境。运行该命令的用户必须在 `wheel` 组中，否则命令会执行失败。该用户还必须知道 `root` 用户的账户密码。

[191] https://docs.freebsd.org/en/books/handbook/security/index.html#users-limiting

[192] https://www.freebsd.org/cgi/man.cgi?query=pw&sektion=8&format=html

[193] https://www.freebsd.org/cgi/man.cgi?query=su&sektion=1&format=html

在这个例子中，用户只有成为超级用户才能运行 `make install`，因为这个步骤需要超级用户的权限。命令完成后，用户输入 `exit`，退出超级用户账户，返回到普通用户的权限。

例 **1.** *以超级用户身份安装一个程序*

```
% configure
% make
% su -
Password:
# make install
# exit
%
```

内置的 su(1)[194] 框架对于只有一个系统管理员的单一系统或小型网络来说效果很好。另一个选择是通过软件包或 port 安装 security/sudo[195]。这个软件提供活动日志，并允许管理员配置哪些用户可以作为超级用户运行哪些命令。

3.3.2.管理账户

FreeBSD 提供了各种不同的命令来管理用户账户。最常见的命令在管理用户帐户的实用工具[196]中进行了总结，后面还有一些使用实例。关于更多的细节和使用例子，请参见每个工具的手册页面。

表 1. 管理用户账户的实用工具

命令	摘要
adduser(8)[197]	推荐用于添加新用户的命令行程序。
rmuser(8)[198]	推荐用于删除用户的命令行程序。
chpass(1)[199]	一个灵活的，用于改变用户数据库信息的工具。
passwd(1)[200]	用于更改用户密码的命令行工具。
pw(8)[201]	一个强大而灵活的工具，用于修改用户账户的方方面面。
bsdconfig(8)	一个支持账户管理的系统配置工具。

[194] https://www.freebsd.org/cgi/man.cgi?query=su&sektion=1&format=html
[195] https://cgit.freebsd.org/ports/tree/security/sudo/pkg-descr
[196] https://docs.freebsd.org/en/books/handbook/book/#users-modifying-utilities
[197] https://www.freebsd.org/cgi/man.cgi?query=adduser&sektion=8&format=html
[198] https://www.freebsd.org/cgi/man.cgi?query=rmuser&sektion=8&format=html
[199] https://www.freebsd.org/cgi/man.cgi?query=chpass&sektion=1&format=html
[200] https://www.freebsd.org/cgi/man.cgi?query=passwd&sektion=1&format=html
[201] https://www.freebsd.org/cgi/man.cgi?query=pw&sektion=8&format=html

3.3.2.1.添加用户

推荐用于添加新用户的程序是 adduser(8)[202]。当一个新用户被添加时，这个程序会自动更 `/etc/passwd` 和 `/etc/group`。它还为新用户创建一个主目录，从 `/usr/share/skel` 复制默认的配置文件，并可以选择给新用户发送欢迎信息。这个工具必须以超级用户的身份运行。

adduser(8)[203] 工具是交互式的，它将引导你完成创建一个新用户账户的步骤。正如在 FreeBSD 上添加用户中所看到的，要么输入所需的信息，要么按回车键接受方括号中的默认值。在这个例子中，用户被邀请进入 `wheel` 组，允许他们用 su(1)[204] 成为超级用户。完成后，该工具将提示创建另一个用户或退出。

例 **2.** 在 **FreeBSD** 上添加一个用户

```
# adduser
```

输出结果应类似于以下内容：

```
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.
```

(continues on next page)

[202] https://www.freebsd.org/cgi/man.cgi?query=adduser&sektion=8&format=html

[203] https://www.freebsd.org/cgi/man.cgi?query=adduser&sektion=8&format=html

[204] https://www.freebsd.org/cgi/man.cgi?query=su&sektion=1&format=html

(continued from previous page)

```
Add another user? (yes/no): no
Goodbye!
```

注意
由于密码在输入时并不会显示，所以在创建用户账户时要注意不要打错密码。

3.3.2.2.删除用户

要从系统中完全删除一个用户，请以超级用户身份运行 rmuser(8)[205]。该命令会执行以下步骤：

1. 删除该用户的 crontab(1)[206] 条目，如果存在的话。
2. 删除属于该用户的任何 at(1)[207] 作业。
3. 发送 SIGKILL 信号给所有属于该用户的进程。
4. 从系统的本地密码文件中删除该用户。
5. 删除用户的主目录（如果它是由该用户拥有的），包括处理实际主目录路径中的符号链接。
6. 从 `/var/mail` 中删除属于该用户的邮件文件。
7. 从 `/tmp`、`/var/tmp` 和 `/var/tmp/vi.recover` 中删除所有由用户拥有的文件。
8. 将用户名从其在 `/etc/group` 中所属的所有组中移除。(如果一个组变成空的，并且组名与用户名相同，这个组就会被删除；这是对 adduser[8][208] 的每个用户唯一组的补充规则)。
9. 移除用户拥有的所有消息队列、共享内存和信号量。

rmuser(8)[209] 不能用来删除超级用户账户，因为这几乎代表着大规模破坏。

默认情况下，使用的是交互式模式，如下面的例子所示。

例 **3. rmuser** 交互式删除账户

```
# rmuser jru
```

输出结果应类似于以下内容：

```
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
```

(continues on next page)

[205] https://www.freebsd.org/cgi/man.cgi?query=rmuser&sektion=8&format=html
[206] https://www.freebsd.org/cgi/man.cgi?query=crontab&sektion=1&format=html
[207] https://www.freebsd.org/cgi/man.cgi?query=at&sektion=1&format=html
[208] https://www.freebsd.org/cgi/man.cgi?query=adduser&sektion=8&format=html
[209] https://www.freebsd.org/cgi/man.cgi?query=rmuser&sektion=8&format=html

(continued from previous page)

```
Remove user's home directory (/home/jru)? y
Removing user (jru): mailspool home passwd.
```

3.3.2.3.修改用户信息

任何用户都可以使用 chpass(1)[210] 来改变他们的默认 shell 和与用户帐户相关的个人信息。超级用户可以使用这个工具来改变任何用户的额外账户信息。

当除了一个可选的用户名外，没有其他选项时，chpass(1)[211] 会显示一个包含用户信息的编辑器。当用户从编辑器中退出时，用户数据库被更新为新的信息。

> 注意
> 在退出编辑器时该工具将提示用户输入密码，除非以超级用户身份运行该工具。

在以超级用户身份使用 chpass[212] 中，超级用户已经输入了 `chpass jru`，现在正在查看这个用户可以修改的字段。如果以 `jru` 来运行这个命令，只会显示且可编辑最后六个字段。这在以普通用户身份使用chpass[213]中概述。

例 **4.** 使用 **chpass** 作为超级用户

```
# chpass
```

输出结果应类似于以下内容：

```
#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:
```

[210] https://www.freebsd.org/cgi/man.cgi?query=chpass&sektion=1&format=html
[211] https://www.freebsd.org/cgi/man.cgi?query=chpass&sektion=1&format=html
[212] https://docs.freebsd.org/en/books/handbook/book/#users-modifying-chpass-su
[213] https://docs.freebsd.org/en/books/handbook/book/#users-modifying-chpass-ru

例 **5.** 使用 **chpass** 作为普通用户

```
#Changing user database information for jru.
Shell: /usr/local/bin/zsh
Full Name: J. Random User
Office Location:
Office Phone:
Home Phone:
Other information:
```

注意

命令 chfn(1)[214] 和 chsh(1)[215] 是 chpass(1)[216] 的链接，ypchpass(1)[217]、ypchfn(1)[218] 和 ypchsh(1)[219] 也是如此。NIS 的支持是系统自带的，不需要在命令前专门指定 yp。如何配置 NIS 在网络服务器[220]中有所介绍。

3.3.2.4.修改用户密码

任何用户都可以使用 passwd(1)[221] 轻松地修改他们的密码。为了防止意外或未经授权的更改，该命令在设置新密码之前会提示用户的原始密码：

例 **6.** 改变你的密码

```
% passwd
```

输出结果应类似于以下内容：

```
Changing local password for jru.
Old password:
New password:
Retype new password:
passwd: updating the database...
passwd: done
```

超级用户可以通过在运行 passwd(1)[222] 时指定用户名来改变任何用户的密码。当这个工具以超级用户身份运行时，它将不会提示用户的当前密码。这允许在用户不记得原来的密码时修改密码。

例 **7.** 以超级用户身份修改其他用户的密码

[214] https://www.freebsd.org/cgi/man.cgi?query=chfn&sektion=1&format=html
[215] https://www.freebsd.org/cgi/man.cgi?query=chsh&sektion=1&format=html
[216] https://www.freebsd.org/cgi/man.cgi?query=chpass&sektion=1&format=html
[217] https://www.freebsd.org/cgi/man.cgi?query=ypchsh&sektion=1&format=html0
[218] https://www.freebsd.org/cgi/man.cgi?query=ypchfn&sektion=1&format=html
[219] https://www.freebsd.org/cgi/man.cgi?query=ypchsh&sektion=1&format=html
[220] https://docs.freebsd.org/en/books/handbook/network-servers/index.html#network-servers
[221] https://www.freebsd.org/cgi/man.cgi?query=passwd&sektion=1&format=html
[222] https://www.freebsd.org/cgi/man.cgi?query=passwd&sektion=1&format=html

```
# passwd jru
```

输出结果应类似于以下内容：

```
Changing local password for jru.
New password:
Retype new password:
passwd: updating the database...
passwd: done
```

注意
与 chpass(1)[223] 一样，yppasswd(1)[224] 是 passwd(1)[225] 的链接，所以 NIS 可以使用任何一条命令。

3.3.2.5.创建、删除、修改和显示系统用户和群组

pw(8)[226] 可以删除、修改和显示用户和组。它的功能是作为系统用户和组文件的前端。pw(8)[227] 有一组非常强大的命令行选项，使它适合于在 shell 脚本中使用，但新用户可能会发现它比本节介绍的其他命令更复杂。

3.3.3.组管理

组是一个用户的列表。组由其组名和 GID 来识别。在 FreeBSD 中，内核使用一个进程的 UID 和它所属的组的列表来决定这个进程被允许做什么。大多数时候，一个用户或进程的 GID 通常意味着列表中的第一个组。

组名与 GID 的对应列表被列在 `/etc/group` 中。这是一个纯文本文件，有四个以冒号分隔的字段。第一个字段是组名，第二个是加密的密码，第三个是 GID，第四个是以逗号分隔的成员列表。关于语法的更完整说明，请参考 group(5)[228]。

超级用户可以使用文本编辑器修改 `/etc/group`，但是建议使用vigr[8][229] 来编辑组文件，因为它可以捕捉到一些常见的错误。另外，可以用 pw(8)[230] 来添加和编辑组。例如，添加一个名为 `teamtwo` 的组，然后确认它的存在：

警告
使用 operator 组时必须小心，因为可能会被授予类似超级用户的访问权限，包括但不限于关机、重启，以及访问组内 `/dev` 中的所有项目。

[223] https://www.freebsd.org/cgi/man.cgi?query=chpass&sektion=1&format=html
[224] https://www.freebsd.org/cgi/man.cgi?query=yppasswd&sektion=1&format=html
[225] https://www.freebsd.org/cgi/man.cgi?query=passwd&sektion=1&format=html
[226] https://www.freebsd.org/cgi/man.cgi?query=pw&sektion=8&format=html
[227] https://www.freebsd.org/cgi/man.cgi?query=pw&sektion=8&format=html
[228] https://www.freebsd.org/cgi/man.cgi?query=group&sektion=5&format=html
[229] https://www.freebsd.org/cgi/man.cgi?query=vigr&sektion=8&format=html
[230] https://www.freebsd.org/cgi/man.cgi?query=pw&sektion=8&format=html

例 **8.** 使用 pw(8)[231] 添加一个组

```
# pw groupadd teamtwo
# pw groupshow teamtwo
```

输出结果应类似于以下内容：

```
teamtwo:*:1100:
```

在这个例子中，1100 是 teamtwo 的 GID。现在，teamtwo 没有成员。这个命令将添加 jru 为 teamtwo 的成员。

例 **9.** 使用 pw(8)[232] 将用户账户添加到一个新组中

```
# pw groupmod teamtwo -M jru
# pw groupshow teamtwo
```

输出结果应类似于以下内容：

```
teamtwo:*:1100:jru
```

参数 -M 会显示一个以逗号分隔的用户列表，用来添加到一个新的（空的）组或替换现有组的成员。对用户来说，这个组的成员资格与密码文件中列出的用户的主组不同（额外的）。这意味着在使用 pw(8)[233] 的 groupshow 时，用户不会显示为成员，但在通过 id(1)[234] 或类似工具查询信息时，会显示出来。当 pw(8)[235] 被用来将一个用户添加到一个组时，它只操作 /etc/group，而不试图从 /etc/passwd 读取其他数据。

例 **10.** 使用 pw(8)[236] 向一个组添加新成员

```
# pw groupmod teamtwo -m db
# pw groupshow teamtwo
```

输出结果应类似于以下内容：

```
teamtwo:*:1100:jru,db
```

在这个例子中，参数 -m 会显示一个以逗号分隔的用户列表，这些用户将被添加到该组。与前面的例子不同，这些用户被添加到组中，并不取代组中的现有用户。

例 **11.** 使用 id(1)[237] 来确定群组成员资格

[231] https://www.freebsd.org/cgi/man.cgi?query=pw&sektion=8&format=html
[232] https://www.freebsd.org/cgi/man.cgi?query=pw&sektion=8&format=html
[233] https://www.freebsd.org/cgi/man.cgi?query=pw&sektion=8&format=html
[234] https://www.freebsd.org/cgi/man.cgi?query=id&sektion=1&format=html
[235] https://www.freebsd.org/cgi/man.cgi?query=pw&sektion=8&format=html
[236] https://www.freebsd.org/cgi/man.cgi?query=pw&sektion=8&format=html
[237] https://www.freebsd.org/cgi/man.cgi?query=id&sektion=1&format=html

```
% id jru
```

输出结果应类似于以下内容：

```
uid=1001(jru) gid=1001(jru) groups=1001(jru), 1100(teamtwo)
```

在这个例子中，`jru` 是 `jru` 和 `teamtwo` 组的成员。

关于这个命令和 **/etc/group** 格式的更多信息，请参考 pw(8)[238] 和 group(5)[239]。

3.4.权限

在 FreeBSD 中，每个文件和目录都有一组相关的权限，有几个工具可以用来查看和修改这些权限。为了确保用户能够访问他们需要的文件，并且不能非法访问操作系统使用的或其他用户拥有的文件，了解权限的工作方式是必要的。

这一节讨论了 FreeBSD 中使用的传统 UNIX® 权限。对于细粒度的文件系统访问控制，请参考“访问控制列表”[240]。

在 UNIX® 中，基本权限由三种访问类型进行分配：读、写和执行。这些访问类型用于确定文件所有者、组和其他人（其他所有人）对文件的访问。读、写和执行权限可以分别用字母`r`、`w` 和 `x` 表示，也可以用二进制数字表示，因为每个权限都是开或关（`0`）。当用数字表示时，阅读顺序为 `rwx`，其中 `r` 的开启值为 `4`，`w` 的开启值为`2`，`x` 的开启值为 `1`。

表 4.1 总结了可能的数字和字母的可能性。在阅读“参数”一栏时，– 被用来代表一个被设置为关闭的权限。

表 2. UNIX® 权限

值	权限	参数
0	不可读，不可写，不可执行	—
1	不可读，不可写，可执行	–x
2	不可读、可写、不可执行	-w-
3	不可读、可写、可执行	-wx
4	可读，不可写，不可执行	r–
5	可读，不可写，可执行	r-x
6	可读、可写，不可执行	rw-
7	可读、可写、可执行	rwx

[238] https://www.freebsd.org/cgi/man.cgi?query=pw&sektion=8&format=html

[239] https://www.freebsd.org/cgi/man.cgi?query=group&sektion=5&format=html

[240] https://docs.freebsd.org/en/books/handbook/security/index.html#fs-acl

使用 ls(1)[241] 的参数 `-l` 可以查看详细的目录列表，其中包括一列关于文件的所有者、组和其他人的权限的信息。例如，在一个任意的目录中，`ls -l`可能显示：

```
% ls -l
```

输出结果应类似于下面的内容：

```
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
```

请看 `myfile` 这一行，第一个“(最左边)”字符表示这个文件是一个普通文件、一个目录、一个特殊字符设备、一个套接字或任何其他特殊的伪文件设备。在这个例子中，`-` 表示一个普通文件。接下来的三个字符，即本例中的 `rw-`，给出了文件所有者的权限。再接下来的三个字符，`r--`，给出了文件所属组的权限。最后三个字符，`r--`，代表其他人的权限。一个破折号意味着该权限被关闭。在这个例子中，权限被设置为所有者可以读写文件，组可以读取文件，而其他人只能读取文件。根据上表，这个文件的权限是 `644`，其中每个数字代表文件权限的三个部分。

系统是如何控制设备的权限的？FreeBSD 将大多数硬件设备视为是一个程序可以打开、读取并写入数据的文件。这些特殊的设备文件被存放在 `/dev/`。

目录也被视为文件。它们有读、写和执行的权限。目录的可执行位与文件的意义略有不同。当一个目录被标记为可执行时，它意味着可以使用 cd(1)[242] 切换到该目录。这也意味着可以访问该目录中的文件，但要取决于文件本身的权限。

为了列出目录内容，必须在目录上设置读取权限。为了删除一个知道名字的文件，必须对包含该文件的目录有写和执行的权限。

还有更多的权限位，但它们主要是在特殊情况下使用，如 `setuid` 二进制文件和粘滞位目录。关于文件权限的更多信息以及如何设置它们，请参考 chmod(1)[243]。

3.4.1.权限符号

符号权限使用字符而非八进制值来给文件或目录分配权限。符号权限使用的语法是（用户）（动作）（权限），其中有下列参数：

[241] https://www.freebsd.org/cgi/man.cgi?query=ls&sektion=1&format=html
[242] https://www.freebsd.org/cgi/man.cgi?query=cd&sektion=1&format=html
[243] https://www.freebsd.org/cgi/man.cgi?query=chmod&sektion=1&format=html

选项	参数	意义
（用户）	u	用户
（用户）	g	组
（用户）	o	其他
（用户）	a	All（“全部”）
（动作）	+	增加权限
（动作）	-	移除权限
（动作）	=	指定权限
（权限）	r	读
（权限）	w	写
（权限）	x	执行
（权限）	t	粘滞位
（权限）	s	设置 UID 或 GID

这些值与 chmod(1)[244] 一起使用，但用字母而非数字。例如，下面的命令会阻止与 *FILE* 相关的组的成员和所有其他用户访问 *FILE*。

```
% chmod go= FILE
```

当必须对一个文件进行多组修改时，可以提供一个逗号分隔的列表。例如，下面的命令移除了组和所有人对 *FILE* 的写入权限，并为每个人增加了执行权限：

```
% chmod go-w,a+x FILE
```

3.4.2.FreeBSD 的文件标志

除了文件权限之外，FreeBSD 还支持使用“文件标志”。这些标志为文件（不含目录），增加了额外的安全和控制能力。通过文件标志，甚至可以阻止 `root` 删除或更改文件。

文件标志是用 chflags(1)[245] 修改的。例如，要在文件 `file1` 上启用系统禁止删除的标志，执行以下命令：

```
# chflags sunlink file1
```

要移除系统禁止删除标志，请在 `sunlink` 前面加一个“no”：

```
# chflags nosunlink file1
```

要查看文件的标志，可以使用 ls(1)[246] 中 `-l` 参数：

[244] https://www.freebsd.org/cgi/man.cgi?query=chmod&sektion=1&format=html
[245] https://www.freebsd.org/cgi/man.cgi?query=chflags&sektion=1&format=html
[246] https://www.freebsd.org/cgi/man.cgi?query=ls&sektion=1&format=html

```
# ls -lo file1
```

```
-rw-r--r-- 1 trhodes trhodes sunlnk 0 Mar 1 05:54 file1
```

一些文件标志只能由 root 用户添加或移除。在其他情况下，文件所有者可以设置其文件标志。更多信息请参考 chflags(1)[247] 和 chflags(2)[248]。

3.4.3.setuid、setgid 和 sticky 权限

除了已经讨论过的权限之外，所有管理员都应该知道还有三个特定的设置：它们是 setuid、setgid 和 sticky 权限。

这些设置对某些 UNIX® 操作很重要，因为它们提供了通常不授予普通用户的功能。为了理解它们，必须注意真实用户 ID 和有效用户 ID 之间的区别。

真实用户 ID 是拥有或启动该进程的 UID。有效 UID 是进程运行的用户 ID。举个例子，当用户修改密码时，passwd(1)[249] 以真实用户 ID 运行。然而，为了更新密码数据库，该命令以 root 用户的有效 ID 运行。这使得用户在修改密码时不会看到 Permission Denied 的错误。

setuid 权限可以通过为用户添加符号 s 权限来添加，如下所示：

```
# chmod u+s suidexample.sh
```

setuid 权限可以通过在权限集前加上数字四（4）来设置，如下例所示：

```
# chmod 4755 suidexample.sh
```

suidexample.sh 的权限现在显示如下：

```
-rwsr-xr-x   1 trhodes  trhodes    63 Aug 29 06:36 suidexample.sh
```

请注意，s 现在是为文件所有者指定的权限集的一部分，取代了可执行位。他允许提高软件的权限，如 passwd(1)[250]。

> 注意
>
> nosuid mount(8)[251] 选项将导致这种二进制文件执行失败而不警告用户。这个选项并不完全可靠，因为 nosuid wrapper 可能会绕过它。

[247] https://www.freebsd.org/cgi/man.cgi?query=chflags&sektion=1&format=html
[248] https://www.freebsd.org/cgi/man.cgi?query=chflags&sektion=2&format=html
[249] https://www.freebsd.org/cgi/man.cgi?query=passwd&sektion=1&format=html
[250] https://www.freebsd.org/cgi/man.cgi?query=passwd&sektion=1&format=html
[251] https://www.freebsd.org/cgi/man.cgi?query=mount&sektion=8&format=html

要实时查看这个，请打开两个终端。在其中一个终端上，以普通用户的身份输入 passwd。在它等待新密码的时候，检查进程表并查看 passwd(1)[252] 的用户信息。

在终端 A：

```
Changing local password for trhodes
Old Password:
```

在终端 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
```

尽管 passwd(1)[253] 是以普通用户的身份运行的，但它使用的是有效 UID，即 root。

setgid 权限的功能与 setuid 权限的功能相同；只是它改变了组的设置。当一个应用程序或实用程序以这种设置执行时，它将被授予基于拥有文件的组的权限，而不是启动该进程的用户。

要在文件上地设置 setgid 权限标志，用 chmod[1][254] 来为该组添加 s 权限：

```
# chmod g+s sgidexample.sh
```

或者，提供 chmod[1][255]，前面加个二（2）：

```
# chmod 2755 sgidexample.sh
```

注意在下面的列表中，s 现在位于指定组权限设置的字段：

```
-rwxr-sr-x   1 trhodes  trhodes    44 Aug 31 01:49 sgidexample.sh
```

> 注意
>
> 在这些例子中，即使有关的 shell 脚本是一个可执行文件，它也不会以不同的 EUID 或有效用户 ID 运行。这是因为 shell 脚本不能使用 setuid(2)[256] 系统调用。

setuid 和 setgid 权限位可能会因为允许提升权限而降低系统的安全性。第三个特殊权限，粘滞位，可以加强系统的安全性。

当一个目录上的粘滞位被设置时，它只允许文件所有者删除文件。这对于防止不拥有文件的用户在公共目录（如 /tmp）中删除文件很有用。要利用这个权限，在文件中添加t 权限：

[252] https://www.freebsd.org/cgi/man.cgi?query=passwd&sektion=1&format=html
[253] https://www.freebsd.org/cgi/man.cgi?query=passwd&sektion=1&format=html
[254] https://www.freebsd.org/cgi/man.cgi?query=chmod&sektion=1&format=html
[255] https://www.freebsd.org/cgi/man.cgi?query=chmod&sektion=1&format=html
[256] https://www.freebsd.org/cgi/man.cgi?query=setuid&sektion=2&format=html

```
# chmod +t /tmp
```

或者，在权限集前加一个一（1）：

```
# chmod 1777 /tmp
```

粘滞位权限将在权限集的最末端显示为 t：

```
# ls -al / | grep tmp
```

```
drwxrwxrwt  10 root  wheel         512 Aug 31 01:49 tmp
```

3.5.目录结构

认识 FreeBSD 的目录结构是对系统整体理解的基础。最重要的目录是根目录即 “/”。这个目录是启动时挂载的第一个目录，它包含了为多用户操作准备的基本系统。根目录还包含其他文件系统的挂载点，这些文件系统在过渡到多用户操作时被挂载。

挂载点是一个目录，其他文件系统可以挂载到一个父文件系统（通常是根文件系统）上。这将在磁盘结构[257]中进一步说明。标准挂载点包括 /usr/、/var/、/tmp/、/mnt/ 和 /cdrom/。这些目录通常被记录到 /etc/fstab 的条目中。这个文件是一个包含各种文件系统和挂载点的表格，由系统读取。/etc/fstab 大多数文件系统都是在启动时由脚本 rc(8)[258] 自动挂载的，除非它们设置了 noauto 参数。详细内容可以在 fstab 文件[259]中找到。

关于文件系统层次结构的完整介绍可在 hier(7)[260] 中找到。下表提供了最常见的目录的简要概述。

目录	说明
/	文件系统的根目录。
/bin/	对单用户和多用户环境都很重要的用户工具。
/boot/	操作系统启动时使用的程序和配置文件。
/boot/defaults/	默认的启动配置文件。详情请参考 loader.conf(5)[261]。
/dev/	设备节点。详情请参考 intro(4)[262]。
/etc/	系统的配置文件和脚本。
/etc/defaults/	默认的系统配置文件。详情请参考 rc(8)[263]。

continues on next page

[257] https://docs.freebsd.org/en/books/handbook/book/#disk-organization

[258] https://www.freebsd.org/cgi/man.cgi?query=rc&sektion=8&format=html

[259] https://docs.freebsd.org/en/books/handbook/book/#disks-fstab

[260] https://www.freebsd.org/cgi/man.cgi?query=hier&sektion=7&format=html

表 3 – continued from previous page

目录	说明
/etc/periodic/	通过 cron(8)[264] 每日、每周和每月需要运行的脚本。详情请参考 periodic(8)[265]。
/lib/	`/bin` 和 `/sbin` 中的二进制文件需要的关键系统库
/libexec/	关键的系统文件
/media/	包含子目录，用作可移动媒体（如 CD、USB 驱动器和软盘）的挂载点。
/mnt/	空目录，通常由系统管理员作为临时挂载点使用。
/net/	自动安装的 NFS 共享,见 auto_master(5)[266]
/proc/	进程文件系统。详情请参考 procfs(5)[267]，mount_procfs(8)[268]。
/rescue/	rescue(8)[269] 中说明的用于紧急恢复的静态链接程序。
/root/	`root` 用户的主目录。
/sbin/	对单用户和多用户环境都很重要的系统程序和管理实用程序。
/tmp/	临时文件，通常在系统重启时会被清空。基于内存的文件系统通常被挂载在 **/tmp**。这可以通过 rc.conf(5)[270] 的 tmpmfs 相关变量或 **/etc/fstab** 中的条目来自动实现；详情请参考 mdmfs(8)[271]。
/usr/	大多数用户的实用程序和应用程序。
/usr/bin/	常用的实用程序、编程工具和应用程序。
/usr/include/	标准的 C 语言头文件。
/usr/lib/	库文件。
/usr/libdata/	杂项工具数据文件。
/usr/libexec/	由其他程序执行的系统守护程序和系统实用程序。
/usr/local/	本地可执行文件和库。也被用作 FreeBSD ports 框架的默认路径。在 **/usr/local** 中，应使用 hier(7)[272] 为 **/usr** 预设的一般布局。man 目录则是例外，它直接位于 **/usr/local** 而不是 **/usr/local/share** 下，而 ports 文档则于 **share/doc/port**。
/usr/obj/	通过编译 **/usr/src** 产生的特定架构目标文件。
/usr/ports/	FreeBSD ports（可选）。
/usr/sbin/	由用户执行的系统守护程序和系统实用程序。
/usr/share/	各个架构的通用文件。
/usr/src/	BSD 或本地源代码文件。
/var/	多用途的日志、临时、暂存和 spool 文件。
/var/log/	杂项系统日志文件。
/var/tmp/	除非 **/var** 是一个基于内存的文件系统，否则临时文件通常会在系统重启后保留下来。

3.6.磁盘结构

FreeBSD 用来查找文件的最小组织单位是文件名。文件名区分大小写，这意味着 **readme.txt** 和 **README.TXT** 是两个独立的文件。FreeBSD 并不使用文件的扩展名来确定该文件是程序、文档还是其他形式的数据。

文件被存储在目录中。一个目录可能不包含任何文件，也可能包含数百个文件。目录之中也可以包含其他目录，在目录之间构建层次结构，以便组织数据。

文件和目录的引用方法是给出文件或目录的名称，后面是一个正斜杠，/，后面是任何其他必要的目录名称。例如，如果目录 **foo** 包含了一个目录 **bar**，这个目录又包含了一个文件叫 **readme.txt**，那么该文件的全名或 *路径*就是 **foo/bar/readme.txt**。请注意，这与 Windows® 不同，后者使用 \ 来分隔文件和目录名。FreeBSD 在路径中不使用磁盘字母或其磁盘名称。例如，在 FreeBSD 上，人们不会输入 ``**c:\foo\bar\readme.txt**``。

3.6.1.文件系统

目录和文件存储在文件系统中。每个文件系统在最顶层都有唯一的一个目录，称为该文件系统的 *根目录*。这个根目录可以包含其他目录。会有一个文件系统被指定为 *根文件系统*或 /，然后其他所有的文件系统都 *挂载*在根文件系统下。无论 FreeBSD 系统上有多少个磁盘，所有目录看起来都是同一个磁盘的一部分。

假设有三个文件系统，分别称为 A、B 和 C。每个文件系统有一个根目录，其中包含另外两个目录，称为 A1、A2（同样地有，B1、B2 和 C1、C2）。

称 A 为根文件系统。如果用 ls(1)[273] 来查看这个目录的内容，它将显示两个子目录，A1 和 A2。该目录树看起来像这样：

[261] https://www.freebsd.org/cgi/man.cgi?query=loader.conf&sektion=5&format=html
[262] https://www.freebsd.org/cgi/man.cgi?query=intro&sektion=4&format=html
[263] https://www.freebsd.org/cgi/man.cgi?query=rc&sektion=8&format=html
[264] https://www.freebsd.org/cgi/man.cgi?query=cron&sektion=8&format=html
[265] https://www.freebsd.org/cgi/man.cgi?query=periodic&sektion=8&format=html
[266] https://www.freebsd.org/cgi/man.cgi?query=auto_master&sektion=5&format=html
[267] https://www.freebsd.org/cgi/man.cgi?query=procfs&sektion=5&format=html
[268] https://www.freebsd.org/cgi/man.cgi?query=mount_procfs&sektion=8&format=html
[269] https://www.freebsd.org/cgi/man.cgi?query=rescue&sektion=8&format=html
[270] https://www.freebsd.org/cgi/man.cgi?query=rc.conf&sektion=5&format=html
[271] https://www.freebsd.org/cgi/man.cgi?query=mdmfs&sektion=8&format=html
[272] https://www.freebsd.org/cgi/man.cgi?query=hier&sektion=7&format=html
[273] https://www.freebsd.org/cgi/man.cgi?query=ls&sektion=1&format=html

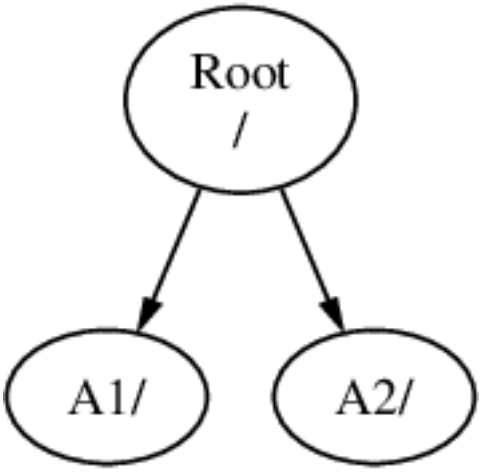


图45: 有根目录和两个子目录（A1 和 A2）的目录树

一个文件系统必须以目录形式被挂载到另一个文件系统上。当把文件系统 `B` 挂载到 `A1` 的目录上时，`B` 的根目录变成了 `A1`，`B` 中的目录也相应地发生改变：

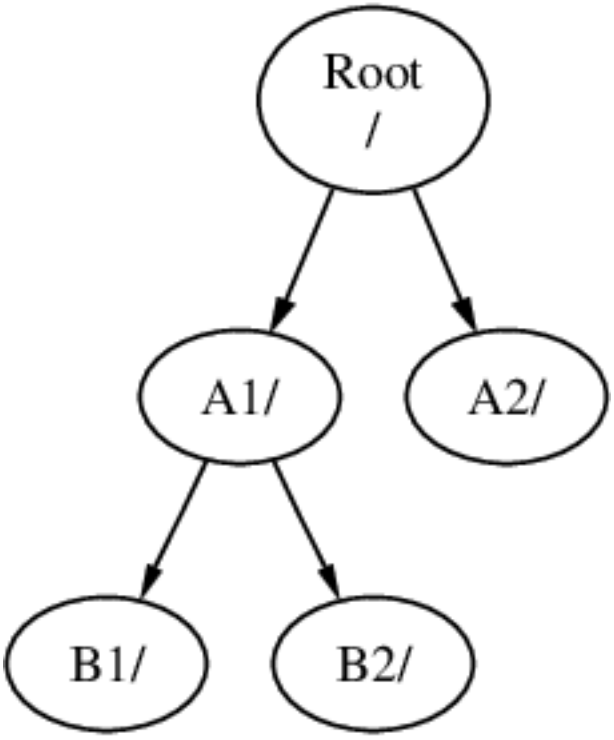


图46: 目录树有根目录和两个子目录，A1 和 A2。还有更多的子目录，B1 和 B2 挂在 A1 上

任何位于 `B1` 或 `B2` 目录下的文件都必须经过路径 **/A1/B1** 或 **/A1/B2** 才能到达。任何在 `/A1` 中原有的文

件都被暂时隐藏了。如果把 B 从 A 上 卸载下来，它们将重新出现。

如果 B 被挂载在 A2 上，那么图片会是这样的：

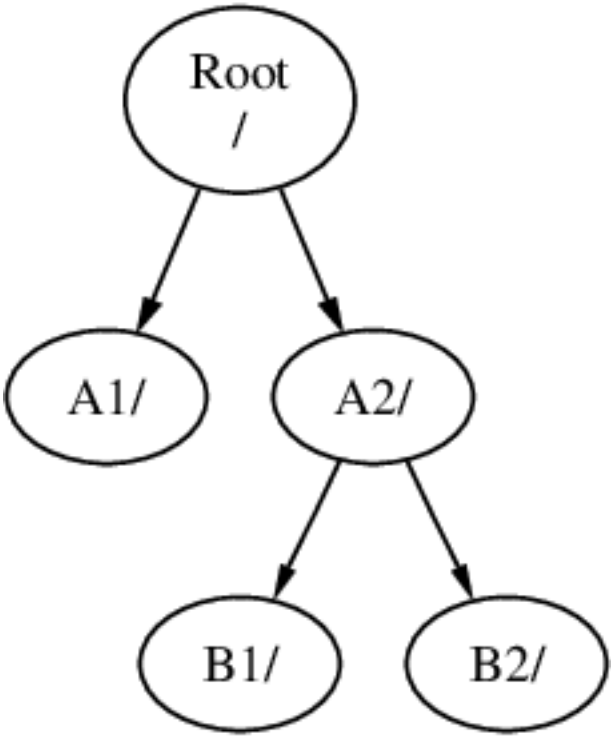


图47: 目录树有根目录和两个子目录，A1 和 A2。还有更多的子目录，B1 和 B2 挂在 A2 上

而路径将分别为 **/A2/B1** 和 **/A2/B2**。

文件系统可以被挂载在其他文件的之上。继续上一个例子，文件系统 C 可以挂载在文件系统 B 的 B1 目录之上，从而形成这种安排：

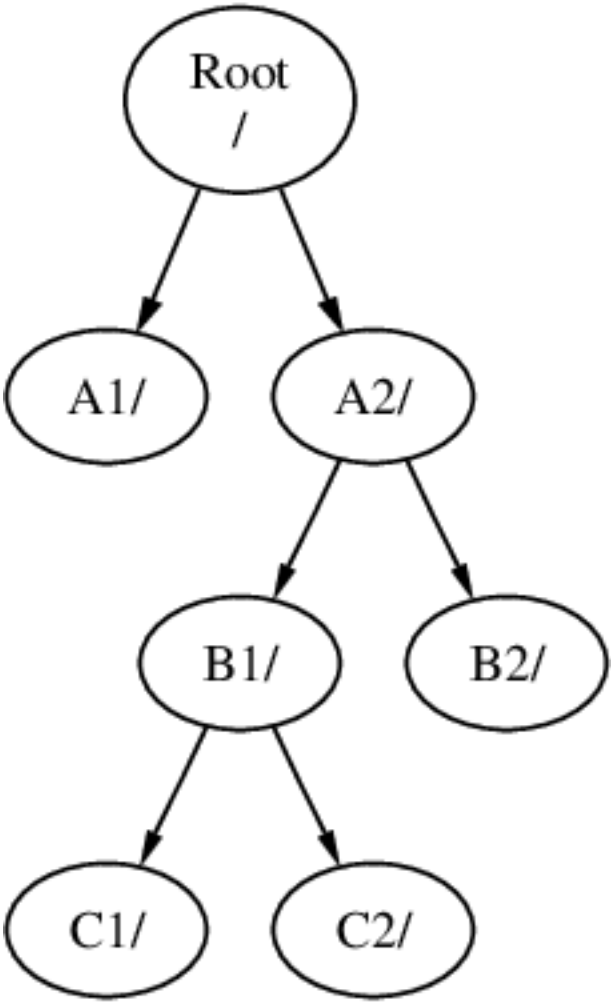


图48: 一个复杂的目录树。有不同的子目录挂在根目录上。

或者可以直接挂载 C 到文件系统 A 的 A1 目录下:

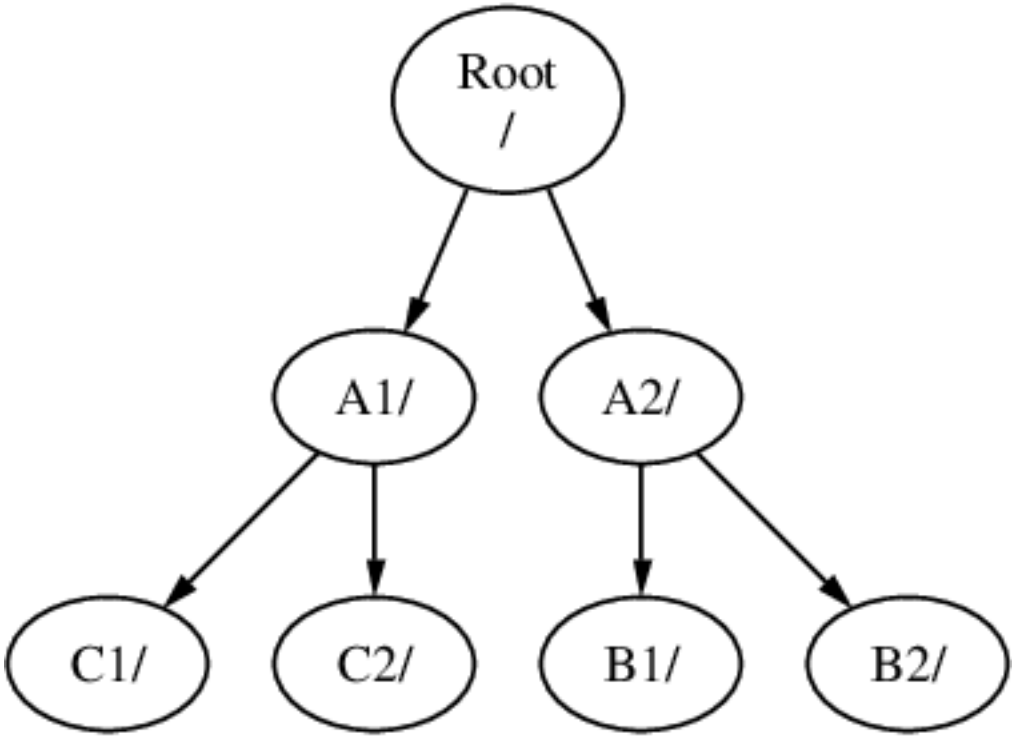


图49: 一个复杂的目录树。有不同的子目录挂在根目录上。

完全可以有一个大的根文件系统，而不需要创建任何其他的文件系统。这种方法有既有缺点，也有优点。

多个文件系统的好处

- 不同的文件系统可以有不同的 *挂载选项*。例如，根文件系统可以被挂载为只读，避免用户在无意中删除或编辑一个关键文件。将用户可写的文件系统，如 **/home**，与其他文件系统分开，允许它们被挂载为 *nosuid*。该选项可以使存储在文件系统上的可执行文件的 *suid/guid* 位失效，从而可能提高安全性。
- FreeBSD 会根据文件系统的使用情况，自动优化文件系统上的文件分布。因此，包含许多经常被写入小文件的文件系统与只包含几个大文件的文件系统的优化方式是不同的。如果一个大的文件系统，这种优化将不存在。
- FreeBSD 的文件系统在断电的情况下依旧能保持稳定。然而，在关键时刻上断电仍然可能会破坏文件系统的结构。通过将数据分散存储到多个文件系统中，会提高文件系统在意外断电后仍能正常工作的可能性，并且在必要时更容易从备份中恢复。

单一文件系统的好处

- 文件系统的大小是固定的。如果你在安装 FreeBSD 时创建了一个文件系统，并给了它一个特定的大小，你可能会在后来发现你需要把这个分区做得更大。如果不进行备份，以新的大小重新创建文件系统，然后再恢复备份的数据，这是很难做到的。

  重要

FreeBSD 的特色命令 growfs(8)[274]，使其可以即时增加文件系统的大小，从而消除这一限制。。一个文件系统只能被扩展到它所在的分区中的空闲空间。如果分区之后还有空间，可以用 gpart(8)[275] 来扩展该分区。如果该分区是虚拟磁盘上的最后一个分区，并且该磁盘被扩展，那么该分区就可以被扩展。

3.6.2.磁盘分区

文件系统包含在分区中。磁盘是用几种分区方案中的一种划分为分区的；见手动分区[276]。较新的方案是 GPT；基于 BIOS 的老式计算机使用 MBR。GPT 支持将磁盘划分为具有一定大小、偏移量和类型的分区。它支持大量的分区和分区类型，只要有可能就推荐使用它。GPT 分区使用带有后缀的磁盘名称，其中后缀为 `p1` 代表第一个分区，`p2` 代表第二个分区，以此类推。然而，MBR 只支持少量的分区。MBR 分区在 FreeBSD 中被称为 `slices`。slices 可以用于不同的操作系统。FreeBSD 的分片是用 BSD 的标签（见 bsdlabel(8)[277]）来细分分区的。

slices 编号跟随设备名称，以 `s` 为前缀，从 1 开始。所以 “da0s1” 是第一个 SCSI 驱动器上的第一个 slices。一个磁盘上只能有四个物理片，但在适当类型的物理片内可以有逻辑片。这些扩展片的编号从 5 开始，所以 “ada0s5” 是第一个 SATA 磁盘上的第一个扩展分区。这些设备被期望占据一个 slices 的文件系统所使用。

每个 GPT 或 BSD 分区只能包含一个文件系统，这意味着文件系统通常由它们在文件系统层次结构中的典型挂载点或它们所包含的分区的名称来描述。

FreeBSD 还使用磁盘空间作为 *交换空间* 来提供 *虚拟内存*。这使你的计算机表现得好像它的内存比实际的要多得多。当 FreeBSD 的内存用完时，它会把一些目前没有被使用的数据移到交换空间，当它需要时再把它移回来（把别的东西移出来）。这被称为分页。

有些 BSD 分区有某些与之相关的惯例。

分区	惯例
`a`	通常包含根文件系统。
`b`	通常包含交换空间。
`c`	通常情况下，它的大小与主分区相同。这可使需要在整个主分区上工作的工具，如坏块扫描器，在分区 `c` 上工作。通常不会在这个分区上创建文件系统。
`d`	分区 `d` 曾经有一个与之相关的特殊含义，虽然现在已经消失了，所以现在 `d` 和一般的分区作用相同。

slice、“危险专用” 物理驱动器和其他驱动器包含分区，这些分区用从 `a` 到 `h` 的字母表示。这个字母被附加到设备名称上，所以 `da0a` 是第一个da 磁盘上的 `a` 分区，它是 “危险专用”。“ada1s3e” 是第二个 SATA 磁盘驱动器第三个 slice 中的第五个分区。

[274] https://www.freebsd.org/cgi/man.cgi?query=growfs&sektion=8&format=html
[275] https://www.freebsd.org/cgi/man.cgi?query=gpart&sektion=8&format=html
[276] https://docs.freebsd.org/en/books/handbook/book/#bsdinstall-part-manual
[277] https://www.freebsd.org/cgi/man.cgi?query=bsdlabel&sektion=8&format=html

最后，系统中的每个磁盘都被识别。磁盘名称的开头是一个表示磁盘类型的代码，然后是一个数字，表示它是哪一个磁盘。与分区和 Slice 不同，磁盘的编号从 0 开始，常见的代码在磁盘设备名称[278]中列出。

当提及一个分区时，应包括磁盘名称、s、slice 编号，然后是分区字母。示例为磁盘、Slice 和分区名称[279]。GPT 分区包括磁盘名称，p，然后是分区编号。

磁盘的概念模型[280]展示了一个磁盘布局的概念模型。

在安装 FreeBSD 时，如果使用 MBR，请配置磁盘 slices，并在 slices 内创建用于 FreeBSD 的分区。如果使用 GPT，为每个文件系统配置分区。无论哪种情况，都要在每个分区中创建一个文件系统或交换空间，并决定每个文件系统的挂载位置。请参阅 gpart(8)[281] 以了解关于操作分区的信息。

表 4. 磁盘设备名称

设备类型	驱动器设备名称
SATA 和 IDE 磁盘驱动器	`ada`
SCSI 磁盘驱动器和 USB 存储设备	`da`
NVMe 存储	`nvd` 或 `nda`
SATA 与 IDE CD-ROM 驱动器	`cd`
SCSI CD-ROM 驱动器	`cd`
软盘驱动器	`fd`
SCSI 磁带机	`sa`
RAID 驱动器	例如，Adaptec® AdvancedRAID 的 `aacd`，Mylex® 的 `mlxd` 和 `mlyd`，AMI MegaRAID® 的 `amrd`，Compaq Smart RAID 的 `idad`，3ware® RAID 的 `twed`。

表 **5.** 磁盘、**Slice** 和分区名称示例

命名	意义
`ada0s1a`	第一块 SATA 硬盘（`ada0`）上第一个 slice（`s1`）的第一个分区（`a`）。
`da1s2e`	第二块 SCSI 磁盘（`da1`）上的第二个 slice（`s2`）的第五个分区（`e`）。

例 **13.** 磁盘的概念模型

这张图显示了 FreeBSD 对连接到系统的第一个 SATA 磁盘的内部视角。假设该磁盘的大小为 250 GB，包含一个 80 GB 的分区和一个 170 GB 的 slice（MS-DOS® 分区）。第一个 slice 包含一个 Windows® NTFS 文件系统 **C**:，第二个 slice 包含一个 FreeBSD 系统。这个 FreeBSD 系统实例有四个数据分区和一个交换分区。

[278] https://docs.freebsd.org/en/books/handbook/book/#disks-naming

[279] https://docs.freebsd.org/en/books/handbook/book/#basics-disk-slice-part

[280] https://docs.freebsd.org/en/books/handbook/book/#basics-concept-disk-model

[281] https://www.freebsd.org/cgi/man.cgi?query=gpart&sektion=8&format=html

这四个分区分别存放一个文件系统。分区 a 用于 / 文件系统，d 用于 **/var/**，e 用于 **/tmp/**，f 用于 **/usr/**。分区字母 c 指的是整个 slice，所以不用于普通分区。

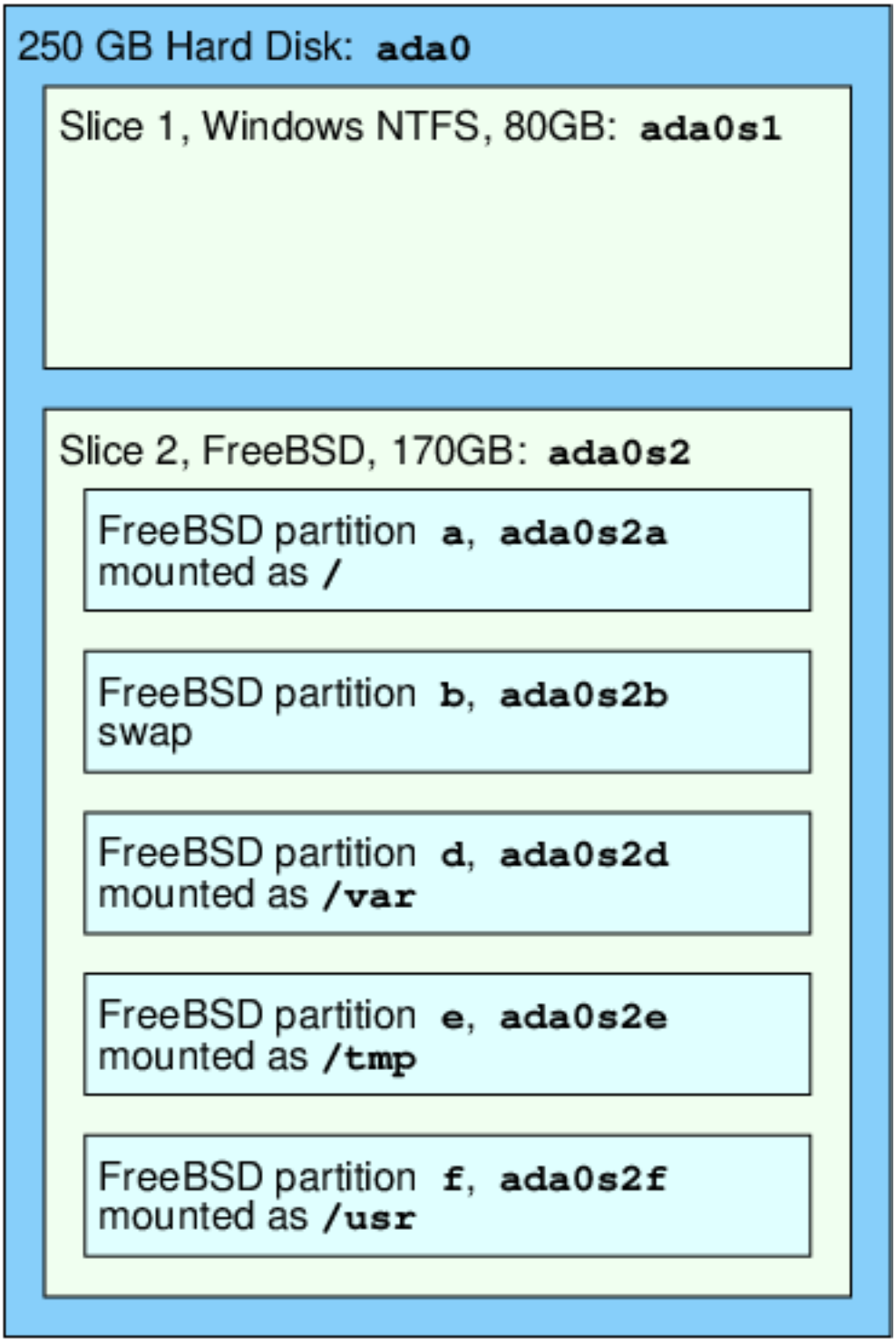


图50: Windows 和 FreeBSD 之间共享驱动器的布局

3.7.文件系统的挂载与卸载

文件系统就像是一棵树，它的根是 **/**。**/dev**、**/usr** 和根目录中的其他目录是其分支，它们可能还有自己的分支，如 **/usr/local** 等等。

将一些目录放在不同的文件系统中是有各种原因的。**/var** 包含了 **log/**、**spool/** 和各种类型的临时文件，因此，可能会被填满。填满根文件系统并不是一个好主意，所以将 **/var** 从 **/** 中剥离出来通常是有好处的。

将一些目录放在不同的文件系统中的另一个常见原因是，它们要被放置在单独的物理磁盘或者是单独的虚拟磁盘上，比如 “网络文件系统（NFS）”[282]中说明的挂载的网络文件系统，或者 CDROM 驱动器。

3.7.1. fstab 文件

在引导过程中（FreeBSD 的引导过程[283]），除了包含 `noauto` 的条目外，**/etc/fstab** 中列出的文件系统都会自动挂载。这个文件包含以下格式的条目：

```
device       /mount-point fstype     options      dumpfreq     passno
```

device

一个现有的设备名称，如磁盘设备名称[284]中所解释的。

mount-point

一个现有的目录，用来挂载文件系统。

fstype

要传递给 mount(8)[285] 的文件系统类型。FreeBSD 默认的文件系统是 `ufs`。

options

`rw` 表示可读写文件系统，`ro` 表示只读文件系统，后面是可能需要的任何其他选项。一个常见的选项是 `noauto`，用于不需要在启动过程中进行挂载的文件系统。其他选项在 mount(8)[286] 中列出。

dumpfreq

被 dump(8)[287] 用来确定哪些文件系统需要转储。如果该字段缺失，则假定其值为 `0`。

passno

决定 UFS 文件系统在重启后应被 fsck(8)[288] 检查的顺序。应该被跳过的文件系统应该把它们的 `passno` 设置为 `0`。根文件系统需要在其他所有文件系统之前被检查，它的 `passno` 应该被设置为 `1`。其他文件系统

[282] https://docs.freebsd.org/en/books/handbook/network-servers/index.html#network-nfs
[283] https://docs.freebsd.org/en/books/handbook/boot/index.html#boot
[284] https://docs.freebsd.org/en/books/handbook/book/#disks-naming
[285] https://www.freebsd.org/cgi/man.cgi?query=mount&sektion=8&format=html
[286] https://www.freebsd.org/cgi/man.cgi?query=mount&sektion=8&format=html
[287] https://www.freebsd.org/cgi/man.cgi?query=dump&sektion=8&format=html
[288] https://www.freebsd.org/cgi/man.cgi?query=fsck&sektion=8&format=html

应该被设置为大于 1 的值。如果多个文件系统有相同的 passno，fsck(8)[289] 将尽可能地尝试并行检查文件系统。

关于 **/etc/fstab** 的格式和选项的更多信息，请参考 fstab(5)[290]。

3.7.2.使用 mount(8)[291]

文件系统是用 mount(8)[292] 挂载的。最基本的语法如下：

```
# mount device mountpoint
```

在 /etc/fstab 中列出的文件系统也可以通过提供挂载点来挂载。

该命令提供了许多选项，在 mount(8)[293] 中有所概述，最常用的选项包括：

挂载选项

-a

挂载 **/etc/fstab** 中列出的所有文件系统，除了那些标记为 noauto、被 -t 标志排除的文件系统，或者已经挂载的文件系统。

-d

执行所有操作但不进行实际的 mount 系统调用。这个选项和 -v 标志一起使用，可以确定 mount(8)[294] 预期要做什么。

-f

挂载文件系统为只读。这与使用 -o ro 是一样的。

-t fstype

挂载指定的文件系统类型或只挂载指定类型的文件系统，如果包括 -a。默认的文件系统类型是“ufs”。

-u

更新文件系统的挂载选项。

-v

详细模式。

-w

对文件系统进行读写挂载。

[289] https://www.freebsd.org/cgi/man.cgi?query=fsck&sektion=8&format=html
[290] https://www.freebsd.org/cgi/man.cgi?query=fstab&sektion=5&format=html
[291] https://www.freebsd.org/cgi/man.cgi?query=mount&sektion=8&format=html
[292] https://www.freebsd.org/cgi/man.cgi?query=mount&sektion=8&format=html
[293] https://www.freebsd.org/cgi/man.cgi?query=mount&sektion=8&format=html
[294] https://www.freebsd.org/cgi/man.cgi?query=mount&sektion=8&format=html

以下选项可以作为逗号分隔的列表传递给 `-o`。

nosuid

不要读取文件系统上的 `setuid` 或 `setgid` 标志。这也是一个有用的安全选项。

3.7.3.使用 umount(8)[Page 121, 295]

要卸载文件系统，可使用 umount(8)[296]。这个命令需要一个参数，可以是挂载点、设备名称、`-a` 或 `-A`。

所有的命令形式都是用 `-f` 来强制卸载，用 `v` 来表示详细情况。请注意，`-f` 通常不是一个好主意，因为它可能使计算机崩溃或损坏文件系统上的数据。

要卸载所有挂载的文件系统，或只卸载 `-t` 后面列出的文件系统类型，请使用 `-a` 或 `-A`。注意，`-A` 不会试图卸载根文件系统。

3.8.进程和守护进程

FreeBSD 是一个多任务的操作系统。每个正在运行的程序都被称为 *进程*。每个正在运行的命令都至少会启动一个新的进程，并且有许多系统进程因 FreeBSD 而运行。

每个进程都由一个称为进程 *ID*（*PID*）的唯一数字来识别。与文件类似，每个进程都有一个所有者和组，所有者和组的权限被用来决定进程可以打开哪些文件和设备。大多数进程也有一个启动它们的父进程。例如，shell 是一个进程，在 shell 中启动的任何命令都是一个以 shell 为父进程的进程。例外的是一个叫做 init(8)[297] 的特殊进程，它是在启动时第一个启动的进程，它的 PID 总是为 `1`。

有些程序的设计不是为了在用户连续输入的情况下运行，而是一有机会就断开与终端的连接。例如，网络服务器响应的是网络请求，而非用户输入。邮件服务器是这种类型的应用程序的另一个例子。这些类型的程序被称为 *守护进程*。守护进程这一术语来自希腊神话，代表了一个既非善也非恶的存在，却在无形中执行着有用的任务。这就是为什么 BSD 的吉祥物是一个穿着帆布鞋、拿着干草叉、看起来很欢快的小恶魔。

有一个惯例是那些作为守护进程运行的程序通常在尾部会加一个字母的“d”。例如，BIND 是 Berkeley Internet Name Domain，但实际执行的程序被命名为 `named`。Apache 网络服务器程序是 `httpd`，行式打印机的 spooling daemon 是 `lpd`。这只是一个命名惯例，现实情况并不总是这样。例如，sendmail 应用程序的主要邮件守护进程是 `sendmail`，而不是 `maild`。

[295] https://www.freebsd.org/cgi/man.cgi?query=umount&sektion=8&format=html

[296] https://www.freebsd.org/cgi/man.cgi?query=umount&sektion=8&format=html

[297] https://www.freebsd.org/cgi/man.cgi?query=init&sektion=8&format=html

3.8.1.查看进程

要查看系统中运行的进程，使用 ps(1)[298] 或 top(1)[299]。要显示当前正在运行的进程的静态列表、它们的 PID、它们正在使用多少内存以及它们是用什么命令启动的，使用 ps(1)[300]。要显示所有正在运行的进程并每隔几秒钟更新一次，以便交互式地查看计算机正在做什么，使用 top(1)[301]。

默认情况下，ps(1)[302] 只显示正在运行并由用户拥有的命令。比如：

```
% ps
```

输出结果应类似于下面的内容：

```
 PID TT  STAT    TIME COMMAND
8203  0  Ss   0:00.59 /bin/csh
8895  0  R+   0:00.00 ps
```

ps(1)[303] 的输出被组织成若干列。`PID` 列显示进程的 ID。PID 从 `1` 开始分配，一直到 `99999`，然后再绕回开头。然而，如果一个 PID 已经被使用，则不会被重新分配。`TT` 列显示了程序正在运行的 tty，`STAT` 显示了程序的状态。`TIME` 是该程序在 CPU 上运行的时间。这通常不是程序启动后所经过的时间，因为大多数程序在需要在 CPU 上花费时间之前，会花很多时间来等待事情的发生。最后，`COMMAND` 是用来启动程序的命令。

有许多不同的选项可以改变所显示的信息。其中最有用的一组是 `auxww`，`a` 显示所有用户的所有运行进程的信息，`u` 显示进程所有者的用户名和内存使用情况，`x` 显示守护进程的信息，`ww` 强制使 ps(1)[304] 显示每个进程的完整命令行，不因为它太长、无法在屏幕上显示而截断它。

top(1)[305] 的输出也类似：

```
% top
```

输出结果应类似于下面的内容：

```
last pid:  9609;  load averages:  0.56,  0.45,  0.36              up 0+00:20:03 ␣
↪10:21:46
107 processes: 2 running, 104 sleeping, 1 zombie
CPU:  6.2% user,  0.1% nice,  8.2% system,  0.4% interrupt, 85.1% idle
Mem: 541M Active, 450M Inact, 1333M Wired, 4064K Cache, 1498M Free
```

(continues on next page)

[298] https://www.freebsd.org/cgi/man.cgi?query=ps&sektion=1&format=html
[299] https://www.freebsd.org/cgi/man.cgi?query=top&sektion=1&format=html
[300] https://www.freebsd.org/cgi/man.cgi?query=ps&sektion=1&format=html
[301] https://www.freebsd.org/cgi/man.cgi?query=top&sektion=1&format=html
[302] https://www.freebsd.org/cgi/man.cgi?query=ps&sektion=1&format=html
[303] https://www.freebsd.org/cgi/man.cgi?query=ps&sektion=1&format=html
[304] https://www.freebsd.org/cgi/man.cgi?query=ps&sektion=1&format=html
[305] https://www.freebsd.org/cgi/man.cgi?query=top&sektion=1&format=html

(continued from previous page)

```
ARC: 992M Total, 377M MFU, 589M MRU, 250K Anon, 5280K Header, 21M Other
Swap: 2048M Total, 2048M Free

  PID USERNAME    THR PRI NICE   SIZE    RES STATE   C   TIME   WCPU COMMAND
  557 root          1 -21  r31   136M 42296K select  0   2:20  9.96% Xorg
 8198 dru           2  52    0   449M 82736K select  3   0:08  5.96% kdeinit4
 8311 dru          27  30    0  1150M   187M uwait   1   1:37  0.98% firefox
  431 root          1  20    0 14268K  1728K select  0   0:06  0.98% moused
 9551 dru           1  21    0 16600K  2660K CPU3    3   0:01  0.98% top
 2357 dru           4  37    0   718M   141M select  0   0:21  0.00% kdeinit4
 8705 dru           4  35    0   480M    98M select  2   0:20  0.00% kdeinit4
 8076 dru           6  20    0   552M   113M uwait   0   0:12  0.00% soffice.bin
 2623 root          1  30   10 12088K  1636K select  3   0:09  0.00% powerd
 2338 dru           1  20    0   440M 84532K select  1   0:06  0.00% kwin
 1427 dru           5  22    0   605M 86412K select  1   0:05  0.00% kdeinit4
```

输出被分成两部分。标题（前五或六行）显示了最后一个运行的进程的 PID，系统负载平均数（衡量系统有多忙），系统运行时间（自上次重启后的时间）和当前时间。标题中的其他数字涉及到有多少进程正在运行，有多少内存和交换空间被使用，以及系统在不同的 CPU 状态下花费了多少时间。如果已经加载 ZFS 文件系统模块，ARC 行表明有多少数据是从内存缓存中而不是从磁盘中读取的。

在标题下面是一系列列，包含与 ps(1)[306] 输出的类似信息，例如 PID、用户名、CPU 时间和启动进程的命令。默认情况下，top(1)[307] 还显示进程占用的内存空间。这被分成两列：一列是总大小，一列是常驻大小。总大小是指应用程序需要多少内存，常驻大小是指它现在实际使用多少内存。

top(1)[308] 每两秒钟自动更新一次显示。可以用 `-s` 指定不同的间隔时间。

3.8.2.结束进程

与正在运行的进程或守护进程沟通的一种方式是使用 kill(1)[309] 发送 信号。有许多不同的信号；一些信号有特定的含义，而其他信号则在应用程序的文档中介绍。用户只能向自己的进程发送信号，向别人的进程发送信号会导致权限拒绝的错误。但 `root` 用户是个例外，他可以向任何人的进程发送信号。

操作系统也可以向进程发送一个信号。如果一个应用程序写得不好，并试图访问它不应该访问的内存，FreeBSD 将向该进程发送 `Segmentation Violation` 信号（`SIGSEGV`）。如果一个应用程序被写成使用 alarm(3)[310] 系统调用，在一段时间后被提醒，它将被发送“Alarm”信号（`SIGALRM`）。

有两个信号可以用来停止一个进程：`SIGTERM` 和 `SIGKILL`。`SIGTERM` 是杀死一个进程的温和方式，因为该进程可以读取该信号，关闭它可能打开的任何日志文件，并试图在关闭前完成它正在做的事情。在某些

[306] https://www.freebsd.org/cgi/man.cgi?query=ps&sektion=1&format=html
[307] https://www.freebsd.org/cgi/man.cgi?query=top&sektion=1&format=html
[308] https://www.freebsd.org/cgi/man.cgi?query=top&sektion=1&format=html
[309] https://www.freebsd.org/cgi/man.cgi?query=kill&sektion=1&format=html
[310] https://www.freebsd.org/cgi/man.cgi?query=alarm&sektion=3&format=html

情况下，如果一个进程正在进行一些不能被中断的任务，它可能会忽略 SIGTERM。

SIGKILL 不能被进程所忽略。向一个进程发送 SIGKILL 通常会使该进程立即停止。[注 1]

其他常用的信号有 SIGHUP、SIGUSR1 和 SIGUSR2。由于这些是通用的信号，不同的应用程序会有不同的反应。

例如，在改变了一个 Web 服务器的配置文件后，需要告诉 Web 服务器重新读取其配置。重启 httpd 会导致网络服务器出现短暂的中断期。相反，向守护进程发送 SIGHUP 信号可以取代重启。请注意，不同的守护进程会有不同的行为，所以请参考守护进程的文档，以确定 SIGHUP 是否能达到预期效果。

重要

杀死系统中的随机进程是一个坏主意。特别是 init(8)[311]，PID 为 1，很特别。运行 /bin/kill -s KILL 1 是一个快速的、不推荐的关闭系统的方法。在按下回车键之前，一定要仔细检查 kill(1)[312] 的参数。

注 **1**

有几个任务是不能被中断的。例如，如果该进程试图从网络中的另一台计算机上读取文件，而另一台计算机不可用，那么该进程就被称为不可中断的。最终该进程将超时（通常在两分钟后）。一旦发生超时，该进程将被杀死。

3.9.Shell

shell 提供了一个与操作系统进行交互的命令行界面。shell 从输入通道接收命令并执行它们。许多 shell 提供了内置的功能来帮助完成日常任务，如文件管理、文件通配符、命令行编辑、命令宏和环境变量。FreeBSD 自带了几个 shell，包括 Bourne shell（sh(1)[313]）和扩展 C shell（tcsh(1)[314]）。其他 shell 可以从 FreeBSD ports 中获得，例如 zsh 和 bash。

shell 的选用实际上是一个品味问题。C 语言程序员可能觉得使用类似 C 语言的 shell 更舒服，比如 tcsh(1)[315]。Linux® 用户可能更喜欢 bash。每个 shell 都有独特的属性，可能与用户喜欢的工作环境配合，也可能不配合，这就是选择使用哪个 shell 的原因了。

一个常见的 shell 的功能是文件名完成。当用户输入一个命令或文件名的前几个字母并按下 Tab 键后，shell 将完成该命令或文件名的其余部分。假设有两个分别名为 **foobar** 和 **football** 的文件。要删除 **foobar**，用户可以输入 rm foo 并按 Tab 键来完成文件名。

但 shell 只显示 rm foo。它无法完成文件名，因为 **foobar** 和 **football** 都以 foo 开头。如果有多个名字匹配，有些 shell 会发出哔哔音效或显示所有的选择。然后用户必须输入更多的字符来确定所需的文件名。输入一个 t 并再次按 Tab 键，就足以让 shell 确定所需的文件名并填入其余部分。

[311] https://www.freebsd.org/cgi/man.cgi?query=init&sektion=8&format=html
[312] https://www.freebsd.org/cgi/man.cgi?query=kill&sektion=1&format=html
[313] https://www.freebsd.org/cgi/man.cgi?query=sh&sektion=1&format=html
[314] https://www.freebsd.org/cgi/man.cgi?query=tcsh&sektion=1&format=html
[315] https://www.freebsd.org/cgi/man.cgi?query=tcsh&sektion=1&format=html

shell 的另一个特点是使用环境变量。环境变量是存储在 shell 环境中的一对变量或键值。这个环境可以被 shell 调用的任何程序所读取，因此包含了很多程序的配置。常见的环境变量[316]提供了一个常见的环境变量的列表和它们的含义。注意，环境变量的名称总是大写的。

表 6. 常见的环境变量

环境变量	含义
USER	当前用户的登录名。
PATH	用冒号分隔的目录列表，以搜索二进制文件。
DISPLAY	如果有的话，则代表要连接的 Xorg 显示器的网络名称
SHELL	当前的 shell。
TERM	用户的终端类型的名称。用于确定终端的功能。
TERMCAP	在数据库中输入终端转义代码以执行各种终端功能。
OSTYPE	操作系统的类型。
MACHTYPE	系统的 CPU 架构。
EDITOR	用户的首选文本编辑器。
PAGER	用户每次查看文本的首选工具。
MANPATH	以冒号分隔的目录列表，用于搜索手册页面。

不同的 shell 设置环境变量的方法也是不同的。在 tcsh(1)[317] 和 csh(1)[318] 中，使用 `setenv` 来设置环境变量。在 sh(1)[319] 和 `bash` 中，使用 export 来设置当前环境变量。这个例子将 shell tcsh(1)[320] 的默认 `EDITOR` 设置为 **/usr/local/bin/emacs**：

```
% setenv EDITOR /usr/local/bin/emacs
```

`bash` 的对应命令是：

```
% export EDITOR="/usr/local/bin/emacs"
```

要展开一个环境变量以查看其当前设置，请在命令行上的名称前输入 `$` 字符。例如，`echo $TERM` 显示当前的 `$TERM` 设置。

shell 将特殊字符（称为元字符）视为数据的特殊代表。最常见的元字符是 `*`，它代表文件名中的任何数量的字符。元字符可以用来执行文件名正则化。例如，`echo *` 等同于 `ls`，因为 shell 将所有与 `*` 相匹配的文件拿出来，`echo` 将它们打印在命令行上。

为了防止 shell 解释一个特殊的字符，可以用反斜杠 () 开头来转义。例如，`echo $TERM` 打印终端设置，而 `echo\$TERM` 则是打印字符串`$TERM`。

[316] https://docs.freebsd.org/en/books/handbook/book/#shell-env-vars

[317] https://www.freebsd.org/cgi/man.cgi?query=tcsh&sektion=1&format=html

[318] https://www.freebsd.org/cgi/man.cgi?query=csh&sektion=1&format=html

[319] https://www.freebsd.org/cgi/man.cgi?query=sh&sektion=1&format=html

[320] https://www.freebsd.org/cgi/man.cgi?query=tcsh&sektion=1&format=html

3.9.1.改变 shell

永久改变默认 shell 的最简单方法是使用 `chsh`。运行这个命令将打开在 `EDITOR` 环境变量中配置的编辑器，默认情况下，它被设置为 vi(1)[321]。需要将 `Shell:` 一行改为新 Shell 的完整路径。

另外，使用 `chsh -s` 可以在不打开编辑器的情况下设置指定的 shell。例如，要把 shell 改为 `bash`：

```
% chsh -s /usr/local/bin/bash
```

在提示下输入密码，然后按 回车键更改 shell。注销并重新登录以开始使用新的 shell。

> 注意
>
> 新的 shell 必须被列在 **/etc/shells** 中。如果 shell 是按照安装应用程序：软件包和 Ports[322] 中说明的从 FreeBSD ports 安装的。它应该会被自动添加到这个文件中。如果没有，请用这个命令添加它（用 shell 路径代替下方路径）：
>
> ```
> # echo /usr/local/bin/bash >> /etc/shells
> ```
>
> 然后，重新运行 chsh(1)[323]。

3.9.2.高级 shell 技巧

UNIX® shell 不仅仅是一个命令解释器，作为一个强大的工具，他允许用户执行命令，重定向其输出，重定向其输入，并将命令连锁起来，以提高最终的命令输出。当这种功能与内置的命令混合在一起时，就为用户提供了一个可以使效率最大化的环境。

shell 重定向是将一条命令的输出或输入发送到另一条命令或文件中的行为。例如，要把 ls(1)[324] 命令的输出抓到一个文件中去，就要把输出重定向：

```
% ls > directory_listing.txt
```

现在目录内容将被列在 **directory_listing.txt** 中。有些命令可以用来读取输入，比如 sort(1)[325]。要对这个列表进行排序，请重定向输入：

```
% sort < directory_listing.txt
```

输入将被排序并放在屏幕上。为了将该输入重定向到另一个文件，可以通过混合方向来重定向 sort(1)[326] 的输出：

[321] https://www.freebsd.org/cgi/man.cgi?query=vi&sektion=1&format=html
[322] https://docs.freebsd.org/en/books/handbook/ports/index.html#ports
[323] https://www.freebsd.org/cgi/man.cgi?query=chsh&sektion=1&format=html
[324] https://www.freebsd.org/cgi/man.cgi?query=ls&sektion=1&format=html
[325] https://www.freebsd.org/cgi/man.cgi?query=sort&sektion=1&format=html
[326] https://www.freebsd.org/cgi/man.cgi?query=sort&sektion=1&format=html

```
% sort < directory_listing.txt > sorted.txt
```

在前面所有的例子中，命令都是使用文件描述符进行重定向的。每个 UNIX® 系统都有文件描述符，包括标准输入（stdin）、标准输出（stdout）和标准错误（stderr）。每一个都有一个目的，其中输入可以是键盘或鼠标，是提供输入的东西。输出可以是屏幕或打印机中的纸张。而错误则是任何用于诊断或错误信息的东西。这三个都被认为是基于 I/O 的文件描述符，有时也被认为是流。

通过使用这些描述符，shell 可通过各种命令传递输出和输入，并重定向到一个文件或从一个文件输出。另一种重定向的方法是管道操作符。

UNIX® 管道操作符 “|” 允许将一个命令的输出直接传递或引导到另一个程序。基本上，管道允许将一个命令的标准输出作为标准输入传递给另一个命令，例如：

```
% cat directory_listing.txt | sort | less
```

在这个例子中，**directory_listing.txt** 的内容将被排序，输出结果传递给 less(1)[327]。这使得用户可以按照自己的节奏滚动浏览输出，并防止其滚出屏幕。

3.10.文本编辑器

大多数 FreeBSD 的配置是通过编辑文本文件完成的，因此熟悉文本编辑器是一个好主意。FreeBSD 的基本系统中自带了一些文本编辑器，而在 ports 中还有很多可选。

一个简单易学的编辑器是 ee(1)[328]，其意为 easy editor。要启动这个编辑器，键入 `ee filename`，其中 *filename* 是要编辑的文件的名称。进入编辑器后，所有用于操作编辑器功能的命令都列在显示屏的顶部。^符号代表`Ctrl`，所以`^e`扩展为`Ctrl+e`。要离开 ee(1)[329]，按 `Esc`，然后在主菜单中选择 “leave editor” 选项。如果文件被修改过了，编辑器会提示保存所有修改。

FreeBSD 在基本系统中还提供了更强大的文本编辑器，如 vi(1)[330]。其他的编辑器，像 editors/emacs[331] 和 editors/vim[332]，是 FreeBSD ports 的一部分。这些编辑器提供了更多的功能，但代价是学习起来更加复杂。学习一个更强大的编辑器，如 vim 或 Emacs，从长远来看可以节省更多时间。

许多修改文件或需要输入的应用程序会自动打开一个文本编辑器。要改变默认的编辑器，请设置 `EDITOR` 环境变量，如 Shell[333] 中所述。

[327] https://www.freebsd.org/cgi/man.cgi?query=less&sektion=1&format=html
[328] https://www.freebsd.org/cgi/man.cgi?query=ee&sektion=1&format=html
[329] https://www.freebsd.org/cgi/man.cgi?query=ee&sektion=1&format=html
[330] https://www.freebsd.org/cgi/man.cgi?query=vi&sektion=1&format=html
[331] https://cgit.freebsd.org/ports/tree/editors/emacs/pkg-descr
[332] https://cgit.freebsd.org/ports/tree/editors/vim/pkg-descr
[333] https://docs.freebsd.org/en/books/handbook/book/#shells

3.11.设备和设备节点

设备是一个术语，主要用于系统中的硬件相关活动，包括磁盘、打印机、显卡和键盘。当 FreeBSD 启动时，大部分的启动信息都是指正在检测的设备。开机信息的副本被保存在 **/var/run/dmesg.boot** 中。

每个设备都有一个设备名称和编号。例如，**ada0** 是第一块 SATA 硬盘，而 **kbd0** 则代表键盘。

FreeBSD 中的大多数设备必须通过特殊的文件进行访问，这些文件被称为设备节点，位于 **/dev**。

3.12.手册页

FreeBSD 上最详细的文档是以手册页的形式出现的。几乎系统中的每一个程序都有一个简短的参考手册，解释其基本操作和可用参数。可以用 `man` 来查看这些手册：

```
% man command
```

其中 *command* 是要了解的命令的名称。例如，要了解更多关于 ls(1)[334] 的信息，请输入：

```
% man ls
```

手册页面被划分成多个章节，每个章节有不同的主题。在 FreeBSD 中，有以下几个部分：

1. 用户指令。
2. 系统调用和错误编号。
3. C 库中的函数。
4. 设备驱动程序。
5. 文件格式。
6. 游戏和其他娱乐程序。
7. 杂项信息。
8. 系统维护和操作命令。
9. 系统内核接口。

在某些情况下，同一个主题可能会出现于在线手册的多个部分。例如，有= `chmod` 用户命令和 `chmod()` 系统调用。要告诉 man(1)[335] 显示哪一节，需要指定节号：

```
% man 1 chmod
```

[334] https://www.freebsd.org/cgi/man.cgi?query=ls&sektion=1&format=html

[335] https://www.freebsd.org/cgi/man.cgi?query=man&sektion=1&format=html

这将显示用户命令 chmod(1)[336] 的手册页面。在书面文档中，对在线手册特定章节的引用通常放在括号里，所以 chmod(1)[337] 指的是用户命令，而 chmod(2)[338] 指的是系统调用。

如果手册页的名称不详，使用 `man -k` 来搜索手册页说明中的关键词：

```
% man -k mail
```

这条命令显示了在描述中含有关键字 `mail` 的命令列表。这等同于使用 apropos(1)[339]。

要阅读 **/usr/sbin** 中所有命令的说明，请输入：

```
% cd /usr/sbin
% man -f * | more
```

或

```
% cd /usr/sbin
% whatis * |more
```

3.12.1. GNU Info 文件

FreeBSD 包括由自由软件基金会 (FSF) 制作的一些应用程序和实用程序。除了手册页面之外，这些程序还可能包括称为 `info` 文件的超文本文件。这些文件可以用 info(1)[340] 查看，如果安装了 editors/emacs[341]，则可以用 emacs 的 `info` 模式。

要使用 info(1)[342]，请输入：

```
% info
```

要想了解简单的介绍，请键入 `h`；要想快速获得命令参考，请键入 `?`。

[336] https://www.freebsd.org/cgi/man.cgi?query=chmod&sektion=1&format=html
[337] https://www.freebsd.org/cgi/man.cgi?query=chmod&sektion=1&format=html
[338] https://www.freebsd.org/cgi/man.cgi?query=man&sektion=1&format=html
[339] https://www.freebsd.org/cgi/man.cgi?query=apropos&sektion=1&format=html
[340] https://www.freebsd.org/cgi/man.cgi?query=info&sektion=1&format=html
[341] https://cgit.freebsd.org/ports/tree/editors/emacs/pkg-descr
[342] https://www.freebsd.org/cgi/man.cgi?query=info&sektion=1&format=html

第四章安装应用程序：软件包和 Ports

4.1.概述

FreeBSD 在基本系统中自带了丰富的系统工具。此外，FreeBSD 还提供了两种互补的技术来安装第三方软件：通过源代码安装的 FreeBSD ports，以及从预先构建的二进制软件包安装。这两种方法都可以使用本地镜像或网络安装软件。

读完本章后，你将知道：

- 二进制软件包与 ports 之间的区别。
- 如何找到已经移植到 FreeBSD 上的第三方软件。
- 如何使用 pkg 来管理二进制包。
- 如何使用 ports 从源代码编译第三方软件。
- 如何找到与应用程序一起安装的文件以进行安装后的配置。
- 如果软件安装失败该怎么办。

4.2.软件安装的概述

FreeBSD *port* 是一组文件的集合，旨在自动完成从源代码编译应用程序的过程。组成 port 的文件包含了自动下载、解压缩、打补丁、编译和安装应用程序的所有必要信息。

如果该软件尚未在 FreeBSD 上进行移植和测试，则可能需要对源代码进行编辑，以使其能够正常安装和运行。

然而，已经有超过 36000[343] 个第三方应用程序被移植到了 FreeBSD。在可行的情况下，这些应用程序会以预编译 软件包的形式提供给大家下载。

软件包可以通过 FreeBSD 的软件包管理命令来操作。

[343] https://www.freebsd.org/ports/

软件包和 ports 都能解析依赖关系。如果使用软件包或 port 安装一个应用程序，而所依赖的库还没有安装，那么就会先自动安装这个库。

一个 FreeBSD 软件包包含了该应用程序的所有命令的预编译副本，以及任何配置文件和文档。可以用 pkg(8)[344] 命令来操作软件包，比如 `pkg install`。

虽然这两种技术是相似的，但软件包和 ports 都各有长处。应选择符合你安装特定应用程序的要求的技术。

软件包的好处

- 软件包文件通常比包含应用程序的源代码的压缩文件小。
- 软件包不需要花时间编译。对于大型的应用程序，如 Firefox、KDE Plasma 或 GNOME，这在运行较慢的系统中可能很重要。
- 软件包不需要了解在 FreeBSD 上编译软件的过程。

port 的好处

- 软件包通常是用保守的选项编译的，因为它们必须在尽可能多的系统上运行。通过从 port 编译，人们可以改变编译选项。
- 一些应用程序有与安装哪些功能有关的编译时选项。例如，NGINX® 可以被配置成各种不同的内置选项。

  在某些情况下，同一个应用程序会存在多个软件包来指定某些设置。例如，NGINX® 可以被拆分为一个 **nginx** 软件包和一个 **nginx-lite** 软件包，这取决于是否安装了 Xorg。如果一个应用程序有超过一个或两个不同的编译时选项，创建多个软件包很快就变得非常困难。
- 一些软件的许可证禁止二进制发布。这种软件必须以源代码的形式发布，必须由终端用户编译源代码。
- 有些人不信任二进制分发，或者喜欢通读源代码，以寻找潜在的问题。
- 为了应用定制的补丁，需要源代码。

要跟踪 ports 的更新，可以订阅 FreeBSD ports 邮件列表[345] 和 FreeBSD ports bugs 邮件列表[346]。

> 警告
>
> 在安装任何应用程序之前，请检查 https://vuxml.freebsd.org/ 以了解有关的安全问题。使用 `pkg audit -F` 来检查已安装的应用程序是否存在已知的漏洞。

本章的其余章节将解释如何使用软件包和 ports 来安装和管理 FreeBSD 上的第三方软件。

[344] https://www.freebsd.org/cgi/man.cgi?query=pkg&sektion=8&format=html

[345] https://lists.freebsd.org/subscription/freebsd-ports

[346] https://lists.freebsd.org/subscription/freebsd-ports-bugs

4.3.寻找所需的应用程序

FreeBSD 的可用应用程序列表一直在增加。有很多方法可以找到要安装的软件：

- FreeBSD 网站在 Ports Portal[347] 维护着一份最新的可搜索的所有可用应用程序的列表。可以通过应用程序名称或软件类别来搜索这些 port。
- Dan Langille 维护着 FreshPorts[348]，它提供了一个全面的搜索工具，也跟踪了 Ports 中应用程序的变化。注册用户可以创建一个自定义的监控列表，以便在他们所监控的 port 被更新时收到一封自动的电子邮件。
- 如果寻找某个特定的应用程序变得具有挑战性，可以尝试搜索像 SourceForge[349] 或 GitHub[350] 这样的网站，然后再到 Ports Portal[351] 上查看该应用程序是否被移植了。
- 使用 pkg(8)[352] 命令搜索应用程序的二进制包库。

4.4.使用 pkg 进行二进制包管理

pkg(8)[353] 为操作软件包提供了接口：注册、添加、删除和更新软件包。

对于那些只希望使用来自 FreeBSD 镜像站的预编译二进制包的网站来说，用 pkg(8)[354] 管理软件包就足够了。

然而，对于那些从源代码构建的网站，将需要一个单独的 port 管理工具[355]。

由于 pkg(8)[356] 只适用于二进制包，它并不能替代这些工具。那些工具可以用来安装二进制包和 ports 中的软件，而 pkg(8)[357] 仅可安装二进制包。

[347] https://www.freebsd.org/ports/
[348] https://www.freshports.org/
[349] https://sourceforge.net/
[350] https://github.com/
[351] https://www.freebsd.org/ports/
[352] https://www.freebsd.org/cgi/man.cgi?query=pkg&sektion=8&format=html
[353] https://www.freebsd.org/cgi/man.cgi?query=pkg&sektion=8&format=html
[354] https://www.freebsd.org/cgi/man.cgi?query=pkg&sektion=8&format=html
[355] https://docs.freebsd.org/en/books/handbook/book/#ports-upgrading-tools
[356] https://www.freebsd.org/cgi/man.cgi?query=pkg&sektion=8&format=html
[357] https://www.freebsd.org/cgi/man.cgi?query=pkg&sektion=8&format=html

4.4.1.开始使用 pkg

所有支持的 FreeBSD 版本现在都包含 `/usr/sbin/pkg`，又称 pkg(7)[358]。这是一个小的占位符，它只具有用来安装真正的 pkg(8)[359] 所需的最小功能。

> 注意
> 要使安装过程成功，需要连接到互联网。

在命令行中运行 pkg(8)[360]：

```
# pkg
```

其输出应该类似于以下内容：

```
The package management tool is not yet installed on your system.
Do you want to fetch and install it not? [y/N]
```

pkg(7)[361] 会拦截这个命令，如果你确认这是你的意图，就会下载 pkg(8)[362] 软件包，从中安装 pkg(8)[363]，引导本地软件包数据库，然后继续运行你最初要求的命令。

较新版本的 pkg(7)[364] 将 `pkg -N` 理解为测试是否安装了 pkg(8)[365] 而不触发安装，反之，`pkg bootstrap[-f]` 可以安装 pkg(8)[366] (或强制它重新安装) 而不执行任何其他操作。

可以在 pkg(8)[367] 的手册中找到 pkg 的使用信息，或者通过运行 `pkg` 而不需要其他参数。其他 pkg 配置选项在 pkg.conf(5)[368] 中有介绍。

每个 pkg 命令的参数都在特定命令的手册页中有记录。

例如，要阅读 `pkg install` 的手册页，请运行这个命令：

```
# pkg help install
```

本节的其余部分演示了可以使用 pkg(8)[369] 执行的常见二进制包管理任务。每条演示的命令都提供了许多开关来定制其用途。有关细节和更多的例子，请参考某个命令的帮助或手册页。

[358] https://www.freebsd.org/cgi/man.cgi?query=pkg&sektion=7&format=html
[359] https://www.freebsd.org/cgi/man.cgi?query=pkg&sektion=8&format=html
[360] https://www.freebsd.org/cgi/man.cgi?query=pkg&sektion=8&format=html
[361] https://www.freebsd.org/cgi/man.cgi?query=pkg&sektion=7&format=html
[362] https://www.freebsd.org/cgi/man.cgi?query=pkg&sektion=8&format=html
[363] https://www.freebsd.org/cgi/man.cgi?query=pkg&sektion=8&format=html
[364] https://www.freebsd.org/cgi/man.cgi?query=pkg&sektion=7&format=html
[365] https://www.freebsd.org/cgi/man.cgi?query=pkg&sektion=8&format=html
[366] https://www.freebsd.org/cgi/man.cgi?query=pkg&sektion=8&format=html
[367] https://www.freebsd.org/cgi/man.cgi?query=pkg&sektion=8&format=html
[368] https://www.freebsd.org/cgi/man.cgi?query=pkg.conf&sektion=5&format=html
[369] https://www.freebsd.org/cgi/man.cgi?query=pkg&sektion=8&format=html

4.4.2.Quarterly 和 Latest ports 分支

Quarterly（季度）分支为用户提供了一个更可预测、更稳定的 ports 和软件包安装和更新的体验。这基本上是通过只允许非特性更新来实现的。Quarterly 分支的目标是接收安全修复（可能是版本更新，或者是提交的回溯）、bug 修复和 ports 合规性或框架变化。季度分支在每个（年度）季度的开始，即 1 月、4 月、7 月和 10 月，从 main 中分离。根据创建的年份（YYYY）和季度（Q1-4）来命名该分支。例如，在 2023 1 月创建的季度分支，被命名为 2023Q1。而 Latest（最新）分支为用户提供最新版本的软件。

要让 pkg.conf(8)[370] 从季度分支切换到最新分支，请首先运行以下命令：

```
# mkdir -p /usr/local/etc/pkg/repos
# echo 'FreeBSD: { url: "pkg+http://pkg.FreeBSD.org/${ABI}/latest" }' > /usr/local/
↪etc/pkg/repos/FreeBSD.conf
```

然后运行这个命令，更新本地软件仓库存储目录到最新分支：

```
# pkg update -f
```

4.4.3.配置 pkg

pkg.conf(5)[371] 是 pkg(8)[372] 工具所使用的系统级配置文件。这个文件的默认位置是 /usr/local/etc/pkg.conf。

> 注意
>
> FreeBSD 不需要有 pkg.conf 文件。许多系统在没有 pkg.conf 或是空的 pkg.conf (除了注释行之外) 的情况下也能正常工作。

文件中以“#”开头的行是注释，会被忽略。

该文件为 UCL 格式。关于 libucl(3)[373] 语法的更多信息，请访问 UCL 官方网站[374]。

可以识别以下类型的选项——布尔值、字符串和列表选项。

如果在配置文件中指定了下列数值之一，则布尔选项被标记为启用——YES、TRUE 和 ON。

[370] https://man.freebsd.org/cgi/man.cgi?query=pkg&sektion=8&format=html
[371] https://www.freebsd.org/cgi/man.cgi?query=pkg.conf&sektion=5&format=html
[372] https://www.freebsd.org/cgi/man.cgi?query=pkg&sektion=8&format=html
[373] https://www.freebsd.org/cgi/man.cgi?query=libucl&sektion=3&format=html
[374] https://github.com/vstakhov/libucl

4.4.4.查找软件包

要搜索一个软件包，可以使用 pkg-search(8)[375]：

```
# pkg search nginx
```

其输出应该类似于以下内容:

```
modsecurity3-nginx-1.0.3       Instruction detection and prevention engine / nginx␣
↪Wrapper
nginx-1.22.1_2,3               Robust and small WWW server
nginx-devel-1.23.2_4           Robust and small WWW server
nginx-full-1.22.1_1,3          Robust and small WWW server (full package)
nginx-lite-1.22.1,3            Robust and small WWW server (lite package)
nginx-naxsi-1.22.1,3           Robust and small WWW server (plus NAXSI)
nginx-prometheus-exporter-0.10.0_7 Prometheus exporter for NGINX and NGINX Plus stats
nginx-ultimate-bad-bot-blocker-4.2020.03.2005_1 Nginx bad bot and other things blocker
nginx-vts-exporter-0.10.7_7    Server that scraps NGINX vts stats and export them via␣
↪HTTP
p5-Nginx-ReadBody-0.07_1       Nginx embeded perl module to read and evaluate a␣
↪request body
p5-Nginx-Simple-0.07_1         Perl 5 module for easy to use interface for Nginx Perl␣
↪Module
p5-Test-Nginx-0.30             Testing modules for Nginx C module development
py39-certbot-nginx-2.0.0       NGINX plugin for Certbot
rubygem-passenger-nginx-6.0.15 Modules for running Ruby on Rails and Rack applications
```

4.4.5.安装和更新软件包

要安装二进制软件包，可以使用 pkg-install(8)[376]。这个命令使用软件库的数据来确定要安装哪个版本的软件，以及它是否有任何未安装的依赖项。例如，要安装 `curl`:

```
# pkg install curl
```

其输出应该类似于以下内容:

```
Updating FreeBSD repository catalogue...
FreeBSD repository is up to date.
All repositories are up to date.
The following 9 package(s) will be affected (of 0 checked):
```

(continues on next page)

[375] https://www.freebsd.org/cgi/man.cgi?query=pkg-search&sektion=8&format=html

[376] https://www.freebsd.org/cgi/man.cgi?query=pkg-install&sektion=8&format=html

(continued from previous page)

```
New packages to be INSTALLED:
        ca_root_nss: 3.83
        curl: 7.86.0
        gettext-runtime: 0.21
        indexinfo: 0.3.1
        libidn2: 2.3.3
        libnghttp2: 1.48.0
        libpsl: 0.21.1_4
        libssh2: 1.10.0.3
        libunistring: 1.0

Number of packages to be installed: 9

The process will require 11 MiB more space.
3 MiB to be downloaded

Proceed with this action? [y/N]
```

在已安装的软件包列表中可以看到新的软件包和任何作为依赖项安装的额外软件包：

```
# pkg info
```

其输出应该类似于以下内容：

```
ca_root_nss-3.83               Root certificate bundle from the Mozilla Project
curl-7.86.0                    Command line tool and library for transferring data␣
↪with URLs
gettext-runtime-0.21.1         GNU gettext runtime libraries and programs
indexinfo-0.3.1                Utility to regenerate the GNU info page index
libidn2-2.3.3                  Implementation of IDNA2008 internationalized domain␣
↪names
libnghttp2-1.48.0              HTTP/2.0 C Library
libpsl-0.21.1_6                C library to handle the Public Suffix List
libssh2-1.10.0.3               Library implementing the SSH2 protocol
libunistring-1.0               Unicode string library
pkg-1.18.4                     Package manager
```

要获取软件包并在以后或在其他地方安装它，请使用 pkg-fetch(8)[377]。例如，下载 `nginx-lite`：

```
# pkg fetch -d -o /usr/home/user/packages/ nginx-lite
```

[377] https://www.freebsd.org/cgi/man.cgi?query=pkg-fetch&sektion=8&format=html

- `-d`：用来获取所有的依赖项
- `-o`：用于指定下载目录

其输出应该类似于以下内容：

```
Updating FreeBSD repository catalogue...
FreeBSD repository is up to date.
All repositories are up to date.
The following packages will be fetched:

New packages to be FETCHED:
        nginx-lite: 1.22.1,3 (342 KiB: 22.20% of the 2 MiB to download)
        pcre: 8.45_3 (1 MiB: 77.80% of the 2 MiB to download)

Number of packages to be fetched: 2

The process will require 2 MiB more space.
2 MiB to be downloaded.

Proceed with fetching packages? [y/N]:
```

要安装所下载的软件包，可以使用 pkg-install(8)[378]，如下所示：

```
# cd /usr/home/user/packages/
```

```
# pkg install nginx-lite-1.22.1,3.pkg
```

4.4.6.获得关于已安装软件包的信息

可以通过运行 pkg-info(8)[379] 来查看有关系统中所安装的软件包的信息，当没有任何参数运行时，它将列出所有安装的软件包或指定的软件包的版本。

例如，要查看安装了哪个版本的 pkg，请运行：

```
# pkg info pkg
```

其输出应该类似于以下内容：

```
pkg-1.19.0
Name           : pkg
```

(continues on next page)

[378] https://www.freebsd.org/cgi/man.cgi?query=pkg-install&sektion=8&format=html

[379] https://www.freebsd.org/cgi/man.cgi?query=pkg-info&sektion=8&format=html

(continued from previous page)

```
Version        : 1.19.0
Installed on   : Sat Dec 17 11:05:28 2022 CET
Origin         : ports-mgmt/pkg
Architecture   : FreeBSD:13:amd64
Prefix         : /usr/local
Categories     : ports-mgmt
Licenses       : BSD2CLAUSE
Maintainer     : pkg@FreeBSD.org
WWW            : https://github.com/freebsd/pkg
Comment        : Package manager
Options        :
        DOCS           : on
Shared Libs provided:
        libpkg.so.4
Annotations    :
        FreeBSD_version: 1301000
        repo_type      : binary
        repository     : FreeBSD
Flat size      : 33.2MiB
Description    :
Package management tool

WWW: https://github.com/freebsd/pkg
```

4.4.7.更新已安装的软件包

可以使用 pkg-upgrade(8)[380] 升级到已安装的软件包最新版本：

```
# pkg upgrade
```

这个命令将比较已安装的版本和版本库目录中的可用版本，并从版本库中更新它们。

4.4.8.审计已安装的软件包

在第三方应用程序中经常发现软件漏洞。为了解决这个问题，pkg 包含了内置的审计机制。要确定系统上安装的软件是否有任何已知的漏洞，请使用 pkg-audit(8)[381]。

```
# pkg audit -F
```

[380] https://www.freebsd.org/cgi/man.cgi?query=pkg-upgrade&sektion=8&format=html

[381] https://www.freebsd.org/cgi/man.cgi?query=pkg-audit&sektion=8&n=1

其输出应该类似于以下内容：

```
Fetching vuln.xml.xz: 100%  976 KiB 499.5kB/s    00:02
chromium-108.0.5359.98 is vulnerable:
  chromium -- multiple vulnerabilities
  CVE: CVE-2022-4440
  CVE: CVE-2022-4439
  CVE: CVE-2022-4438
  CVE: CVE-2022-4437
  CVE: CVE-2022-4436
  WWW: https://vuxml.FreeBSD.org/freebsd/83eb9374-7b97-11ed-be8f-3065ec8fd3ec.html
```

4.4.9.删除软件包

可以用 pkg-delete(8)[382] 删除不再需要的软件包。

比如说：

```
# pkg delete curl
```

其输出应该类似于以下内容：

```
Checking integrity... done (0 conflicting)
Deinstallation has been requested for the following 1 packages (of 0 packages in the␣
↪universe):

Installed packages to be REMOVED:
        curl :7.86.0

Number of packages to be removed: 1

The operation will free 4 MiB.

Proceed with deinstallation packages? [y/N]: y
[1/1] Deinstalling curl-7.86.0...
[1/1] Deleting files for curl-7.86.0: 100%
```

[382] https://www.freebsd.org/cgi/man.cgi?query=pkg-delete&sektion=8&format=html

4.4.10.自动删除未使用的软件包

删除一个软件包可能会留下不再需要的依赖项。可以通过 pkg-autoremove(8)[383] 自动检测并删除作为依赖关系安装的不需要的包（叶子包）：

```
# pkg autoremove
```

其输出应该类似于以下内容：

```
Checking integrity... done (0 conflicting)
Deinstallation has been requested for the following 1 packages:

Installed packages to be REMOVED:
        ca_root_nss-3.83

Number of packages to be removed: 1

The operation will free 723 KiB.

Proceed with deinstalling packages? [y/N]:
```

作为依赖关系安装的包被称为 *自动*软件包。非自动软件包，即明确安装的软件包不是作为另一个软件包的依赖关系，可以用以下方法列出：

```
# pkg prime-list
```

其输出应该类似于以下内容：

```
nginx
openvpn
sudo
```

`pkg prim-list` 是 `/usr/local/etc/pkg.conf` 中声明的一个别名命令。还有许多其他命令可以用来查询系统的软件包数据库。例如，命令 `pkg prim-origins` 可以用来获取上述列表的初始 port 目录。

```
# pkg prime-origins
```

其输出应该类似于以下内容：

```
www/nginx
security/openvpn
security/sudo
```

[383] https://www.freebsd.org/cgi/man.cgi?query=pkg-autoremove&sektion=8&format=html

这个列表可以用来重建所有使用 ports-mgmt/poudriere[384] 或 ports-mgmt/synth[385] 等编译工具安装在系统中的软件包。

将已安装的软件包标记为自动，可以使用：

```
# pkg set -A 1 devel/cmake
```

如果一个软件包成为叶子包并被标记为自动，它就会被 pkg autoremove 选中。

将一个已安装的软件包标记为非自动，可以使用：

```
# pkg set -A 0 devel/cmake
```

4.4.11.删除陈旧的软件包

默认情况下，pkg 将软件包存储在 pkg.conf(5)[386] 中由 `PKG_CACHEDIR` 定义的缓存目录中。只有最新安装的软件包的副本被保存。旧版本的 pkg 会保留所有以前的软件包。要删除这些过时的软件包，请运行:

```
# pkg clean
```

可以通过运行来清理全部缓存：

```
# pkg clean -a
```

4.4.12.锁定和解锁软件包

pkg-lock(8)[387] 用来锁定软件包，以防止重新安装、修改或删除。pkg-unlock(8)[388] 则是解锁指定的软件包。两种变体都只对当前安装的软件包有影响。因此，不可能通过这个机制来阻止一个新包的安装，除非这样的安装意味着要更新一个被锁定的包。

例如，要锁定 `nginx-lite`：

```
# pkg lock nginx-lite
```

解锁 `nginx-lite`：

```
# pkg unlock nginx-lite
```

[384] https://cgit.freebsd.org/ports/tree/ports-mgmt/poudriere/pkg-descr
[385] https://cgit.freebsd.org/ports/tree/ports-mgmt/synth/pkg-descr
[386] https://www.freebsd.org/cgi/man.cgi?query=pkg.conf&sektion=5&format=html
[387] https://www.freebsd.org/cgi/man.cgi?query=pkg-lock&sektion=8&format=html
[388] https://www.freebsd.org/cgi/man.cgi?query=pkg-unlock&sektion=8&format=html

4.4.13.修改软件包的元数据

FreeBSD Ports 中的软件可能会历经重大的版本号变化。为了解决这个问题，pkg 有一个内置的命令来更新软件包的来源。这可能很有用，例如如果 lang/python3[389] 被改名为 lang/python311[390]，那么 lang/python3[391] 现在可以代表 `3.11` 版本。

要改变上述例子中的软件包来源，请运行：

```
# pkg set -o lang/python3:lang/python311
```

再比如，要将 lang/ruby31[392] 更新为 lang/ruby32[393]，运行：

```
# pkg set -o lang/ruby31:lang/ruby32
```

> 注意
>
> 当改变软件包的来源时，重要的是要重新安装那些依赖于修改了来源的软件包的软件包。要强制重新安装依赖的软件包，请运行：
>
> ```
> # pkg install -Rf lang/ruby32
> ```

4.5.使用 Ports

ports 是由多组 **Makefile**、补丁和描述文件构成的。每一组文件都可用来在 FreeBSD 上编译和安装一个单独的应用程序，并被称为 *port*。

默认情况下，ports 本身是作为一个子目录存储在 **/usr/ports** 下。

> 警告
>
> 在安装和使用 ports 之前，请注意通常不建议将 ports 与通过 pkg 提供的二进制包一起使用来安装软件。pkg 跟踪的是 port tree 的季度分支版本，而不是 HEAD。HEAD 分支中的 port 与季度分支中的相比，依赖关系可能有所不同，这可能导致 pkg 安装的依赖关系与 ports 中的依赖关系发生冲突。如果必须同时使用 ports 和 pkg，那么请确保你的 ports 和 pkg 在的同一个 ports 分支上。

ports 包含软件类别的目录。在每一个类别中，都有针对各个应用程序的子目录。每个应用程序的子目录都包含了一组文件，这些文件告诉 FreeBSD 如何编译和安装这个程序，这被称为 *ports* 框架。每个 port 框架都包括这些文件和目录：

- **Makefile**：包含指定应用程序应如何被编译以及其组件应安装在何处的代码。
- **distinfo**: 包含为编译 port 而必须下载的文件的名称和校验和。

[389] https://cgit.freebsd.org/ports/tree/lang/python3/pkg-descr
[390] https://cgit.freebsd.org/ports/tree/lang/python311/pkg-descr
[391] https://cgit.freebsd.org/ports/tree/lang/python3/pkg-descr
[392] https://cgit.freebsd.org/ports/tree/lang/ruby31/pkg-descr
[393] https://cgit.freebsd.org/ports/tree/lang/ruby32/pkg-descr

- **files/**: 这个目录包含了程序在 FreeBSD 上编译和安装所需的任何补丁。这个目录还可能包含用于编译 port 的其他文件。
- **pkg-descr**: 提供了关于对该程序更详细的说明。
- **pkg-plist**: 包含所有将被 port 安装的文件的列表。它也会告诉 ports 系统在卸载时要删除哪些文件。

一些 ports 还包含 **pkg-message** 或其他文件来处理特殊情况。关于这些文件以及一般的 ports 的更多细节，请参考 FreeBSD Porter 的手册[394]。

port 中并不包括实际的源代码，也就是所谓的 **distfile**。编译 port 的 extract 部分会自动将下载的源代码保存到 **/usr/ports/distfiles**。

4.5.1.安装 Ports

在使用 port 编译应用程序之前，必须首先安装 ports。如果在安装 FreeBSD 时没有安装 ports，可以使用下列方法来安装它：

Git 方法

如果需要对 ports 进行更多的控制，或需要维护本地的修改，或者运行 FreeBSD-CURRENT，可以使用 Git 来获取 ports。请参阅 The Git Primer[395] 以了解关于 Git 的详细介绍。

1. 在用来检查 ports 之前，必须先安装 Git。如果已经有了 ports 的副本，请像这样安装 Git：

```
# cd /usr/ports/devel/git
# make install clean
```

如果 ports 不可用，或者正在使用 pkg 管理软件包，那么可以将 Git 作为一个软件包来安装：

```
# pkg install git
```

2. 查看一下 ports tree 的 HEAD 分支副本：

```
# git clone https://git.FreeBSD.org/ports.git /usr/ports
```

3. 或者，查看一份季度分支的副本：

```
# git clone https://git.FreeBSD.org/ports.git -b 2023Q1 /usr/ports
```

4. 根据需要，在初始化 Git 检出后更新 **/usr/ports**：

```
# git -C /usr/ports pull
```

5. 根据需要，将 **/usr/ports** 切换到不同的季度分支：

[394] https://docs.freebsd.org/en/books/porters-handbook/

[395] https://docs.freebsd.org/en/articles/committers-guide/#git-primer

```
# git -C /usr/ports switch 2023Q1
```

4.5.2.安装 Ports

本节提供了关于使用 ports 来安装或删除软件的基本说明。关于可用的 `make` 目标和环境变量的详细的说明，可以在 ports(7)[396] 中找到。

> 警告
>
> 在编译任何 port 之前，请务必按照前一节所述更新 ports。由于安装任何第三方软件都可能引入安全漏洞，因此建议首先检查 https://vuxml.freebsd.org/，以了解与该 port 有关的已知安全问题。另外，在安装新的 port 之前，可以运行 `pkg audit -F`。这个命令可以被配置为在每日安全系统检查中自动执行安全审计和漏洞数据库的更新。更多信息，请参阅 pkg-audit(8)[397] 和 periodic(8)[398]。

使用 ports 的前提是有一个正常的互联网连接。它还需要超级用户的权限。

要编译和安装这个 port，请切换到要安装的 port 的目录，然后在提示符下输入 `make install`。提示消息会显示进度：

```
# cd /usr/ports/sysutils/lsof
# make install
>> lsof_4.88D.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-4.88
...
[extraction output snipped]
...
>> Checksum OK for lsof_4.88D.freebsd.tar.gz.
===>  Patching for lsof-4.88.d,8
===>  Applying FreeBSD patches for lsof-4.88.d,8
===>  Configuring for lsof-4.88.d,8
...
[configure output snipped]
...
===>  Building for lsof-4.88.d,8
...
[compilation output snipped]
...

===>  Installing for lsof-4.88.d,8
...
```

(continues on next page)

[396] https://www.freebsd.org/cgi/man.cgi?query=ports&sektion=7&format=html

[397] https://www.freebsd.org/cgi/man.cgi?query=pkg-audit&sektion=8&format=html

[398] https://www.freebsd.org/cgi/man.cgi?query=periodic&sektion=8&format=html

(continued from previous page)

```
[installation output snipped]
...
===>   Generating temporary packing list
===>   Compressing manual pages for lsof-4.88.d,8
===>   Registering installation for lsof-4.88.d,8
===>  SECURITY NOTE:
      This port has installed the following binaries which execute with
      increased privileges.
/usr/local/sbin/lsof
#
```

由于 `lsof` 是一个需要提升权限才能运行的程序，所以在安装时将显示一个安全警告。安装完成后，将返回提示符。

一些 shell 保留了 `PATH` 环境变量中列出的目录中可用的命令的缓存，以加快对这些命令的可执行文件的查找操作。`tcsh` shell 的用户应该输入 `rehash`，这样就可以在不指定完整路径的情况下使用一个新安装的命令。在 `sh` shell 中使用 `hash -r` 来代替。更多信息请参考 shell 的文档。

在安装过程中，会创建一个工作子目录，其中包含了编译过程中使用的所有临时文件。删除这个目录可以节省磁盘空间，并在以后更新到较新版本的 port 时尽量减少出现问题的机会：

```
# make clean
===>  Cleaning for lsof-88.d,8
#
```

> 注意
>
> 为了省去这个多余的步骤，在编译 port 时应使用 `make install clean`。

4.5.2.1.定制 Port 安装

一些 port 提供了编译选项，可以用来启用或禁用应用程序组件，提供安全选项，或允许其他定制。例子包括 www/firefox[399]、security/gpgme[400]。如果这个 port 依赖于其他有可配置选项的 port，它可能会因为用户的互动而暂停数次，因为默认行为是提示用户从菜单中选择选项。为了避免这种情况，并在一个批次中完成所有的配置，可以在 port 的框架中运行 `make config-recursive`。然后，运行 `make install [clean]` 来编译和安装这个 port。

> 提示
>
> 当使用 `config-recursive` 时，要配置的 port 列表是由 `all-depends-list` 目标收集的。建议运行 `make config-recursive` 直到所有依赖的 port 选项都被设定，并且不再出现 port 选项菜单，以确定所有的依赖选项都被配置了。

[399] https://cgit.freebsd.org/ports/tree/www/firefox/pkg-descr

[400] https://cgit.freebsd.org/ports/tree/security/gpgme/pkg-descr

有几种方法可以重新进入 port 的编译选项菜单，以便在 port 编译完成后添加、删除或修改这些选项。一种方法是 `cd` 进入包含 port 的目录，然后输入 `make config`。另一个方法是使用 `make showconfig`。还有一个方法是执行 `make rmconfig`，它将删除所有选定的选项并让你重新开始设定。所有这些选项，以及其他选项，在 ports(7)[401] 中都有详细的解释。

ports 系统使用 fetch(1)[402] 来下载源文件，它支持各种环境变量。如果 FreeBSD 系统位于防火墙或 FTP/HTTP 代理之后，可能需要设置 `FTP_PASSIVE_MODE`、`FTP_PROXY` 和 `FTP_PASSWORD` 这些变量。请参阅 fetch(3)[403] 以了解支持的变量的完整列表。

对于那些不能保持连接到互联网的用户来说，可以在 **/usr/ports** 中运行 `make fetch`，以获取所有的 distfiles，或在一个类别中，例如 **/usr/ports/net**，或在特定的 port 框架中运行。请注意，如果一个 port 有任何依赖关系，在一个类别或 port 框架中运行此命令将 不会获取另一个类别中的 port 的 distfiles。相反，应该使用 `make fetch-recursive` 来获取一个 port 的所有依赖关系的 distfiles。

在少数情况下，例如当一个组织有个本地的 distfiles 仓库时，可以用变量 `MASTER_SITES` 来覆盖 **Makefile** 中指定的下载位置。在使用时，应指定备用的位置：

```
# cd /usr/ports/directory
# make >MASTER_SITE_OVERRIDE= \
ftp://ftp.organization.org/pub/FreeBSD/ports/distfiles/ fetch
```

变量 `WRKDIRPREFIX` 和 `PREFIX` 可以覆盖默认的工作目录和目标目录。比如说：

```
# make WRKDIRPREFIX=/usr/home/example/ports install
```

将在 **/usr/home/example/ports** 中编译 port，并将一切文件安装到 **/usr/local** 中：

```
# make PREFIX=/usr/home/example/local install
```

将在 **/usr/ports** 中编译这个 port，并将其安装在 **/usr/home/example/local**。然后使用：

```
# make WRKDIRPREFIX=../ports PREFIX=../local install
```

将把这两者结合起来。

也可以把这些设置为环境变量。关于如何设置环境变量的说明，请参考你的 shell 的手册页。

[401] https://www.freebsd.org/cgi/man.cgi?query=ports&sektion=7&format=html
[402] https://www.freebsd.org/cgi/man.cgi?query=fetch&sektion=1&format=html
[403] https://www.freebsd.org/cgi/man.cgi?query=ports&sektion=7&format=html

4.5.3.移除已安装的 Port

可以通过 `pkg delete` 来卸载已安装的 port。可以在 pkg-delete(8)[404] 手册中找到使用这一命令的例子。

另外，也可以在 port 的目录中运行 `make deinstall`：

```
# cd /usr/ports/sysutils/lsof
# make deinstall
===>  Deinstalling for sysutils/lsof
===>   Deinstalling
Deinstallation has been requested for the following 1 packages:

    >lsof-4.88.d,8

The deinstallation will free 229 kB
[1/1] Deleting lsof-4.88.d,8... done
```

建议在卸载 port 的过程中阅读这些信息。如果这个 port 有任何依赖它的应用程序，这些信息将被显示出来，但卸载将继续进行。在这种情况下，为了防止依赖关系被破坏，建议重新安装该应用程序。

4.5.4.更新 Port

随着时间的推移，ports 中会出现更新的软件版本。本节介绍了如何确定哪些软件可以更新，以及如何执行更新。

要确定所安装的 port 是否有更新的版本，应使用“Git 方法”[405]中说明的更新命令，确保安装了最新版本的 ports。下面的命令将列出已经安装的过时的 port：

```
# pkg version -l "<"
```

> 重要
>
> 在尝试更新之前，请从文件的顶部阅读 **/usr/ports/UPDATING**，以确定与上次更新 port 或安装系统最接近的日期。这个文件涉及了用户在更新 port 时可能遇到的各种问题和需要执行的额外步骤，包括诸如文件格式的变化、配置文件位置的变化，以及与先前版本不兼容的情况。请注意任何与需要更新的 port 相匹配的说明，并在执行更新时遵循这些说明。

[404] https://www.freebsd.org/cgi/man.cgi?query=pkg-delete&sektion=8&format=html

[405] https://docs.freebsd.org/en/books/handbook/book/#ports-using-git-method

4.5.4.1.更新和管理 Port 的工具

ports 包含了几个实用程序来执行实际的更新。每一个都有它的优势和劣势。

过去，大多数安装都使用 Portmaster 或 Portupgrade。Synth 是一个较新的选择。

哪种工具最适合某个特定的系统，都由系统管理员来选择。在使用这些工具之前，建议先备份你的数据。

4.5.4.2.使用 Portmaster 更新 Port

ports-mgmt/portmaster[406] 是一个非常迷你的工具，用于更新已安装的 port。它被设计用来使用 FreeBSD 基本系统中安装的工具，而不需要依赖其他 port 或数据库。要把这个工具作为一个 port 来安装：

```
# cd /usr/ports/ports-mgmt/portmaster
# make install clean
```

Portmaster 定义了四种类型的 port：

- Root（根）port：没有依赖关系，也不是任何其他 port 的依赖关系。
- Trunk（主干）port：没有依赖关系，但其他 port 依赖它。
- Branch（分支）port：有依赖关系，其他 port 依赖于它。
- Leaf（子）port：有依赖关系但没有其他 port 依赖它。

要列出这些类别并搜索更新：

```
# 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
...
```

(continues on next page)

[406] https://cgit.freebsd.org/ports/tree/ports-mgmt/portmaster/pkg-descr

(continued from previous page)

```
===>>> 32 leaf ports

===>>> 137 total >installed ports
       >===>>> 83 have new versions available
```

该命令用于更新所有过时的 port：

```
# portmaster -a
```

> 注意
>
> 在默认情况下，Portmaster 在删除现有的 port 之前会做备份。如果新版本安装成功，Portmaster 将删除备份。使用 `-b` 可以指示 Portmaster 不自动删除备份。添加 `-i` 可以在交互式模式下启动 Portmaster，在更新每个 port 之前提示确认。还有许多其他选项可用。请阅读 portmaster(8)[407] 的手册以了解它们的详细用法。

如果在更新过程中遇到错误，可以添加 `-f` 来更新和重建所有的 port：

```
# portmaster -af
```

Portmaster 也可以用来在系统中安装新的 port，在联编和安装新 port 之前更新所有的依赖关系。要使用这个功能，需要在 ports 中指定 port 的位置。

```
# portmaster shells/bash
```

关于 ports-mgmt/portmaster[408] 的更多信息可以在其 **pkg-descr** 中找到。

4.5.4.3.使用 Portupgrade 更新 Port

ports-mgmt/portupgrade[409] 是另一个可以用来更新 port 的工具。它安装了一套可以用来管理 port 的应用程序。而且，它依赖于 Ruby。要安装这个 port：

```
# cd /usr/ports/ports-mgmt/portupgrade
# make install clean
```

在使用这个工具进行更新之前，建议使用 `pkgdb -F` 扫描已安装的 port 列表，并修正它报告的所有不一致之处。

要更新系统中安装的所有过时的 port，请使用 `portupgrade -a`。另外，也可以使用`-i`来确认每一次的更新：

[407] https://www.freebsd.org/cgi/man.cgi?query=portmaster&sektion=8&format=html

[408] https://cgit.freebsd.org/ports/tree/ports-mgmt/portmaster/pkg-descr

[409] https://cgit.freebsd.org/ports/tree/ports-mgmt/portupgrade/pkg-descr

```
# portupgrade -ai
```

要更新一个指定的应用程序而不是所有可用的 port，请使用 `portupgrade pkgname`。包括 `-R` 是非常重要的，它可以首先更新指定应用程序所需的所有 port：

```
# portupgrade -R firefox
```

如果包含 `-P`，Portupgrade 会在 `PKG_PATH` 中列出的本地目录中搜索可用的软件包。如果本地没有可用的软件包，它就会从远程站点获取软件包。如果在本地找不到软件包或无法从远程获取，Portupgrade 将使用 ports。要完全避免使用 ports，请指定 `-PP`。这最后一组选项告诉 Portupgrade，如果没有可用的包，就放弃：

```
# portupgrade -PP gnome3
```

如果指定了 `-P`，则只需获取 port distfiles 或 packages，而不需要编译或安装任何东西，请使用 `-F`。关于所有可用开关的进一步信息，请参考 `portupgrade` 的手册。

关于 ports-mgmt/portupgrade[410] 的更多信息可以在其 **pkg-descr** 中找到。

4.5.5.Ports 和磁盘空间

使用 ports 会随着时间的推移而耗尽磁盘空间。在编译和安装了一个 port 之后，在 ports 框架中运行 `make clean` 将清理临时 **工作**目录。如果使用 Portmaster 来安装一个 port，它将自动删除这个目录，除非指定了 `-K`。如果安装了 Portupgrade，这个命令将删除在 ports 本地副本中发现的所有 **工作**目录：

```
# portsclean -C
```

此外，过时的源码包文件会随着时间的推移在 **/usr/ports/distfiles** 中积累起来。要使用 Portupgrade 来删除所有不再被任何 port 引用的 distfiles：

```
# portsclean -D
```

Portupgrade 可以删除所有不被当前安装在系统上的任何 port 引用的 distfiles：

```
# portsclean -DD
```

如果安装了 Portmaster，请使用：

```
# portmaster --clean-distfiles
```

默认情况下，该命令是交互式的，提示用户确认是否应该删除 distfile。

[410] https://cgit.freebsd.org/ports/tree/ports-mgmt/portupgrade/pkg-descr

除了这些命令之外，ports-mgmt/pkg_cutleaves[411] 还会自动删除已安装的、不再需要的 port。

4.6.使用 Poudriere 构建软件包

Poudriere 是一个采用 BSD 许可证的工具，用于创建和测试 FreeBSD 软件包。它使用 FreeBSD jail 来建立隔离的编译环境。这些 jail 可以用来为与已安装系统不同的 FreeBSD 版本构建软件包，如果主机是 amd64 系统，还可以为 i386 构建软件包。构建了这些包后，它们的目录就与官方镜像站相同了。这些软件包可以被 pkg(8)[412] 和其他软件包管理工具使用。

Poudriere 可通过软件包或 port ports-mgmt/poudriere[413] 安装。该安装包括一个配置文件样本 **/usr/local/etc/poudriere.conf.sample**。将此文件复制到 **/usr/local/etc/poudriere.conf**。编辑复制的文件以满足本地配置需求。

尽管在系统上运行 poudriere 不依赖于 ZFS，但使用 ZFS 是有好处的。当使用 ZFS 时，必须在 **/usr/local/etc/poudriere.conf** 中指定 `ZPOOL`，并将 `FREEBSD_HOST` 设置为地理位置上相近的镜像站。定义 `CCACHE_DIR` 可以启用 devel/ccache[414]，以缓存缓存编译后的资源、减少频繁构建代码的编译时间。有一种方便快捷的做法，那就是将一个孤立树挂载在 **/poudriere**，并将 poudriere 数据集放在该孤立树中。其它参数使用默认值即可。

检测到的处理器内核数量用于定义将并行运行多少个编译项目。可以用内存或交换空间以提供足够的虚拟内存。如果虚拟内存用完了，jail 的编译就会停止并被清除，引发奇怪的错误信息。

4.6.1.初始化 Jail 和 Ports

配置完成后，初始化 poudriere，使其安装一个带有所需 FreeBSD 基本系统和 ports 的 jail。用 `-j` 指定 Jail 的名称，用 `-v` 指定 FreeBSD 版本。在运行 FreeBSD/amd64 的系统上，可以用 `-a` 设置架构为 i386 或 amd64。默认是 `uname` 显示的架构。

```
# poudriere jail -c -j 13amd64 -v 13.1-RELEASE
[00:00:00] Creating 13amd64 fs at /poudriere/jails/13amd64... done
[00:00:00] Using pre-distributed MANIFEST for FreeBSD 13.1-RELEASE amd64
[00:00:00] Fetching base for FreeBSD 13.1-RELEASE amd64
/poudriere/jails/13amd64/fromftp/base.txz              125 MB 4110 kBps    31s
[00:00:33] Extracting base... done
[00:00:54] Fetching src for FreeBSD 13.1-RELEASE amd64
/poudriere/jails/13amd64/fromftp/src.txz               154 MB 4178 kBps    38s
[00:01:33] Extracting src... done
[00:02:31] Fetching lib32 for FreeBSD 13.1-RELEASE amd64
/poudriere/jails/13amd64/fromftp/lib32.txz              24 MB 3969 kBps    06s
```

(continues on next page)

[411] https://cgit.freebsd.org/ports/tree/ports-mgmt/pkg_cutleaves/pkg-descr

[412] https://www.freebsd.org/cgi/man.cgi?query=pkg&sektion=8&format=html

[413] https://cgit.freebsd.org/ports/tree/ports-mgmt/poudriere/pkg-descr

[414] https://cgit.freebsd.org/ports/tree/devel/ccache/pkg-descr

(continued from previous page)

```
[00:02:38] Extracting lib32... done
[00:02:42] Cleaning up... done
[00:02:42] Recording filesystem state for clean... done
[00:02:42] Upgrading using ftp
/etc/resolv.conf -> /poudriere/jails/13amd64/etc/resolv.conf
Looking up update.FreeBSD.org mirrors... 3 mirrors found.
Fetching public key from update4.freebsd.org... done.
Fetching metadata signature for 13.1-RELEASE from update4.freebsd.org... done.
Fetching metadata index... done.
Fetching 2 metadata files... done.
Inspecting system... done.
Preparing to download files... done.
Fetching 124 patches.....10....20....30....40....50....60....70....80....90....100....
↪110....120.. done.
Applying patches... done.
Fetching 6 files... done.
The following files will be added as part of updating to
13.1-RELEASE-p1:
/usr/src/contrib/unbound/.github
/usr/src/contrib/unbound/.github/FUNDING.yml
/usr/src/contrib/unbound/contrib/drop2rpz
/usr/src/contrib/unbound/contrib/unbound_portable.service.in
/usr/src/contrib/unbound/services/rpz.c
/usr/src/contrib/unbound/services/rpz.h
/usr/src/lib/libc/tests/gen/spawnp_enoexec.sh
The following files will be updated as part of updating to
13.1-RELEASE-p1:
[…]
Installing updates...Scanning //usr/share/certs/blacklisted for certificates...
Scanning //usr/share/certs/trusted for certificates...
 done.
13.1-RELEASE-p1
[00:04:06] Recording filesystem state for clean... done
[00:04:07] Jail 13amd64 13.1-RELEASE-p1 amd64 is ready to be used
```

```
# poudriere ports -c -p local -m git+https
[00:00:00] Creating local fs at /poudriere/ports/local... done
[00:00:00] Checking out the ports tree... done
```

poudriere 可以在一台电脑上的多个 jail 中，从不同的 ports 上建立多个配置的 ports。这些组合的自定义

配置被称为 集。在安装了 ports-mgmt/poudriere[415] 或 ports-mgmt/poudriere-devel[416] 之后，请参阅 poudriere(8)[417] 的 CUSTOMIZATION 部分来了解详情。

这里的示例文件的基本配置是在 **/usr/local/etc/poudriere.d** 中放置一个特定于 jail、ports 和设置的 **make.conf**。在构建时系统的 **make.conf** 和这个新文件被合并起来，以创建构建 jail 所使用的 **make.conf**。

要构建的软件包被输入 **13amd64-local-workstation-pkglist** 中：

```
editors/emacs
devel/git
ports-mgmt/pkg
...
```

配置了指定 ports 的选项和依赖关系：

```
# poudriere options -j 13amd64 -p local -z workstation -f 13amd64-local-workstation-
↪pkglist
```

最后，软件包被构建，并创建了一个软件包仓库：

```
# poudriere bulk -j 13amd64 -p local -z workstation -f 13amd64-local-workstation-
↪pkglist
```

在运行过程中，按 `Ctrl+t` 会显示当前的构建状态。Poudriere 还在 **/poudriere/logs/bulk/jailname** 中建立了一些文件，这些文件可以用网络服务器来显示构建信息。

完成后，现在可以从 poudriere 仓库中安装新的软件包。

关于使用 poudriere 的更多信息，见 poudriere(8)[418] 和主站 https://github.com/freebsd/poudriere/wiki。

4.6.2.配置 pkg 客户端使用 Poudriere 仓库

虽然可以同时使用自定义版本库和官方版本库，但有时需要禁用官方版本库。这可以通过创建一个配置文件来实现，使用该文件覆盖并禁用官方配置文件。创建 **/usr/local/etc/pkg/repos/FreeBSD.conf**，其中包含以下内容：

```
FreeBSD: {
    enabled: no
}
```

[415] https://cgit.freebsd.org/ports/tree/ports-mgmt/poudriere/pkg-descr
[416] https://cgit.freebsd.org/ports/tree/ports-mgmt/poudriere-devel/pkg-descr
[417] https://www.freebsd.org/cgi/man.cgi?query=poudriere&sektion=8&format=html
[418] https://www.freebsd.org/cgi/man.cgi?query=poudriere&sektion=8&format=html

通常情况下，通过 HTTP 向客户机提供软件包库是最简单的。设置一个网络服务器来提供软件包目录，例如：**/usr/local/poudriere/data/packages/13amd64**，其中 **13amd64** 是编译的名称。

如果软件包仓库的 URL 是：http://pkg.example.com/13amd64，那么 **/usr/local/etc/pkg/repos/custom.conf** 中的仓库配置文件会是这样的：

```
custom: {
    url: "http://pkg.example.com/13amd64",
    enabled: yes,
}
```

如果不希望将包存储库暴露在互联网上，则可以使用 `file://` 协议直接指向存储库：

```
custom: {
    url: "file:///usr/local/poudriere/data/packages/11amd64",
    enabled: yes,
}
```

4.7.安装后的注意事项

不管软件是通过二进制包还是 ports 安装的，大多数第三方应用程序在安装后都需要一定程度的配置。下面的命令和路径可以用来帮助确定应用程序的安装内容。

- 大多数应用程序在 **/usr/local/etc** 中至少预置了一个默认配置文件。如果这个应用程序有大量的配置文件，将创建一个子目录来存放这些文件。通常，安装的样本配置文件以 **.sample** 等后缀结尾。应该查阅并编辑这些配置文件以满足系统的需要。要编辑一个样本文件，首先需要复制该文件并去除其 **.sample** 扩展名。
- 应用程序提供的文档将会被安装到 **/usr/local/share/doc**，许多应用程序还会安装手册页。在使用之前，应该查阅这些文档。
- 一些应用程序运行的服务必须在启动应用程序之前被添加到 **/etc/rc.conf** 中。这些应用程序通常在 **/usr/local/etc/rc.d** 中安装了一个启动脚本，更多信息见启动服务[419]。

  注意

  根据设计，在安装时不运行应用程序的启动脚本，在卸载或升级时也不运行其停止脚本。这个决定是留给各个系统管理员的。

- 使用 csh(1)[420] 的用户应该运行 `rehash` 来重建 shell `PATH` 中的已知二进制列表。
- 可使用 `pkg info` 来确定哪些文件、手册和二进制文件是与应用程序一起安装的。

[419] https://docs.freebsd.org/en/books/handbook/config/index.html#configtuning-starting-services

[420] https://www.freebsd.org/cgi/man.cgi?query=csh&sektion=1&format=html

4.8.如何处理损坏的 port

当一个 port 不能编译或安装时，请尝试以下方法：

1. 搜索一下问题报告数据库[421]中是否有针对该 port 的待修复问题。如果有的话，采纳推荐的修复方案可能会解决这个问题。

2. 向 port 的维护者寻求帮助。在 ports 框架中输入 `make maintainer` 或阅读 port 的 **Makefile** 来找到维护者的电子邮件地址。记得在发给维护者的邮件中包含导致错误的输出。

   注意

   有些 port 不是由个人维护的，而是由邮件列表[422]所代表的团体维护者维护的。有许多，但非全部的这些地址看起来像 freebsd-listname@FreeBSD.org 的都是。在发送电子邮件时请考虑到这一点。

   特别是，由 ports@FreeBSD.org[423] 所维护的 port 都不是由某个特定的人维护的。但是任何修复和支持都来自于订阅了该邮件列表的广大社区。我们总是需要更多的志愿者！

   如果电子邮件没有得到回应，请使用 Bugzilla 按照撰写 FreeBSD 问题报告[424]中的说明提交一份错误报告。

3. 修复它！Port 用户手册[425]中有包括关于 ports 基础设施的详细信息，这样你就可以修复偶然坏掉的 port，甚至提交你自己的 port！

4. 使用 pkg 二进制管理程序[426]中的说明来安装软件包，而非 ports。

[421] https://www.freebsd.org/support/

[422] https://docs.freebsd.org/en/articles/mailing-list-faq/

[423] https://docs.freebsd.org/en/articles/problem-reports/

[424] https://docs.freebsd.org/en/articles/problem-reports/

[425] https://docs.freebsd.org/en/books/porters-handbook/

[426] https://docs.freebsd.org/en/books/handbook/book/#pkgng-intro

第五章 X Window 系统

5.1. 概述

通过 bsdinstall 安装的 FreeBSD 并不会自动安装图形化用户界面。本章说明了如何安装和配置 Xorg，它提供了用于提供图形环境的开源 X Window 系统。然后介绍了如何寻找并安装桌面环境和窗口管理器。

在阅读本章之前，你应该：

- 知道如何安装第三方软件，如安装应用程序: 软件包和 Ports[427] 中所述。

读完本章后，你会知道：

- X Window 系统的各种组件，以及它们如何相互交互。
- 如何安装和配置 Xorg。
- 如何在 Xorg 中使用 TrueType® 字体。
- 如何为你的系统设置图形界面登录（XDM）。

5.2. 安装 Xorg

在 FreeBSD 上，可以用软件包或 port 等方式来安装 Xorg。

使用二进制元包可以实现快速安装，但可自定义选项较少：

```
# pkg install xorg
```

通过以上任一安装方式，均可安装完整的 Xorg 系统。

当前用户必须是 `video` 组的成员。要将用户添加到 `video` 组，请执行以下命令：

[427] https://docs.freebsd.org/en/books/handbook/book/#ports

```
# pw groupmod video -m username
```

提示

x11/xorg-minimal[428] 提供了适合于较有经验的用户使用的精简的 X 系统。该版本不会安装大量的文档、库和软件，但某些软件需要这些附加组件才能运行。

提示

显卡、显示器和输入设备是自动检测的，无需任何手动配置。除非自动配置失败，否则不要创建 xorg.conf 或执行 -configure 操作。

5.3. 显卡驱动

下表展示了 FreeBSD 所支持的不同的显卡，应该安装哪个软件包以及其对应的模块。

表 7.显卡软件包

品牌	类型	软件包	模块
Intel®	开源	drm-kmod	i915kms
AMD®	开源	drm-kmod	amdgpu 和 radeonkms
NVIDIA®	专有	nvidia-driver	nvidia 或 nvidia-modeset
VESA	开源	xf86-video-vesa	vesa
SCFB	开源	xf86-video-scfb	scfb
Virtualbox	开源	virtualbox-ose-additions	Virtualbox OSE 增强功能包中的 vboxvideo 驱动
VMware®	开源	xf86-video-vmware	vmwgfx

下面的命令可以用来识别系统中安装的是哪种显卡：

```
% pciconf -lv|grep -B4 VGA
```

输出应类似于以下内容：

```
vgapci0@pci0:0:2:0:     class=0x030000 rev=0x07 hdr=0x00 vendor=0x8086 device=0x2a42␣
↪subvendor=0x17aa subdevice=0x20e4
    vendor     = 'Intel Corporation'
    device     = 'Mobile 4 Series Chipset Integrated Graphics Controller'
    class      = display
    subclass   = VGA
```

警告

[428] https://cgit.freebsd.org/ports/tree/x11/xorg-minimal/

如果显卡不被 Intel®、AMD® 或 NVIDIA® 驱动程序所支持，则应使用 VESA 或 SCFB 模块。以 BIOS 模式引导时必须使用 VESA 模块，以 UEFI 模式引导时必须使用 SCFB 模块。

这条命令用来检查启动模式：

```
% sysctl machdep.bootmethod
```

输出应类似于以下内容：

```
machdep.bootmethod: BIOS
```

5.3.1. Intel®

Intel® Graphics 是指与 Intel® CPU 集成在同一芯片上的图形芯片。维基百科提供了英特尔高清显卡各代产品的变体和名称概览[429]。

graphics/drm-kmod[430] 软件包间接提供了一系列内核模块，可用于 Intel® 显卡。执行以下命令来安装 Intel® 驱动：

```
# pkg install drm-kmod
```

然后将模块添加到 `/etc/rc.conf` 文件中，执行以下命令：

```
# sysrc kld_list+=i915kms
```

提示

如果发现 CPU 使用率过高或高清视频过度撕裂，安装 multimedia/libva-intel-driver[431] 可能有所帮助。执行以下命令来安装该软件包：

```
# pkg install libva-intel-driver mesa-libs mesa-dri
```

5.3.2. AMD®

graphics/drm-kmod[432] 软件包间接提供了一系列内核模块，可用于 AMD® 显卡。`amdgpu` 和 `radeonkms` 模块可以根据硬件的级别使用。FreeBSD 项目维护了一个用来匹配对应驱动的 AMD 图形支持列表[433]。

执行以下命令来安装 AMD® 驱动：

[429] https://en.wikipedia.org/wiki/List_of_Intel_graphics_processing_units
[430] https://cgit.freebsd.org/ports/tree/graphics/drm-kmod/
[431] https://cgit.freebsd.org/ports/tree/multimedia/libva-intel-driver/
[432] https://cgit.freebsd.org/ports/tree/graphics/drm-kmod/
[433] https://wiki.freebsd.org/Graphics/AMD-GPU-Matrix

```
# pkg install drm-kmod
```

对于 HD7000 或 Tahiti 之后的显卡，将该模块添加到 /etc/rc.conf 文件中，执行以下命令：

```
# sysrc kld_list+=amdgpu
```

对于旧显卡（HD7000 之前或 Tahiti 之前），将该模块添加到 /etc/rc.conf 文件中，执行以下命令：

```
# sysrc kld_list+=radeonkms
```

5.3.3. NVIDIA®

FreeBSD 支持多个版本的 NVIDIA® 专有驱动程序。使用较新显卡的用户应该安装软件包 x11/nvidia-driver[434] 。那些使用旧显卡的用户必须在下面查看哪个版本支持它们。

表 8.支持的 NVIDIA® 驱动程序版本

软件包	支持的硬件
x11/nvidia-driver-304	支持的硬件[435]
x11/nvidia-driver-340	支持的硬件[436]
x11/nvidia-driver-390	支持的硬件[437]
x11/nvidia-driver-470	支持的硬件[438]
x11/nvidia-driver	支持的硬件[439]

警告

NVIDIA® 显卡驱动 304 版本（nvidia-driver-304）不支持 xorg-server 1.20 及以上版本。

执行以下命令来安装最新的 NVIDIA® 驱动：

```
# pkg install nvidia-driver
```

然后将模块添加到 /etc/rc.conf 文件中，执行以下命令：

```
# sysrc kld_list+=nvidia-modeset
```

[434] https://cgit.freebsd.org/ports/tree/x11/nvidia-driver/

[435] https://www.nvidia.com/Download/driverResults.aspx/123712/en-us/

[436] https://www.nvidia.com/Download/driverResults.aspx/156167/en-us/

[437] https://www.nvidia.com/Download/driverResults.aspx/191122/en-us/

[438] https://www.nvidia.com/Download/driverResults.aspx/191234/en-us/

[439] https://www.nvidia.com/Download/driverResults.aspx/205466/en-us/

警告

如果安装了 x11/nvidia-driver-304 或 x11/nvidia-drivers-340 软件包，则必须使用 nvidia 驱动程序。

```
# sysrc kld_list+=nvidia
```

5.4. Xorg 配置

5.4.1. 快速开始

Xorg 支持大多数常见的显卡、键盘和定点设备。

警告

会自动检测显卡、显示器和输入设备，无需任何手动配置。除非自动配置失败，否则不要创建 xorg.conf 或执行 Xorg -configure 操作。

5.4.1. 配置文件

Xorg 在多个目录中查找配置文件。FreeBSD 推荐将这些文件存放于 **/usr/local/etc/X11/**。使用此目录有助于将应用程序文件与操作系统文件相分离。

5.4.2. 单个或多个文件

使用多个文件，每个文件配置一个特定的设置，比使用传统的单个 **xorg.conf** 更方便。这些文件存储在主配置文件目录的 **/usr/local/etc/X11/xorg.conf.d/** 子目录中。

提示

传统的单个 **xorg.conf** 仍然有效，但不像 **/usr/local/etc/X11/xorg.conf.d/** 子目录中的多个文件那样清晰和灵活。

5.4.3. 显卡

可以在目录 **/usr/local/etc/X11/xorg.conf.d/** 中指定显卡的驱动程序。

在配置文件中配置 Intel® 驱动：

例 **14.** 在文件中选择 **Intel®** 显卡驱动程序
/usr/local/etc/X11/xorg.conf.d/20-intel.conf

```
Section "Device"
        Identifier "Card0"
        Driver     "intel"
EndSection
```

在配置文件中配置 AMD® 驱动：

例 **15.** 在文件中选择 **AMD®** 显卡驱动程序
/usr/local/etc/X11/xorg.conf.d/20-radeon.conf

```
Section "Device"
        Identifier "Card0"
        Driver     "radeon"
EndSection
```

在配置文件中配置 NVIDIA® 驱动：

例 **16.** 在文件中选择 **NVIDIA®** 显卡驱动程序
/usr/local/etc/X11/xorg.conf.d/20-nvidia.conf

```
Section "Device"
        Identifier "Card0"
        Driver     "nvidia"
EndSection
```

在配置文件中设置 VESA 驱动：

例 **17.** 在文件中选择 **VESA** 显卡驱动程序
/usr/local/etc/X11/xorg.conf.d/20-vesa.conf

```
Section "Device"
        Identifier "Card0"
      Driver     "vesa"
EndSection
```

在配置文件中设置 SCFB 驱动：

例 **18.** 在文件中选择 **SCFB** 显卡驱动程序
/usr/local/etc/X11/xorg.conf.d/20-scfb.conf

```
Section "Device"
        Identifier "Card0"
      Driver     "scfb"
EndSection
```

如果需要配置多个显卡，可以添加 BusID。执行以下命令可以显示显卡总线 ID 列表:

```
% pciconf -lv | grep -B3 display
```

输出应类似于以下内容:

```
vgapci0@pci0:0:2:0:     class=0x030000 rev=0x07 hdr=0x00 vendor=0x8086 device=0x2a42␣
↪subvendor=0x17aa subdevice=0x20e4
    vendor     = 'Intel Corporation'
    device     = 'Mobile 4 Series Chipset Integrated Graphics Controller'
    class      = display
--
vgapci1@pci0:0:2:1:     class=0x038000 rev=0x07 hdr=0x00 vendor=0x8086 device=0x2a43␣
↪subvendor=0x17aa subdevice=0x20e4
    vendor     = 'Intel Corporation'
    device     = 'Mobile 4 Series Chipset Integrated Graphics Controller'
    class      = display
```

例 19. 在文件中选择 **Intel®** 显卡驱动程序和 **NVIDIA®** 显卡驱动程序 **/usr/local/etc/X11/xorg.conf.d/20-drivers.conf**

```
Section "Device"
        Identifier "Card0"
      Driver     "intel"
      BusID      "pci0:0:2:0"
EndSection

Section "Device"
        Identifier "Card0"
      Driver     "nvidia"
      BusID      "pci0:0:2:1"
EndSection
```

5.4.4. 显示器

几乎所有的显示器都支持扩展显示识别数据标准（EDID）。Xorg 使用 EDID 与显示器通信，检测支持的分辨率和刷新率。然后它选择最合适的设置组合来使用该显示器。

显示器支持的其他分辨率可以通过在配置文件中设置所需的分辨率来选择，或者在启动 X server 服务器之后用 xrandr(1)[440] 配置。

[440] https://man.freebsd.org/cgi/man.cgi?query=xrandr&sektion=1&format=html

5.4.4.1. 使用 RandR (调整大小和方向)

在不加任何参数的情况下运行 xrandr(1)[441]，可以看到视频输出和检测到的显示器模式的列表：

```
% xrandr
```

输出应类似于以下内容：

```
Screen 0: minimum 320 x 200, current 2560 x 960, maximum 8192 x 8192
LVDS-1 connected 1280x800+0+0 (normal left inverted right x axis y axis) 261mm x 163mm
   1280x800      59.99*+  59.81    59.91    50.00
   1280x720      59.86    59.74
   1024x768      60.00
   1024x576      59.90    59.82
   960x540       59.63    59.82
   800x600       60.32    56.25
   864x486       59.92    59.57
   640x480       59.94
   720x405       59.51    58.99
   640x360       59.84    59.32
VGA-1 connected primary 1280x960+1280+0 (normal left inverted right x axis y axis)␣
↪410mm x 257mm
   1280x1024     75.02    60.02
   1440x900      74.98    60.07
   1280x960      60.00*
   1280x800      74.93    59.81
   1152x864      75.00
   1024x768      75.03    70.07    60.00
   832x624       74.55
   800x600       72.19    75.00    60.32    56.25
   640x480       75.00    72.81    66.67    59.94
   720x400       70.08
HDMI-1 disconnected (normal left inverted right x axis y axis)
DP-1 disconnected (normal left inverted right x axis y axis)
HDMI-2 disconnected (normal left inverted right x axis y axis)
DP-2 disconnected (normal left inverted right x axis y axis)
DP-3 disconnected (normal left inverted right x axis y axis)
```

这表明，VGA-1 在使用屏幕分辨率为 1280x960 像素，刷新率约为 60 赫兹来进行显示输出。LVDS-1 在使用屏幕分辨率为 1280x960 像素，刷新率约为 60 赫兹作为副屏进行显示输出。显示器没有连接到 HDMI-1，HDMI-2，DP-1，DP-2 和 DP-3 接口。

其他任何显示模式都可以用 xrandr(1)[442] 选择。例如，要切换到 60 赫兹的 1280x1024：

[441] https://man.freebsd.org/cgi/man.cgi?query=xrandr&sektion=1&format=html

[442] https://man.freebsd.org/cgi/man.cgi?query=xrandr&sektion=1&format=html

```
% xrandr --output LVDS-1 --mode 1280x1024 --rate 60
```

5.4.4.2. 使用 Xorg 配置文件

显示器配置也可以在配置文件中进行设置。

要在配置文件中设置 1024x768 的屏幕分辨率，请执行以下操作：

例 **20.** 在文件中设置屏幕分辨率
/usr/local/etc/X11/xorg.conf.d/10-monitor.conf

```
Section "Device"
      Identifier "Screen0"
      Device     "Card0"
      SubSection "Display"
      Modes      "1024x768"
      EndSubSection
EndSection
```

5.4.5. 输入设备

Xorg 通过 x11/libinput[443] 支持绝大多数输入设备。

提示
某些桌面环境(比如 KDE Plasma)提供了一个图形界面来设置这些参数。在进行手动配置编辑之前，请检查是否存在这种情况。

例如，配置键盘布局：

例 **21.** 设置键盘布局
/usr/local/etc/X11/xorg.conf.d/00-keyboard.conf

```
Section "InputClass"
        Identifier "Keyboard1"
        MatchIsKeyboard "on"
        Option "XkbLayout" "es, fr"
        Option "XkbModel" "pc104"
        Option "XkbVariant" ",qwerty"
        Option "XkbOptions" "grp:win_space_toggle"
EndSection
```

[443] https://cgit.freebsd.org/ports/tree/x11/libinput/

5.5. 在 Xorg 中使用字体

Xorg 附带的默认字体不太适合典型的桌面出版应用程序。大字体显得参差不齐，看起来不专业，小字体几乎完全看不懂。但是，有一些免费的、高质量的 Type1 (PostScript®) 字体可以很容易地与 Xorg 一起使用。

5.5.1. Type1 字体

URW 字体（x11-fonts/urwfonts[444]）包括标准 Type1 字体（Times Roman™、Helvetica™、Palatino™ 和其他）的高质量版本。Freefonts 字体（x11-fonts/freefonts[445]）包括了更多的字体，但其中大多数是为了在图形软件（如 Gimp）中使用，不够完备，无法作为屏幕字体使用。此外，只需花费很少的精力，就可以配置 Xorg 使用 TrueType® 字体。有关这方面的更多细节，请参见 X(7)[446] 手册页或 TrueType® 字体[447]。

要从二进制软件包中安装上述 Type1 字体，请运行下列命令：

```
# pkg install urwfonts
```

同样地，freefont 或其它字体也是如此。要想让 X 服务器检测到这些字体，请在 X 服务器配置文件（**/usr/local/etc/X11/xorg.conf.d/90-fonts.conf**）中适当的添加一行，内容如下：

```
Section "Files"
  FontPath "/usr/local/share/fonts/urwfonts/"
EndSection
```

或者，在 X 会话的命令行中运行：

```
% xset fp+ /usr/local/share/fonts/urwfonts
% xset fp rehash
```

这样即可生效，但在关闭 X 会话时将会失效，除非将它添加到启动文件中（常规的 `startx` 会话在 **~/.xinitrc** 中，用于 XDM 这样的图形登录管理器登录在 **~/.xsession**）。第三种方法是使用新的 **/usr/local/etc/fonts/local.conf**，正如在抗锯齿字体[448]中演示的那样。

[444] https://cgit.freebsd.org/ports/tree/x11-fonts/urwfonts/

[445] https://cgit.freebsd.org/ports/tree/x11-fonts/freefonts/

[446] https://www.freebsd.org/cgi/man.cgi?query=X&sektion=7&format=html

[447] https://docs.freebsd.org/en/books/handbook/x11/#truetype

[448] https://docs.freebsd.org/en/books/handbook/book/#antialias

5.5.2. TrueType® 字体

Xorg 已经内置了对 TrueType® 字体的渲染支持。有两个不同的模块可以实现这个功能。本例中使用了 freetype 模块，因为它与其他字体渲染后端更加一致。要启用 freetype 模块，只需在 **/usr/local/etc/X11/xorg.conf.d/90-fonts.conf** 的 "Module"部分添加以下一行。

```
Load  "freetype"
```

现在为 TrueType® 字体建立一个目录（如 **/usr/local/share/fonts/TrueType**），将所有的 TrueType® 字体复制到这个目录中。请记住，不能直接从 Apple® Mac® 中提取 TrueType® 字体；它们必须是 UNIX®/MS-DOS®/Windows® 中的格式，才能被 Xorg 使用。文件被复制到这个目录中后，使用 mkfontscale 创建一个 **fonts.dir**，这样 X 字体渲染器就知道这些新文件已经被安装好了。可以作为一个软件包安装 mkfontscale：

```
# pkg install mkfontscale
```

然后在目录中创建 X 字体文件的索引：

```
# cd /usr/local/share/fonts/TrueType
# mkfontscale
```

现在将 TrueType® 目录添加到字体路径中。这和在 Type1 字体[449]中所述的一样：

```
% xset fp+ /usr/local/share/fonts/TrueType
% xset fp rehash
```

或在 **xorg.conf** 中添加 FontPath 行。

现在，Gimp、LibreOffice 和所有其他的 X 应用程序应该可以识别已安装的 TrueType® 字体。极小的字体（如网页上高分辨率显示的文本）和极大的字体（在 LibreOffice 中）现在看起来会好得多。

5.5.3. 抗锯齿字体

Xorg 中所有在 **/usr/local/share/fonts/** 和 **~/.fonts/** 中找到的字体都会自动提供给采用 Xft-aware 技术的应用程序用于抗锯齿。大多数最新的应用程序都支持 Xft-aware，包括 KDE、GNOME 和 Firefox。

要控制哪些字体开启抗锯齿，或者配置抗锯齿属性，可以创建（或编辑，如果文件已经存在）文件 **/usr/local/etc/fonts/local.conf**。可以通过这个文件进行微调 Xft 字体系统的一些高级功能；本节只说明一些简单的功能。详细内容请参见 fonts-conf(5)[450]。

这个文件必须是 XML 格式的。请仔细注意大小写，并确保所有的标签都被正确配对。该文件以通常的 XML 头开始，后面是 DOCTYPE 定义，然后是<fontconfig> 标签：

[449] https://docs.freebsd.org/en/books/handbook/book/#type1

[450] https://man.freebsd.org/cgi/man.cgi?query=fonts-conf&sektion=5&format=html

```
<?xml version="1.0"?>
      <!DOCTYPE fontconfig SYSTEM "fonts.dtd">
      <fontconfig>\
```

如前所述，**/usr/local/share/fonts/** 以及 **~/.fonts/** 中的所有字体已经被提供给支持 Xft-aware 的应用程序。要在这两个目录树之外添加另一个目录，请在 **/usr/local/etc/fonts/local.conf** 中添加这样一行:

```
<dir>/path/to/my/fonts</dir>
```

在添加新的字体，特别是新的字体目录之后，要更新字体缓存:

```
# fc-cache -f
```

抗锯齿使边界略显模糊，这使非常小的文本更易阅读，并消除了大文本的“梯形”，但如果应用于正常文本，可能会导致眼睛疲劳。要将小于 14 号的字体排除在抗锯齿之外，请加入这几行:

```
<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>
```

一些等宽字体的间距也可能因抗锯齿而不合适。需要指出的是，在 KDE 环境中，这似乎是一个未解决的问题。一个可行的解决方法是强制这类字体的间距为 100。需要添加这几行:

```
<match target="pattern" name="family">
   <test qual="any" name="family">
       <string>fixed</string>
   </test>
   <edit name="family" mode="assign">
       <string>mono</string>
   </edit>
```

(continues on next page)

(continued from previous page)

```
</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>
```

（这为其它一些常规固定字体创建了 "mono" 的别名），然后再添加:

```
<match target="pattern" name="family">
    <test qual="any" name="family">
        <string>mono</string>
    </test>
    <edit name="spacing" mode="assign">
        <int>100</int>
    </edit>
</match>
```

某些特定字体，如 Helvetica，在应用抗锯齿时可能会出现问题。通常情况下，这表现为字体在垂直方向上看起来被切成两半。在最坏的情况下，它可能会导致应用程序的崩溃。为了避免这种情况，可以考虑在 **local.conf** 中加入以下内容:

```
<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>
```

在编辑完 **local.conf** 后，确保用 </fontconfig> 标签结束该文件。不这样做会导致更改被忽略。

用户可以通过创建自己的 **~/.config/fontconfig/fonts.conf** 来添加个性化的设置。这个文件使用和上面说明相同的 XML 格式。

最后一点：对于 LCD 屏幕，可能需要进行子像素采样。这基本上是分别处理（水平分离的）红色、绿色和蓝色分量，以提高水平分辨率；其结果可能非常显而易见。要启用这一点，请在 **local.conf** 的空白之处添加这几行:

```
<match target="font">
    <test qual="all" name="rgba">
        <const>unknown</const>
    </test>
    <edit name="rgba" mode="assign">
        <const>rgb</const>
    </edit>
</match>
```

注意

根据显示器的种类，rgb 可能需要改为 bgr、vrgb 或 vbgr：试一下，看看哪种效果最好。

想要了解在 FreeBSD 上安装和配置字体的更多信息，请阅读字体和 FreeBSD[451] 这篇文章。

[451] https://docs.freebsd.org/en/articles/fonts/

第六章 FreeBSD 中的 Wayland

6.1.FreeBSD 中的 Wayland

Wayland 是一种支撑图形用户界面的新软件，但它在几个重要方面与 Xorg 有所不同。首先，Wayland 只是一种协议，充当客户端之间的中介，使用了不同的机制来消除对 X 服务器的依赖。Xorg 包含了 X11 协议，用于运行远程显示，X 服务器可接受连接并显示窗口。在 Wayland 下，显示服务器由混成器或窗口管理器提供而非传统的 X 服务器。

由于 Wayland 不是 X 服务器，需要利用其他方法才能实现传统的 X 屏幕连接（例如用 VNC 或 RDP 进行远程桌面管理）。其次，Wayland 可以将客户端和混成器之间的复合通信作为无需支持 X 协议的独立实体来管理。

Wayland 相对较新，并非所有软件都已更新为在没有 Xwayland 支持的情况下本地运行。因为 Wayland 不提供 X 服务器，并且期望混成器提供这种支持，所以尚不支持 Wayland 的 X11 窗口管理器需要不以 `-rootless` 参数启动 `Xwayland`。在删除 `-rootless` 参数后将恢复对 X11 窗口管理器支持。

> 注意
>
> 当前的 NVidia 驱动程序应该可以与大多数 wl-roots 混成器一起使用，但它可能有点不稳定并且目前不支持所有功能。请求志愿者帮助开发 NVidia DRM。

目前，很多软件都可以在 Wayland 上正常运行，包括 Firefox。还有一些桌面也可用，例如 Compiz Fusion 的替代品，名为 Wayfire，以及 i3 窗口管理器替代品——Sway。

> 注意
>
> 截至 2021 年 5 月，在 FreeBSD 中的 plasma5-kwin 可支持 Wayland。要在 Wayland 下使用 Plasma，请使用 `startplasma-wayland` 参数来 `ck-launch-session`，并将 dbus 与 `:ck-launch-session dbus-run-session startplasma-wayland` 匹配以使其工作。

对于混成器，必须有支持 evdev(4)[452] 驱动程序的内核才能使用快捷键功能。这是默认内置在 **GENERIC** 内核中的；但是，如果它被自定义并且 evdev(4)[453] 支持被移除，则需要加载 evdev(4)[454] 模块。此外，`Wayland`

[452] https://www.freebsd.org/cgi/man.cgi?query=evdev&sektion=4&format=html

[453] https://www.freebsd.org/cgi/man.cgi?query=evdev&sektion=4&format=html

[454] https://www.freebsd.org/cgi/man.cgi?query=evdev&sektion=4&format=html

的用户需要是 video 组的成员。要快速进行此更改，请使用 pw 命令：

```
pw groupmod video -m user
```

安装 Wayland 很简单，协议本身没有太多的配置。大部分的组件取决于所选的混成器。下面安装的 seatd 可作为混成器安装和配置的通用步骤，使其省略一步，因为需要使用 seatd 来提供对某些设备的非 root 访问权限。此处说明的所有混成器都应与 graphics/drm-kmod[455] 开源驱动程序一起使用；但是，NVidia 显卡在使用专有驱动程序时可能会出现问题。首先安装以下软件包：

```
# pkg install wayland seatd
```

在安装了协议和支持软件之后，混成器得创建用户界面。以下部分将介绍几个混成器。所有使用 Wayland 的混成器都需要在环境变量中定义一个运行时目录，这可以在 sh 中使用以下命令来实现：

```
% export XDG_RUNTIME_DIR=/var/run/user/`id -u`
```

需要注意的是，大多数混成器会在 XDG_RUNTIME_DIR 目录中搜索配置文件。在此处包含的示例中，将使用一个参数来指定 **~/.config** 中的配置文件，以将临时文件和配置文件分开。建议为每个混成器配置一个别名来加载指定的配置文件。

> 警告
>
> 据报告，ZFS 用户可能会遇到一些 Wayland 客户端的问题，因为他们需要访问运行时目录中的 posix_fallocate()。虽然作者无法在他们的 ZFS 系统上重现此问题，但推荐的解决方法是不要将 ZFS 用于运行时目录，而是将 tmpfs 用于 **/var/run** 目录。在这种情况下，tmpfs 文件系统用于 **/var/run** 并通过命令 mount -t tmpfs tmpfs /var/run 命令挂载，然后通过 **/etc/fstab** 使此更改在重新启动后保持不变。XDG_RUNTIME_DIR 环境变量可以配置为使用 **/var/run/user/$UID** 以避免 ZFS 的潜在陷阱。在查看以下部分中的配置示例时，请考虑该场景。

守护进程 seatd 帮助管理混成器中非 root 用户访问共享系统设备，比如显卡。对于传统的 X11 管理器（例如 Plasma 和 GNOME）不需要 seatd，但对于此处讨论的 Wayland 混成器，它需要在系统上启用并在启动混成器环境之前运行。要立即启动 seatd 守护程序，并在系统初始化时自启：

```
# sysrc seatd_enable="YES"
# service seatd start
```

之后，需要为图形界面环境安装一个类似于 X11 桌面的混成器。这里讨论了三个，包括基本配置选项、设置屏幕锁定以及更多信息的建议。

[455] https://cgit.freebsd.org/ports/tree/graphics/drm-kmod/pkg-descr

6.2.Wayfire 混成器

Wayfire 是一个旨在实现轻量级和可定制的混成器。有几个功能可用，它实现了以前发布的 Compiz Fusion 桌面的几个元素。所有组件在现代硬件上看起来都很漂亮。要启动并运行 Wayfire，首先要安装所需的软件包：

```
# pkg install wayfire wf-shell alacritty swaylock-effects swayidle wlogout kanshi␣
↪mako wlsunset
```

软件包 `alacritty` 提供了一个终端模拟器。尽管如此，并不是完全需要它，因为其他终端模拟器（例如 `kitty` 和 XFCE-4 `Terminal`）已经过测试和验证，可以在 Wayfire 混成器下工作。Wayfire 配置比较简单；它使用一个文件，任何自定义行为均应该对其进行修改。首先，将示例文件复制到运行时环境配置目录，然后编辑该文件：

```
% mkdir ~/.config/wayfire
% cp /usr/local/share/examples/wayfire/wayfire.ini ~/.config/wayfire
```

大多数用户的默认设置应该没问题。在配置文件中，像著名的 `cube` 这样的项目都是预先配置的，还有一些说明来帮助进行可用的设置。一些值得注意的主要设置包括：

```
[output]
mode = 1920x1080@60000
position = 0,0
transform = normal
scale = 1.000000
```

在这个例子中，从配置文件中，屏幕的输出应该是列出的刷新率。例如，刷新率应设置为 `widthx-height@refresh_rate`。该位置将输出放置在指定的特定像素位置。对于大多数用户来说，默认值应该没问题。最后，transform 设置背景变换，scale 将输出缩放到指定的比例因子。这些选项的默认值通常是可以接受的；有关详细信息，请参阅文档。

如前所述，Wayland 是新软件，并非所有应用程序都使用该协议。目前，`sddm` 似乎不支持在 Wayland 中启动和管理混成器。在这些示例中，已改为使用 swaylock 实用程序。配置文件包含运行 `swayidle` 和 `swaylock` 用于空闲和锁定屏幕的选项。此选项用于定义系统空闲时要执行的操作，如下所示：

```
idle = swaylock
```

并使用以下行配置屏幕超时锁定：

```
[idle]
toggle = <super> KEY_Z
screensaver_timeout = 300
dpms_timeout = 600
```

第一个选项会在 300 秒后锁屏，再过 300 秒后，屏幕会通过 `dpms_timeout` 选项关闭。

最后要注意的是 `<super>` 键。大部分配置都提到了这个键，就是键盘上传统的 `Windows` 键。大多数键盘都有这个 super 键可用；但是，如果它不可用，则应在此配置文件中重新映射。例如，要锁定屏幕，请按住 super 键、`shift` 键，然后按 `escape` 键。除非映射已更改，否则这将执行 `swaylock` 应用程序。`swaylock` 的默认配置会显示灰屏；但是，该应用程序高度可定制的并且有据可查。此外，由于 swaylock-effects 是被预装的版本，所以有几个参数可用，例如模糊效果，可以使用以下命令查看：

```
% swaylock --effect-blur 7x5
```

还有 `--clock` 参数将在锁屏上显示带有日期和时间的时钟。安装 x11/swaylock-effects[456] 时，包含默认的 **pam.d** 配置。它提供了适合大多数用户的默认选项。有关提供更高级的选项的详细信息，请参阅 PAM 文档。

此时，是时候测试 Wayfire 了，看看它是否可以在系统上启动。只需键入以下命令：

```
% wayfire -c ~/.config/wayfire/wayfire.ini
```

混成器现在应该启动并在屏幕顶部显示背景图像和菜单栏。Wayfire 将尝试列出已安装的桌面兼容应用程序并在此下拉菜单中显示它们；例如，如果安装了 XFCE-4 文件管理器，它将显示在此下拉菜单中。如果一个特定的应用程序是兼容的并且其键盘快捷键是必要的，则可以使用 **wayfire.ini** 配置文件将其映射到键盘序列。Wayfire 还有一个名为 Wayfire Config Manager 的配置工具。它位于下拉菜单栏中，但也可以通过终端通过执行以下命令来启动：

```
% wcm
```

可以通过此应用程序启用、禁用或配置各种 Wayfire 配置选项，包括复合特效。此外，为了更人性化的体验，可以在配置文件中启用后台管理器、面板和停靠应用程序：

```
panel = wf-panel
dock = wf-dock
background = wf-background
```

> 警告
>
> 通过 `wcm` 所做的更改将覆盖在 **wayfire.ini** 配置文件中的自定义更改。强烈建议对 **wayfire.ini** 文件进行备份，以便可以恢复任何重要的更改。

最后，**wayfire.ini** 中列出的默认启动器是 x11/wf-shell[457]，如果用户有需要，可以将其替换为其他面板。

[456] https://cgit.freebsd.org/ports/tree/x11/swaylock-effects/pkg-descr

[457] https://cgit.freebsd.org/ports/tree/x11/wf-shell/pkg-descr

6.3.Hikari 混成器

Hikari 混成器使用了几个以生产力为中心的概念，例如工作表、工作区等。这样，它类似于平铺窗口管理器。打破这一点，混成器从单个工作区开始，类似于虚拟桌面。Hikari 使用单个工作区或虚拟桌面进行用户交互。工作区由多个视图组成，这些视图是混成器中的工作窗口，分为工作表或组。工作表和组都由一组视图组成；再次，组合在一起的窗口。在工作表或组之间切换时，活动工作表或组将统称为工作区。手册页会将其分解为有关每个功能的更多信息，但对于本文档，仅考虑使用单个工作表的单个工作区。Hikari 安装将包含一个单独的包 x11-wm/hikari[458] 和一个终端模拟器 `alacritty`：

```
# pkg install hikari alacritty
```

> 注意
>
> 其他 shell，例如 `kitty` 或 Plasma `Terminal`，可在 Wayland 下运行。用户可尝试使用他们最喜欢的终端编辑器来验证兼容性。

Hikari 使用配置文件 `hikari.conf`，它可以放在 XDG_RUNTIME_DIR 中，也可以在启动时使用 `-c` 参数指定。不需要自动启动配置文件，但可能会使迁移到此混成器更容易一些。开始配置是创建 Hikari 配置目录并复制配置文件进行编辑：

```
% mkdir ~/.config/hikari
% cp /usr/local/etc/hikari/hikari.conf ~/.config/hikari
```

配置分为不同节，例如 ui、outputs、layouts 等。对于大多数用户来说，默认设置可以正常工作；但是，应该做出一些重要的改变。例如，$TERMINAL 变量通常不在用户环境中设置。更改此变量或更改 **hikari.conf** 文件以读取：

```
terminal = "/usr/local/bin/alacritty"
```

将使用快捷键启动 `alacritty` 终端。在浏览配置文件时，应该注意大写字母用于为用户映射键。比如启动终端的 `L` 键，`L+Return` 其实就是前面讲的 super 键或者 Windows 徽标键。因此，按住 `L/super/Windows` 键并按回车键将使用默认配置打开指定的终端仿真器。将其他键映射到应用程序需要创建操作定义。为此，操作项应列在 actions 节中，例如：

```
actions {
  terminal = "/usr/local/bin/alacritty"
  browser = "/usr/local/bin/firefox"
}
```

然后可以在键盘分节下映射一个动作，该节在 bindings 节中定义：

[458] https://cgit.freebsd.org/ports/tree/x11-wm/hikari/pkg-descr

```
bindings {
  keyboard {
SNIP
    "L+Return" = action-terminal
    "L+b" = action-browser
SNIP
```

重启 Hikari 后，按住 Windows 徽标按钮并按键盘上的 b 键将启动 Web 浏览器。混成器没有菜单栏，建议用户在迁移前至少设置一个终端模拟器。手册页包含大量文档，在执行完整迁移之前应该阅读它。关于 Hikari 的另一个积极方面是，在迁移到混成器时，Hikari 可以在 Plasma 和 GNOME 桌面环境中启动，允许在完全迁移之前进行测试。

在 Hikari 中锁定屏幕很容易，因为软件包中捆绑了默认的 **pam.d** 配置文件和解锁实用程序。锁定屏幕的键绑定是 L（Windows 徽标键）+ Shift+退格键。需要注意的是，所有未标记为公开的视图都将被隐藏。这些视图在锁定时永远不会接受输入，但要注意敏感信息是可见的。对于某些用户来说，迁移到本节讨论的其他屏幕锁定实用程序（如 swaylock-effects）可能更容易。要启动 Hikari，请使用以下命令：

```
% hikari -c ~/.config/hikari/hikari.conf
```

6.4.Sway 混成器

Sway 混成器是一种平铺式混成器，其试图取代 i3 窗口管理器。它应该兼容于用户当前的 i3 配置；但是，新功能可能需要一些额外的设置。在接下来的示例中，将假定全新安装而无需迁移任何 i3 配置。要安装 Sway 和有用的组件，请以 root 用户身份执行以下命令：

```
# pkg install sway swayidle swaylock-effects alacritty dmenu-wayland dmenu
```

对于基本配置文件，执行以下命令，然后在复制配置文件后对其进行编辑：

```
% mkdir ~/.config/sway
% cp /usr/local/etc/sway/config ~/.config/sway
```

基本配置文件有许多默认参数，这对大多数用户来说都很好。应该进行一些重要的更改，如下所示：

```
# Logo key. Use Mod1 for Alt.
input * xkb_rules evdev
set $mod Mod4
# Your preferred terminal emulator
set $term alacritty
set $lock swaylock -f -c 000000
output "My Workstation" mode 1366x786@60Hz position 1366 0
```

(continues on next page)

(continued from previous page)

```
output * bg ~/wallpapers/mywallpaper.png stretch
### Idle configuration
exec swayidle -w \
          timeout 300 'swaylock -f -c 000000' \
          timeout 600 'swaymsg "output * dpms off"' resume 'swaymsg "output * dpms on"
↪' \
          before-sleep 'swaylock -f -c 000000'
```

在前面的示例中，加载了 evdev(4)[459] 事件的 `xkb` 规则，并将 $mod 键设置为快捷键的 Windows 徽标键。接下来将终端模拟器设置为 `alacritty`，并定义锁屏命令，在稍后会详细介绍；输出关键字、模式、位置、背景壁纸和 Sway 也被告知拉伸此壁纸以填充屏幕；最后，`swaylock` 设置为在 300 秒超时后守护并锁定屏幕，在 600 秒后将屏幕或显示器置于睡眠模式；锁屏的背景颜色为 000000，即黑色，也在这里定义。使用 swaylock-effects 时，还可以使用 `--clock` 参数显示时钟。有关更多选项，请参阅手册页。还应查看 sway-output(5)[460] 手册页，它包含大量有关自定义可用输出选项的信息。

在 Sway 中，要调出应用程序菜单，请按住 Windows 徽标键（mod）并按 `d` 键。可以使用键盘上的箭头键导航菜单。还有一种方法可以操作栏的布局并添加托盘；阅读 sway-bar(5)[461] 手册页以获取更多信息。默认配置会在右上角添加日期和时间。有关示例，请参见配置文件中的 `Bar` 节。默认情况下，配置不包括在上述示例之外锁定屏幕，启用锁定计时器。创建锁定键快捷键需要在 `Key bindings` 部分添加以下行：

```
# Lock the screen manually
bindsym $mod+Shift+Return exec $lock
```

现在可以使用按住 Windows 徽标键、按住 shift 并最后按 return 的组合来锁定屏幕。安装 Sway 时，无论是来自软件包还是 FreeBSD ports，都会安装 **pam.d** 的默认文件。大多数用户应该可以接受默认配置，但可以使用更高级的选项。阅读 PAM 文档以获取更多信息。

最后，要退出 Sway 并返回到 shell，请按住 Windows 徽标键、shift 键，然后按 `e` 键。将显示一个提示，其中包含退出 Sway 的选项。在迁移期间，可以通过 Plasma 等 X11 桌面上的终端模拟器启动 Sway。这使得在完全迁移到此混成器之前测试不同的更改和快捷键更容易一些。要启动 Sway，请执行以下命令：

```
% sway -c ~/.config/sway/config
```

[459] https://www.freebsd.org/cgi/man.cgi?query=evdev&sektion=4&format=html
[460] https://www.freebsd.org/cgi/man.cgi?query=sway-output&sektion=5&format=html
[461] https://www.freebsd.org/cgi/man.cgi?query=sway-bar&sektion=5&format=html

6.5.使用 Xwayland

安装 Wayland 时，应该已经安装了 Xwayland 二进制文件，除非是在没有 X11 支持的情况下编译的 Wayland。如果不存在 **/usr/local/bin/Xwayland** 这个文件，请使用以下命令安装 Xwayland：

```
# pkg install xwayland-devel
```

> 注意
>
> 推荐使用 Xwayland 的开发版本，并且应该与 Wayland 软件包一起安装。每个混成器都有一种启用或禁用此功能的方法。

安装 Xwayland 后，在所选混成器中对其进行配置。对于 Wayfire，**wayfire.ini** 文件中需要以下行：

```
xwayland = true
```

对于 Sway 混成器，应默认启用 Xwayland。尽管如此，还是建议在 **~/.config/sway/config** 中手动添加配置行，如下所示：

```
xwayland enable
```

最后，对于 Hikari，不需要进行任何更改。默认情况下内置了对 Xwayland 的支持。要禁用该支持，请使用 ports 重新编译软件包并在那时禁用 Xwayland support。

进行这些更改后，在命令行启动混成器并使用快捷键运行终端。在此终端中，执行 env 命令并搜索 DISPLAY 变量。如果混成器能够正确启动 Xwayland X 服务器，输出的环境变量应该类似于以下内容：

```
% env | grep DISPLAY
```

```
WAYLAND_DISPLAY=wayland-1
DISPLAY=:0
```

在此输出中，有一个默认的 Wayland 显示和 Xwayland 服务器的显示。验证 Xwayland 是否正常运行的另一种方法是使用安装和测试小程序：[x11/eyes] 并检查输出。如果 xeyes 应用程序启动并且眼睛跟随鼠标指针，则 Xwayland 运行正常。如果出现如下错误，则说明 Xwayland 初始化过程中发生了一些事情，可能需要重新安装：

```
Error: Cannot open display wayland-0
```

> 警告
>
> Wayland 的一个安全特性是，在没有运行 X 服务器的情况下，没有其他的网络监听器。启用 Xwayland 后，此安全功能将不再适用于该系统。

对于某些混成器，例如 Wayfire，可能无法正常启动 Xwayland。因此，env 将显示关于环境变量 DISPLAY 的如下信息：

```
% env | grep DISPLAY
```

```
DISPLAY=wayland-1
WAYLAND_DISPLAY=wayland-1
```

有时即使安装并配置了 Xwayfire，X11 应用程序也不能运行并出现显示问题。要解决此问题，请通过这两种方法验证是否已经存在使用 UNIX 套接字的 Xwayland 实例。首先，检查 sockstat 的输出并搜索 X11-unix：

```
% sockstat | grep x11
```

应该有类似于以下信息的内容：

```
trhodes  Xwayland   2734  8  stream /tmp/.X11-unix/X0
trhodes  Xwayland   2734  9  stream /tmp/.X11-unix/X0
trhodes  Xwayland   2734  10 stream /tmp/.X11-unix/X0
trhodes  Xwayland   2734  27 stream /tmp/.X11-unix/X0_
trhodes  Xwayland   2734  28 stream /tmp/.X11-unix/X0
```

这表明存在 X11 套接字。这可以通过尝试在混成器下运行的终端仿真器中手动执行 Xwayland 来进一步验证：

```
% Xwayland
```

如果 X11 套接字已经可用，则应向用户显示以下错误：

```
(EE)
Fatal server error:
(EE) Server is already active for display 0
    If this server is no longer running, remove /tmp/.X0-lock
    and start again.
(EE)
```

由于使用显示 0 的活动 X 显示，环境变量设置不正确，要解决此问题，请将 DISPLAY 环境变量更改为 :0 并尝试再次执行应用程序。以下示例使用 mail/claws-mail[462] 作为需要 Xwayland 服务的应用程序：

```
export DISPLAY=:0
```

[462] https://cgit.freebsd.org/ports/tree/mail/claws-mail/pkg-descr

在此更改之后，mail/claws-mail[463] 应用程序现在应该开始使用 Xwayland 并按预期运行。

6.6.使用 VNC 进行远程连接

在本文档的前面部分提到，Wayland 不提供与 Xorg 提供的相同的 X 服务器样式访问。相反，用户可以自由选择远程桌面协议，例如 RDP 或 VNC。FreeBSD ports 包含 wayvnc，它支持基于 wlroots 的混成器，例如这里讨论的混成器。可以使用以下方式安装此应用程序：

```
# pkg install wayvnc
```

与其他一些软件包不同，wayvnc 不附带配置文件。值得庆幸的是，手册页记录了重要的选项，并且可以将它们外推到一个简单的配置文件中：

```
address=0.0.0.0
enable_auth=true
username=username
password=password
private_key_file=/path/to/key.pem
certificate_file=/path/to/cert.pem
```

强烈建议生成密钥文件，以提高连接的安全性。调用时，wayvnc 会在 **~/.config/wayvnc/config** 中搜索配置文件。可以在启动服务器时使用 -C configuration_file 参数进行覆盖。因此，要启动 wayvnc 服务器，请执行以下命令：

```
% wayvnc -C ~/.config/wayvnc/config
```

> 注意
>
> 在撰写本文时，还没有 rc.d 脚本可以在系统初始化时启动 wayvnc。如果需要该功能，则需要创建本地启动文件。这大概是对 port 维护者的一个功能需求。

6.7. Wayland 登录管理器

虽然当前存在多个登录管理器并且正在慢慢迁移到 Wayland，但一种选择是 x11/ly[464] 文本用户界面 (TUI) 管理器。ly 使用最小配置就可以在系统初始化时呈现一个登录窗口，从而启动 Sway、Wayfire 和其他系统。要安装 ly，请执行以下命令：

```
# pkg install ly
```

会有一些配置提示出现，导入步骤是在 **/etc/gettytab** 中加入以下几行：

[463] https://cgit.freebsd.org/ports/tree/mail/claws-mail/pkg-descr

[464] https://cgit.freebsd.org/ports/tree/x11/ly/pkg-descr

```
Ly:\
  :lo=/usr/local/bin/ly:\
  :al=root:
```

然后修改 **/etc/ttys** 中的 ttyv1 行以匹配以下行：

```
ttyv1 "/usr/libexec/getty Ly" xterm onifexists secure
```

系统重新启动后，应该会出现登录信息。配置特定的设置，例如语言可编辑 **/usr/local/etc/ly/config.ini**。至少，该文件应包含先前在 **/etc/ttys** 中指定的指定 tty。

> 注意
> 如果将 ttyv0 设置为登录终端，可能需要按 alt 和 F1 键才能正确看到登录窗口。

当登录窗口出现时，使用左右箭头将在不同的、受支持的窗口管理器之间进行交换。

6.8. 实用工具

所有混成器都可以使用的一个实用的 Wayland 软件是 waybar。虽然 Wayfire 的确带有启动菜单，但易于使用且快速的任务栏对于任何混成器或桌面管理器来说都是一个很好的配件。一个快速且易于配置的 Wayland 兼容任务栏是 waybar。要安装软件包和支持的音频控制实用程序，请执行以下命令：

```
# pkg install pavucontrol waybar
```

要创建配置目录并复制默认配置文件，请执行以下命令：

```
% mkdir ~/.config/waybar
% cp /usr/local/etc/xdg/waybar/config ~/.config/waybar
```

`lalauncher` 实用程序为各种应用程序提供了一个启动栏。软件包没有提供示例配置文件，因此必须采取以下措施：

```
mkdir ~/.config/lavalauncher
```

仅包含 Firefox 的示例配置文件位于右侧，如下所示：

```
global-settings {
    watch-config-file = true;
}
bar {
    output            = eDP-1;
    position          = bottom;
```

(continues on next page)

(continued from previous page)

```
    background-colour = "#202020";
    # Condition for the default configuration set.
    condition-resolution = wider-than-high;
    config {
        position = right;
    }
    button {
        image-path          =     /usr/local/lib/firefox/browser/chrome/icons/default/
↪default48.png;
        command[mouse-left] =     /usr/local/bin/firefox;
    }
    button {
      image-path           =   /usr/local/share/pixmaps/thunderbird.png;
      command[mouse-left]  =   /usr/local/bin/thunderbird;
}
```

第二部分：常见任务

现在基础知识已经讲完了，本书的这一部分将讨论一些在 FreeBSD 中经常使用的特性。这些章节：

- 介绍了流行和实用的桌面应用程序：浏览器、生产力工具、文档查看器等等。
- 介绍了一些可用于 FreeBSD 的多媒体工具。
- 解释了编译定制 FreeBSD 内核以实现额外功能的过程。
- 详细介绍了打印系统，包括桌面和网络连接的打印机设置。
- 展示了如何在 FreeBSD 系统上运行 Linux 应用程序。

其中的一些章节建议预读相关文件，这一点在每章开头的提要中都有说明。

第七章网络

7.1.概述

本章将深入探讨网络配置和性能，展示 FreeBSD 操作系统强大的网络功能。无论是使用有线网络还是无线网络，本章都提供了在 FreeBSD 中配置和优化网络连接的全面指导。

在深入了解细节之前，读者最好对网络概念（如协议、网络接口和寻址）有基本的了解。

本章内容包括：

- 在 FreeBSD 中配置有线网络的能力，包括网络接口设置、寻址和自定义选项。
- 在 FreeBSD 中配置无线网络的技能，包括无线网络接口设置、安全协议和故障排除技术。
- FreeBSD 的网络功能及其卓越的网络性能享有盛誉。
- 了解 FreeBSD 支持的各种网络服务和协议，以及 DNS、DHCP 等的配置说明。

在高级网络[465]章节查找有关如何进行高级网络配置的更多信息。

7.2.设置网络

设置有线或无线连接是 FreeBSD 用户的一项常见任务。本节将介绍如何识别有线和无线网络适配器以及如何配置它们。

在开始配置之前，有必要了解以下网络情况：

- 网络是否启用 DHCP
- 如果网络没有启用 DHCP，需要使用静态 IP
- 网络掩码
- 默认网关的 IP 地址

[465] https://docs.freebsd.org/en/books/handbook/advanced-networking/#advanced-networking

提示
网络连接可能是在安装时由 bsdinstall(8)[466] 配置的。

7.2.1.识别网络适配器

FreeBSD 支持多种有线和无线网络适配器。请查看所用 FreeBSD 版本[467]的硬件兼容性列表，以确定网络适配器是否受支持。

要获取系统使用的网络适配器，请执行以下命令：

```
% pciconf -lv | grep -A1 -B3 network
```

输出结果应类似于下面的内容：

```
em0@pci0:0:25:0:        class=0x020000 rev=0x03 hdr=0x00 vendor=0x8086 device=0x10f5␣
↪subvendor=0x17aa subdevice=0x20ee
    vendor     = 'Intel Corporation'
    device     = '82567LM Gigabit Network Connection'
    class      = network
    subclass   = ethernet
--
iwn0@pci0:3:0:0:        class=0x028000 rev=0x00 hdr=0x00 vendor=0x8086 device=0x4237␣
↪subvendor=0x8086 subdevice=0x1211
    vendor     = 'Intel Corporation'
    device     = 'PRO/Wireless 5100 AGN [Shiloh] Network Connection'
    class      = networ
```

@ 符号前的文字是控制设备的驱动程序名称。在本例中，它们分别是 em(4)[468] 和 iwn(4)[469]。

注意
只有在 FreeBSD 没有正确检测到网络接口卡模块时，才需要加载该模块。
例如，要加载 alc(4)[470] 模块，请执行以下命令：

```
# kldload if_alc
```

或者，要在启动时将驱动程序作为模块加载，可在 **/boot/loader.conf** 中加入以下一行：

```
if_alc_load="YES"
```

[466] https://man.freebsd.org/cgi/man.cgi?query=bsdinstall&sektion=8&format=html
[467] https://www.freebsd.org/releases/
[468] https://man.freebsd.org/cgi/man.cgi?query=em&sektion=4&format=html
[469] https://man.freebsd.org/cgi/man.cgi?query=iwn&sektion=4&format=html
[470] https://man.freebsd.org/cgi/man.cgi?query=alc&sektion=4&format=html

7.3.有线网络

一旦加载了正确的驱动程序，就需要对网络适配器进行配置。FreeBSD 使用驱动程序名称后面的单元号来命名网络适配器。单元号代表适配器在启动时被检测到或随后被发现的顺序。

例如，`em0` 是系统中使用 em(4)[471] 驱动程序的第一个网络接口卡 (NIC)。

要显示网络接口配置，请输入以下命令：

```
% ifconfig
```

输出结果应类似于下面的内容：

```
em0: flags=8863<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
        options=481249b<RXCSUM,TXCSUM,VLAN_MTU,VLAN_HWTAGGING,VLAN_HWCSUM,LRO,WOL_
↪MAGIC,VLAN_HWFILTER,NOMAP>
        ether 00:1f:16:0f:27:5a
        inet6 fe80::21f:16ff:fe0f:275a%em0 prefixlen 64 scopeid 0x1
        inet 192.168.1.19 netmask 0xffffff00 broadcast 192.168.1.255
        media: Ethernet autoselect (1000baseT <full-duplex>)
        status: active
        nd6 options=23<PERFORMNUD,ACCEPT_RTADV,AUTO_LINKLOCAL>
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> metric 0 mtu 16384
        options=680003<RXCSUM,TXCSUM,LINKSTATE,RXCSUM_IPV6,TXCSUM_IPV6>
        inet6 ::1 prefixlen 128
        inet6 fe80::1%lo0 prefixlen 64 scopeid 0x2
        inet 127.0.0.1 netmask 0xff000000
        groups: lo
        nd6 options=21<PERFORMNUD,AUTO_LINKLOCAL>
```

在此示例中，显示了以下设备：

- `em0`：以太网接口。
- `lo0`：环路接口是一种软件环回机制，可用于性能分析、软件测试和/或本地通信。更多信息请参见 lo(4)[472]。

示例显示 `em0` 已启动并运行。

主要指标包括：

1. `UP` 表示接口已配置就绪。
2. 接口的 IPv4 (`inet`) 地址是 `192.168.1.19`。
3. 接口的 IPv6 (`inet6`) 地址是 `fe80::21f:16ff:fe0f:275a%em0`。

[471] https://man.freebsd.org/cgi/man.cgi?query=em&sektion=4&format=html

[472] https://man.freebsd.org/cgi/man.cgi?query=lo&sektion=4&format=html

4. 它有一个有效的子网掩码（netmask），其中 0xffffff00 与 255.255.255.0 相同。
5. 它有一个有效的广播地址，即 192.168.1.255。
6. 接口（以太网）的 MAC 地址是 00:1f:16:0f:27:5a。
7. 物理介质选择为自动选择模式（介质：以太网自动选择（1000baseT ））。
8. 链路状态（status）为活动状态，表示已检测到载波信号。对于 em0，当以太网电缆未插入接口时，状态：无载波状态是正常的。

如果 ifconfig(8)[473] 的输出显示与下一个输出类似，则表明接口尚未配置：

```
em0: flags=8822<BROADCAST,SIMPLEX,MULTICAST> metric 0 mtu 1500
        options=481249b<RXCSUM,TXCSUM,VLAN_MTU,VLAN_HWTAGGING,VLAN_HWCSUM,LRO,WOL_
↪MAGIC,VLAN_HWFILTER,NOMAP>
        ether 00:1f:16:0f:27:5a
        media: Ethernet autoselect
        status: no carrier
        nd6 options=29<PERFORMNUD,IFDISABLED,AUTO_LINKLOCAL>
```

7.3.1.配置静态 IPv4 地址

本节将介绍如何在 FreeBSD 系统上配置静态 IPv4 地址。

网络接口卡配置可通过 ifconfig(8)[474] 命令行执行，但重启后将无法保留，除非配置也添加到 **/etc/rc.conf** 中。

> 注意
>
> 如果网络是在安装过程中通过 bsdinstall(8)[475] 配置的，则网络接口卡 (NIC) 的某些条目可能已经存在。请在执行 sysrc(8)[476] 之前仔细检查 **/etc/rc.conf**。

可以执行以下命令设置 IP 地址：

```
# ifconfig em0 inet 192.168.1.150/24
```

要使更改在重启后仍然有效，请执行以下命令：

```
# sysrc ifconfig_em0="inet 192.168.1.150 netmask 255.255.255.0"
```

执行以下命令添加默认路由器：

[473] https://man.freebsd.org/cgi/man.cgi?query=ifconfig&sektion=8&format=html
[474] https://man.freebsd.org/cgi/man.cgi?query=ifconfig&sektion=8&format=html
[475] https://man.freebsd.org/cgi/man.cgi?query=bsdinstall&sektion=8&format=html
[476] https://man.freebsd.org/cgi/man.cgi?query=sysrc&sektion=8&format=html

```
# sysrc defaultrouter="192.168.1.1"
```

将 DNS 记录添加到 **/etc/resolv.conf** 中：

```
nameserver 8.8.8.8
nameserver 8.8.4.4
```

然后重启 `netif` 并执行以下 `routing` 命令：

```
# service netif restart && service routing restart
```

可以使用 ping(8)[477] 测试连接：

```
% ping -c2 www.FreeBSD.org
```

输出结果应类似于下面的内容：

```
PING web.geo.FreeBSD.org (147.28.184.45): 56 data bytes
64 bytes from 147.28.184.45: icmp_seq=0 ttl=51 time=55.173 ms
64 bytes from 147.28.184.45: icmp_seq=1 ttl=51 time=53.093 ms

--- web.geo.FreeBSD.org ping statistics ---
2 packets transmitted, 2 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 53.093/54.133/55.173/1.040 ms
```

7.3.2.配置动态 IPv4 地址

如果网络上有 DHCP 服务器，那么配置网络接口使用 DHCP 就非常简单了。FreeBSD 使用 dhclient(8)[478] 作为 DHCP 客户端。dhclient(8)[479] 会自动提供 IP、掩码和默认路由器。

要使接口使用 DHCP，请执行以下命令：

```
# sysrc ifconfig_em0="DHCP"
```

可以通过运行以下命令手动使用 dhclient(8)[480]：

```
# dhclient em0
```

[477] https://man.freebsd.org/cgi/man.cgi?query=ping&sektion=8&format=html
[478] https://man.freebsd.org/cgi/man.cgi?query=dhclient&sektion=8&format=html
[479] https://man.freebsd.org/cgi/man.cgi?query=dhclient&sektion=8&format=html
[480] https://man.freebsd.org/cgi/man.cgi?query=dhclient&sektion=8&format=html

输出结果应类似于下面的内容：

```
DHCPREQUEST on em0 to 255.255.255.255 port 67
DHCPACK from 192.168.1.1
unknown dhcp option value 0x7d
bound to 192.168.1.19 -- renewal in 43200 seconds.
```

这样就可以验证使用 DHCP 分配地址是否正确。

> 提示
>
> dhclient(8)[481] 客户端可以在后台启动。这可能会给依赖于工作网络的应用程序带来麻烦，但在许多情况下，它能提供更快的启动速度。
>
> 要在后台执行 dhclient(8)[482]，请执行以下命令：
>
> ```
> # sysrc background_dhclient="YES"
> ```

然后执行以下命令重启 `netif`：

```
# service netif restart
```

可以使用 ping(8)[483] 测试连接：

```
% ping -c2 www.FreeBSD.org
```

输出结果应类似于下面的内容：

```
PING web.geo.FreeBSD.org (147.28.184.45): 56 data bytes
64 bytes from 147.28.184.45: icmp_seq=0 ttl=51 time=55.173 ms
64 bytes from 147.28.184.45: icmp_seq=1 ttl=51 time=53.093 ms

--- web.geo.FreeBSD.org ping statistics ---
2 packets transmitted, 2 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 53.093/54.133/55.173/1.040 ms
```

[481] https://man.freebsd.org/cgi/man.cgi?query=dhclient&sektion=8&format=html
[482] https://man.freebsd.org/cgi/man.cgi?query=dhclient&sektion=8&format=html
[483] https://man.freebsd.org/cgi/man.cgi?query=ping&sektion=8&format=html

7.3.3.IPv6

IPv6 是熟知的 IP 协议（又称 IPv4）的新版本。

与 IPv4 相比，IPv6 具有多项优势和许多新功能：

- 其 128 位地址空间可容纳 340,282,366,920,938,463,463,374,607,431,768,211,456 个地址。这解决了 IPv4 地址短缺和最终 IPv4 地址耗尽的问题。
- 路由器只在路由表中存储网络聚合地址，从而将路由表的平均空间减少到 8192 个条目。这解决了与 IPv4 相关的可扩展性问题，因为 IPv4 要求互联网路由器之间交换每个分配的 IPv4 地址块，导致路由表过于庞大，无法进行有效的路由选择。
- 地址自动配置（RFC2462）。
- 强制组播地址。
- 内置 IPsec（IP 安全）。
- 简化首部结构。
- 支持移动 IP。
- IPv6 到 IPv4 过渡机制。

FreeBSD 包含 KAME 项目[484] IPv6 参考实现，并提供使用 IPv6 所需的一切。

本节主要介绍如何配置和运行 IPv6。

IPv6 地址有三种不同类型：

Unicast

发送到单播地址的数据包会到达属于该地址的接口。

Anycast

这些地址在语法上与单播地址没有区别，但它们针对的是一组接口。指向任播地址的数据包将到达最近的路由器接口。任播地址只用于路由器。

Multicast

这些地址标识一组接口。发送到组播地址的数据包将到达属于组播组的所有接口。IPv4 广播地址（通常为 `xxx.xxx.xxx.255`）在 IPv6 中用组播地址表示。

读取 IPv6 地址时，标准形式表示为 `x:x:x:x:x:x:x:x:x:x:x:x:x:x:x:x`，其中每个 `x` 代表一个 16 位十六进制值。例如 `FEBC:A574:382B:23C1:AA49:4592:4EFE:9982`。

通常情况下，一个地址会有很长的全为零的子串。可以使用 `::`（双冒号）来替换每个地址的一个子串。此外，每个十六进制值最多可以省略三个前导 0。例如，`fe80::1` 对应于标准格式 `fe80:0000:0000:0000:0000:0000:0000:0001`。

[484] http://www.kame.net/

第三种形式是使用众所周知的 IPv4 符号书写最后 32 位。例如，`2002::10.0.0.1` 相当于十六进制的规范表示 `2002:0000:0000:0000:0000:0000:0a00:0001`，反过来又相当于 `2002::a00:1`。

要查看 FreeBSD 系统的 IPv6 地址，请执行以下命令：

```
# ifconfig
```

输出结果应类似于下面的内容：

```
em0: flags=8863<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
        options=481249b<RXCSUM,TXCSUM,VLAN_MTU,VLAN_HWTAGGING,VLAN_HWCSUM,LRO,WOL_
↪MAGIC,VLAN_HWFILTER,NOMAP>
        ether 00:1f:16:0f:27:5a
        inet 192.168.1.150 netmask 0xffffff00 broadcast 192.168.1.255
        inet6 fe80::21f:16ff:fe0f:275a%em0 prefixlen 64 scopeid 0x1
        media: Ethernet autoselect (1000baseT <full-duplex>)
        status: active
        nd6 options=23<PERFORMNUD,ACCEPT_RTADV,AUTO_LINKLOCAL>
```

在本例中，em0 接口使用的是 `fe80::21f:16ff:fe0f:275a%em0`，这是一个自动配置的链路本地地址，由 MAC 地址自动生成。

某些 IPv6 地址已被保留。保留地址列表见下表：

表 1.IPv6 保留地址示例

IPv6 地址	前缀长度（比特）	描述	备注
`::`	128 比特	未指明	相当于 IPv4 中的 `0.0.0.0`。
`::1`	128 比特	回环地址	相当于 IPv4 中的 `127.0.0.1`。
`::00:xx:xx:xx:xx`	96 比特	内置 IPv4	低 32 位是兼容的 IPv4 地址。
`::ff:xx:xx:xx:xx`	96 比特	IPv4 映射 IPv6 地址	低 32 位是不支持 IPv6 的主机的 IPv4 地址。
`fe80::/10`	10 比特	链接本地	相当于 IPv4 中的 169.254.0.0/16。
`fc00::/7`	7 比特	唯一本地	唯一的本地地址用于本地通信，只能在一组合作站点内路由。
`ff00::`	8 比特	组播	
`2000::-3fff::`	3 比特	全局单播	所有全局单播地址都从该地址池中分配。前 3 位是 `001`。

有关 IPv6 地址结构的更多信息，请参阅 RFC3513[485]。

[485] http://www.ietf.org/rfc/rfc3513.txt

7.3.4.配置静态 IPv6 地址

要将 FreeBSD 系统配置为具有静态 IPv6 地址的 IPv6 客户端，必须设置 IPv6 地址。

执行以下命令以满足要求：

```
# sysrc ifconfig_em0_ipv6="inet6 2001:db8:4672:6565:2026:5043:2d42:5344 prefixlen 64"
```

要指定默认路由器，请执行以下命令指定其地址：

```
# sysrc ipv6_defaultrouter="2001:db8:4672:6565::1"
```

7.3.5.配置动态 IPv6 地址

如果网络有 DHCP 服务器，则很容易将网络接口配置为使用 DHCP。dhclient(8)[486] 会自动提供 IP、掩码和默认路由器。

要使接口启用 DHCP，请执行以下命令：

```
# sysrc ifconfig_em0_ipv6="inet6 accept_rtadv"
# sysrc rtsold_enable="YES"
```

7.3.6.路由器通告和主机自动配置

本节演示如何在 IPv6 路由器上设置 rtadvd(8)[487]，以公告 IPv6 网络前缀和默认路由。

要启用 rtadvd(8)[488]，请执行以下命令：

```
# sysrc rtadvd_enable="YES"
```

指定 IPv6 路由器通告的接口非常重要。例如，告诉 rtadvd(8)[489] 使用 em0：

```
# sysrc rtadvd_interfaces="em0"
```

接下来，创建配置文件 **/etc/rtadvd.conf**，如本例所示：

```
em0:\
    :addrs#1:addr="2001:db8:1f11:246::":prefixlen#64:tc=ether:
```

[486] https://man.freebsd.org/cgi/man.cgi?query=dhclient&sektion=8&format=html
[487] https://man.freebsd.org/cgi/man.cgi?query=rtadvd&sektion=8&format=html
[488] https://man.freebsd.org/cgi/man.cgi?query=rtadvd&sektion=8&format=html
[489] https://man.freebsd.org/cgi/man.cgi?query=rtadvd&sektion=8&format=html

用要使用的接口替换 em0，用分配的前缀替换 2001:db8:1f11:246::。

对于专用的 /64 子网，无需更改任何其他内容。否则，请将 prefixlen# 更改为正确的值。

7.3.7.IPv6 和 IPv4 地址映射

当服务器启用 IPv6 时，可能需要启用 IPv4 映射 IPv6 地址通信。这种兼容性选项允许将 IPv4 地址表示为 IPv6 地址。允许 IPv6 应用程序与 IPv4 通信，反之亦然，这可能是一个安全问题。

大多数情况下可能不需要此选项，仅为兼容而提供。该选项允许纯 IPv6 应用程序在双协议栈环境中与 IPv4 一起工作。这对可能不支持纯 IPv6 环境的第三方应用程序最有用。

要启用此功能，请执行以下命令：

```
# sysrc ipv6_ipv4mapping="YES"
```

7.4.无线网络

大多数无线网络都基于 IEEE® 802.11 标准[490]。

FreeBSD 支持使用 802.11a[491]、802.11b[492]、802.11g[493] 和 802.11n[494] 的网络。

> 注意
> FreeBSD 上的 802.11ac[495] 支持目前正在开发中。

一个基本的无线网络由多个无线通信站组成，这些无线通信站使用 2.4GHz 或 5GHz 频段的无线电进行广播，但具体频段因地而异，而且也在不断变化，以实现 2.3GHz 和 4.9GHz 频段的通信。

配置无线网络有三个基本步骤：

1. 扫描并选择接入点
2. 验证工作站
3. 配置 IP 地址或使用 DHCP

下文将讨论每个步骤。

[490] https://en.wikipedia.org/wiki/IEEE_802.11
[491] https://en.wikipedia.org/wiki/IEEE_802.11a-1999
[492] https://en.wikipedia.org/wiki/IEEE_802.11b-1999
[493] https://en.wikipedia.org/wiki/IEEE_802.11g-2003
[494] https://en.wikipedia.org/wiki/IEEE_802.11n-2009
[495] https://en.wikipedia.org/wiki/IEEE_802.11ac-2013

7.4.1.连接到无线网络的快速入门

将 FreeBSD 连接到现有的无线网络是一种非常常见的情况。

本程序显示了所需的步骤：

- 第一步是从网络管理员处获取无线网络的 SSID（服务集标识符）和 PSK（预共享密钥）。
- 第二步是在 **/etc/wpa_supplicant.conf** 中为该网络添加一个条目。如果该文件不存在，请创建它：

```
network={
 ssid="myssid"
 psk="mypsk"
}
```

- 第三步是添加网络条目，以便在启动时配置网络：

```
# sysrc wlans_iwn0="wlan0"
# sysrc ifconfig_wlan0="WPA DHCP"
```

- 最后一步是执行以下命令重启 `netif` 服务：

```
# service netif restart
```

7.4.2.基本无线配置

第一步是将无线网卡配置到一个接口上。要了解系统中有哪些无线网卡，请查看识别网络适配器[496]章节。

```
# ifconfig wlan0 create wlandevice iwm0
```

要使更改在重启后仍然有效，请执行以下命令：

```
# sysrc wlans_iwn0="wlan0"
```

> 注意
>
> 由于世界各地的监管情况不尽相同，因此有必要正确设置适用于您所在地区的域，以便获得有关可使用哪些频道的正确信息。
>
> 可用的区域定义可在 **/etc/regdomain.xml** 中找到。要在运行时设置数据，请使用 `ifconfig`：
>
> ```
> # ifconfig wlan0 regdomain etsi2 country AT
> ```
>
> 要持久保存设置，请将其添加到 **/etc/rc.conf** 中：

[496] https://docs.freebsd.org/en/books/handbook/network/#config-identify-network-adapter

```
# sysrc create_args_wlan0="country AT regdomain etsi2"
```

7.4.3.扫描无线网络

可用 ifconfig(8)[497] 扫描可用的无线网络。

要列出无线网络，请执行以下命令：

```
# ifconfig wlan0 up list scan
```

输出结果应类似于下面的内容：

```
SSID/MESH ID                      BSSID              CHAN RATE    S:N     INT CAPS
FreeBSD                           e8:d1:1b:1b:58:ae     1   54M  -47:-96   100 EP  ␣
↪RSN BSSLOAD HTCAP WPS WME
NetBSD                            d4:b9:2f:35:fe:08     1   54M  -80:-96   100 EP  ␣
↪RSN BSSLOAD HTCAP WPS WME
OpenBSD                           fc:40:09:c6:31:bd    36   54M  -94:-96   100 EPS ␣
↪VHTPWRENV APCHANREP RSN WPS BSSLOAD HTCAP VHTCAP VHTOPMODE WME
GNU-Linux                         dc:f8:b9:a0:a8:e0    44   54M  -95:-96   100 EP  ␣
↪WPA RSN WPS HTCAP VHTCAP VHTOPMODE WME VHTPWRENV
Windows                           44:48:b9:b3:c3:ff    44   54M  -84:-96   100 EP  ␣
↪BSSLOAD VHTPWRENV HTCAP WME RSN VHTCAP VHTOPMODE WPS
MacOS                             46:48:b9:b3:c3:ff    44   54M  -84:-96   100 EP  ␣
↪BSSLOAD VHTPWRENV HTCAP WME RSN VHTCAP VHTOPMODE WPS
```

1. SSID/MESH ID 标识网络名称。
2. BSSID 标识接入点的 MAC 地址。
3. CAPS 字段标识每个网络的类型以及在其中运行的台站的功能（更多详情请参见 ifconfig(8)[498] 中的 `list scan` 的定义）。

7.4.4.连接和验证无线网络

从扫描的网络列表中选择无线网络后，就需要进行连接和验证。在绝大多数无线网络中，身份验证是通过路由器配置的密码完成的。其他方案则需要在数据传输前完成加密握手，要么使用预共享密钥或秘密，要么使用涉及 RADIUS 等后端服务的更复杂方案。

[497] https://man.freebsd.org/cgi/man.cgi?query=ifconfig&sektion=8&format=html
[498] https://man.freebsd.org/cgi/man.cgi?query=ifconfig&sektion=8&format=html

7.4.4.1.使用 WPA2/WPA/Personal 验证

无线网络的身份验证过程由 wpa_supplicant(8)[499] 管理。

wpa_supplicant(8)[500] 配置将在 **/etc/wpa_supplicant.conf** 文件中进行。更多信息，请参阅 wpa_supplicant.conf(5)[501]。

扫描无线网络、选择网络并设置密码 (PSK) 后，该信息将被添加到 **/etc/wpa_supplicant.conf** 文件中，如下例所示：

```
network={
        scan_ssid=1
        ssid="FreeBSD"
        psk="12345678"
}
```

下一步是在 **/etc/rc.conf** 文件中配置无线连接。

要使用静态地址，必须执行以下命令：

```
# sysrc ifconfig_wlan0="inet 192.168.1.20 netmask 255.255.255.0"
```

要使用动态地址，必须执行以下命令：

```
# ifconfig_wlan0="WPA DHCP"
```

然后执行以下命令重启网络：

```
# service netif restart
```

> 注意
> 有关如何执行更高级身份验证方法的更多信息，请访问无线高级身份验证。

7.4.4.2.使用开放网络进行身份验证

> 提示
> 重要的是，用户在没有任何认证的情况下连接开放网络时要非常小心。

完成无线网络扫描并选择无线网络的 SSID 后，执行以下命令：

```
# ifconfig wlan0 ssid SSID
```

[499] https://man.freebsd.org/cgi/man.cgi?query=wpa_supplicant&sektion=8&format=html
[500] https://man.freebsd.org/cgi/man.cgi?query=wpa_supplicant&sektion=8&format=html
[501] https://man.freebsd.org/cgi/man.cgi?query=wpa_supplicant.conf&sektion=5&format=html

然后执行 dhclient(8)[502] 获取配置的地址：

```
# dhclient wlan0
```

7.4.5.同时使用有线和无线连接

有线连接具有更好的性能和可靠性，而无线连接则具有灵活性和移动性。笔记本电脑用户通常希望在这两种连接之间无缝漫游。

在 FreeBSD 上，可以将两个甚至更多的网络接口以“故障切换”的方式组合在一起。这种配置使用一组网络接口中最优先的可用连接，当链路状态发生变化时，操作系统会自动切换。

链路聚合和故障切换[503]中介绍了链路聚合和故障切换，以太网和无线接口之间的故障切换模式[504]中提供了同时使用有线和无线连接的示例。

7.5.主机名

主机名代表网络上主机的完全合格域名（FQDN）。

> 提示
> 如果没有为主机设置主机名，FreeBSD 将为其赋值 Amnesiac。

7.5.1.检查当前主机名

hostname(1)[505] 可用于检查当前主机名：

```
$ hostname
```

输出结果应类似于下面的内容：

```
freebsdhostname.example.com
```

[502] https://man.freebsd.org/cgi/man.cgi?query=dhclient&sektion=8&format=html
[503] https://docs.freebsd.org/en/books/handbook/advanced-networking/#network-aggregation
[504] https://docs.freebsd.org/en/books/handbook/advanced-networking/#networking-lagg-wired-and-wireless
[505] https://man.freebsd.org/cgi/man.cgi?query=hostname&sektion=1&format=html

7.5.2.更改主机名

要更改主机名并在重启后继续使用，请执行以下命令：

```
# sysrc hostname="freebsdhostname.example.com"
```

7.6.DNS

DNS 可以理解为电话簿[506]，在这个电话簿中，IP 被识别为主机名，反之亦然。

有三个文件处理 FreeBSD 系统与 DNS 的交互。这三个文件是 hosts(5)[507]、resolv.conf(5)[508] 和 nsswitch.conf(5)[509]

除非 **/etc/nsswitch.conf** 文件中另有说明，否则 FreeBSD 将首先查看 **/etc/hosts** 文件中的地址，然后查看 **/etc/resolv.conf** 文件中的 DNS 信息。

> 注意
> nsswitch.conf(5)[510] 文件规定了 nsdispatch（名称服务交换调度程序）的运行方式。
> 默认情况下，**/etc/nswitch.conf** 文件的 hosts 部分如下：
>
> ```
> hosts: files dns
> ```
>
> 例如，在使用 nscd(8)[511] 服务时。可以通过保留以下行来更改优先顺序：
>
> ```
> hosts: files cache dns
> ```

7.6.1.本地地址

/etc/hosts 文件是一个简单的文本数据库，提供主机名与 IP 地址的映射。可将通过局域网连接的本地计算机的条目添加到该文件中，以实现简单的命名目的，而无需设置 DNS 服务器。此外，**/etc/hosts** 还可用于提供互联网名称的本地记录，从而减少为常用名称查询外部 DNS 服务器的需要。

例如，如果要在本地环境中创建一个 www/gitlab-ce[512] 实例，可以在 **/etc/hosts** 文件中添加如下内容：

```
192.168.1.150 git.example.com git
```

[506] https://en.wikipedia.org/wiki/Telephone_directory
[507] https://man.freebsd.org/cgi/man.cgi?query=hosts&sektion=5&format=html
[508] https://man.freebsd.org/cgi/man.cgi?query=resolv.conf&sektion=5&format=html
[509] https://man.freebsd.org/cgi/man.cgi?query=nsswitch.conf&sektion=5&format=html
[510] https://man.freebsd.org/cgi/man.cgi?query=nsswitch.conf&sektion=5&format=html
[511] https://man.freebsd.org/cgi/man.cgi?query=nscd&sektion=8&format=html
[512] https://cgit.freebsd.org/ports/tree/www/gitlab-ce/

7.6.2.配置名称服务器

FreeBSD 系统访问 Internet 域名系统 (DNS) 的方式由 resolv.conf(5)[513] 控制。

/etc/resolv.conf 中最常见的条目是

名称	解释
`nameserver`	解析器应查询的名称服务器的 IP 地址。服务器按所列顺序查询，最多查询三个。
`search`	主机名查询的搜索列表。这通常由本地主机名的域决定。
`domain`	本地域名。

典型的 **/etc/resolv.conf** 文件如下：

```
search example.com
nameserver 147.11.1.11
nameserver 147.11.100.30
```

注意
只能使用其中一个搜索和域选项。

使用 DHCP 时，dhclient(8)[514] 通常会根据从 DHCP 服务器接收到的信息重写 **/etc/resolv.conf**。

提示
如果进行配置的机器不是 DNS 服务器，则可以使用 local-unbound(8)[515] 来提高 DNS 查找性能。
要在启动时启用它，请执行以下命令：

```
# sysrc local_unbound_enable="YES"
```

要启动 local-unbound(8)[516] 服务，请执行以下命令：

[513] https://man.freebsd.org/cgi/man.cgi?query=resolv.conf&sektion=5&format=html
[514] https://man.freebsd.org/cgi/man.cgi?query=dhclient&sektion=8&format=html
[515] https://man.freebsd.org/cgi/man.cgi?query=local-unbound&sektion=8&format=html
[516] https://man.freebsd.org/cgi/man.cgi?query=local-unbound&sektion=8&format=html

```
# service local_unbound start
```

7.7.故障排除

在排除硬件和软件配置故障时，先检查简单的东西。

- 网线是否插好?
- 网络服务配置是否正确?
- 防火墙配置是否正确?
- 网卡是否受 FreeBSD 支持?
- 路由器是否正常工作?

> 提示
>
> 在发送错误报告之前，请务必查看 FreeBSD 发行版页面中的硬件说明，将 FreeBSD 版本更新到最新的稳定版，查看邮件列表存档，并在互联网上搜索。

7.7.1.有线网络的故障排除

如果网卡正常工作，但性能不佳，请阅读 tuning(7)[517]。此外，请检查网络配置，因为不正确的网络设置会导致连接缓慢。

如果系统无法将数据包路由到目标主机，则会出现 `No route to host` 提示。如果没有指定默认路由或拔掉电缆，就会出现这种情况。检查 `netstat -rn` 的输出，确保有有效路由到达主机。如果没有，请阅读网关和路由[518]。

`ping: sendto: Permission denied` 错误信息通常是由于防火墙配置错误造成的。如果在 FreeBSD 上启用了防火墙但没有定义任何规则，默认的策略是拒绝所有流量，即使是 ping(8)[519]。请参阅防火墙[520]了解更多信息。

[517] https://man.freebsd.org/cgi/man.cgi?query=tuning&sektion=7&format=html

[518] https://docs.freebsd.org/en/books/handbook/advanced-networking/#network-routing

[519] https://man.freebsd.org/cgi/man.cgi?query=ping&sektion=8&format=html

[520] https://docs.freebsd.org/en/books/handbook/firewalls/#firewalls

7.7.2.无线网络的故障排除

本节介绍了一些帮助排除常见无线网络问题的步骤。

- 如果扫描时没有列出接入点，请检查配置是否将无线设备限制在一组有限的信道上。
- 如果设备无法与接入点关联，请检查配置是否与接入点的设置相匹配。这包括验证方案和任何安全协议。尽可能简化配置。如果使用 WPA2 或 WPA 等安全协议，可将接入点配置为开放式身份验证和无安全性，看看流量是否能通过。
- 一旦系统可以与接入点关联，就可以使用 ping(8)[521] 等工具诊断网络配置。
- 还有许多低级调试工具。可以使用 wlandebug(8)[522] 在 802.11 协议支持层启用调试信息。

[521] https://man.freebsd.org/cgi/man.cgi?query=ping&sektion=8&format=html

[522] https://man.freebsd.org/cgi/man.cgi?query=wlandebug&sektion=8&format=html

第八章桌面环境

8.1. 概述

虽然 FreeBSD 常常由于其性能和稳定性而被用作服务器，但是它也非常适合用作日常使用的桌面。通过 FreeBSD 提供的软件包和 ports，你可以获取超过 36000 个软件，并直接搭建起一个可以运行各种桌面应用程序的自定义桌面环境。本章将演示如何安装流行的桌面环境以及桌面应用程序，例如网络浏览器、生产力工具、文档浏览器和财务软件。

要求：

- 本章的读者应该已经了解了如何在 FreeBSD 上安装 X Window 系统[523] 或 Wayland[524]。
- 本章将指导读者安装官方软件包。要从 ports 编译自定义软件包，请参阅使用 Ports[525] 一节。

8.2. 桌面环境

本节介绍如何在 FreeBSD 系统上安装和配置一些流行的桌面环境。桌面环境可以是从简单的窗口管理器到一整套桌面应用程序。

表 1.受支持的桌面环境

[523] https://docs.freebsd.org/en/books/handbook/book/#x11

[524] https://docs.freebsd.org/en/books/handbook/book/#wayland

[525] https://docs.freebsd.org/en/books/handbook/book/#ports-using

名称	许可证	软件包
KDE Plasma	GPL 2.0 或更高版本	x11/kde5
GNOME	GPL 2.0 或更高版本	x11/gnome
XFCE	GPL、LGPL、BSD	x11-wm/xfce4
MATE	GPL 2.0、LGPL 2.0	x11/mate
Cinnamon	GPL 2.0 或更高版本	x11/cinnamon
LXQT	GPL、LGPL	x11-wm/lxqt

8.2.1. KDE Plasma

KDE Plasma 是一个易于使用的桌面环境。此桌面提供了一套具有一致外观、标准化菜单和工具栏、快捷键、配色方案、国际化以及集中式、对话框驱动的桌面配置的软件。有关 KDE 的更多信息，请访问 KDE 主页[526]。有关 FreeBSD 的具体信息，请参阅 KDE 上的 FreeBSD 主页[527]。

8.2.1.1. 安装 KDE Plasma 元包

要随 KDE Plasma 元包一起安装 KDE 框架、Plasma 桌面和应用程序，请执行：

```
# pkg install kde5
```

8.2.1.2. 最简化安装 KDE Plasma

要安装最精简的 KDE Plasma，请执行：

```
# pkg install plasma5-plasma
```

> 提示
> 这种安装 过于简化，以致于必须单独安装 Konsole，请执行：
>
> ```
> # pkg install konsole
> ```

[526] https://kde.org/
[527] https://freebsd.kde.org/

8.2.1.3. 配置 KDE Plasma

KDE Plasma 使用 dbus-daemon(1)[528] 作为消息总线和硬件抽象层。此软件是作为 KDE Plasma 的依赖项自动安装的。

在 `/etc/rc.conf` 中启用 D-BUS 服务以在系统启动时启动：

```
# sysrc dbus_enable="YES"
```

增加消息大小，请执行：

```
sysctl net.local.stream.recvspace=65536
sysctl net.local.stream.sendspace=65536
```

8.2.1.4. 启动 KDE Plasma

KDE Plasma 首选的显示管理器是 x11/sddm[529]。要安装 x11/sddm[530]，请执行：

```
# pkg install sddm
```

在 `/etc/rc.conf` 中启用 SDDM 服务以在系统启动时启动：

```
# sysrc sddm_enable="YES"
```

通过运行以下命令（本例为西班牙语），可以在 SDDM 中设置键盘语言：

```
# sysrc sddm_lang="es_ES"
```

第二种启动 KDE Plasma 的方法是手动调用 startx(1)[531]。要使其工作，请在 `~/.xinitrc` 中需要添加以下行：

```
% echo "exec ck-launch-session startplasma-x11" > ~/.xinitrc
```

[528] https://man.freebsd.org/cgi/man.cgi?query=dbus-daemon&sektion=1&format=html
[529] https://cgit.freebsd.org/ports/tree/x11/sddm/pkg-descr
[530] https://cgit.freebsd.org/ports/tree/x11/sddm/pkg-descr
[531] https://man.freebsd.org/cgi/man.cgi?query=startx&sektion=1&format=html

8.2.2. GNOME

GNOME 是一个用户友好的桌面环境。它包括一个用于启动应用程序和显示状态的面板、一个桌面、一组工具和应用程序，以及一组使应用程序易于协作和保持一致的约定。

8.2.2.1. 安装 GNOME 元包

要随 GNOME 元包安装 GNOME 桌面和应用程序，请执行：

```
# pkg install gnome
```

8.2.2.2. 最简化 GNOME 安装

要安装 GNOME-lite 元包，并将 GNOME 桌面精简为仅用于基本功能，请执行：

```
# pkg install gnome-lite
```

8.2.2.3. 配置 GNOME

GNOME 需要挂载 `/proc`。将此行添加到 `/etc/fstab` 中，以便在系统启动期间自动挂载此文件系统：

```
# Device                Mountpoint      FStype  Options         Dump    Pass#
proc                    /proc           procfs  rw              0       0
```

GNOME 使用 dbus-daemon(1)[532] 作为消息总线和硬件抽象层。此软件是作为 GNOME 的依赖项自动安装的。

在 `/etc/rc.conf` 中启用 D-BUS 服务以在系统启动时启动：

```
# sysrc dbus_enable="YES"
```

8.2.2.4. 启动 GNOME

GNOME 显示管理器是 GNOME 的首选显示管理器。GDM 是作为 GNOME 软件包的一部分被安装的。

在 `/etc/rc.conf` 中启用 GDM 以在系统启动时启动：

```
# sysrc gdm_enable="YES"
```

[532] https://man.freebsd.org/cgi/man.cgi?query=dbus-daemon&sektion=1&format=html

第二种启动 GNOME 的方法是手动调用 startx(1)[533]。要使其工作，在 `~/.xinitrc` 中需要添加以下行：

```
% echo "exec gnome-session" > ~/.xinitrc
```

8.2.3. XFCE

XFCE 是一个基于 GTK+ 的桌面环境，轻量化，并提供了一个简单、高效、易用的桌面。它是完全可自定义的，有一个带有菜单、小程序和应用程序启动器的主面板，提供文件管理器和声音管理器，并且可以进行主题化。由于它速度快、轻量化、效率高，因此非常适合有内存限制的较老或较慢的机器。

8.2.3.1. 安装 XFCE

要安装 XFCE 元包，请执行：

```
# pkg install xfce
```

8.2.3.2. 配置 XFCE

XFCE 要求挂载 `/proc`。将此行添加到 `/etc/fstab` 中，以便在系统启动期间自动挂载此文件系统：

```
# Device                Mountpoint      FStype  Options         Dump    Pass#
proc                    /proc           procfs  rw              0       0
```

XFCE 使用 dbus-daemon(1)[534] 作为消息总线和硬件抽象层。此软件是作为 XFCE 的依赖项自动安装的。

在 `/etc/rc.conf` 中启用 D-BUS 以在系统启动时启动：

```
# sysrc dbus_enable="YES"
```

8.2.3.3. 启动 XFCE

x11/lightdm[535] 是一款支持不同显示技术的显示管理器，是一个很好的选择，因为它非常轻，只需要很少的内存占用，并且性能卓越。

要安装它，请执行：

[533] https://man.freebsd.org/cgi/man.cgi?query=startx&sektion=1&format=html
[534] https://man.freebsd.org/cgi/man.cgi?query=dbus-daemon&sektion=1&format=html
[535] https://cgit.freebsd.org/ports/tree/x11/lightdm/pkg-descr

```
# pkg install lightdm lightdm-gtk-greeter
```

在 /etc/rc.conf 中启用 lightdm 以在系统启动时启动：

```
# sysrc lightdm_enable="YES"
```

第二种启动 XFCE 的方法是手动调用 startx(1)[536]。要使其工作，在 ~/.xinitrc 中需要添加以下行：

```
% echo '. /usr/local/etc/xdg/xfce4/xinitrc' > ~/.xinitrc
```

8.2.4. MATE

MATE 桌面环境是 GNOME2 的延续。它使用传统的风格提供了一个直观而有吸引力的桌面环境。

8.2.4.1. 安装 MATE 元包

要随 MATE 元包一起安装包含 MATE 桌面和一些额外应用程序（如文本编辑器、归档管理器等），请执行：

```
# pkg install mate
```

8.2.4.2. 最简化 MATE 安装

要安装 MATE-lite 元包，MATE 桌面经过精简，仅用于基本功能，请执行：

```
# pkg install mate-base
```

8.2.4.3. 配置 MATE

MATE 需要挂载 /proc。将此行添加到 /etc/fstab 中，以便在系统启动期间自动挂载此文件系统：

```
# Device                Mountpoint      FStype  Options         Dump    Pass#
proc                    /proc           procfs  rw              0       0
```

MATE 使用 dbus-daemon(1)[537] 作为消息总线和硬件抽象层。此软件是作为 MATE 的依赖项自动安装的。在 /etc/rc.conf 中启用 D-BUS 以在系统启动时启动：

[536] https://man.freebsd.org/cgi/man.cgi?query=startx&sektion=1&format=html

[537] https://man.freebsd.org/cgi/man.cgi?query=dbus-daemon&sektion=1&format=html

```
# sysrc dbus_enable="YES"
```

8.2.4.4. 启动 MATE

x11/lightdm[538] 是一款支持不同显示技术的显示管理器，是一个很好的选择，因为它非常轻，只需要很少的内存占用，并且性能卓越。

要安装它，请执行：

```
# pkg install lightdm lightdm-gtk-greeter
```

在 `/etc/rc.conf` 中启用 lightdm 以在系统启动时启动：

```
# sysrc lightdm_enable="YES"
```

第二种启动 MATE 的方法是手动调用 startx(1)[539]。要使其工作，在 `~/.xinitrc` 中需要添加以下行：

```
% echo "exec ck-launch-session mate-session" > ~/.xinitrc
```

8.2.5. Cinnamon

Cinnamon 是一款 UNIX® 桌面，提供了先进的创新功能和传统的用户体验。桌面布局类似于 Gnome 2。底层技术是从 Gnome Shell 派生出来的。重点是让用户有宾至如归的感觉，并为他们提供易于使用和舒适的桌面体验。

8.2.5.1 安装 Cinnamon

要安装 Cinnamon 软件包，请执行以下操作：

```
# pkg install cinnamon
```

[538] https://cgit.freebsd.org/ports/tree/x11/lightdm/pkg-descr

[539] https://man.freebsd.org/cgi/man.cgi?query=startx&sektion=1&format=html

8.2.5.2 配置 Cinnamon

Cinnamon 需要挂载 /proc。将此行添加到 /etc/fstab 中，以便在系统启动期间自动挂载此文件系统：

```
# Device                Mountpoint      FStype  Options         Dump    Pass#
proc                    /proc           procfs  rw              0       0
```

Cinnamon 使用 dbus-daemon(1)[540] 作为消息总线和硬件抽象层。此软件是作为 Cinnamon 的依赖项自动安装的。在 /etc/rc.conf 中启用 D-BUS 以在系统启动时启动：

```
# sysrc dbus_enable="YES"
```

8.2.5.3 启动 Cinnamon

x11/lightdm[541] 是一款支持不同显示技术的显示管理器，是一个很好的选择，因为它非常轻，只需要很少的内存占用，并且性能卓越。

要安装它，请执行以下操作：

```
# pkg install lightdm lightdm-gtk-greeter
```

在 /etc/rc.conf 中启用 lightdm 以在系统启动时启动：

```
# sysrc lightdm_enable="YES"
```

第二种启动 Cinnamon 的方法是手动调用 startx(1)[542]。要使其工作，在 ~/.xinitrc 中需要添加以下行：

```
% echo "exec ck-launch-session cinnamon-session" > ~/.xinitrc
```

8.2.6 LXQT

LXQt 是一个基于 Qt 技术的高级、易用、快速的桌面环境。它是为那些重视简单、快速和直观界面的用户量身定制的。与大多数桌面环境不同，LXQt 在性能较差的机器上也能很好地工作。

[540] https://man.freebsd.org/cgi/man.cgi?query=dbus-daemon&sektion=1&format=html
[541] https://cgit.freebsd.org/ports/tree/x11/lightdm/pkg-descr
[542] https://man.freebsd.org/cgi/man.cgi?query=startx&sektion=1&format=html

8.2.6.1. 安装 LXQT

要安装 LXQT 元软件包，请执行以下操作：

```
# pkg install lxqt
```

8.2.6.2. 配置 LXQT

LXQT 需要安装 /proc。将此行添加到 /etc/fstab 中，以便在系统启动期间自动挂载此文件系统：

```
# Device                Mountpoint      FStype  Options         Dump    Pass#
proc                    /proc           procfs  rw              0       0
```

LXQT 使用 dbus-daemon(1)[543] 作为消息总线和硬件抽象层。此软件是作为 LXQT 的依赖项自动安装的。

在 /etc/rc.conf 中启用 D-BUS 以在系统启动时启动：

```
# sysrc dbus_enable="YES"
```

8.2.6.3. 启动 LXQT

首选的 LXQT 显示管理器是 x11/sddm[544]。要安装 x11/sddm[545]，请执行：

```
# pkg install sddm
```

在 /etc/rc.conf 中启用 SDDM 服务以在系统启动时启动：

```
# sysrc sddm_enable="YES"
```

通过运行以下命令（本示例为西班牙语），可以在 SDDM 中设置键盘语言：

```
# sysrc sddm_lang="es_ES"
```

第二种启动 LXQT 的方法是手动调用 startx(1)[546]。要使其工作，在 ~/.xinitrc 中需要添加以下行：

```
% echo "exec ck-launch-session startlxqt" > ~/.xinitrc
```

[543] https://man.freebsd.org/cgi/man.cgi?query=dbus-daemon&sektion=1&format=html
[544] https://cgit.freebsd.org/ports/tree/x11/sddm/pkg-descr
[545] https://cgit.freebsd.org/ports/tree/x11/sddm/pkg-descr
[546] https://man.freebsd.org/cgi/man.cgi?query=startx&sektion=1&format=html

8.3. 浏览器

本节介绍了如何在 FreeBSD 系统上安装和配置一些流行的网络浏览器，依次从资源消耗高的高级网络浏览器到资源使用减少的命令行网络浏览器。

表 2.受支持的浏览器

名称	许可证	软件包	资源消耗
Firefox	MPL 2.0	www/firefox[547]	大量
Chromium	BSD-3 等	www/chromium[548]	大量
Iridium browser	BSD-3 等	www/iridium[549]	大量
Falkon	MPL 2.0	www/falkon[550]	大量
Konqueror	GPL 2.0 或更高版本	x11-fm/konqueror[551]	中等
Gnome Web (Epiphany)	GPL 3.0 或更高版本	www/epiphany[552]	中等
qutebrowser	GPL 3.0 或更高版本	www/qutebrowser[553]	中等
Dillo	GPL 3.0 或更高版本	www/dillo2[554]	轻量
Links	GPL 2.0 或更高版本	www/links[555]	轻量
w3m	MIT	www/w3m[556]	轻量

8.3.1. Firefox

Firefox 是一个以标准化 HTML 渲染引擎、多标签页浏览，弹窗拦截、可选扩展、更高的安全性等著名的开源浏览器。Firefox 基于 Mozilla 的代码库。

若要安装最新版本的 Firefox，请执行：

```
# pkg install firefox
```

若要安装 Firefox 长期支持版（即 ESR），请执行：

```
# pkg install firefox-esr
```

[547] https://cgit.freebsd.org/ports/tree/www/firefox/pkg-descr
[548] https://cgit.freebsd.org/ports/tree/www/chromium/pkg-descr
[549] https://cgit.freebsd.org/ports/tree/www/iridium/pkg-descr
[550] https://cgit.freebsd.org/ports/tree/www/falkon/pkg-descr
[551] https://cgit.freebsd.org/ports/tree/x11-fm/konqueror/pkg-descr
[552] https://cgit.freebsd.org/ports/tree/www/epiphany/pkg-descr
[553] https://cgit.freebsd.org/ports/tree/www/qutebrowser/pkg-descr
[554] https://cgit.freebsd.org/ports/tree/www/dillo2/pkg-descr
[555] https://cgit.freebsd.org/ports/tree/www/links/pkg-descr
[556] https://cgit.freebsd.org/ports/tree/www/w3m/pkg-descr

8.3.2. Chromium

Chromium 是一个开源的浏览器项目，旨在提供更加安全、快速、稳定的网络浏览体验，同时具有标签式浏览、弹出窗口拦截、浏览器扩展等功能，谷歌的 Chrome 浏览器就基于该项目。

若要安装 Chromium，请执行：

```
# pkg install chromium
```

> 注意
> Chromium 的可执行文件是 **/usr/local/bin/chrome**，而非 **/usr/local/bin/chromium**。

8.3.3. Iridium browser

Iridium 是基于 Chromium 代码库的免费、开放和自由的修改版，在几个关键领域增强了隐私。禁止向中央服务自动传输部分查询、关键字和指标，只有在征得同意的情况下才会发送。

若要安装 Iridium browser，请执行：

```
# pkg install iridium
```

8.3.4. Falkon

Falkon 是一款新的、非常快速的 QtWebEngine 浏览器。其目标是成为一款可在所有主流平台上使用的轻量级网络浏览器。Falkon 具有您期望从网络浏览器获得的所有标准功能。它包括书签、历史记录（也在侧边栏中）和选项卡。除此之外，您还可以使用内置的 AdBlock 插件屏蔽广告，使用 Click2Flash 屏蔽 Flash 内容，并使用 SSL 管理器编辑本地 CA 证书数据库。

若要安装 Falkon，请执行：

```
# pkg install falkon
```

8.3.5. Konqueror

Konqueror 不仅是一个网页浏览器，它还是一个文件管理器和多媒体查看器。它支持 WebKit，包括 Chromium 在内的许多现代浏览器使用的渲染引擎，以及它自己的 KHTML 引擎。

若要安装 Falkon，请执行：

```
# pkg install konqueror
```

8.3.6. Gnome Web（Epiphany）

Gnome Web（Epiphany）是一款以牺牲其他浏览器中的许多功能为代价，设计尽可能轻量级和快速的网络浏览器。

若要安装 Gnome Web（Epiphany），请执行：

```
# pkg install epiphany
```

8.3.7. qutebrowser

Qutebrowser 是一款以键盘为中心的浏览器，具有最简的图形界面。它基于 Python 和 PyQt5 并且是使用 GPL 许可发布的自由软件。它的灵感来自其他浏览器/插件，如 dwb 和 Vierator/Pentdactyl。

若要安装 qutebrowser，请执行：

```
# pkg install qutebrowser
```

8.3.8. Dillo

Dillo 目的在于成为一款小型、稳定、对开发人员友好、可用、快速和可扩展的多平台替代浏览器。这个新的 Dillo 实验版本基于 FLTK 工具包，而非 GTK1 ，并且已经被大量重写。

若要安装 Dillo，请执行：

```
# pkg install dillo2
```

8.3.9. Links

一个类似 lynx 的 WWW 浏览器，具有文本和图形模式，具有许多功能，如显示表格、菜单等。

若要安装 Links，请执行：

```
# pkg install links
```

8.3.10. w3m

w3m 是一个基于分页器/文本的网络浏览器。它是一个类似于 Lynx 的应用程序，但它有几个 Lynx 没有的功能，比如渲染表格和渲染帧。

若要安装 w3m，请执行：

```
# pkg install w3m
```

8.4. 开发工具

本节介绍如何在 FreeBSD 系统上安装和配置一些流行的开发工具。

表 3.受支持的开发工具

名称	许可证	软件包	资源消耗
Visual Studio Code	MIT	editors/vscode[557]	大量
Qt Creator	QtGPL	devel/qtcreator[558]	大量
Kdevelop	GPL 2.0 或更高版本和 LGPL 2.0 或更高版本	devel/kdevelop[559]	大量
Eclipse IDE	EPL	java/eclipse[560]	大量
Vim	VIM	editors/vim[561]	轻量
Neovim	Apache 2.0	editors/neovim[562]	轻量
GNU Emacs	GPL 3.0 或更高版本	editors/emacs[563]	轻量

8.4.1. Visual Studio Code

Visual Studio Code 是一款工具，它简单易用，同时又满足了开发人员进行代码编辑、构建和调试等核心操作的需求。它提供了全面的编辑和调试支持、可扩展性模型以及与现有工具的轻量级集成。

若要安装 Visual Studio Code，请执行：

```
# pkg install vscode
```

[557] https://cgit.freebsd.org/ports/tree/editors/vscode/pkg-descr
[558] https://cgit.freebsd.org/ports/tree/devel/qtcreator/pkg-descr
[559] https://cgit.freebsd.org/ports/tree/devel/kdevelop/pkg-descr
[560] https://cgit.freebsd.org/ports/tree/java/eclipse/pkg-descr
[561] https://cgit.freebsd.org/ports/tree/editors/vim/pkg-descr
[562] https://cgit.freebsd.org/ports/tree/editors/neovim/pkg-descr
[563] https://cgit.freebsd.org/ports/tree/editors/emacs/pkg-descr

8.4.2. Qt Creator

Qt Creator 是一个跨平台的集成开发环境（IDE），可满足 Qt 开发人员的需求。Qt Creator 的功能包括：

- 支持 C++、QML 和 ECMAscript 的代码编辑器；
- 快速代码导航工具；
- 键入时进行静态代码检查和格式提示；
- 上下文相关帮助；
- 可视化调试器；
- 集成的图形界面布局和表单设计器。

若要安装 Qt Creator，请执行：

```
# pkg install qtcreator
```

8.4.3. kdevelop

用于 C/C++ 和其他编程语言的开源、功能完整、插件可扩展的 IDE。它基于 KDevPlatform 以及 KDE 和 Qt 库，自 1998 年以来一直在开发中。

若要安装 kdevelop，请执行：

```
# pkg install kdevelop
```

8.4.4. Eclipse IDE

Eclipse 平台是一个开放、可扩展的 IDE，可以用于各种应用场景。它提供了构建和运行集成软件开发工具所需的基础和组件，同时也允许开发人员独立开发工具并与其他人的工具进行集成。简单来说，Eclipse 平台为软件开发提供了一个灵活而强大的基础设施。

若要安装 Eclipse IDE，请执行：

```
# pkg install eclipse
```

8.4.5. Vim

Vim 是一个高度可配置的文本编辑器，旨在实现高效的文本编辑。它是与大多数 UNIX 系统一起分发的 vi 编辑器的改进版本。

Vim 通常被称为“程序员的编辑器”，对编程非常有用，以至于许多人认为它是一个完整的 IDE。不过，这不仅仅是针对程序员来说。Vim 非常适合各种文本编辑——从撰写电子邮件到编辑配置文件。

若要安装 Vim，请执行：

```
# pkg install vim
```

8.4.6. Neovim

Neovim 是激进的，基于editors/vim[564] 重构。它代表了对代码库的彻底审查，并进行了许多健全性改进，包括合理的默认值、内置的终端模拟器、异步插件架构以及为速度和可扩展性而设计的强大 API。它保留了与几乎所有 Vim 插件和脚本的完全兼容性。

若要安装 Neovim，请执行：

```
# pkg install neovim
```

8.4.7. GNU Emacs

GNU Emacs 是一个可扩展、可定制、免费/自由的文本编辑器。它的核心是 Emacs Lisp 的解释器，其为 Lisp 编程语言的一种方言，具有支持文本编辑的扩展。

若要安装 GNU Emacs，请执行：

```
# pkg install emacs
```

8.5. 桌面生产力工具

当提到生产力工具时，用户往往倾向于使用办公套件或者易于使用的文字处理软件。虽然有些桌面环境（比如 KDE Plasma[565]）提供了一个默认的办公套件，但是 FreeBSD 并没有提供默认的生产力工具套件，你需要自己安装想要的办公套件和其他图形化的文字处理程序。FreeBSD 提供的生产力工具与你安装的桌面环境无关。

这一节中我们将演示如何安装下列流行的生产力软件，并指出了该应用程序对资源的消耗程度，从 ports 编译所需要的时间以及主要的依赖关系。

[564] https://cgit.freebsd.org/ports/tree/editors/vim/pkg-descr

[565] https://github.com/Chinese-FreeBSD-Community/FreeBSD-En-Handbook/blob/main/desktop/_index.adoc#kde-environment

表 4.受支持的桌面生产力工具

名称	许可证	安装包	资源消耗
LibreOffice	MPL 2.0	editors/libreoffice[566]	大量
Calligra Suite	LGPL 和 GPL	editors/calligra[567]	中等
AbiWord	GPL 2.0 或更高版本	editors/abiword[568]	中等

8.5.1. LibreOffice

LibreOffice 是一个由文档基金会[569]开发的免费办公软件套件。它与其他的主流办公套件相兼容，支持多种平台。它是 Apache OpenOffice 的一个重命名复刻，包含一个完整的办公生产力套件中所应含有的应用程序：文字处理器、电子表格、幻灯片管理器绘图程序、数据库管理程序，以及用于创建和编辑数学公式的工具。它有许多不同的语言版本，并且支持界面、拼写检查器和字典等的本地化。关于 LibreOffice 的更多信息，可以参考 libreoffice.org[570]。

若要安装 LibreOffice，请执行：

```
# pkg install libreoffice
```

LibreOffice 软件包默认情况下只有英文版。要获得 LibreOffice 的本地化版本，必须安装语言包。例如，对于西班牙语本地化的版本，需要使用以下命令安装软件包 edits/libreoffice es[571]：

```
# pkg install libreoffice-es
```

8.5.2. Calligra

KDE Plasma 桌面环境包含了一个可以被单独安装的办公套件，即 Calligra。Calligra 包含了在其他办公套件中可以找到的标准组件，其中，Words 是文字处理器，Sheets 是电子表格程序，Stage 是幻灯片演示程序，Karbon 可用于绘制图形文档。

若要安装 Calligra，请执行：

```
# pkg install calligra
```

[566] https://cgit.freebsd.org/ports/tree/editors/libreoffice/pkg-descr
[567] https://cgit.freebsd.org/ports/tree/editors/calligra/pkg-descr
[568] https://cgit.freebsd.org/ports/tree/editors/abiword/pkg-descr
[569] http://www.documentfoundation.org/
[570] http://www.libreoffice.org/
[571] https://cgit.freebsd.org/ports/tree/editors/libreoffice-es/pkg-descr

8.5.3. AbiWord

AbiWord 是一个免费的文字处理程序，在观感上与 Microsoft® Word 相似。它速度很快，包含许多功能，而且对用户很友好。

AbiWord 可以导入或导出许多文件格式，包括一些专有格式，如 Microsoft® 的 **.rtf**。

若要安装 AbiWord，请执行：

```
# pkg install abiword
```

8.6. 文档阅读器

自 UNIX® 问世以来，有一些新的文件格式得到了普及，它们可能无法直接使用基本系统的组件来查看。本节将演示如何安装下列文档查看器：

表 5.受支持的文档阅读器

名称	许可证	软件包	资源消耗
Okular	GPL 2.0	graphics/okular[572]	大量
Evince	GPL 2.0	graphics/evince[573]	中等
ePDFView	GPL 2.0	graphics/epdfview[574]	中等
Xpdf	GPL 2.0	graphics/xpdf[575]	轻量

8.6.1. Okular

Okular 是一个通用的文档查看器，是 KDE Plasma 项目的一部分。

Okular 将出色的功能与支持不同类型文档的多功能性相结合，如 PDF、Postscript、DjVu、CHM、XPS、ePub 等。

要安装 Okular，请执行：

```
# pkg install okular
```

[572] https://cgit.freebsd.org/ports/tree/graphics/okular/pkg-descr
[573] https://cgit.freebsd.org/ports/tree/graphics/evince/pkg-descr
[574] https://cgit.freebsd.org/ports/tree/graphics/epdfview/pkg-descr
[575] https://cgit.freebsd.org/ports/tree/graphics/xpdf/pkg-descr

8.6.2. Evince

Evince 是一个用于多种文档格式的文档查看器，包括 PDF 和 Postscript。GNOME 项目的一部分。evince 的目标是用一个简单的软件取代 ggv 和 gpdf 等多个文档查看器。

要安装 Evince，请执行：

```
# pkg install evince
```

8.6.3. ePDFView

ePDFView 是一个轻量级的 PDF 文档查看器，只使用 Gtk+ 和 Poppler 库。ePDFView 的目的是制作一个简单的 PDF 文档查看器，类似于 Evence，但不使用 GNOME 库。

要安装 ePDFView，请执行：

```
# pkg install epdfview
```

8.6.4. Xpdf

对于那些更喜欢小型 FreeBSD PDF 查看器的用户，Xpdf 提供了一个轻量级和高效率的查看器。Xpdf 使用标准的 X 字体，且不需要任何额外工具。

要安装 Xpdf，请执行：

```
# pkg install xpdf
```

8.7. 财务

要在 FreeBSD 上管理个人财务，可以安装一些强大的、易于使用的应用。其中一些可与常用文件格式相兼容，比如 Quicken 和 Excel 所使用的格式。

这一节包含了这些程序：

表 6.受支持的财务软件

名称	许可证	软件包	资源消耗
KMyMoney	GPL 2.0	finance/kmymoney[576]	大量
GnuCash	GPL 2.0 和 GPL 3.0	finance/gnucash[577]	大量

8.7.1. KMyMoney

KMyMoney 是一个由 KDE 社区创建的个人财务应用程序，其目标是提供商业版个人财务管理中的主要功能。它突出了易用性和方便的复式记账功能。KMyMoney 可以由标准的 Quicken QIF 文件导入，跟踪投资，处理多种货币，并提供丰富的报告。

要安装 KMyMoney，请执行：

```
# pkg install kmymoney
```

8.7.2. GnuCash

GnuCash 是 GNOME 努力提供的用户友好且功能强大的软件的一部分，能用来记录收入、支出、银行账户和股票，在具有直观的界面保持了其专业性。

GnuCash 提供了智能的存款计算器、分层的账户系统，以及许多键盘快捷键和自动补全的方式。它可以将一笔交易细化为多个部分，还能导入和合并 Quicken QIF 文件，且能够处理大多数国际日期和货币格式。

要安装 GnuCash，请执行：

```
# pkg install gnucash
```

[576] https://cgit.freebsd.org/ports/tree/finance/kmymoney/pkg-descr

[577] https://cgit.freebsd.org/ports/tree/finance/gnucash/pkg-descr

第九章多媒体

9.1. 概述

多媒体章节整体介绍了 FreeBSD 上的多媒体支持情况。多媒体软件和技术已经成为现代计算机的一个组成部分，FreeBSD 为大部分多媒体软硬件提供了强大而可靠的支持。本章涵盖了各种多媒体组件，如音频、视频和图像处理。它还讨论了各种媒体格式和编解码器，以及用于多媒体创作和播放的工具和软件。此外，本章还包括多媒体系统配置、故障排除和优化。无论您是多媒体爱好者还是专业内容创作者，FreeBSD 都能为你的多媒体工作提供一个强大的平台。本章旨在帮助你充分利用 FreeBSD 的多媒体功能，提供有用的信息和实例来帮助你入门。

9.2. 设置声卡

在默认情况下，FreeBSD 会自动检测系统使用的声卡。FreeBSD 支持各种各样的声卡。可以在硬件兼容列表[578]中查看受支持的音频设备列表。

> 注意
>
> 只有在 FreeBSD 没有正确检测到声卡模块的情况下才有必要加载之。

如果不知道系统有什么声卡，也不知道该使用哪个模块，可以通过执行以下命令加载 `snd_driver` 元驱动：

```
# kldload snd_driver
```

要在启动时自动加载这个驱动程序，请将下面一行添加到 **/boot/loader.conf** 中：

```
snd_driver_load="YES"
```

[578] https://www.freebsd.org/releases/12.0R/hardware/

9.2.1. 测试声卡

运行下列命令来侦测显卡：

```
% dmesg | grep pcm
```

输出结果看起来应该像这样：

```
pcm0: <Conexant CX20561 (Hermosa) (Analog 2.0+HP/2.0)> at nid 26,22 and 24 on hdaa0
pcm1: <Conexant CX20561 (Hermosa) (Internal Analog Mic)> at nid 29 on hdaa0
```

也可以用这个命令检查声卡的状态：

```
# cat /dev/sndstat
```

输出结果看起来应该像这样：

```
Installed devices:
pcm0: <Conexant CX20561 (Hermosa) (Analog 2.0+HP/2.0)> (play/rec) default
pcm1: <Conexant CX20561 (Hermosa) (Internal Analog Mic)> (rec)
```

如果没有 pcm 设备被列出，请仔细检查是否有正确的设备驱动程序被载入或编译到内核里。如果一切顺利，现在在 FreeBSD 中声卡应该可以工作了。

beep(1)[579] 可通过产生一些噪音来确认声卡是否正常工作：

```
% beep
```

9.2.2. 混音器

FreeBSD 系统中内置了不同的工具来设置和展示声卡混音器的属性。

表 15. 支持的混音器软件包

名称	许可证	软件包	工具包
mixer(8)[580]	BSD-2	系统自带	命令行界面
dsbmixer	BSD-2	audio/dsbmixer[581]	Qt
KDE Plasma 声音组件	GPL 2.0	audio/plasma5-plasma-pa[582]	Qt
mixertui	BSD-2	audio/mixertui[583]	文本用户界面

[579] https://man.freebsd.org/cgi/man.cgi?query=beep&sektion=1&format=html

9.2.3. 显卡声音

显卡通常有自己的集成声音设备，可能不清楚哪个设备被用作了默认设备。为了确认，运行 dmesg，寻找 pcm 这些条目，以确定系统是如何列举输出的。执行下面的命令：

```
% dmesg | grep pcm
```

输出结果看起来像这样：

```
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
```

在这个例子中，NVIDIA® 显卡（集成的声卡）的排序先于 Realtek® 声卡，声卡被标识为 `pcm4`。执行下列命令让 Realtek® 声卡成为默认播放设备：

```
# sysctl hw.snd.default_unit=4
```

在 **/etc/sysctl.conf** 中加入以下一行，使这个改变永久化：

```
hw.snd.default_unit=4
```

9.2.4. 自动切换到耳机

有些系统可能会在音频输出之间的切换上遇到困难，但幸运的是 FreeBSD 允许在 **device.hints** 中配置自动切换。

通过执行以下命令，查看系统是如何列举音频输出的：

```
% dmesg | grep pcm
```

输出结果看起来像这样：

[580] https://man.freebsd.org/cgi/man.cgi?query=mixer&sektion=8&format=html
[581] https://cgit.freebsd.org/ports/tree/audio/dsbmixer/
[582] https://cgit.freebsd.org/ports/tree/audio/plasma5-plasma-pa/
[583] https://cgit.freebsd.org/ports/tree/audio/mixertui/

```
pcm0: <Realtek ALC892 Analog> at nid 23 and 26 on hdaa0
pcm1: <Realtek ALC892 Right Analog Headphones> at nid 22 on hdaa0
```

在 **/boot/device.hints** 中添加下列行：

```
hint.hdac.0.cad0.nid22.config="as=1 seq=15 device=Headphones"
hint.hdac.0.cad0.nid26.config="as=2 seq=0 device=speakers"
```

> 注意
> 请记住，这些值是针对上面的例子。它们依据具体的系统而定。

9.2.5. 声卡的故障排除

一些常见的错误信息和它们的解决方案：

表 16. 常见错误信息

错误	解决方案
`xxx: can't open /dev/dsp!`	输入 `fstat | grep dsp` 来检查是否有其他的应用程序在打开设备。值得注意的是 esound 和 KDE 的声音支持常常会出现问题。

要让 `hw.snd.default_unit` 中的变更生效，使用 audio/pulseaudio[584] 的程序可能需要重启守护程序audio/pulseaudio[585]。另一种方式是实时更改 audio/pulseaudio[586] 设置——用 pacmd(1)[587] 定位到守护程序 audio/pulseaudio[588]，打开命令行：

```
# pacmd
Welcome to PulseAudio 14.2! Use "help" for usage information.
>>>
```

下面的命令将默认的 sink 改为前面例子中的声卡编号 4：

```
set-default-sink 4
```

> 警告
> 不要使用 `exit` 命令来退出命令行界面。那会杀死 audio/pulseaudio[589] 守护进程。应该用 `Ctrl+D` 代替之。

[584] https://cgit.freebsd.org/ports/tree/audio/pulseaudio/
[585] https://cgit.freebsd.org/ports/tree/audio/pulseaudio/
[586] https://cgit.freebsd.org/ports/tree/audio/pulseaudio/
[587] https://man.freebsd.org/cgi/man.cgi?query=pacmd&sektion=1&format=html
[588] https://cgit.freebsd.org/ports/tree/audio/pulseaudio/
[589] https://cgit.freebsd.org/ports/tree/audio/pulseaudio/

9.3. 音频播放器

这一节介绍了一些可用于音频播放的 FreeBSD ports 中的软件。

表 17.音频播放器软件包

名称	许可证	软件包	工具包
Elisa	LGPL 3.0	audio/elisa[590]	Qt
GNOME Music	GPL 2.0	audio/gnome-music[591]	GTK+
Audacious	BSD-2	multimedia/audacious[592]	Qt
MOC（music on console）	GPL 2.0	audio/moc[593]	文本用户界面

9.3.1. Elisa

Elisa 是一个由 KDE 社区开发的音乐播放器，它努力做到简单而好用。

运行命令安装 Elisa：

```
# pkg install elisa
```

9.3.2. GNOME Music

GNOME Music 是一个新的 GNOME 音乐播放器。它的目标是将优雅和沉浸式的浏览体验与简单明了的控制相结合。

运行命令安装 GNOME Music：

```
# pkg install gnome-music
```

[590] https://cgit.freebsd.org/ports/tree/audio/elisa/

[591] https://cgit.freebsd.org/ports/tree/audio/gnome-music/

[592] https://cgit.freebsd.org/ports/tree/multimedia/audacious/

[593] https://cgit.freebsd.org/ports/tree/audio/moc/

9.3.3. Audacious

Audacious 是一个开源的音频播放器。作为 XMMS 的后裔，它按照你的要求播放音乐，而不会和其他任务争夺计算机资源。

运行命令安装 Audacious：

```
# pkg install audacious-qt6 audacious-plugins-qt6
```

> 注意
> Audacious 原生支持 OSS，但必须在音频标签的设置中进行配置。

9.3.4. MOC（music on console）

MOC（music on console）是一个功能强大且易于使用的控制台音频播放器。

无论系统或 I/O 负载如何，MOC 都能顺利播放，因为它在一个单独的线程中处理输出缓冲区。它不会造成文件之间的空隙，因为在播放当前文件时，要播放的下一个文件是预先缓存的。

运行命令安装 MOC（music on console）：

```
# pkg install moc
```

9.4. 视频播放器

这一节介绍了一些在 FreeBSD ports 中可用于视频播放的软件。

表 18. 视频播放器软件包

名称	许可证	软件包	工具包
MPlayer	GPL 2.0	multimedia/mplayer[594]	命令行界面
SMPlayer	GPL 2.0	multimedia/smplayer[595]	Qt
VLC media player	GPL 2.0	multimedia/vlc[596]	Qt
Kodi (XBMC)	GPL 2.0	multimedia/kodi[597]	X11

[594] https://cgit.freebsd.org/ports/tree/multimedia/mplayer/
[595] https://cgit.freebsd.org/ports/tree/multimedia/smplayer/
[596] https://cgit.freebsd.org/ports/tree/multimedia/vlc/
[597] https://cgit.freebsd.org/ports/tree/multimedia/kodi/

9.4.1. MPlayer

MPlayer 是一个多媒体播放器和编码器套件，可在许多平台上运行，并在命令行上工作。它支持众多不同的文件格式和编解码器，包括流行的 DivX、XviD、H.264 流以及 DVD 和 SVCD，还有许多流行的音频编解码器。

运行命令安装 MPlayer：

```
# pkg install mplayer
```

在 mplayer(1)[598] 中查看 MPlayer 的工作示例。

9.4.2. SMPlayer

SMPlayer 打算成为 MPlayer 的一个完整的前端，从播放视频、DVD 和 VCD 等基本功能到支持 MPlayer 过滤器等更高级的功能。

运行命令安装 SMPlayer：

```
# pkg install smplayer
```

9.4.3. VLC media player

VLC media player 是一个高度便携的多媒体播放器，可播放各种音频和视频格式（MPEG-1、MPEG-2、MPEG-4、DivX、mp3、ogg 等）以及 DVD、VCD 和各种流媒体协议。它还可以用作服务器，在高带宽网络上以 IPv4 或 IPv6 的单播或多播方式进行流媒体传输。VLC 还能对媒体进行实时转码，以便进行流媒体播放或保存到磁盘。

运行命令安装 VLC：

```
# pkg install vlc
```

9.4.4. Kodi (XBMC)

Kodi（以前称为 XBMC）是一个免费和开源的跨平台媒体播放器和娱乐中心。它允许用户播放和查看大多数视频、音乐、播客和其他来自本地和网络存储媒体及互联网的数字媒体文件。

运行命令安装 Kodi：

```
# pkg install kodi
```

[598] https://man.freebsd.org/cgi/man.cgi?query=mplayer&sektion=1&format=html

9.5. 视频会议

可以用 FreeBSD 的桌面环境来加入视频会议。本节将说明如何配置网络摄像头以及 FreeBSD 支持哪些视频会议软件。

9.5.1. 设置网络摄像头

为了使 FreeBSD 能够访问网络摄像头并进行配置，有必要安装一些软件：

- multimedia/webcamd[599] 是一个守护程序，能够使用数百个不同的基于 USB 的网络摄像头和 DVB USB 设备。
- multimedia/pwcview[600] 是一个可以用来查看网络摄像头视频流的软件。

运行命令安装需要的软件：

```
# pkg install webcamd pwcview
```

在 `/etc/rc.conf` 中启用 webcamd(8)[601] 服务以在系统启动时启动它：

```
# sysrc webcamd_enable="YES"
```

该用户必须属于 `webcamd` 组。要将该用户添加到 `webcamd` 组，请执行以下命令：

```
# pw groupmod webcamd -m username
```

由于 multimedia/webcamd[602] 需要 cuse(3)[603] 模块，必须通过执行以下命令来加载该模块：

```
# kldload cuse
```

要在系统启动时加载 cuse(3)[604]，执行命令：

```
# sysrc kld_list += "cuse"
```

在安装了这些软件后，就可以用 webcamd(8)[605] 来显示可用的网络摄像机的列表：

[599] https://cgit.freebsd.org/ports/tree/multimedia/webcamd/
[600] https://cgit.freebsd.org/ports/tree/multimedia/pwcview/
[601] https://man.freebsd.org/cgi/man.cgi?query=webcamd&sektion=8&format=html
[602] https://cgit.freebsd.org/ports/tree/multimedia/webcamd/
[603] https://man.freebsd.org/cgi/man.cgi?query=cuse&sektion=3&format=html
[604] https://man.freebsd.org/cgi/man.cgi?query=cuse&sektion=3&format=html
[605] https://man.freebsd.org/cgi/man.cgi?query=webcamd&sektion=8&format=html

```
# webcamd -l
```

输出结果看起来像这样：

```
webcamd [-d ugen0.2] -N SunplusIT-Inc-HP-TrueVision-HD-Camera -S unknown -M 0  ①
webcamd [-d ugen1.3] -N Realtek-802-11n-WLAN-Adapter -S 00e04c000001 -M 0
```

① 可用的 webcam

要配置可用的网络摄像头，执行以下命令：

```
# sysrc webcamd_0_flags="-d ugen0.2"
```

> 注意
>
> 请注意，如果这是一个即插即用的 USB 网络摄像头，改变它所连接的 USB 端口将改变 `webcamd -l` 的输出，并且可能需要更新 rc.conf 中的条目。对于使用 USB 集成网络摄像头的笔记本电脑，这不应该是一个问题。

必须通过执行以下命令启动 webcamd(8)[606] 服务：

```
# service webcamd start
```

输出结果看起来应该像这样：

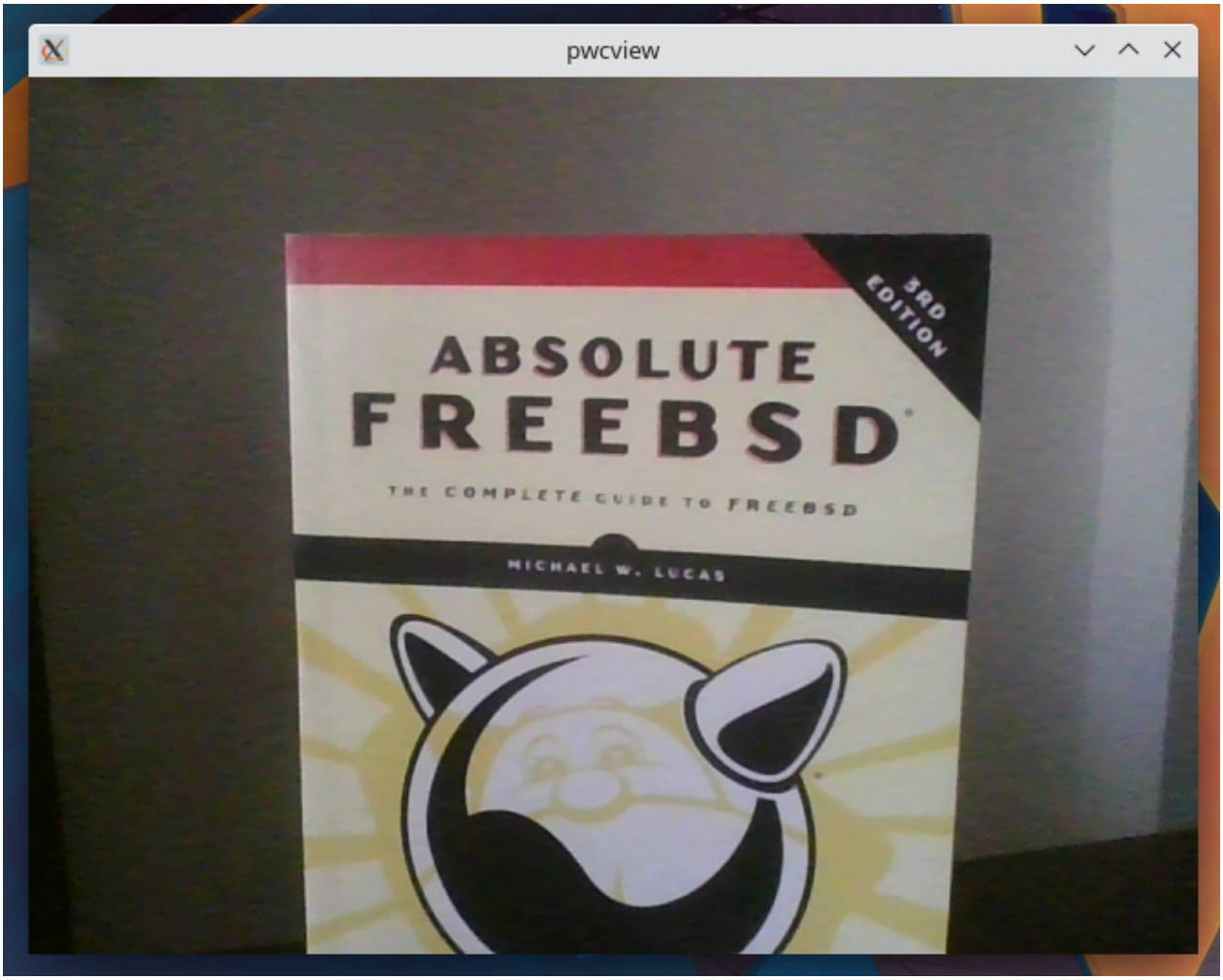
```
Starting webcamd.
webcamd 1616 - - Attached to ugen0.2[0]
```

multimedia/pwcview[607] 可以用来检查网络摄像头的正常功能。下面的命令可以用来执行 multimedia/pwcview[608]：

```
% pwcview -f 30 -s vga
```

然后 multimedia/pwcview[609] 将显示网络摄像头：

[606] https://man.freebsd.org/cgi/man.cgi?query=webcamd&sektion=8&format=html
[607] https://cgit.freebsd.org/ports/tree/multimedia/pwcview/
[608] https://cgit.freebsd.org/ports/tree/multimedia/pwcview/
[609] https://cgit.freebsd.org/ports/tree/multimedia/pwcview/



9.5.2. 会议软件状态

FreeBSD 目前支持以下用于进行视频会议的工具。

表 19. 会议软件

名称	Firefox 状态	Chromium 状态	网址
Microsoft Teams	不支持	支持	https://teams.live.com[610]
Google Meet	不支持	支持	https://meet.google.com/
Zoom	支持	支持	https://zoom.us[611]
Jitsi	不支持	支持	https://meet.jit.si/
BigBlueButton	不支持	支持	https://bigbluebutton.org/

[610] https://teams.live.com/

[611] https://zoom.us/

9.6. 图像扫描仪

在 FreeBSD 中，对图像扫描仪的访问是由 SANE（Scanner Access Now Easy，简易扫描仪访问）[612]提供的，可以在 FreeBSD ports 中找到它。

9.6.1. 检查扫描仪

在尝试任何配置之前，重要的是检查 SANE 是否支持扫描仪。

在连接了扫描仪的情况下，运行以下命令以获得所有连接的 USB 设备：

```
# usbconfig list
```

输出结果看起来像这样：

```
ugen4.2: <LITE-ON Technology USB NetVista Full Width Keyboard.> at usbus4, cfg=0␣
↪md=HOST spd=LOW (1.5Mbps) pwr=ON (70mA)
ugen4.3: <Logitech USB Optical Mouse> at usbus4, cfg=0 md=HOST spd=LOW (1.5Mbps)␣
↪pwr=ON (100mA)
ugen3.2: <HP Deskjet 1050 J410 series> at usbus3, cfg=0 md=HOST spd=HIGH (480Mbps)␣
↪pwr=ON (2mA)
```

运行以下命令以获得 `idVendor` 和 `idProduct`：

```
# usbconfig -d 3.2 dump_device_desc
```

> 注意
>
> 请注意，扫描仪是一个即插即用的设备，改变它所连接的 USB 端口将改变 `usbconfig list` 的输出。

输出结果看起来像这样：

```
ugen3.2: <HP Deskjet 1050 J410 series> at usbus3, cfg=0 md=HOST spd=HIGH (480Mbps)␣
↪pwr=ON (2mA)

bLength = 0x0012
bDescriptorType = 0x0001
bcdUSB = 0x0200
bDeviceClass = 0x0000  <Probed by interface class>
bDeviceSubClass = 0x0000
bDeviceProtocol = 0x0000
bMaxPacketSize0 = 0x0040
```

(continues on next page)

[612] http://www.sane-project.org/

(continued from previous page)

```
idVendor = 0x03f0
idProduct = 0x8911
bcdDevice = 0x0100
iManufacturer = 0x0001  <HP>
iProduct = 0x0002  <Deskjet 1050 J410 series>
bNumConfigurations = 0x0001
```

一旦获得了 `idVendor` 和 `idProduct`，就有必要在 SANE 的支持设备列表[613]中通过过滤 idProduct 来确认是否支持你的扫描仪。

9.6.2. SANE 配置

SANE 通过后端提供对扫描仪的访问。为了能够用 FreeBSD 进行扫描，必须通过运行以下命令来安装 graphics/sane-backends[614] 软件包：

```
# pkg install sane-backends
```

> 提示
>
> 一些 USB 扫描器需要加载固件。像上面的例子中使用的惠普扫描仪，需要安装软件包 print/hplip[615] 。

在安装了必要的软件包之后，必须配置 devd(8)[616] 允许 FreeBSD 访问扫描仪。

将 `saned.conf` 文件添加到 **/usr/local/etc/devd/saned.conf** 中，内容如下：

```
notify 100 {
        match "system" "USB";
        match "subsystem" "INTERFACE";
        match "type" "ATTACH";
        match "cdev" "ugen[0-9].[0-9]";
        match "vendor" "0x03f0";
        match "product" "0x8911";
        action "chown -L cups:saned /dev/\$cdev && chmod -L 660 /dev/\$cdev";
};
```

`vendor`：是之前通过运行 `usbconfig -d 3.2 dump_device_desc` 命令得到的 idVendor。

`product`：是之前通过运行 `usbconfig -d 3.2 dump_device_desc` 命令获得的 idProduct。

之后，必须通过运行下面的命令重新启动 devd(8)[617]：

[613] http://www.sane-project.org/lists/sane-mfgs-cvs.html
[614] https://cgit.freebsd.org/ports/tree/graphics/sane-backends/
[615] https://cgit.freebsd.org/ports/tree/print/hplip/
[616] https://man.freebsd.org/cgi/man.cgi?query=devd&sektion=8&format=html
[617] https://man.freebsd.org/cgi/man.cgi?query=devd&sektion=8&format=html

```
# service devd restart
```

SANE 的后端包括 scanimage(1)[618]，可以用来列出设备并执行图像采集。

用 -L 参数执行 scanimage(1)[619]，以列出扫描仪设备：

```
# scanimage -L
```

输出结果看起来像这样：

```
device `hpaio:/usb/Deskjet_1050_J410_series?serial=XXXXXXXXXXXXXX' is a Hewlett-
↪Packard Deskjet_1050_J410_series all-in-one
```

如果 scanimage(1)[620] 不能识别扫描仪，就会出现这个信息：

```
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).
```

在 scanimage(1)[621] 识别到扫描仪后，配置就完成了，扫描仪就可以使用了。

要激活该服务并在启动时运行，请执行以下命令：

```
# sysrc saned_enable="YES"
```

虽然 scanimage(1)[622] 可以用来从命令行中进行图像采集，但通常更倾向于使用图形界面来进行图像扫描。

表 20. 图形化扫描程序

名称	许可证	软件包
skanlite	GPL 2.0	graphics/skanlite
GNOME Simple Scan	GPL 3.0	graphics/simple-scan
XSANE	GPL 2.0	graphics/xsane

[618] https://man.freebsd.org/cgi/man.cgi?query=scanimage&sektion=1&format=html
[619] https://man.freebsd.org/cgi/man.cgi?query=scanimage&sektion=1&format=html
[620] https://man.freebsd.org/cgi/man.cgi?query=scanimage&sektion=1&format=html
[621] https://man.freebsd.org/cgi/man.cgi?query=scanimage&sektion=1&format=html
[622] https://man.freebsd.org/cgi/man.cgi?query=scanimage&sektion=1&format=html

第十章配置 FreeBSD 内核

10.1.概述

内核是 FreeBSD 操作系统的核心。它负责管理内存、执行安全控制、网络、磁盘访问以及更多的工作。虽然 FreeBSD 的大部分内容都是可以动态配置的，但偶尔还是需要配置和编译一个定制内核。

读完本章后，你将会了解：

- 该在什么时候编译定制内核。
- 如何查看硬件列表。
- 如何调整内核配置文件。
- 如何使用内核配置文件来创建和编译一个新的内核。
- 如何安装新的内核。
- 如果出现问题，如何进行故障排除。

所有在本章示例中列出的命令均应以 `root` 执行。

10.2.为什么要编译定制内核

传统上，FreeBSD 使用的是一个宏内核。内核是一个很大的程序，有一个固定的设备支持列表，如果要改变内核的行为，必须先编译再重新启动进入新的内核。

在今天，FreeBSD 内核中的大部分功能都包含在模块中，必要时可以动态地从内核中加载和卸载模块。这使得运行中的内核能够即时适应新的硬件，并将新的功能带入内核。这就是所谓的模块化内核。

偶尔仍有必要进行静态的内核配置。有时，所需的功能与内核紧密结合，以至于无法动态加载。一些安全环境会阻止内核模块的加载和卸载，并要求只将需要的功能静态地编译到内核中。

对于高级 BSD 用户来说，编译一个定制内核通常是一种仪式。这个过程虽然很耗时，但可以为 FreeBSD 系统带来好处。与必须支持各种硬件的 **GENERIC** 内核不同，定制的内核可以被调整为只为该计算机的硬件提供支持。

这有很多好处，例如：

- 更快的启动时间。由于内核将只探测系统使用的硬件，可以减少系统启动的时间。
- 更少的内存占用。通过省略不使用的功能和设备驱动，定制内核通常比 **GENERIC** 内核使用更少的内存。这一点很重要，因为内核代码在任何时候都驻留在物理内存中，这块空间应用程序无法使用。因此，定制的内核适合运行在拥有少量内存的系统上。
- 额外的硬件支持。定制内核可以增加对 **GENERIC** 内核中没有的设备的支持。

  注意
  在构建定制内核之前，先考虑这样做的原因。如果需要特定的硬件支持，它可能已经作为一个模块存在。

内核模块存在于 **/boot/kernel** 中，并可以使用 kldload(8)[623] 将其动态加载到正在运行的内核中。大多数内核驱动都有一个可加载模块和手册页。

例如，无线以太网驱动 ath(4)[624] 在其手册页中有如下信息：

```
Alternatively, to load the driver as a module at boot time, place the
following line in loader.conf(5):

      if_ath_load="YES"
```

在 **/boot/loader.conf** 中添加 `if_ath_load="YES"` 和 `if_ath_pci_load="YES"` 可在启动时动态加载该模块。

在某些情况下，**/boot/kernel** 没有相关的模块。这主要是针对某些子系统而言的。

10.3.浏览系统硬件

在编辑内核配置文件之前，建议对机器的硬件进行一次清点。在双系统中，可以从另一个操作系统中创建清单。例如，Microsoft Windows 的设备管理器包含了关于已安装设备的信息。

如果 FreeBSD 是唯一被安装的操作系统，使用 dmesg(8)[625] 来确定在启动探测过程中发现并列出的硬件。FreeBSD 上的大多数设备驱动程序都有一个手册页，列出了该驱动程序所支持的硬件。例如，下面几行表示驱动 psm(4)[626] 找到了一个鼠标：

[623] https://www.freebsd.org/cgi/man.cgi?query=kldload&sektion=8&format=html
[624] https://www.freebsd.org/cgi/man.cgi?query=ath&sektion=4&format=html
[625] https://www.freebsd.org/cgi/man.cgi?query=dmesg&sektion=8&format=html
[626] https://www.freebsd.org/cgi/man.cgi?query=psm&sektion=4&format=html

```
psm0: <PS/2 Mouse> irq 12 on atkbdc0
psm0: [GIANT-LOCKED]
psm0: [ITHREAD]
psm0: model Generic PS/2 mouse, device ID 0
```

由于这个硬件的存在，不应该从定制内核配置文件中删除这个驱动。

如果 **dmesg** 的输出没有显示启动探针输出的结果，请改为读取 **/var/run/dmesg.boot** 的内容。

另一个查看硬件的工具是 pciconf(8)[627]，它提供了更多粗略的输出。例如:

```
% pciconf -lv
```

输出应该和下面类似:

```
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
```

这个输出表示驱动 **ath** 找到了一个无线以太网设备:

man(1)[628] 的 `-k` 参数可以用来提供有用的信息。例如，它可以用来显示包含一个特定设备品牌或名称的手册页面的列表:

```
# man -k Atheros
```

输出应该和下面类似:

```
ath(4)                   - Atheros IEEE 802.11 wireless network driver
ath_hal(4)               - Atheros Hardware Access Layer (HAL)
```

在创建硬件清单之后，需要参考它来确保在编辑定制内核配置时，不会删除已安装硬件的驱动程序。

[627] https://www.freebsd.org/cgi/man.cgi?query=pciconf&sektion=8&format=html
[628] https://www.freebsd.org/cgi/man.cgi?query=man&sektion=1&format=html

10.4.配置文件

为了创建定制内核配置文件和编译定制内核，必须首先安装完整的 FreeBSD 源代码。

如果 **/usr/src/** 目录不存在或为空，说明没有安装源代码。可以使用 Git 来安装源代码。使用说明：“使用 Git”[629]。

源代码安装完毕后，检查 **/usr/src/sys** 的内容。这个目录包含许多子目录，包括那些表示 FreeBSD 支持的体系结构的子目录。在一个特定架构的目录中的所有内容只涉及该架构，其余的代码是所有平台通用的独立机器代码。每个支持的架构都有一个 **conf** 子目录，包含该架构的 **GENERIC** 内核配置文件。

请勿直接编辑 **GENERIC**。而应将文件复制并重命名为另一个文件名，并对该副本进行编辑。文件名通常使用全大写字母。当维护多台不同硬件的 FreeBSD 机器时，建议用机器的主机名来命名。在这个例子里，为 `amd64` 架构的 **GENERIC** 配置文件创建了一个名为 **MYKERNEL** 的副本：

```
# cd /usr/src/sys/amd64/conf
# cp GENERIC MYKERNEL
```

现在可以用任何 `ASCII` 文本编辑器来编辑 **MYKERNEL**。默认的编辑器是 vi，不过 FreeBSD 还安装了一个更容易使用的编辑器，名为 ee。

内核配置文件的格式很简单。每一行都包含一个代表设备或子系统的关键字，一个参数，和一个简短的介绍。`#` 后面的任何文字都被认为是注释并被忽略。要删除内核对某个设备或子系统的支持，在代表该设备或子系统的行的开头加上 `#`。不要为任何你不理解的行添加或删除 `#`。关于配置内核文件的更多信息可以在 config(5)[630] 中找到。

> 警告
>
> 移除对某个设备或选项的支持很容易导致内核损坏。例如，如果从内核配置文件中删除驱动 ata(4)[631]，使用 ATA 磁盘驱动的系统可能无法启动。如果不确定，就在内核中留下对设备的支持。

除了在这个文件中提供的简要介绍之外，在 **NOTES** 中还有更多的说明，这些说明可以在该架构的 **GENERIC** 的同一目录中找到。对于独立于架构的选项，请参考 **/usr/src/sys/conf/NOTES**。

> 技巧
>
> 当完成对内核配置文件的定制后，保存一份备份到 **/usr/src** 以外的位置。另外，在其他地方保留内核配置文件，并创建一个符号链接到该文件：

```
# cd /usr/src/sys/amd64/conf
# mkdir /root/kernels
# cp GENERIC /root/kernels/MYKERNEL
# ln -s /root/kernels/MYKERNEL MYKERNEL
```

[629] https://docs.freebsd.org/en/books/handbook/mirrors/index.html#git

[630] https://man.freebsd.org/cgi/man.cgi?query=config&sektion=5&format=html

[631] https://www.freebsd.org/cgi/man.cgi?query=ata&sektion=4&format=html

可以在配置文件中使用 include 指令。这允许将另一个配置文件包含在当前的文件中，从而使维护相对于现有文件的小改动变得容易。如果只需要少量的额外选项或驱动，就可以相对于 **GENERIC** 来说保持一个很小的变化，就像在这个例子中看到的那样：

```
include GENERIC
ident MYKERNEL

options         IPFIREWALL
options         DUMMYNET
options         IPFIREWALL_DEFAULT_TO_ACCEPT
options         IPDIVERT
```

使用这种方法，本地配置文件表达了与 **GENERIC** 内核的局部差异。随着升级的进行，添加到 **GENERIC** 中的新特性也将被添加到本地内核中，除非使用 **nooptions** 或 **nodevice** 来阻止。

在 config(5)[632] 中可以找到一份全面的配置指令清单和它们的简介。

> 注意
> 要创建一个包含所有可用选项的文件，请以 root 身份运行以下命令：

```
# cd /usr/src/sys/arch/conf && make LINT
```

10.5.编译与安装定制内核

步骤：编译内核：

1. 切换到 /usr/src 目录：

```
# cd /usr/src
```

2. 通过指定定制内核配置文件的名称来编译新的内核：

```
# make buildkernel KERNCONF=MYKERNEL
```

3. 安装与指定内核配置文件相关的新内核。这个命令将把新内核复制到 **/boot/kernel/kernel**，并把旧内核保存到 **/boot/kernel.old/kernel**：

```
# make installkernel KERNCONF=MYKERNEL
```

4. 关机重启进入新的内核。如果出了问题，请参考无法启动内核[633].

[632] https://www.freebsd.org/cgi/man.cgi?query=config&sektion=5&format=html
[633] https://docs.freebsd.org/en/books/handbook/kernelconfig/#kernelconfig-noboot

默认情况下，当编译一个定制内核时，所有的内核模块都会被重建。要想更快地更新内核或只编译自定义模块，需要在开始编译内核之前编辑 **/etc/make.conf**。

例如，这个变量指定了要编译的模块列表，而不是默认的编译所有模块：

```
MODULES_OVERRIDE = linux ipfw
```

另外，这个变量还列出了要在编译过程中排除哪些模块：

```
WITHOUT_MODULES = linux acpi sound
```

也可以使用其他变量。详情请参考 make.conf(5)[634]。

10.6.如果发生了一些错误

在建立一个定制的内核时，有四类问题可能发生：

config 失败

如果 `config` 失败，它将打印出配置错误的行号。

例如，对于下面的信息，要比较 **GENERIC** 与 **NOTES**，请确保第 32 行的输入是正确的：

```
config: /usr/src/sys/amd64/conf/GENERIC:32: syntax error
```

make 失败

如果 `make` 失败，通常是由于内核配置文件中的错误，而这个错误对于 `config` 来说还不够严重。查看配置，如果问题不明显，请向包含内核配置文件的 FreeBSD 一般问题邮件列表[635]发送一封电子邮件。

内核无法启动

如果新内核无法启动或无法识别设备，不要惊慌。FreeBSD 有一个优秀的机制即从不能运行的内核中恢复。

只要在 FreeBSD 的引导加载器中选择要启动的内核即可。当系统启动菜单出现时，可以通过选择“Kernel:”选项来访问它，该选项最初显示 “Kernel:default/kernel” 。每次选择该选项时，都会出现另一个选项，例如“kernel.old”。当所需的内核选项出现，按回车键启动。

正常启动后，检查一下配置文件，并尝试再次建立它。可以参考 **/var/log/messages**来获取帮助，它记录了每次成功启动时的内核信息。另外，dmesg(8)[636] 可以打印当前启动时的内核信息。

注意

[634] https://www.freebsd.org/cgi/man.cgi?query=make.conf&sektion=5&format=html

[635] https://lists.freebsd.org/subscription/freebsd-questions

[636] https://www.freebsd.org/cgi/man.cgi?query=dmesg&sektion=8&format=html

在排除内核的故障时，确保有一份 **GENERIC** 的备份，或者其他已知可以工作的内核在下次编译时不会被清除。这一点很重要，因为每次安装新内核时，**kernel.old** 都会被最后安装的内核覆盖，而这个内核可能是可启动的，也可能是不可启动的。尽快重命名包含可以正常启动的内核的目录，并转移工作内核：

```
# mv /boot/kernel /boot/kernel.bad
# mv /boot/kernel.good /boot/kernel
```

内核可以工作，但 ps(1)[637] 不能工作

如果内核版本与系统实用程序所使用的版本不同，例如，在一个 -RELEASE 系统上安装了一个由 -CURRENT 源构建的内核，许多系统状态命令（如 ps(1)[638] 和 vmstat(8)[639]）将无法运行。要解决这个问题，需要重新编译并安装[640]与内核相同版本的源代码树。使用与操作系统其他部分不同版本的内核绝对不是一个好主意。

[637] https://man.freebsd.org/cgi/man.cgi?query=ps&sektion=1&format=html

[638] https://man.freebsd.org/cgi/man.cgi?query=ps&sektion=1&format=html

[639] https://man.freebsd.org/cgi/man.cgi?query=vmstat&sektion=8&format=html

[640] https://docs.freebsd.org/en/books/handbook/cutting-edge/#makeworld

第十一章打印

将信息放在纸上是一项重要的功能，尽管有许多人试图淘汰它。打印由两个基本组成部分：数据必须被传递给打印机，而且必须以打印机能够理解的形式出现。

11.1.快速入门

基本的打印功能可以快速完成设置。打印机必须能够打印纯 ASCII 文本。关于打印其他类型的文件，请参见过滤器[641]。

1. 创建一个目录，在打印文件时转存文件：

```
# mkdir -p /var/spool/lpd/lp
# chown daemon:daemon /var/spool/lpd/lp
# chmod 770 /var/spool/lpd/lp
```

1. 以 root 权限用这些内容创建 **/etc/printcap**：

```
lp:\
lp=/dev/unlpt0:\  ①
sh:\
mx#0:\
sd=/var/spool/lpd/lp:\
lf=/var/log/lpd-errs:
```

① 这一行用于连接到 USB 端口的打印机。

对于连接到并行或“打印机”端口的打印机，使用：

```
:lp=/dev/lpt0:\
```

对于直接连接到网络的打印机，请使用：

[641] https://docs.freebsd.org/en/books/handbook/book/#printing-lpd-filters

```
:lp=:rm=network-printer-name:rp=raw:\
```

用网络打印机的 DNS 主机名代替 *network-printer-name*。

1. 通过编辑 **/etc/rc.conf**，添加这一行来启用 LPD：

```
lpd_enable="YES"
```

启动服务：

```
# service lpd start
Starting lpd.
```

1. 测试打印：

```
# printf "1. This printer can print.\n2. This is the second line.\n" | lpr
```

> 技巧
>
> 如果两行文本都不从左边界开始，而是呈现“阶梯状”，请看防止纯文本打印机的阶梯状打印[642]。

现在可以用 lpr 打印文本文件了。在命令行中给出文件名，或者直接用管道输出到 lpr 中。

```
% lpr textfile.txt
% ls -lh | lpr
```

11.2.连接打印机

打印机与计算机系统的连接方式多种多样。小型桌面打印机通常直接连接到计算机的 USB 端口。较老的打印机则连接到一个并行或”打印机“端口。一些打印机直接连接到网络上，使多台计算机可以很容易地共享它们。少数打印机使用罕见的串行端口连接。

FreeBSD 可以与所有这些类型的打印机进行通信。

- **USB**

USB 打印机可以连接到计算机上任何可用的 USB 端口。

当 FreeBSD 检测到一个 USB 打印机时，会创建两个设备条目：**/dev/ulpt0** 和 **/dev/unlpt0**。发送到任何一个设备的数据都会被转发到打印机。每次打印作业后，**ulpt0** 都会重置 USB 端口。重置端口可能会给某些打印机带来问题，所以通常使用 **unlpt0** 设备来代替。**unlpt0** 不会重置 USB 端口。

- **并行（IEEE-1284）**

并行设备是 **/dev/lpt0**。无论是否连接有打印机，这个设备都会出现，它不是自动检测的。

[642] https://docs.freebsd.org/en/books/handbook/book/#printing-lpd-filters-stairstep

供应商在很大程度上已经放弃了这些“传统”端口，许多计算机不再有这些端口。适配器可以用来将并行打印机连接到 USB 端口。有了这样的适配器，打印机就可以被当作实际上是一台 USB 打印机。被称为打印服务器的设备也可以用来将并行打印机直接连接到网络上。

- **串行（RS232）**

串行端口是另一个传统的端口，除了在某些专项应用中，很少用于打印机。电缆、连接器和所需的布线差别很大。

对于内置于主板的串行端口，串行设备名称为 **/dev/cuau0** 或 **/dev/cuau1**。也可以使用串行 USB 适配器，这些适配器将显示为 **/dev/cuaU0**。

要与串行打印机通信，必须知道几个通信参数。最重要的是 *波特率*或 `BPS`（Bits Per Second）和 *奇偶校验*。数值各不相同，但典型的串行打印机使用波特率 9600 和无奇偶校验。

- **网络**

网络打印机直接连接到本地计算机网络。

必须知道打印机的 DNS 主机名。如果打印机是由 DHCP 分配的动态地址，那么 DNS 应该动态更新，以便主机名总是有正确的 IP 地址。网络打印机通常被赋予静态 IP 地址以避免这个问题。

大多数网络打印机能理解用 LPD 协议发送的打印作业。也可以指定一个打印队列名称。有些打印机对数据的处理方式不同，这取决于使用哪种队列。例如，原始（`raw`）队列打印的数据是不变的，而文本（`text`）队列则在纯文本中加入换行符。

许多网络打印机也可以打印直接发送到 9100 端口的数据。

11.2.1.摘要

有线网络连接通常是最容易设置的，而且打印速度最快。对于与电脑的直接连接，USB 是首选，因为速度快且简单。并行连接也可以，但对电缆长度和速度有限制。串行连接更难配置。不同型号的电缆布线不同，通信参数如波特率和奇偶校验位必然增加了复杂性。所幸串行打印机很少见。

11.3.常见的页面描述语言

发送给打印机的数据必须使用打印机能够理解的语言。这些语言被称为页面描述语言，或称 PDL。

- *ASCII*

纯 ASCII 文本是向打印机发送数据的最简单方式。字符与将被打印的内容一一对应：数据中的 A 会在页面上打印出 A。可用的格式化非常少。没有办法选择字体或比例间距。ASCII 的简单性意味着可以直接从计算机中打印出来文本，几乎不需要编码或翻译。打印出来的结果与发送的内容直接对应。

一些便宜的打印机不能打印纯 ASCII 文本。这使得它们更难设置，但通常的打印机还是可以的。

- **PostScript®**

PostScript® 几乎与 ASCII 相反。PostScript® 程序不是简单的文本，而是一组绘制最终文件的指令。可以使用不同的字体和图形。然而，这种能力是有代价的。必须编写绘制页面的程序。通常这个程序是由应用软件生成的，所以这个过程对用户来说是不可见的。

廉价的打印机有时会省去 PostScript® 的兼容性，作为一种节约成本的措施。

- *PCL* **（打印机命令语言）**

PCL 是 ASCII 的扩展，增加了用于格式化、字体选择和打印图形的转义序列。许多打印机提供 PCL5 支持。有些支持较新的 PCL6 或 PCLXL。这些后来的版本是 PCL5 的超集，可以提供更快的打印。

- **基于主机**

制造商可以通过给打印机配备一个简单的处理器和很少的内存来降低其成本。这些打印机不能够打印纯文本。相反，文本和图形的位图由主机上的驱动程序绘制，然后发送到打印机上。这些打印机被称为基于主机的打印机。

驱动程序和基于主机的打印机之间的通信通常是通过专有的或未记录的协议进行的，这使得它们只能在最常见的操作系统上发挥作用。

11.3.1.将 PostScript® 转换为其他 PDL

许多来自 ports 和 FreeBSD 工具的应用程序会产生 PostScript® 输出。这张表显示了可用于将其转换为其他常见的 PDL 的实用程序。

表 9. 输出 PDL

输出 PDL	由谁产生	说明
PCL or PCL5	print/ghostscript9-base[643]	单色使用 `-sDEVICE=ljet4`，彩色使用 `-sDEVICE=cljet5`
PCLXL or PCL6	print/ghostscript9-base[644]	单色使用 `-sDEVICE=pxlmono`，彩色使用 `-sDEVICE=pxlcolor` for color
ESC/P2	print/ghostscript9-base[645]	`-sDEVICE=uniprint`
XQX	print/foo2zjs[646]	

[643] https://cgit.freebsd.org/ports/tree/print/ghostscript9-base/pkg-descr
[644] https://cgit.freebsd.org/ports/tree/print/ghostscript9-base/pkg-descr
[645] https://cgit.freebsd.org/ports/tree/print/ghostscript9-base/pkg-descr
[646] https://cgit.freebsd.org/ports/tree/print/foo2zjs/pkg-descr

11.3.2.摘要

为了实现最简单的打印，请选择支持 PostScript® 的打印机。支持 PCL 的打印机是下一个首选。通过 print/ghostscript9-base[647]，这些打印机可以像原生的 PostScript® 一样使用。直接支持 PostScript® 或 PCL 的打印机几乎都支持直接打印纯 ASCII 文本文件。

行式打印机，如典型的喷墨打印机，通常不支持 PostScript® 或 PCL。print/ghostscript9-base[648] 支持其中一些打印机所使用的 PDL。然而，由于需要传输和打印大量的数据，在这些打印机上打印整个基于图形的页面时往往非常慢。

基于主机的打印机往往更难设置。有些因为有专有的 PDL 而根本无法使用。尽可能避免使用这些打印机。

许多 PDL 的说明见 http://www.undocprint.org/formats/page_description_languages。各种型号的打印机所使用的特定 PDL 可以在 http://www.openprinting.org/printers 找到。

11.4.直接打印

对于偶尔的打印需求，可以直接将文件发送到打印机设备上，无需任何设置。例如，可以把一个名为 **sample.txt** 的文件发送到一台 USB 打印机上：

```
# cp sample.txt /dev/unlpt0
```

直接打印到网络打印机取决于打印机的支持，但大多数打印机都接受来自 9100 端口的打印作业，nc(1)[649] 可以与它们一起使用。要将同一个文件打印到 DNS 主机名为 *netlaser* 的打印机上：

```
# nc netlaser 9100 < sample.txt
```

11.5.LPD（行式打印机程序）

在后台打印文件被称为 *打印后台处理服务*打印队列。打印后台处理服务程序允许用户继续使用计算机上的其他程序，而不必等待打印机慢慢完成打印任务。

FreeBSD 包括一个叫做 lpd(8)[650] 的打印后台处理服务。打印任务是用 lpr(1)[651] 来提交的。

[647] https://cgit.freebsd.org/ports/tree/print/ghostscript9-base/pkg-descr
[648] https://cgit.freebsd.org/ports/tree/print/ghostscript9-base/pkg-descr
[649] https://www.freebsd.org/cgi/man.cgi?query=nc&sektion=1&format=html
[650] https://www.freebsd.org/cgi/man.cgi?query=lpd&sektion=8&format=html
[651] https://www.freebsd.org/cgi/man.cgi?query=lpr&sektion=1&format=html

11.5.1.初始设置

需要创建用于存储打印任务的目录，设置权限，防止其他用户查看这些文件的内容：

```
# mkdir -p /var/spool/lpd/lp
# chown daemon:daemon /var/spool/lpd/lp
# chmod 770 /var/spool/lpd/lp
```

打印机是在 **/etc/printcap** 中定义的。每台打印机的条目包括名称、所连接的端口和其他各种设置等细节。用这些内容创建 **/etc/printcap**：

```
lp:\                          ①
    :lp=/dev/unlpt0:\             ②
    :sh:\                     ③
    :mx#0:\                   ④
    :sd=/var/spool/lpd/lp:\           ⑤
    :lf=/var/log/lpd-errs:            ⑥
```

① 该打印机的名称。lpr(1)[652] 将打印任务发送到 lp 打印机，除非用 -P 指定其他打印机，所以默认的打印机应该被命名为 `lp`。

② 打印机所连接的设备。可以用合适的连接类型来替换这一行。

③ 在打印任务开始时不打印标题页。

④ 不限制打印任务的最大尺寸。

⑤ 访问该打印机的打印队列目录的路径。每台打印机都有自己的打印队列目录。

⑥ 该打印机上的错误情况的日志文件位置。

创建 **/etc/printcap** 后，使用 chkprintcap(8)[653] 来进行测试，以排除潜在的错误：

```
# chkprintcap
```

在继续下一步之前先解决出现的错误。

在 **/etc/rc.conf** 启用lpd(8)[654] ：

```
lpd_enable="YES"
```

启动打印服务：

[652] https://www.freebsd.org/cgi/man.cgi?query=lpr&sektion=1&format=html
[653] https://www.freebsd.org/cgi/man.cgi?query=chkprintcap&sektion=8&format=html
[654] https://www.freebsd.org/cgi/man.cgi?query=lpd&sektion=8&format=html

```
# service lpd start
```

11.5.2.通过 lpr[Page 247, 655] 进行打印

文件通过 `lpr` 被发送到打印机。需要打印的文件可以用命令行的方式来命名并被传送到 `lpr`。下面这两条命令效果相同，都是把 **doc.txt** 的内容发送到默认打印机打印：

```
% lpr doc.txt
% cat doc.txt | lpr
```

可以用 -P 来指定打印机。如要将分件发送到名为 *laser* 的打印机打印：

```
% lpr -Plaser doc.txt
```

11.5.3.过滤器

之前的例子都是把文本文件的内容直接发送到打印机打印。只要打印机识别那些文件的内容，就可以正确进行打印。

一些打印机不能打印纯文本格式，提交打印机的文件可能根本不是纯文本格式。

过滤器可以转换或处理这些文件。一种常用的情况是把发送进来的文件，如纯文本格式，转换到打印机可以识别的形式，如 PostScript® 或 PCL。过滤器还能提供额外的功能，比如添加页号或突出显示源代码部分来更便于阅读。

这里要讨论的过滤器是 *输入过滤器*和 *字符过滤器*这些过滤器把收到的文件转换为不同格式。在创建文件之前需要先使用 su(1)[656] 切换到 `root` 用户。

过滤器可以通过 `if=identifier` 在 **/etc/printcap** 中进行定义。如要使用 **/usr/local/libexec/lf2crlf** 作为过滤器，修改 **/etc/printcap** 如下：

```
lp:\
    :lp=/dev/unlpt0:\
    :sh:\
    :mx#0:\
    :sd=/var/spool/lpd/lp:\
    :if=/usr/local/libexec/lf2crlf:\    ①
    :lf=/var/log/lpd-errs:
```

① `if=` 指定的输入过滤器会作用于所有接收到的文本

[655] https://www.freebsd.org/cgi/man.cgi?query=lpr&sektion=1&format=html

[656] https://www.freebsd.org/cgi/man.cgi?query=su&sektion=1&format=html

技巧

printcap 中定义的打印机也可以用被反斜杠和冒号等字符分隔的一长行来定义，之前的例子可以用下面单独一行来说明，尽管有些难以阅读：

```
lp:lp=/dev/unlpt0:sh:mx#0:sd=/var/spool/lpd/lp:if=/usr/local/libexec/
↪lf2crlf:lf=/var/log/lpd-errs:
```

11.5.3.1.避免文本打印机出现阶梯状的打印效果

通常 FreeBSD 文本文件在每行末尾只包含一个换行字符，就会在标准的打印机上出现如下阶梯状打印结果：

```
A printed file looks
                    like the steps of a staircase
                                                scattered by the wind
```

过滤器可以把新行的字符转换为回车符和新行。回车符让打印机在打印每一行后回到下一行的最左方。创建 **/usr/local/libexec/lf2crlf** 并增加如下内容：

```
#!/bin/sh
CR=$'\r'
/usr/bin/sed -e "s/$/${CR}/g"
```

设置相关可执行权限：

```
# chmod 555 /usr/local/libexec/lf2crlf
```

修改 **/etc/printcap** 使用新过滤器：

```
:if=/usr/local/libexec/lf2crlf:\
```

再次打印同样的纯文本格式文件来进行测试。顺利的话回车符就会让每行从页面的最左方开始。

11.5.3.2.用 print/enscriptPage 248, 657 让 PostScript® 打印机打出漂亮的纯文本字体

GNUEnscript 可以让支持 PostScript® 的打印机把纯文本格式文件转换为排版优美的 PostScript® 进行打印。它还增加了页码，自动换行和其它许多特性，让打印出来的文本更易于阅读。根据纸张尺寸大小，从 ports 安装 print/enscript-letter[658] 或者 print/enscript-a4[659]。

[657] https://cgit.freebsd.org/ports/tree/print/enscript/pkg-descr

[658] https://cgit.freebsd.org/ports/tree/print/enscript-letter/pkg-descr

[659] https://cgit.freebsd.org/ports/tree/print/enscript-a4/pkg-descr

创建 **/usr/local/libexec/enscript** 并添加以下内容：

```
#!/bin/sh
/usr/local/bin/enscript -o -
```

设置权限为可执行：

```
# chmod 555 /usr/local/libexec/enscript
```

更改 **/etc/printcap** 来使用新过滤器：

```
:if=/usr/local/libexec/enscript:\
```

打印纯文本文件来测试过滤器。

11.5.3.3.打印 PostScript® 到 PCL 打印机

许多程序输出 PostScript® 格式的文档。然而，便宜的打印机通常只能识别纯文本或 PCL 格式。这个过滤器可以把 PostScript® 格式文件转换为 PCL 格式后再发送给打印机。

从 ports 安装 Ghostscript PostScript® 解释器 print/ghostscript9-base[660]。

创建 **/usr/local/libexec/ps2pcl ** 并添加以下内容：

```
#!/bin/sh
/usr/local/bin/gs -dSAFER -dNOPAUSE -dBATCH -q -sDEVICE=ljet4 -sOutputFile=- -
```

设置权限为可执行：

```
# chmod 555 /usr/local/libexec/ps2pcl
```

PostScript® 格式的内容经过该脚本处理后会被转换为 PCL 格式再发送到打印机。

修改 **/etc/printcap** 来使用这个新的过滤器：

```
:if=/usr/local/libexec/ps2pcl:\
```

发送少量 PostScript® 格式内容来进行测试：

```
% printf "%%\!PS \n /Helvetica findfont 18 scalefont setfont \
72 432 moveto (PostScript printing successful.) show showpage \004" | lpr
```

[660] https://cgit.freebsd.org/ports/tree/print/ghostscript9-base/pkg-descr

11.5.3.4.智能过滤器

此类过滤器可以侦测输入内容的类型并自动把它转换到正确的格式让打印机打印，它们非常有用。PostScript® 格式文件的头两个字符通常是 `%!`，过滤器可以侦测到它们。而 PostScript® 格式文件不需要更改就可以被直接发送到支持 PostScript® 的打印机进行打印。使用之前的 Enscript 过滤器文本文件就可以被转换为 PostScript® 格式。创建 **/usr/local/libexec/psif** 并添加以下内容：

```
#!/bin/sh
#
#  psif - Print PostScript or plain text on a PostScript printer
#
IFS="" read -r first_line
first_two_chars=`expr "$first_line" : '\(..\)'`

case "$first_two_chars" in
%!)
    # %! : PostScript job, print it.
    echo "$first_line" && cat && exit 0
    exit 2
    ;;
*)
    # otherwise, format with enscript
    ( echo "$first_line"; cat ) | /usr/local/bin/enscript -o - && exit 0
    exit 2
    ;;
esac
```

设置文件权限为可执行：

```
# chmod 555 /usr/local/libexec/psif
```

修改 **/etc/printcap** 来使用新过滤器：

```
:if=/usr/local/libexec/psif:\
```

打印 PostScript® 格式的纯文本文件来测试过滤器。

11.5.3.5.其它智能过滤器

编写一个能识别许多不同类型的输入并正确进行格式转换是非常有挑战的。ports 中的 print/apsfilter[661] 是一种智能而且神奇的过滤器，它可以识别十几种文件格式类型并自动把它们转换为能被打印机理解的 PDL 打印描述语言。更多详细内容请参考 http://www.apsfilter.org。

11.5.4.多打印队列配置

/etc/printcap 中的内容严格来说就是各种打印机 队列定义。可以为单台打印机增加许多队列定义，当与过滤器同时使用时，这些不同的队列为用户提供了更多的控制来完成打印任务。

举个例子，在某个办公室有一台联网的 PostScript® 激光打印机。大多数用户想打印纯文本格式，但有些更高级的用户希望能直接打印 PostScript® 格式的文件，那么可以在 **/etc/printcap:** 中为这两类群体创建两种打印队列：

```
textprinter:\
    :lp=9100@officelaser:\
    :sh:\
    :mx#0:\
    :sd=/var/spool/lpd/textprinter:\
    :if=/usr/local/libexec/enscript:\
    :lf=/var/log/lpd-errs:

psprinter:\
    :lp=9100@officelaser:\
    :sh:\
    :mx#0:\
    :sd=/var/spool/lpd/psprinter:\
    :lf=/var/log/lpd-errs:
```

发送到打印机 `textprinter` 的文档会被 **/usr/local/libexec/enscript** 过滤器进行格式转换，如之前的例子所述，而高级用户在不需要格式转化的情况下就可以使用 `psprinter` 来直接打印 PostScript® 格式的文件。

这个多打印队列配置的技巧可以用来利用打印机的各种不同功能。带双面器的打印机也可以使用两种配置队列，一种是普通单面打印，另一种是带双面打印过滤控制指令的双面打印。

[661] https://cgit.freebsd.org/ports/tree/print/apsfilter/pkg-descr

11.5.5.监督并控制打印流程

有几种工具可以监督打印任务，查看并控制打印操作。

11.5.5.1.lpq(1)[Page 252, 662]

lpq(1)[663] 显示用户打印任务的状态。其它用户的打印任务不会显示。

下例显示当前用户在单台打印机上还未完成的打印任务：

```
% lpq -Plp
Rank    Owner      Job  Files                                 Total Size
1st     jsmith     0    (standard input)                      12792 bytes
```

下例显示当前用户在所有打印机上的任务状态：

```
% lpq -a
lp:
Rank    Owner      Job  Files                                 Total Size
1st     jsmith     1    (standard input)                      27320 bytes

laser:
Rank    Owner      Job  Files                                 Total Size
1st     jsmith     287  (standard input)                      22443 bytes
```

11.5.5.2.lprm(1)[664]

lprm(1)[665] 用来删除打印任务。普通用户只能删除自己的打印任务。root 用户可以删除任何或所有的打印任务。

删除某台打印机上所有未完成的打印任务：

```
# lprm -Plp -
dfA002smithy dequeued
cfA002smithy dequeued
dfA003smithy dequeued
cfA003smithy dequeued
dfA004smithy dequeued
cfA004smithy dequeued
```

662 https://www.freebsd.org/cgi/man.cgi?query=lpq&sektion=1&format=html
663 https://www.freebsd.org/cgi/man.cgi?query=lpq&sektion=1&format=html
664 https://www.freebsd.org/cgi/man.cgi?query=lprm&sektion=1&format=html
665 https://www.freebsd.org/cgi/man.cgi?query=lprm&sektion=1&format=html

删除某台打印机上单项打印任务。可以用 lpq(1)[666] 来查找打印任务编号。

```
% lpq
Rank   Owner      Job  Files                                 Total Size
1st    jsmith     5    (standard input)                      12188 bytes

% lprm -Plp 5
dfA005smithy dequeued
cfA005smithy dequeued
```

11.5.5.3.lpc(8)[Page 253, 667]

lpc(8)[668] 被用来查看和修改打印机状态。`lpc` 命令可以带命令参数和打印机名称参数。可以使用 `all` 来替代具体的打印机名称，命令就会用于所有打印机。普通用户可以查用 lpc(8)[669] 来查看打印状态。只有 `root` 用户才可以使用修改打印机状态的命令。

显示所有打印机的状态：

```
% lpc status all
lp:
    queuing is enabled
    printing is enabled
    1 entry in spool area
    printer idle
laser:
    queuing is enabled
    printing is enabled
    1 entry in spool area
    waiting for laser to come up
```

阻止打印机接收新打印任务，然后又开始重新接收新打印任务：

```
# lpc disable lp
lp:
    queuing disabled
# lpc enable lp
lp:
    queuing enabled
```

停止打印，但可以继续接收新打印任务。然后开始继续打印：

[666] https://www.freebsd.org/cgi/man.cgi?query=lpq&sektion=1&format=html
[667] https://www.freebsd.org/cgi/man.cgi?query=lpc&sektion=8&format=html
[668] https://www.freebsd.org/cgi/man.cgi?query=lpc&sektion=8&format=html
[669] https://www.freebsd.org/cgi/man.cgi?query=lpc&sektion=8&format=html

```
# lpc stop lp
lp:
    printing disabled
# lpc start lp
lp:
    printing enabled
    daemon started
```

在出现一些出错情况后重新启动打印机：

```
# lpc restart lp
lp:
    no daemon to abort
    printing enabled
    daemon restarted
```

关闭打印队列并使打印机不可用，并将相关信息告知所有用户：

```
# lpc down lp Repair parts will arrive on Monday
lp:
    printer and queuing disabled
    status message is now: Repair parts will arrive on Monday
```

重新恢复被设置为不可用的打印机：

```
# lpc up lp
lp:
    printing enabled
    daemon started
```

更多命令和参数选项请参考 lpc(8)[670]。

11.5.6.共享打印机

在公司和学校中打印机经常被设置为不同用户共同使用。有些额外功能可以让共享打印机变得更加方便。

[670] https://www.freebsd.org/cgi/man.cgi?query=lpc&sektion=8&format=html

11.5.6.1.别名

打印机名称可以在 **/etc/printcap** 中第一行配置里进行设置。额外的名称，或 *别名*可以添加到打印机名称之后。用 | 来区隔打印机名称以及不同的别名（可以有多个别名）：

```
lp|repairsprinter|salesprinter:\
```

在使用打印机名称的地方都可以用别名来替代。例如，销售部门的用户可以如下别名来进行打印：

```
% lpr -Psalesprinter sales-report.txt
```

维修部门的用户也可以使用如下别名来进行打印：

```
% lpr -Prepairsprinter repairs-report.txt
```

所有的文档都在同一台打印机上打印。当销售部门足够扩展到需要他们自己的打印机时，可以把别名从共享打印机配置里删除并把同样的名字用于新的打印机。在两个部门的用户们可以使用之前的命令继续打印，但销售部门的文档就会发送到新打印机进行打印。

11.5.6.2.报头页

在从一台任务繁忙的共享打印机上的一堆纸张中找到自己的打印文档对用户们来说不是一件容易得事情。报头页就是用来解决这个问题的方法。在每项打印任务之前，用户名和文档名称会被打印在报头部分。这些页面有时也被称为 *横幅页*或 *分隔页*。

根据打印机连接到计算机的方式不同（USB 连接，并口连接，串口连接或网络连接等），报头页的设置也有所不同。

直接连接打印机的报头页功能可以通过删除 **/etc/printcap** 中 **:sh:**（不用报头）这一配置行来启用。这些报头页只使用换行字符来进行换行。有些打印机还需要 **/usr/share/examples/printing/hpif** 过滤器来防止出现楼梯状的打印文本。当需要换行打印时，过滤器控制 PCL 格式打印机可以同时正常打印回车和换行字符。

网络打印机报头页设置必须在打印机上进行。**/etc/printcap** 的报头页配置不会起作用。通常这种设置可以在打印机上的控制面板或网页版设置页面上进行。

11.5.7.其它参考

示例文档：**/usr/share/examples/printing/**。

4.3BSD 行式打印机队列手册，**/usr/share/doc/smm/07.lpd/paper.ascii.gz**。

手册页面：printercap(5)[671], lpd(8)[672]，lpr(1)[673]，lpc(8)[674]，lprm(1)[675]，lpq(1)[676]。

11.6.其他打印系统

除了自带的 lpd(8)[677] 以外，还可以使用其它几种打印系统。这些系统支持其它协议和额外的功能。

11.6.1.CUPS (通用 UNIX 打印系统)

CUPS 是一种在许多操作系统中都流行的打印系统。关于在 FreeBSD 中使用 CUPS 请参考单独的文档：CUPS[678]

11.6.2.HPLIP

惠普公司提供了一种支持很多惠普生产的喷墨和激光打印机的打印系统。相关源代码仓库是 print/hplip[679]。网页地址在：https://developers.hp.com/hp-linux-imaging-and-printing。port 包含了所有和 FreeBSD 有关的详细安装说明。配置信息等相关内容请参考 https://developers.hp.com/hp-linux-imaging-and-printing/install

11.6.3.LPRng

LPRng 作为一种可选打印系统，可以看作是升级版的 lpd(8)[680]。它的源代码仓库在 sysutils/LPRng[681]。相关文档和详细信息请参考 http://www.lprng.com/。

[671] https://www.freebsd.org/cgi/man.cgi?query=printcap&sektion=5&format=html
[672] https://www.freebsd.org/cgi/man.cgi?query=lpd&sektion=8&format=html
[673] https://www.freebsd.org/cgi/man.cgi?query=lpr&sektion=1&format=html
[674] https://www.freebsd.org/cgi/man.cgi?query=lpc&sektion=8&format=html
[675] https://www.freebsd.org/cgi/man.cgi?query=lprm&sektion=1&format=html
[676] https://www.freebsd.org/cgi/man.cgi?query=lpq&sektion=1&format=html
[677] https://www.freebsd.org/cgi/man.cgi?query=lpd&sektion=8&format=html
[678] https://docs.freebsd.org/en/articles/cups/
[679] https://cgit.freebsd.org/ports/tree/print/hplip/pkg-descr
[680] https://www.freebsd.org/cgi/man.cgi?query=lpd&sektion=8&format=html
[681] https://cgit.freebsd.org/ports/tree/sysutils/LPRng/pkg-descr

第十二章 Linux 二进制兼容层

12.1.概述

FreeBSD 提供了 **可选的** Linux® 二进制兼容层（通常被称为 Linuxulator），允许用户安装和运行原生的 Linux 二进制程序。它适用于 x86（32 位和 64 位）、AArch64 架构。一些 Linux 特有的功能还没有被支持。这主要是一些硬件特有的功能或与系统管理有关的功能，如 cgroups 和 namespaces。

阅读本章前，你应该知道：

- 了解如何安装其他第三方软件[682]。

读完本章后，你将知道：

- 如何在 FreeBSD 系统上启用 Linux 二进制兼容层。
- 如何安装额外的 Linux 依赖库。
- 如何在 FreeBSD 系统上安装 Linux 应用程序。
- FreeBSD 中的 Linux 兼容层的实现细节。

在阅读本章之前，你应该：

- 知道如何安装额外的第三方软件[683]。

[682] https://docs.freebsd.org/en/books/handbook/ports/#ports

[683] https://docs.freebsd.org/en/books/handbook/ports/index.html#ports

12.2.配置 Linux 二进制兼容层

在默认情况下，linux(4)[684] 二进制兼容层没有启用。

要在系统启动时启用 Linux ABI，请执行下面的命令：

```
# sysrc linux_enable="YES"
```

配置后，就可以通过运行以下命令来启动，而无需重启：

```
# service linux start
```

这足以让静态链接的 Linux 二进制文件工作。

Linux 服务将加载必要的内核模块，并在 **/compat/Linux** 路径下挂载 Linux 软件所需要的文件系统。然后，Linux 二进制文件可以以与 FreeBSD 本地二进制文件相同的方式启动——它们的行为几乎与本地进程完全一样，并且可以以平常的方式进行跟踪和调试。

可以通过执行以下命令对 **/compat/linux** 路径下的当前内容进行检查：

```
# ls -l /compat/linux/
```

输出应该与下面类似：

```
total 1
dr-xr-xr-x  13 root  wheel  512 Apr 11 19:12 dev
dr-xr-xr-x   1 root  wheel    0 Apr 11 21:03 proc
dr-xr-xr-x   1 root  wheel    0 Apr 11 21:03 sys
```

12.3.Linux 用户空间

Linux 软件正常工作所依赖的不只是 ABI。为了运行这样的 Linux 软件，必须首先安装 Linux 用户空间。

技巧

如果只是想运行一些已经包含在 Ports 中的软件，可以通过软件包管理器来安装，pkg(8)[685] 将自动设置所需的 Linux 用户空间。

例如，要安装 Sublime Text 4，以及它所依赖的所有 Linux 库，运行这个命令：

```
# pkg install linux-sublime-text4
```

[684] https://man.freebsd.org/cgi/man.cgi?query=linux&sektion=4&format=html

[685] https://man.freebsd.org/cgi/man.cgi?query=pkg&sektion=8&format=html

12.3.1.从 FreeBSD 软件包安装 CentOS 基础系统

要安装 CentOS 用户空间，请执行以下命令：

```
# pkg install linux_base-c7
```

emulators/linux_base-c7[686] 将衍生自 CentOS 7 的基本系统安装在 **/compat/linux** 路径下。

安装这个包后，通过执行下列命令查看 **/compat/linux** 路径下的内容来确认 CentOS 用户空间已经被成功安装：

```
# ls -l /compat/linux/
```

输出应该和下面类似：

```
total 30
lrwxr-xr-x   1 root  wheel    7 Apr 11  2018 bin -> usr/bin
drwxr-xr-x  13 root  wheel  512 Apr 11 21:10 dev
drwxr-xr-x  25 root  wheel   64 Apr 11 21:10 etc
lrwxr-xr-x   1 root  wheel    7 Apr 11  2018 lib -> usr/lib
lrwxr-xr-x   1 root  wheel    9 Apr 11  2018 lib64 -> usr/lib64
drwxr-xr-x   2 root  wheel    2 Apr 11 21:10 opt
dr-xr-xr-x   1 root  wheel    0 Apr 11 21:25 proc
lrwxr-xr-x   1 root  wheel    8 Feb 18 02:10 run -> /var/run
lrwxr-xr-x   1 root  wheel    8 Apr 11  2018 sbin -> usr/sbin
drwxr-xr-x   2 root  wheel    2 Apr 11 21:10 srv
dr-xr-xr-x   1 root  wheel    0 Apr 11 21:25 sys
drwxr-xr-x   8 root  wheel    9 Apr 11 21:10 usr
drwxr-xr-x  16 root  wheel   17 Apr 11 21:10 var
```

12.3.2.使用 debootstrap 构建 Debian/Ubuntu 基本系统

另一种提供 Linux 共享库的方法是使用 sysutils/debootstrap[687]。这样做的好处是可以提供一个完整的 Debian 或 Ubuntu 发行版。

执行下列命令安装 debootstrap：

```
# pkg install debootstrap
```

debootstrap(8)[688] 需要启用 linux(4)[689] ABI 支持。启用后，执行以下命令在 **/compat/ubuntu** 路径下安装

[686] https://cgit.freebsd.org/ports/tree/emulators/linux_base-c7/

[687] https://cgit.freebsd.org/ports/tree/sysutils/debootstrap/

[688] https://man.freebsd.org/cgi/man.cgi?query=debootstrap&sektion=8&format=html

[689] https://man.freebsd.org/cgi/man.cgi?query=linux&sektion=4&format=html

Ubuntu 或 Debian：

```
# debootstrap focal /compat/ubuntu
```

> 注意
>
> 虽然从技术上来说可以安装到 **/compat/linux** 路径下，但是为了避免和基于 CentOS 的软件包发生冲突，不建议这样做。因此，应该从发行版或版本名称中导出目录名称，例如，**/compat/ubuntu**。

输出应该和下面类似：

```
I: Retrieving InRelease
I: Checking Release signature
I: Valid Release signature (key id F6ECB3762474EDA9D21B7022871920D1991BC93C)
I: Retrieving Packages
I: Validating Packages
I: Resolving dependencies of required packages...
I: Resolving dependencies of base packages...
I: Checking component main on http://archive.ubuntu.com/ubuntu...
[...]
I: Configuring console-setup...
I: Configuring kbd...
I: Configuring ubuntu-minimal...
I: Configuring libc-bin...
I: Configuring ca-certificates...
I: Base system installed successfully.
```

然后在 **/etc/fstab** 中设置挂载。

> 技巧
>
> 如果家目录的内容会被共享，以及为了能够运行 X11 软件，**/home** 和 **/tmp** 目录应该使用 nullfs(5)[690] 挂载在 linux compat 区域用于回环。
>
> 下面的例子可以添加在 **/etc/fstab** 文件中：

```
# Device        Mountpoint              FStype          Options             ␣
↪       Dump    Pass#
devfs           /compat/ubuntu/dev      devfs           rw,late             ␣
↪       0       0
tmpfs           /compat/ubuntu/dev/shm  tmpfs           rw,late,size=1g,
↪mode=1777    0       0
fdescfs         /compat/ubuntu/dev/fd   fdescfs         rw,late,linrdlnk    ␣
↪       0       0
linprocfs       /compat/ubuntu/proc     linprocfs       rw,late             ␣
↪       0       0
```

(continues on next page)

[690] https://man.freebsd.org/cgi/man.cgi?query=nullfs&sektion=5&format=html

(continued from previous page)

```
linsysfs        /compat/ubuntu/sys      linsysfs        rw,late             ␣
↪       0       0
/tmp            /compat/ubuntu/tmp      nullfs          rw,late             ␣
↪       0       0
/home           /compat/ubuntu/home     nullfs          rw,late             ␣
↪       0       0
```

然后执行 mount(8)[691]：

```
# mount -al
```

要使用 chroot(8)[692] 访问系统，请执行以下命令：

```
# chroot /compat/ubuntu /bin/bash
```

然后可以执行 uname(1)[693] 来检查 Linux 环境：

```
# uname -s -r -m
```

输出应该和下面类似：

```
Linux 3.17.0 x86_64
```

进入 chroot 后，系统的表现与正常的 Ubuntu 安装一样。虽然 systemd 没有运行，但 service(8)[694] 命令照常工作。

技巧

想要添加默认源中缺少的软件包库，请编辑 **/compat/ubuntu/etc/apt/sources.list** 文件。

对于 amd64 架构，可以使用下面的例子：

```
deb http://archive.ubuntu.com/ubuntu focal main universe restricted␣
↪multiverse
deb http://security.ubuntu.com/ubuntu/ focal-security universe multiverse␣
↪restricted main
deb http://archive.ubuntu.com/ubuntu focal-backports universe multiverse␣
↪restricted main
deb http://archive.ubuntu.com/ubuntu focal-updates universe multiverse␣
↪restricted main
```

对于 amd64 架构，另一个可以使用的例子：

[691] https://man.freebsd.org/cgi/man.cgi?query=mount&sektion=8&format=html

[692] https://man.freebsd.org/cgi/man.cgi?query=chroot&sektion=8&format=html

[693] https://man.freebsd.org/cgi/man.cgi?query=uname&sektion=1&format=html

[694] https://man.freebsd.org/cgi/man.cgi?query=service&sektion=8&format=html

```
deb http://ports.ubuntu.com/ubuntu-ports bionic main universe restricted␣
↪multiverse
```

12.4.高级主题

可以在 linux(4)[695] 中找到所有与 Linux 有关的 sysctl(8)[696] 变量的清单。

一些应用程序要求挂载特定的文件系统。

这通常由 **/etc/rc.d/linux** 脚本处理，但可以通过执行以下命令在开机时禁用这个脚本：

```
sysrc linux_mounts_enable="NO"
```

由 rc 脚本挂载的文件系统对 chroot 或 jail 内的 Linux 进程不起作用；如果需要，在 **/etc/fstab** 中配置它们：

```
devfs      /compat/linux/dev      devfs      rw,late                    0  0
tmpfs      /compat/linux/dev/shm  tmpfs      rw,late,size=1g,mode=1777  0  0
fdescfs    /compat/linux/dev/fd   fdescfs    rw,late,linrdlnk           0  0
linprocfs  /compat/linux/proc     linprocfs  rw,late                    0  0
linsysfs   /compat/linux/sys      linsysfs   rw,late                    0  0
```

由于 Linux 二进制兼容层已经获得了对运行 32 位和 64 位 Linux 二进制文件的支持，因此不再可能将仿真功能静态地链接到一个定制内核中。

12.4.1.手动安装附加库

> 注意
> 对于用 debootstrap(8)[697] 创建的基本系统子目录，参考上述说明。

如果一个 Linux 应用程序在配置了 Linux 二进制兼容层后抱怨缺少共享库，请确定 Linux 二进制需要哪些共享库，并手动安装它们。

在使用相同 CPU 架构的 Linux 系统中，可以用 `ldd` 来确定应用程序需要哪些共享库。

例如，要检查 `linuxdoom` 需要哪些共享库，可以从安装了 Doom 的 Linux 系统上运行这个命令：

```
% ldd linuxdoom
```

输出应该和下面类似：

[695] https://www.freebsd.org/cgi/man.cgi?query=linux&sektion=4&format=html
[696] https://www.freebsd.org/cgi/man.cgi?query=sysctl&sektion=8&format=html
[697] https://www.freebsd.org/cgi/man.cgi?query=debootstrap&sektion=8&format=html

```
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
```

然后，将 Linux 系统输出的最后一列中的所有文件复制到 FreeBSD 系统的 **/compat/linux** 中。复制完毕后，为第一列中的名字创建符号链接。

这个例子将使得在 FreeBSD 系统上出现以下文件:

```
/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
```

如果 Linux 共享库已经存在，其主要修订号与 `ldd` 输出的第一列相匹配，则不需要将其复制到最后一列命名的文件中，因为现有的库应该可以工作。建议复制新版本的共享库。旧的可以删除——只要符号链接指向新的共享库。

例如，这些库已经存在于 FreeBSD 系统中：

```
/compat/linux/lib/libc.so.4.6.27
/compat/linux/lib/libc.so.4 -> libc.so.4.6.27
```

`ldd` 表示一个二进制文件需要一个更高的版本：

```
libc.so.4 (DLL Jump 4.5pl26) -> libc.so.4.6.29
```

由于现有库的最后一位数字只差了一两个版本，所以程序应该仍然可以使用稍旧的版本。然而，用较新的版本替换现有的 **libc.so** 是稳妥的。

```
/compat/linux/lib/libc.so.4.6.29
/compat/linux/lib/libc.so.4 -> libc.so.4.6.29
```

一般来说，只有在 FreeBSD 上安装 Linux 程序的前几次，才需要寻找 Linux 二进制文件所依赖的共享库。一段时间后，系统中就会有足够的 Linux 共享库，从而能够运行新安装的 Linux 二进制文件，而不需要任何额外的工作。

12.4.2.Linux ELF 二进制文件的 brand

FreeBSD 内核使用几种方法来确定要执行的二进制文件是否是 Linux 的：它检查 ELF 文件头中的 brand，寻找已知的 ELF 解释器路径，并检查 ELF 注释；最后，默认情况下，无 brand 的 ELF 可执行文件被假定为 Linux。

如果所有这些方法都失败了，试图执行该二进制文件可能会导致错误信息：

```
% ./my-linux-elf-binary
```

输出结果应类似于下面的内容：

```
ELF binary type not known
Abort
```

为了帮助 FreeBSD 内核区分 FreeBSD ELF 二进制文件和 Linux 二进制文件，可以使用 brandelf(1)[698]：

```
% brandelf -t Linux my-linux-elf-binary
```

12.4.3.安装基于 RPM 的 Linux 应用程序

要安装基于 RPM 的 Linux 应用程序，首先要安装软件包或 port archivers/rpm4[699]。安装完毕后，`root` 用户可以执行这个命令来安装 **.rpm**：

```
# cd /compat/linux
# rpm2cpio < /path/to/linux.archive.rpm | cpio -id
```

如果有必要，使用 `brandelf` 处理已安装的 ELF 二进制文件。请注意，这将妨碍卸载的彻底性。

12.4.4.配置主机名解析器

如果 DNS 不工作或出现这个错误：

```
resolv+: "bind" is an invalid keyword resolv+:
"hosts" is an invalid keyword
```

配置 **/compat/linux/etc/host.conf** 如下：

```
order hosts, bind
multi on
```

[698] https://www.freebsd.org/cgi/man.cgi?query=brandelf&sektion=1&format=html

[699] https://cgit.freebsd.org/ports/tree/archivers/rpm4/

这规定了首先搜索 **/etc/hosts**，再搜索 DNS。当 **/compat/linux/etc/host.conf** 不存在时，Linux 软件使用主机上的 **/etc/host.conf** 文件并抱怨 FreeBSD 缺少该文件。如果没有使用 **/etc/resolv.conf** 配置域名服务器，则卸载 `bind`。

12.4.5.补充说明

关于如何与 Linux® 进行二进制兼容的更多信息，可以在 FreeBSD 的 Linux 仿真[700]这篇文章中找到。

[700] https://docs.freebsd.org/en/articles/linux-emulation/

第十三章 WINE

13.1.概述

WINE[701] 是 Wine Is Not an Emulator（Wine 不是一个模拟器）的缩写，在技术上是一个软件转换层。它能够在 FreeBSD（和其他）系统上安装和运行一些为 Windows® 编写的软件。

它通过拦截系统调用，或从软件到操作系统的请求，并将它们从 Windows® 调用翻译成 FreeBSD 所理解的调用。它还会根据需要将一切响应翻译成 Windows® 软件所期望的内容。因此在某些方面，它 模拟了 Windows® 环境，因为它提供了许多 Windows® 应用程序所期望的资源。

然而，它并不是传统意义上的仿真器。许多这样的解决方案是通过使用软件进程代替硬件来构建整个其他的计算机。虚拟化（如 port emulators/qemu[702] 提供的）就是以这种方式运作。这种方法的好处之一是能够将有关的操作系统的完整版本安装到模拟器上。对应用程序来说，这意味着看起来的环境与真实的机器没有任何区别，而且很有可能一切都能在上面运行。这种方法的缺点是，作为模拟硬件的软件在本质上比实际的硬件要慢。计算机内置的软件（称为 客户机）需要真正的机器（主机）的资源，并且只要它在运行，就会抓住这些资源。

另一方面，WINE 项目对系统资源的占用要轻得多。它会实时翻译系统调用，所以尽管它很难像真正的 Windows® 电脑一样快，但却非常接近。另一方面，WINE 正试图在所有不同的系统调用和其他需要支持的功能方面跟上前进的脚步。因此，在 WINE 上，可能会有一些应用程序不能像预期的那样工作，或者根本就不能工作，或者甚至一开始就不能安装。

最后，WINE 提供了另一种尝试让特定的 Windows® 软件程序在 FreeBSD 上运行的选择。它总是可以作为首选，如果成功的话，它可以提供良好的体验，而不会不必要地耗尽 FreeBSD 主机的系统资源。

本章将介绍：

- 如何在 FreeBSD 系统上安装 WINE：
- WINE 是如何运行的，以及它与其他替代方案如虚拟化有什么不同。
- 如何对 WINE 进行微调以满足某些应用程序的特殊需要。

[701] https://www.winehq.org/

[702] https://cgit.freebsd.org/ports/tree/emulators/qemu/pkg-descr

- 如何为 WINE 安装图形界面辅助工具。
- 在 FreeBSD 上的常见技巧和解决方案。
- 在 FreeBSD 上的 WINE 在多用户环境方面的考虑。

在阅读本章之前，有必要了解一下：

- 理解 UNIX® 和 FreeBSD 的基础知识[703]。
- 知道如何安装 FreeBSD[704]。
- 知道如何设置网络连接[705]。
- 知道如何安装额外的第三方软件[706]。

13.2.WINE 概述和概念

WINE 是一个复杂的系统，因此在 FreeBSD 系统上运行它之前，值得了解一下它是什么以及它是如何工作的。

13.2.1.什么是 WINE ？

正如本章简介中提到的，WINE 是一个兼容层，允许 Windows® 应用程序在其他操作系统上运行。在理论上，这意味着这些程序应该可以在 FreeBSD、macOS 和 Android 等系统上运行。

当 WINE 运行一个 Windows® 可执行文件时，会发生两件事：

- 首先，WINE 实现了一个模仿各种版本的 Windows® 的环境。例如，如果一个应用程序要求访问一个资源，如内存，WINE 有一个内存接口，看起来和实际上（就应用程序而言）都像 Windows®。
- 然后，假设该应用程序使用该接口，WINE 就会接收到对内存空间的传入请求，并将其转化为与主机系统兼容的东西。以同样的方式，当应用程序检索该数据时，WINE 帮助从主机系统中获取该数据并将其传回给 Windows® 应用程序。

13.2.2.WINE 和 FreeBSD 系统

在 FreeBSD 系统上安装 WINE 将需要一些不同的组件：

- 用于运行 Windows® 可执行文件、配置 WINE 子系统或编译支持 WINE 的程序等任务的 FreeBSD 应用程序。
- 大量用于实现 Windows® 核心功能的库（例如 **/lib/wine/api-ms-core-memory-l1-1.dll.so**，它是上述内存接口的一部分）。

[703] https://docs.freebsd.org/en/books/handbook/basics/index.html#basics
[704] https://docs.freebsd.org/en/books/handbook/bsdinstall/index.html#bsdinstall
[705] https://docs.freebsd.org/en/books/handbook/advanced-networking/index.html#advanced-networking
[706] https://docs.freebsd.org/en/books/handbook/ports/index.html#ports

- 一些 Windows® 可执行文件，它们是（或模仿）常见的实用程序（例如 **/lib/wine/notepad.exe.so**，它提供标准的 Windows® 文本编辑器）。
- 额外的 Windows® 组件，特别是字体（如 Tahoma 字体，它存储在安装根目录的 **share/wine/fonts/tahoma.ttf** 中）。

13.2.3.WINE 中图形与文本模式/终端程序的对比

作为一个终端工具是“头等公民”的操作系统，我们很自然地认为 WINE 将包含对文本模式程序的广泛支持。然而，大多数 Windows® 的应用程序，特别是最流行的，都是以图形用户界面（GUI）为设计理念的。因此，WINE 的实用程序被默认设计为启动图形程序。

然而，有三种方法可以运行这些所谓的控制台用户接口（CUI）程序：

- *裸流*方法将直接显示输出到标准输出。
- *wineconsole* 工具可以与 *用户* 或 *curses* 一起使用，以利用 WINE 系统为 CUI 程序提供的一些增强功能。

这些方法在 WINE Wiki[707] 上有更详细的描述。

13.2.4.WINE 衍生项目

WINE 本身是一个成熟的开源项目，所以它被用作更复杂的解决方案的基础也就不足为奇。

13.2.4.1.商业 WINE 的实施

许多公司已经采用了 WINE，并使其成为他们自己专有产品的核心（WINE 的 LGPL 许可证允许这样做）。如下是其中最著名的两个：

- Codeweavers CrossOver

这个解决方案提供了一个简化的“一键式”WINE 安装，它包含了额外的增强和优化功能（尽管该公司将其中许多功能回馈给 WINE 项目的上游）。Codeweavers 的一个重点领域是使最流行的应用程序能够顺利安装和运行。

虽然该公司曾经为他们的 CrossOver 解决方案制作了一个原生的 FreeBSD 版本，但它似乎早已被放弃了。虽然一些资源（如专门的论坛[708]）仍然存在，但它们也有一段时间没有活动了。

- Steam Proton

游戏公司 Steam 也使用 WINE 使 Windows® 游戏能够在其他系统上安装和运行。它的主要目标是基于 Linux 的系统，尽管对 macOS 也存在一些支持。

虽然 Steam 不提供原生的 FreeBSD 客户端，但有一些使用 FreeBSD 的 Linux 兼容层来运行 Linux® 客户端的选择。

[707] https://wiki.winehq.org/Wine_User's_Guide#Text_mode_programs_.28CUI:_Console_User_Interface.29

[708] https://www.codeweavers.com/compatibility/crossover/forum/freebsd

13.2.4.2.WINE 配套程序

除了专有的产品之外，其他项目也发布了旨在与标准的、开源的 WINE 版本协同工作的应用程序。这些程序的目标包括使安装更容易，以及提供方便的方法来安装流行的软件。

这些解决方案将在后面关于 GUI 前端[709]的章节中详细介绍，包括以下这些：

- winetricks
- Homura

13.2.5.WINE 的替代方案

对于 FreeBSD 用户，使用 WINE 的一些替代方案如下：

- 双系统。一个直接的选择是在该操作系统上原生地运行所需的 Windows® 应用程序。这当然意味着要在现有的 FreeBSD 上启动 Windows®，所以如果需要同时访问两个系统的程序，这个方法是不可行的。
- 虚拟机。虚拟机（VM），正如本章前面提到的，是模拟全套硬件的软件进程，在其上可以安装和运行其他操作系统（包括 Windows®）。现代工具使虚拟机易于创建和管理，但这种方法是有代价的。主机系统的很大一部分资源必须分配给每个虚拟机，而且只要虚拟机在运行，主机就不能回收这些资源。虚拟机管理器的几个例子包括开源解决方案 qemu、bhyve 和 VirtualBox。更多细节请参见虚拟化[710]一章。
- 远程访问。像许多其他类似 UNIX® 的系统一样，FreeBSD 可以运行各种应用程序，使用户能够远程访问 Windows® 计算机并使用他们的程序或数据。除了像 xrdp 这样连接到标准的 Windows® 远程桌面协议的客户端之外，还可以使用其他开源标准，如 vnc（只要另一端有兼容的服务器）。

13.3.在 FreeBSD 上安装 WINE

可以通过 pkg 工具安装 WINE，也可以通过 ports 编译安装。

13.3.1.WINE 的安装需求

在安装 WINE 本身之前，安装以下的先决软件是很有用的。

- 图形化用户界面

大多数 Windows® 程序都希望有一个图形化的用户界面。如果在安装 WINE 时没有图形用户界面，它的依赖关系将包括 Wayland 混成器，因此图形用户界面将与 WINE 一起安装。但在安装 WINE 之前，安装、配置并正确运行所选择的 GUI 是很有用的。

- wine-gecko

[709] https://docs.freebsd.org/en/books/handbook/book/#wine-management-guis
[710] https://docs.freebsd.org/en/books/handbook/book/#virtualization

长期以来，Windows® 操作系统预装了一个默认的网络浏览器：Internet Explorer。因此，一些应用程序在工作时假定总是有能够显示网页的东西。为了提供这种功能，WINE 层包括一个使用 Mozilla 项目的 Gecko 引擎的网络浏览器组件。当首次启动 WINE 时，它将提供下载和安装，而且用户可能希望它这样做（这些将在后面的章节中介绍）。但他们也可以在安装 WINE 之前安装它，或者在安装 WINE 的同时进行安装。

用下面的方法安装这个软件包：

```
# pkg install wine-gecko
```

或者，用下面的方法编译 port：

```
# cd /usr/ports/emulator/wine-gecko
# make install
```

- wine-mono

这个 port 安装了 MONO 框架，一个微软.NET 的开源实现。在 WINE 的安装中包括这个，将使任何用.NET 编写的应用程序更有可能在系统上安装和运行。

要安装该软件包：

```
# pkg install win-mono
```

从 ports 进行编译：

```
# cd /usr/ports/emulator/win-mono
# make install
```

13.3.2.通过 FreeBSD 软件包安装 WINE

有了这些先决条件，用下面的命令通过软件包来安装 WINE：

```
# pkg install wine
```

或者用下面的命令从源代码编译 WINE 子系统：

```
# cd /usr/ports/emulator/wine
# make install
```

13.3.3.WINE 安装中的 32 位与 64 位的关注点

像大多数软件一样，Windows® 应用程序从旧的 32 位架构升级到 64 位。而且大多数最新的软件都是为 64 位操作系统编写的，尽管现代操作系统有时也可以继续运行旧的 32 位程序。FreeBSD 也不例外，从 5.x 系列开始就有对 64 位的支持。

然而，使用默认情况下不再支持的旧软件是模拟器的一个常见用途，用户通常转向 WINE 来玩游戏和使用其他在现代硬件上不能正常运行的程序。幸运的是，FreeBSD 可以支持所有这三种情况：

- 在现代的 64 位机器上，想要运行 64 位的 Windows® 软件，只需安装上面几节中提到的 ports。ports 系统会自动安装 64 位版本。
- 另外，用户可能有一台旧的 32 位机器，他们不想用原来的、现在不支持的软件运行。他们可以安装 32 位（i386）版本的 FreeBSD，然后再安装上面几节中的 port。

13.4.在 FreeBSD 上运行第一个 WINE 程序

现在 WINE 已经安装好了，下一步是通过运行一个简单的程序来试试。简单的方法是下载一个独立的应用程序，也就是说，可以简单地解压和运行程序而无需任何复杂的安装过程。

所谓的“可移植”版本的应用程序是这种测试的好选择，只用一个可执行文件运行的程序也是如此。

13.4.1.从命令行运行一个程序

有两种不同的方法可以从终端启动一个 Windows 程序。第一种，也是最直接的一种，是导航到包含程序可执行文件（**.EXE**）的目录，然后执行以下命令：

```
% wine program.exe
```

对于需要命令行参数的应用程序，像往常一样在可执行文件后添加参数：

```
% wine program2.exe -file file.txt
```

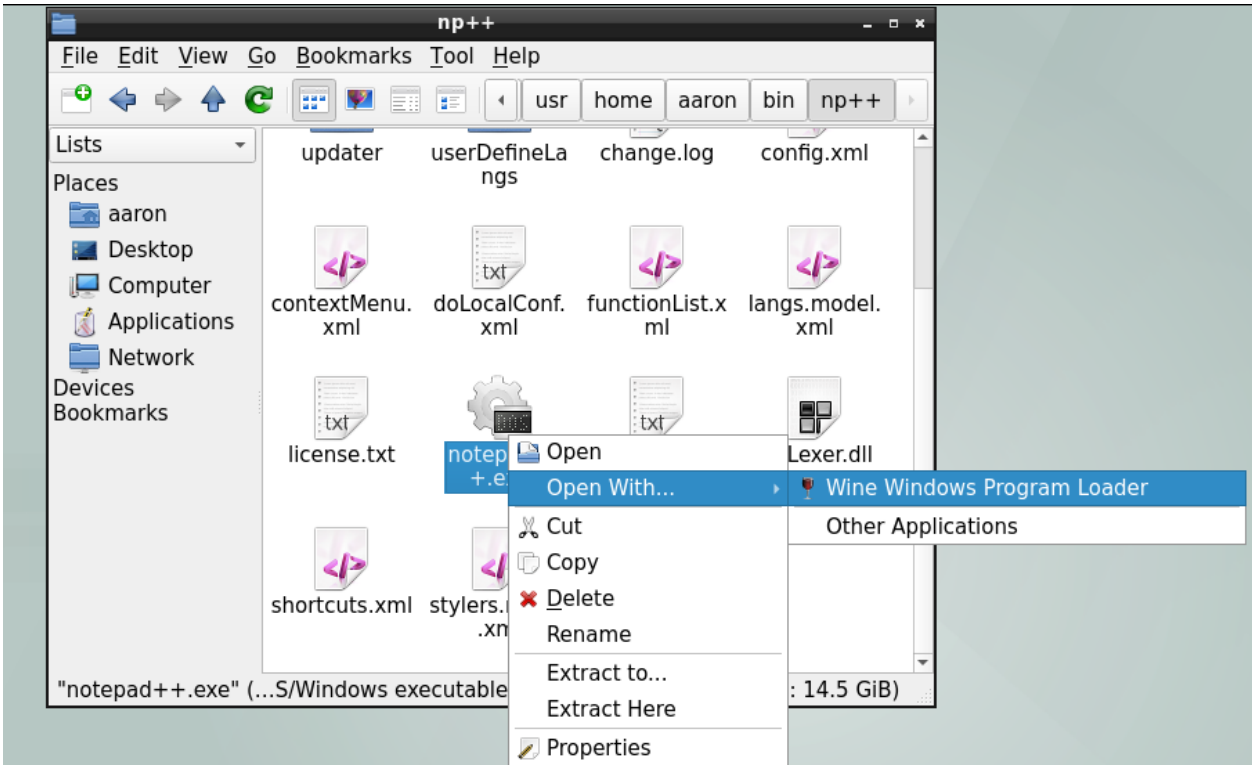
另外，提供可执行文件的完整路径，以便在脚本中使用它，例如：

```
% wine /home/user/bin/program.exe
```

13.4.2.使用 GUI 运行一个程序

安装后，图形 shell 应更新为与 Windows 可执行文件（**.EXE**）有关的新关联。现在可以使用文件管理器浏览系统，并以与其他文件和程序相同的方式启动 Windows 应用程序（单击或双击，取决于桌面的设置）。

在大多数桌面上，通过右键点击文件，并在上下文菜单中寻找打开文件的条目，来检查确保这种关联是正确的。其中一个选项（希望是默认选项）将是与 **Wine Windows 程序加载器**，如下面的截图所示：



如果程序不能按预期运行，请尝试从命令行启动它，并查看终端显示的所有信息以排除故障。

如果 WINE 在安装后不是 **.EXE** 文件的默认应用程序，请检查当前桌面环境、图形化 shell 或文件管理器中该扩展名的 MIME 关联。

13.5.配置 WINE 安装

在了解了什么是 WINE 以及它如何在高层次上工作之后，在 FreeBSD 上有效使用它的下一步就是熟悉它的配置。下面几节将介绍 *WINE Prefixe* 的关键概念，并说明如何用它来控制通过 WINE 运行的应用程序的行为。

13.5.1.WINE Prefixe

WINE Prefixe 是一个目录，通常位于 **$HOME/.wine** 的默认位置之下，尽管它可以位于其他地方。WINE Prefixe 是一组配置和支持文件，由 wine 用来配置和运行特定应用程序所需的 Windows® 环境。默认情况下，一个全新的 WINE 安装在用户首次启动时将创建以下结构：

- **.update-timestamp**：包含文件 **/usr/share/wine/wine.inf** 的最后修改日期。它被 WINE 用来判断一个 WINE Prefixe 是否过时，并在需要时自动更新它。
- **dosdevices/**：包含 Windows® 资源与主机 FreeBSD 系统上的资源的映射信息。例如，在新的 WINE 安装之后，这里应该至少包含两个条目，可以使用 Windows® 风格的驱动器字母来访问 FreeBSD 的文件系统。
  - **c:@**：与下面说明的 **drive_c** 的链接。
  - **z:@**：到系统根目录的链接。
- **drive_c/**：模拟 Windows® 系统的主驱动器（即 **C:**）。它包含一个目录结构和相关文件，与标准的 Windows® 系统一样。一个新的 WINE Prefixe 将包含 Windows® 10 目录，如 *Users* 和 *Windows*，其中包含操作系统本身。此外，安装在 WINE Prefixe 中的应用程序将位于 *Program Files* 或 *Program Files (x86)* 中，这取决于其架构。
- **system.reg**：这个注册表文件包含了 Windows® 安装的信息，在 WINE 的情况下，它是 **drive_c** 中的环境。
- **user.reg**：这个注册表文件包含了当前用户的个人配置，由各种软件或通过使用注册表编辑器来完成。
- **userdef.reg**：这个注册表文件是新创建的用户的默认配置集。

13.5.2.创建和使用 WINE Prefixe

虽然 WINE 会在用户的 **$HOME/.wine/** 中创建一个默认 WINE Prefixe，但也可以设置多个 WINE Prefixe。这样做有几个原因：

- 最常见的原因是根据有关软件的兼容性需要，模拟不同版本的 Windows®。
- 此外，经常会遇到一些在默认环境下不能正常工作的软件，需要进行特殊的配置。将这些软件隔离在自己的自定义 WINE Prefixe 中是很有用的，这样的改变不会影响其他应用程序。
- 同样，为了评估一个应用程序的兼容性，将 default 或 “main” Prefixe 复制到一个单独的 “testing” Prefixe 中，可以减少出错的可能性。

从终端创建一个 WINE Prefixe 需要使用以下命令：

```
% WINEPREFIX="/home/username/.win-new" winecfg
```

这将运行 `winecfg` 程序，该程序可用于配置 WINE Prefixe（在后面的章节中会有更多介绍）。但通过为 `WINEPREFIX` 环境变量提供一个目录路径值，如果一个 WINE Prefixe 还不存在，就会在该位置创建一个新的 WINE Prefixe：

向 wine 程序提供同样的变量，同样会导致所选程序以指定的 WINE Prefixe 运行：

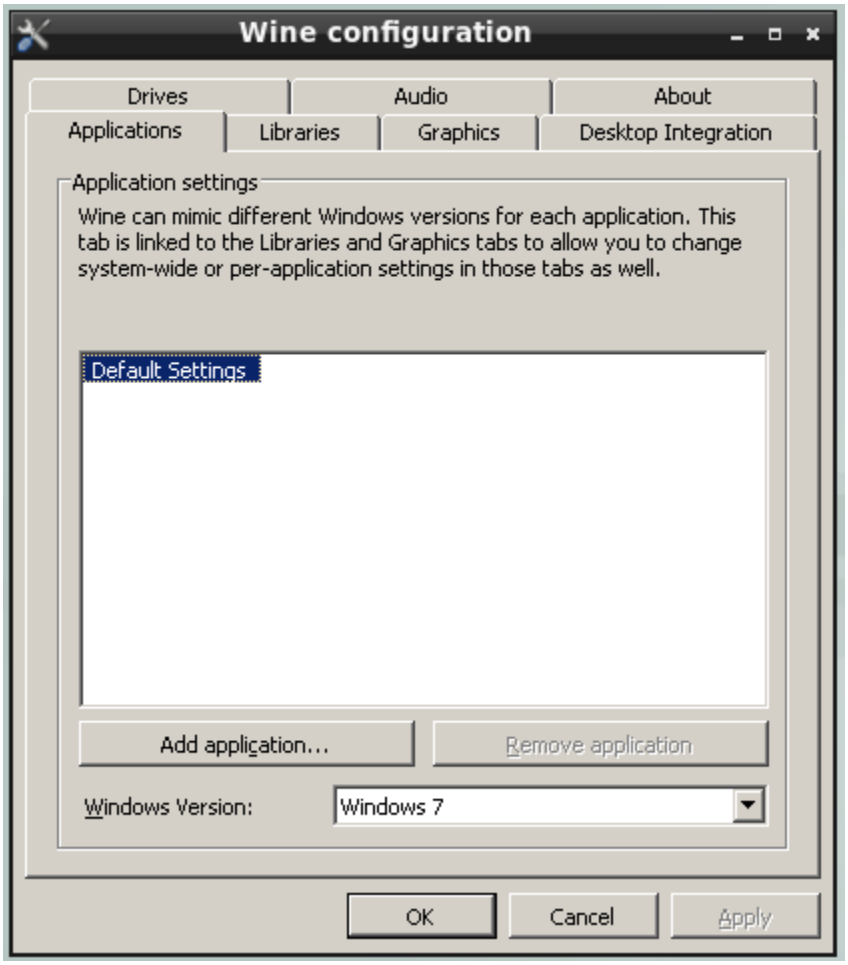
```
% WINEPREFIX="/home/username/.win-new" wine program.exe
```

13.5.3.用 winecfg 配置 WINE Prefixe

如上所述，WINE 包括一个名为 `winecfg` 的工具，可以在图形用户界面中配置 WINE Prefixe。它包含各种功能，在下面的章节中会详细介绍。当 `winecfg` 从一个 WINE Prefixe 中运行时，或者在 `WINEPREFIX` 变量中提供一个 WINE Prefixe 的位置时，它就可以对所选的 Prefixes 进行配置，如以下各节所述。

在 *Applications* 选项卡上进行的选择将影响在 *Libraries* 和 *Graphics* 选项卡上的更改范围，这些更改将被限制在所选的应用程序中。更多细节请参见 WINE Wiki 中关于使用 Winecfg[711] 的部分。

13.5.3.1.Applications



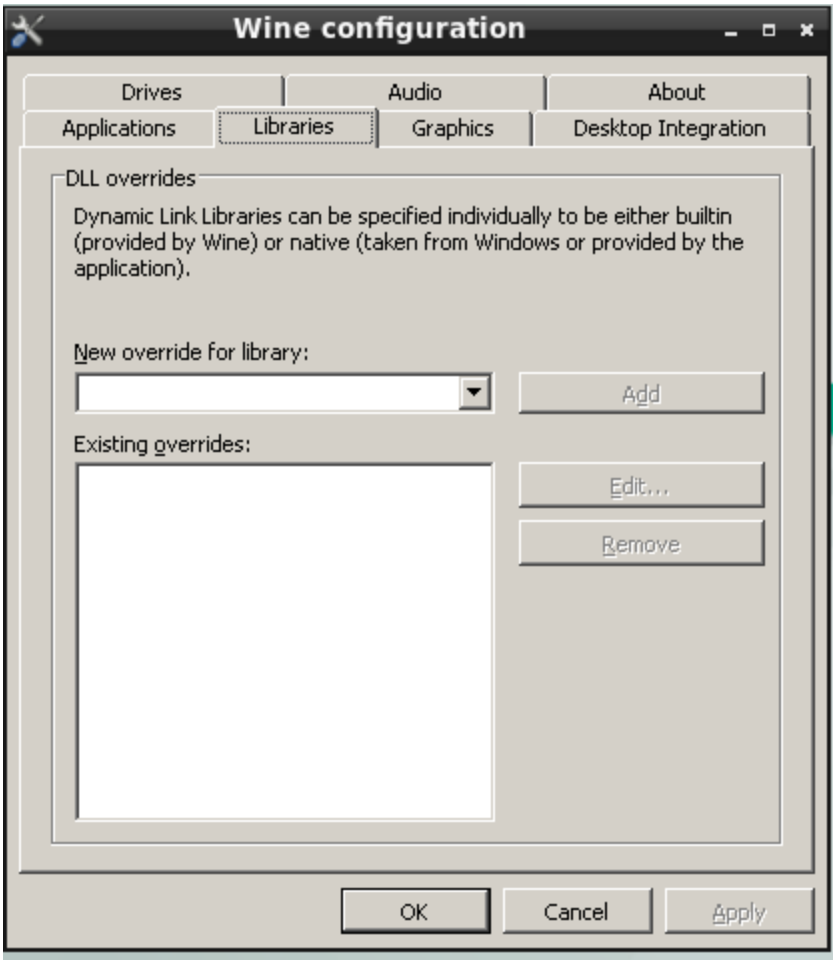
[711] https://wiki.winehq.org/Wine_User's_Guide#Using_Winecfg

Applications 包含了使程序与特定版本的 Windows® 相关联的控制。第一次启动时，应用程序设置部分将包含一个条目：*Default Settings*。这相当于 WINE Prefixe 的所有默认配置，（正如禁用 *Remove application* 按钮所暗示的）不能被删除。

但可以通过以下过程添加额外的应用程序：

1. 单击 *Add application* 按钮。
2. 使用提供的对话框选择所需的程序的可执行文件。
3. 选择要与所选程序一起使用的 Windows® 版本。

13.5.3.2.Libraries



WINE 提供了一组开源的库文件作为其发行的一部分，这些库文件提供了与 Windows® 对应的相同功能。然而，正如本章前面所提到的，WINE 项目一直在努力跟上这些库的新更新。因此，与 WINE 一起发行的版本可能缺少最新 Windows® 程序所期望的功能。

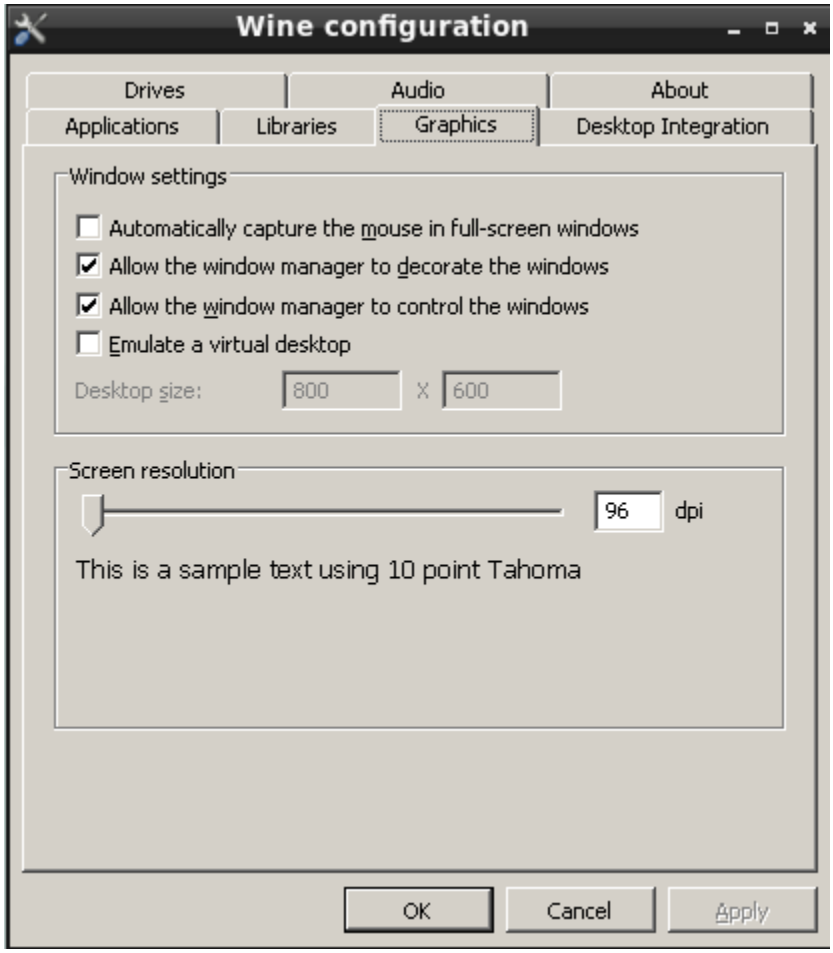
然而，`winecfg` 可以为内置的库指定重写，特别是在安装 FreeBSD 的同一台机器上有 Windows® 的版

本。对于每一个要被覆盖的库，请执行以下操作：

1. 打开 *New override for library* 下拉菜单，选择要替换的库。
2. 点击 *Add* 按钮。
3. 新的覆盖将出现在 *Existing overrides* 列表中，注意括号中的 *native*、*builtin* 指定。
4. 点击选择该库。
5. 点击 *Edit* 按钮。
6. 使用提供的对话框选择一个相应的库来代替内置库。

   请确保选择一个真正与内置库对应的文件，否则可能会有意想不到的行为。

13.5.3.3.Graphics

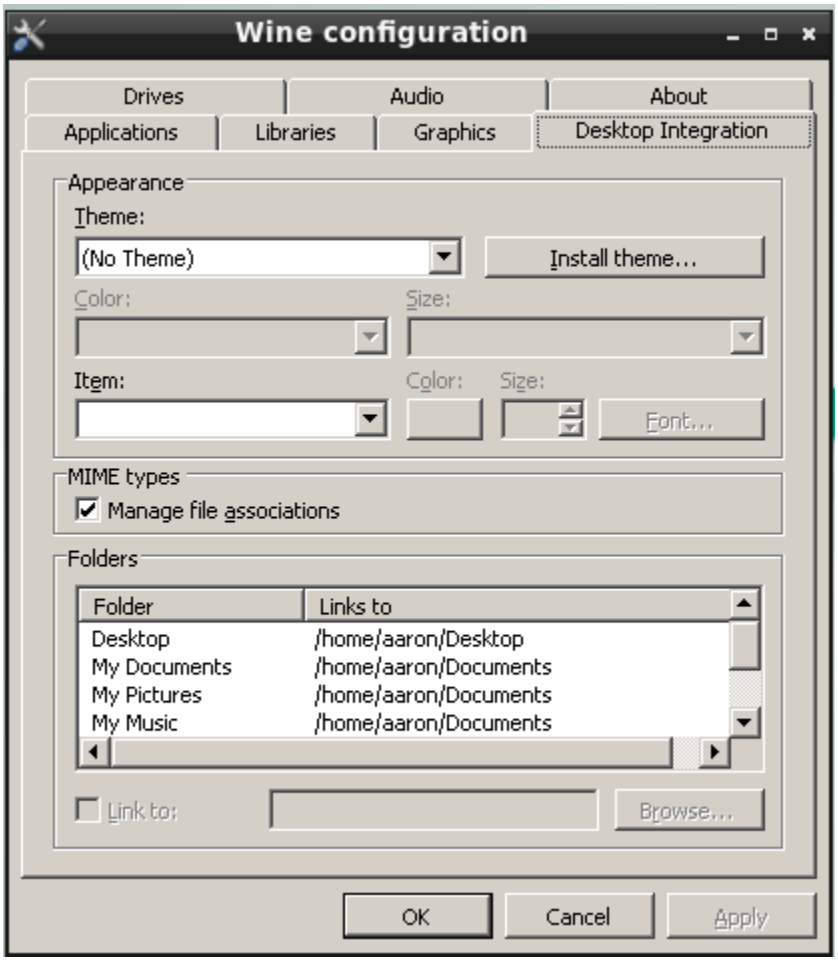


图形选项卡提供了一些选项，使通过 WINE 运行的程序的窗口能够在 FreeBSD 中顺利运行：

- 当窗口为全屏时自动捕捉鼠标。

- 允许 FreeBSD 的窗口管理器装饰窗口，例如通过 WINE 运行的程序的标题栏。
- 允许窗口管理器为通过 WINE 运行的程序控制窗口，例如对它们运行调整大小的功能。
- 创建一个模拟的虚拟桌面，所有 WINE 程序将在其中运行。如果选择了这个项目，虚拟桌面的大小可以用桌面大小输入框来指定。
- 为通过 WINE 运行的程序设置屏幕分辨率。

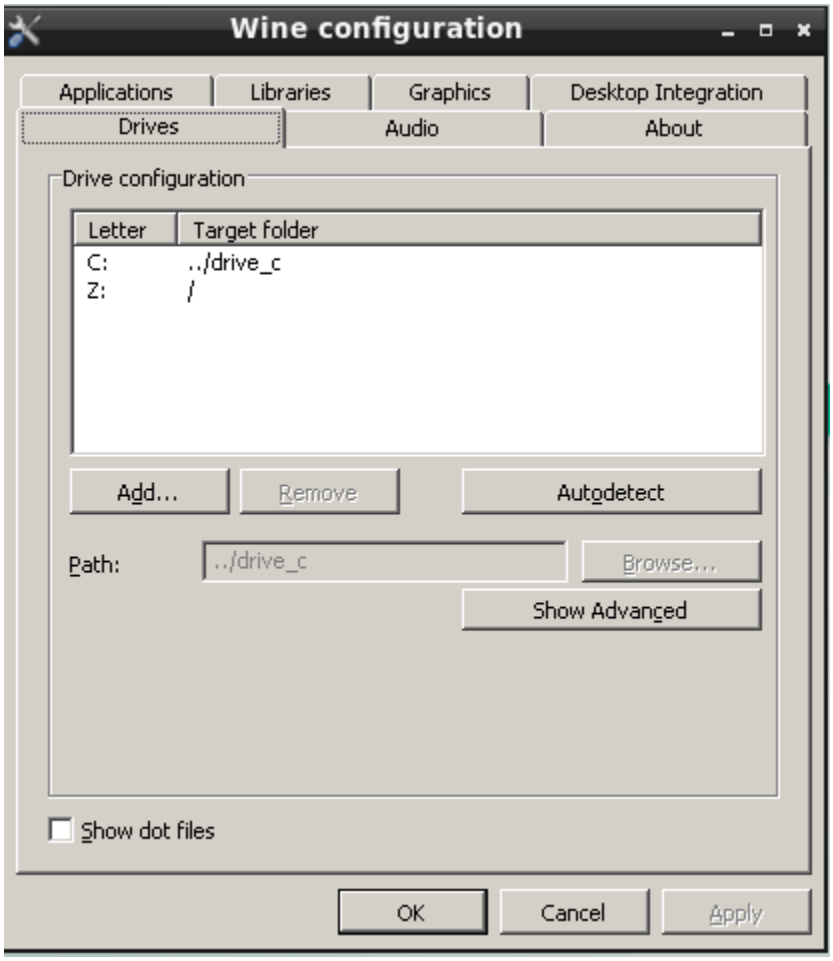
13.5.3.4.Desktop Integration



这个标签允许配置以下项目：

- 通过 WINE 运行的程序要使用的主题和相关的视觉设置。
- WINE 子系统是否应该在内部管理 MIME 类型（用于确定哪个应用程序打开特定的文件类型）。
- FreeBSD 主机系统中的目录与 Windows® 环境中的有用文件夹的映射。要改变现有的关联，选择所需的项目并点击浏览，然后使用提供的对话框选择一个目录。

13.5.3.5.Drives



Drives 选项卡允许将主机 FreeBSD 系统中的目录与 Windows® 环境中的驱动器字母联系起来。这个选项卡中的默认值应该看起来很熟悉，因为它们显示的是当前 WINE Prefixe 中 **dosdevices/** 的内容。通过这个对话框所做的改变将反映在 **dosdevices** 中，在该目录中创建的正确格式的链接将显示在这个标签中。

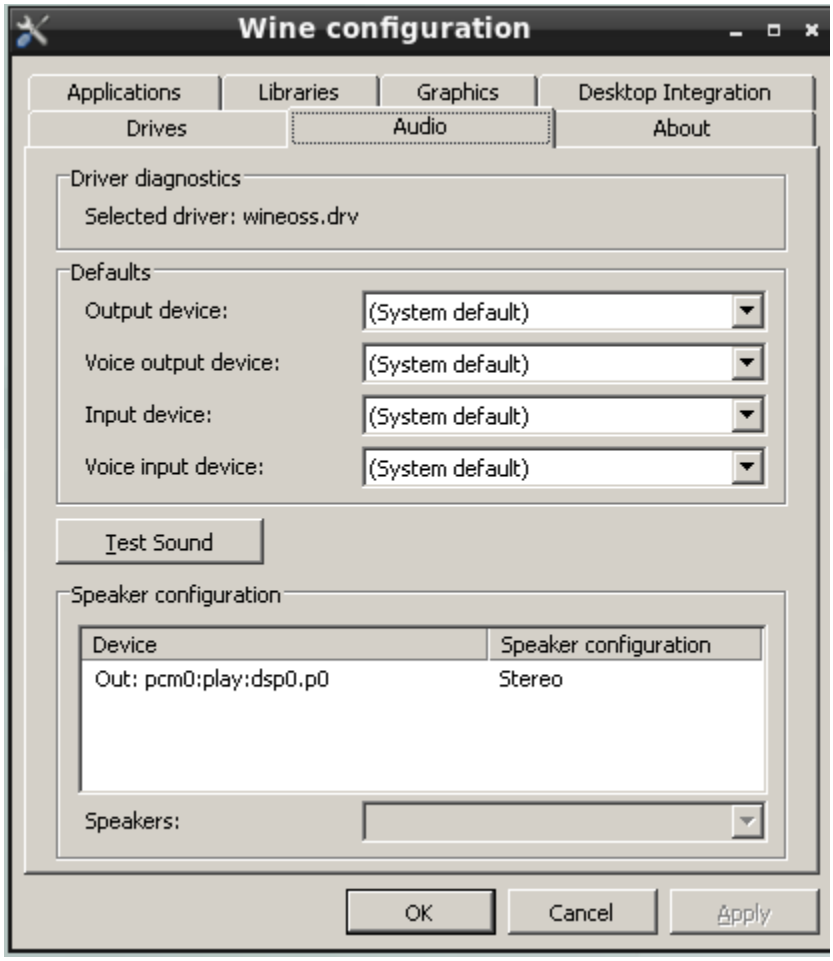
要创建一个新的条目，例如 CD-ROM（挂载在 **/mnt/cdrom**），采取以下步骤：

1. 点击 *Add* 按钮。
2. 在提供的对话框中，选择一个空闲的盘符。
3. 点击 *OK*。
4. 在 *Path* 输入框中输入资源的路径，或者点击 *Browse* 并使用提供的对话框来选择它。

默认情况下，WINE 会自动检测所链接的资源类型，但这可以被手动覆盖。关于高级选项的更多细节，请参见 WINE Wiki 中的章节[712]。

[712] https://wiki.winehq.org/Wine_User's_Guide#Drive_Settings

13.5.3.6.Audio



这个选项卡包含了一些可配置的选项，用于将 Windows® 程序的声音路由到原生的 FreeBSD 声音系统，包括：

- 驱动程序选择
- 默认设备选择
- 声音测试

13.5.3.7.About

最后一个标签包含了关于 WINE 项目的信息，包括一个网站的链接。它还允许输入（完全可选的）用户信息，尽管这些信息不会像其他操作系统那样被发送到任何地方。

13.6.WINE 图形管理用户界面

虽然 WINE 的基本安装包含一个 `winecfg` 的 GUI 配置工具，但它的主要目的只是：严格配置现有的 WINE Prefixe。然而，还有一些更高级的应用程序，它们将协助应用程序的初始安装，以及优化它们的 WINE 环境。下面的章节包括了一些最流行的选择。

13.6.1.Winetricks

`winetricks` 工具是一个跨平台的、通用的 WINE 辅助程序。它不是由 WINE 项目本身开发的，而是由一群贡献者在 Github[713] 上维护。它包含了一些自动的 “recipes”，通过优化设置以及自动获取一些 DLL 库，使普通应用程序在 WINE 上运行。

13.6.1.1.安装 winetricks

要在 FreeBSD 上使用软件包安装 `winetricks`，请使用以下命令（注意 `winetricks` 需要软件包 i386-wine 或 i386-wine-devel，因此不会与其他依赖项一起自动安装）：

```
# pkg install i386-wine winetricks
```

要从源代码编译它，请在终端执行以下命令：

```
# cd /usr/ports/emulators/i386-wine
# make install
# cd /usr/ports/emulators/winetricks
# make install
```

如果需要手动安装，请参考 Github[714] 账户的说明。

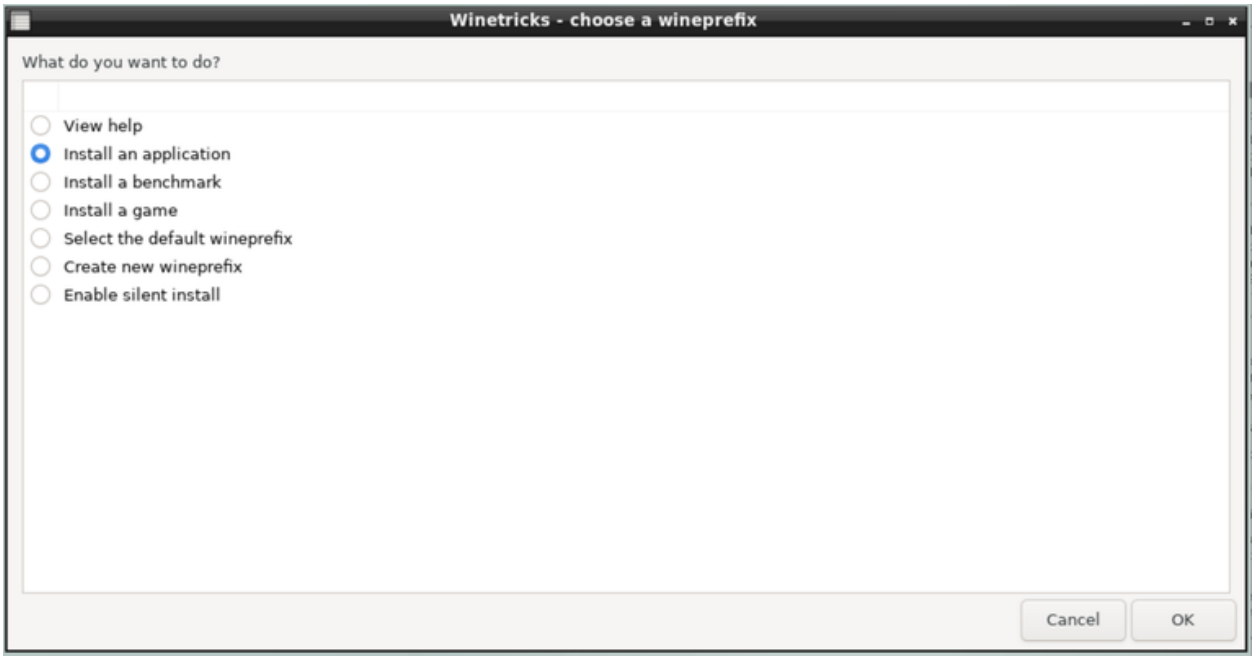
13.6.1.2.使用 winetricks

用以下命令运行 winetricks：

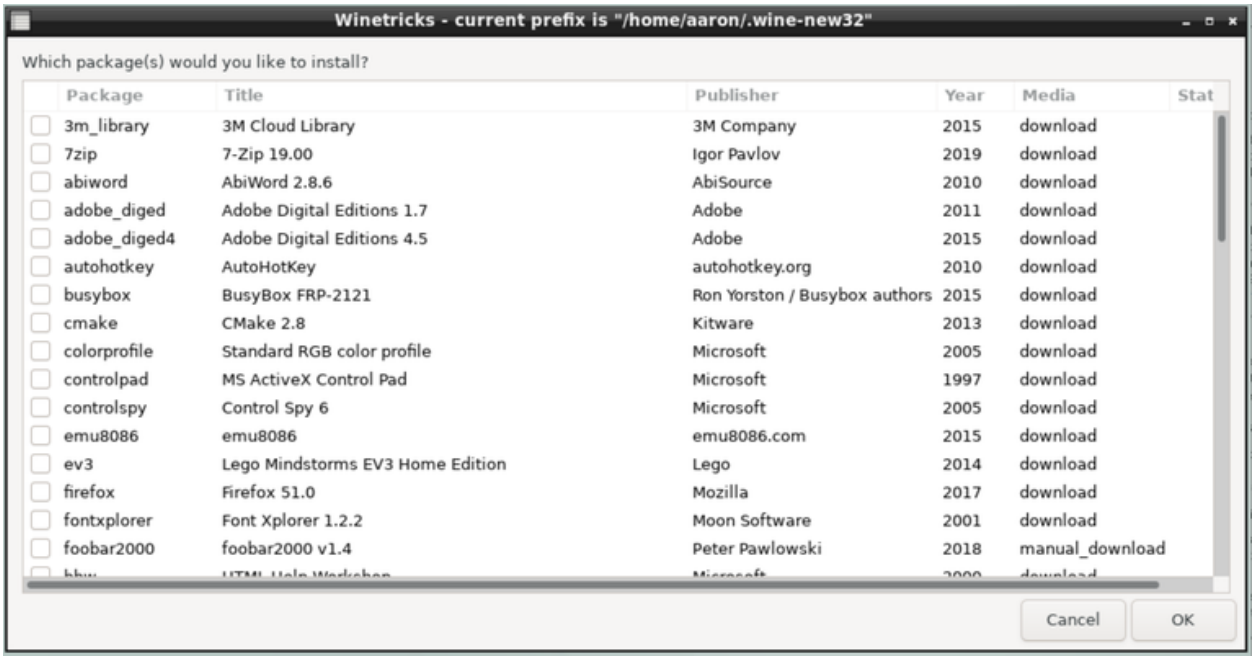
```
% winetricks
```

注意：这应该是在 32 位的 WINE Prefixes 下运行 `winetricks`。启动 `winetricks` 会显示一个有许多选择的窗口，如下所示:

[713] https://github.com/Winetricks/winetricks
[714] https://github.com/Winetricks/winetricks

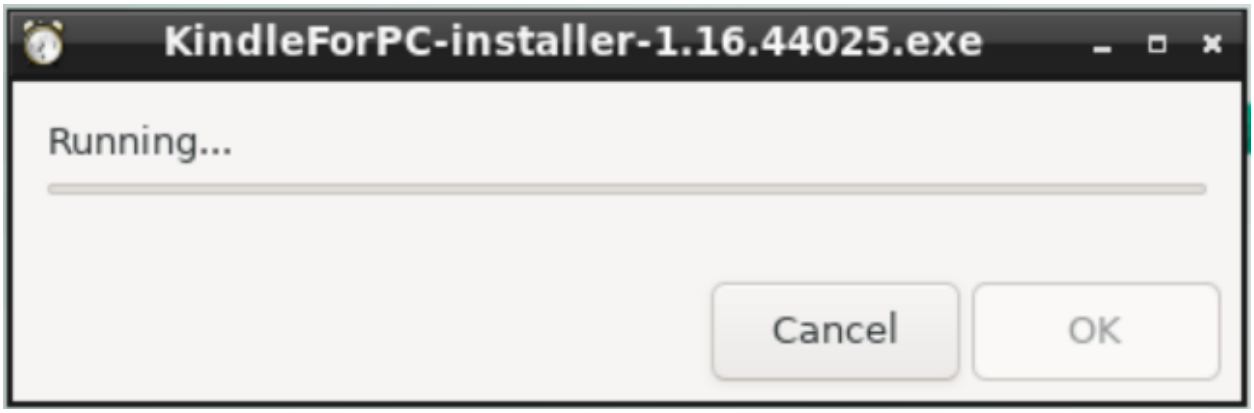


选择 *Install an application* 、*Install a benchmark* 或 *Install a game* 会显示一个支持选项的列表，比如下面的应用程序列表:

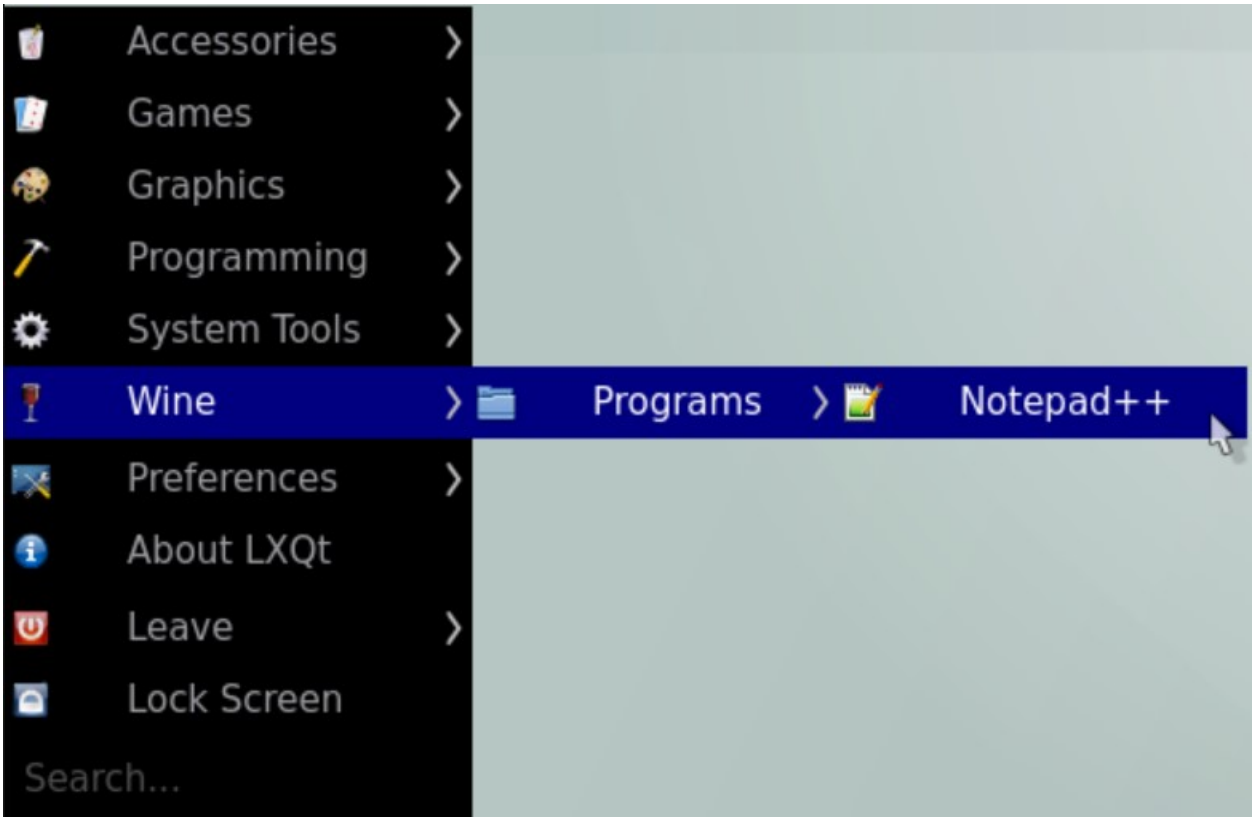


选择一个或多个项目并点击 *OK* 将开始它们的安装过程。最初，一些看似错误的信息可能会出现，但它们实际上是 `winetricks` 在配置 WINE 环境以解决应用程序的已知问题时发出的信息提示:

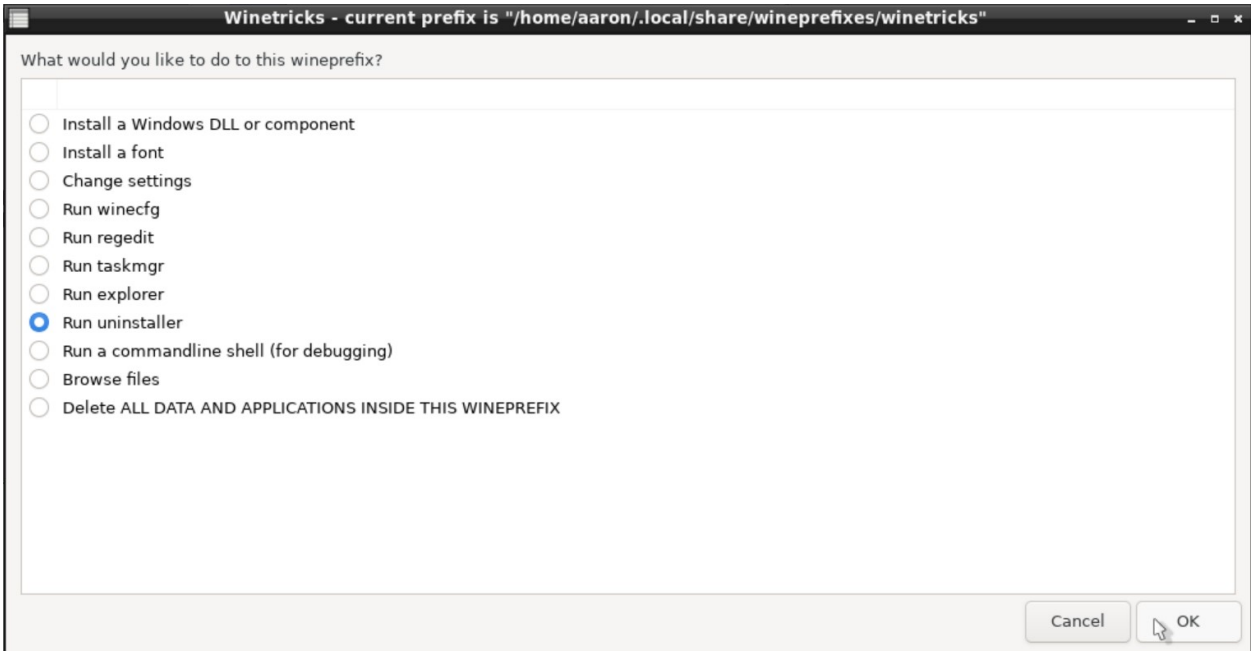
这些被规避后，应用程序的实际安装程序将被运行：



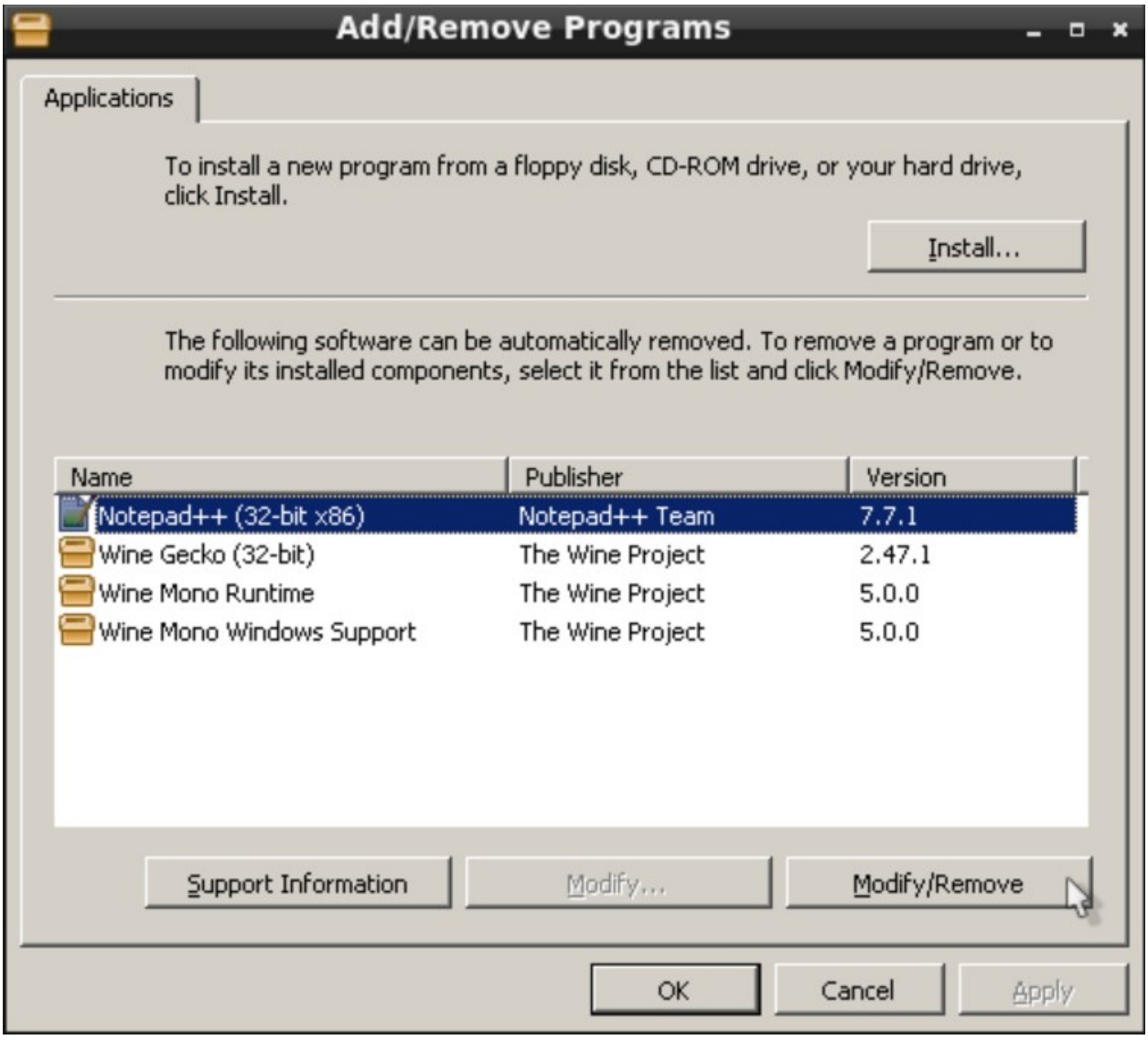
安装完成后，应该可以从桌面环境的标准菜单中找到新的 Windows 应用程序（如下图 LXQT 桌面环境的截图）：



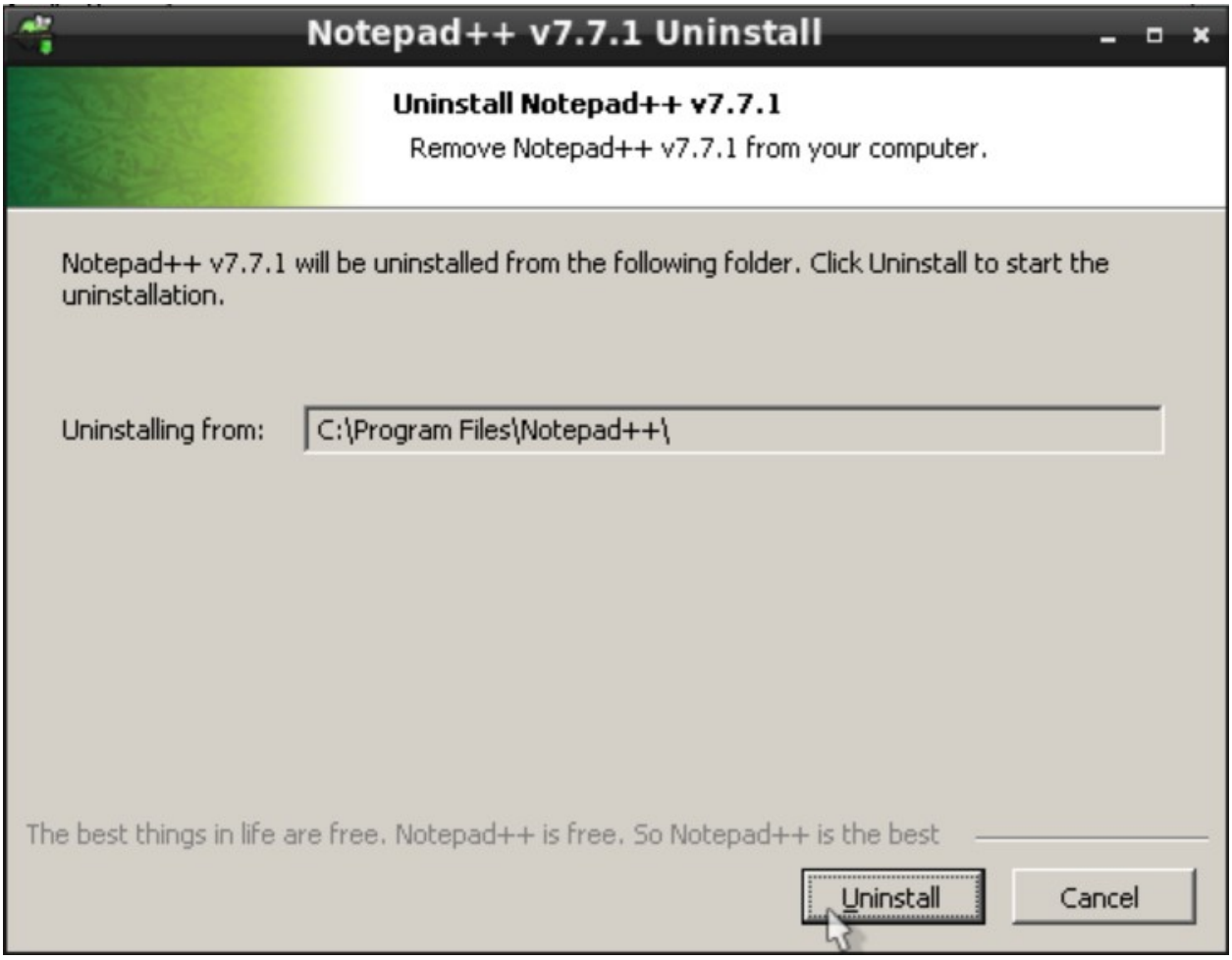
要删除该应用程序，再次运行 `winetricks`，并选择 *Run an uninstaller*。



一个 Windows® 风格的对话框将出现，其中有一个已安装的程序和组件的列表。选择要删除的应用程序，然后点击 *Modify/Remove* 按钮。



这将运行应用程序的内置安装程序，它也应该有卸载的选项：



13.6.2.Homura

Homura 是一个类似于 `winetricks` 的应用程序，尽管它的灵感来自 Linux 的 Lutris[715] 游戏系统。但是，虽然它专注于游戏，但也有一些非游戏的应用程序可以通过 Homura 安装。

13.6.2.1.安装 Homura

要安装 Homura 的软件包，执行以下命令：

```
# pkg install homura
```

Homura 在 FreeBSD ports 系统中是可用的。然而，他不在软件包或 ports 的 *emulators* 分类下，应该在 *games* 部分寻找它：

[715] https://lutris.net/

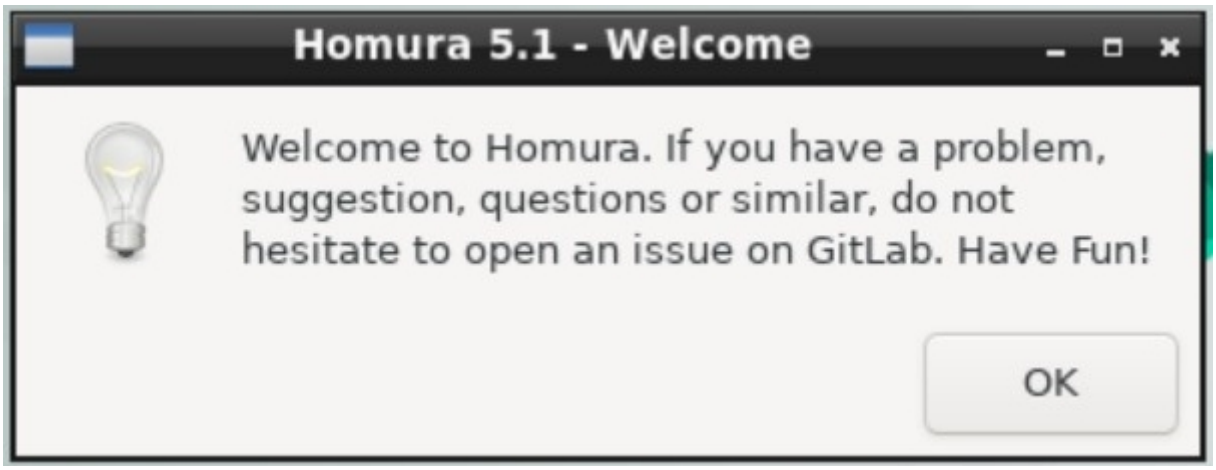
```
# cd /usr/ports/games/homura
# make install
```

13.6.2.2.使用 Homura

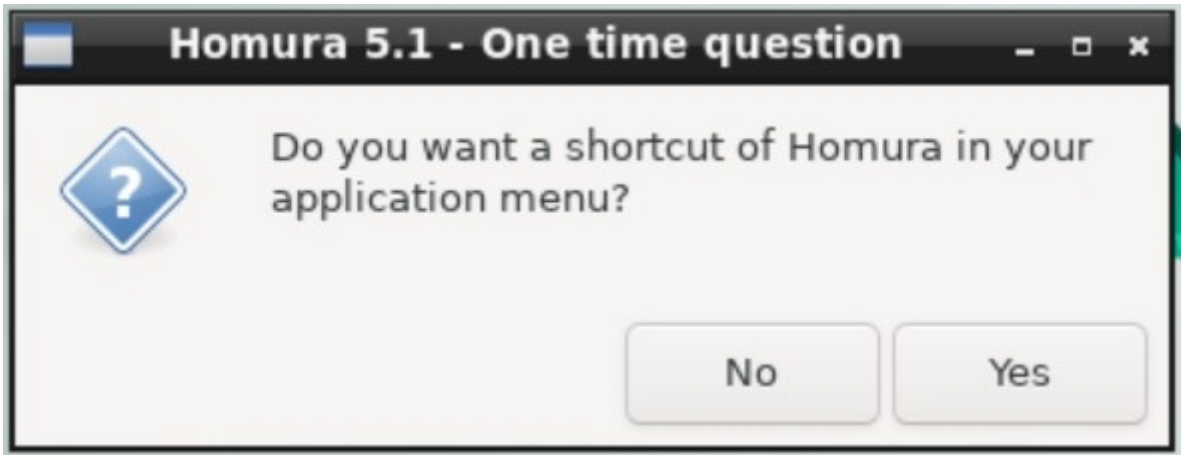
Homura 的使用与 `winetricks` 的使用非常相似。第一次使用它时，从命令行（或桌面环境的运行程序）启动它：

```
% Homura
```

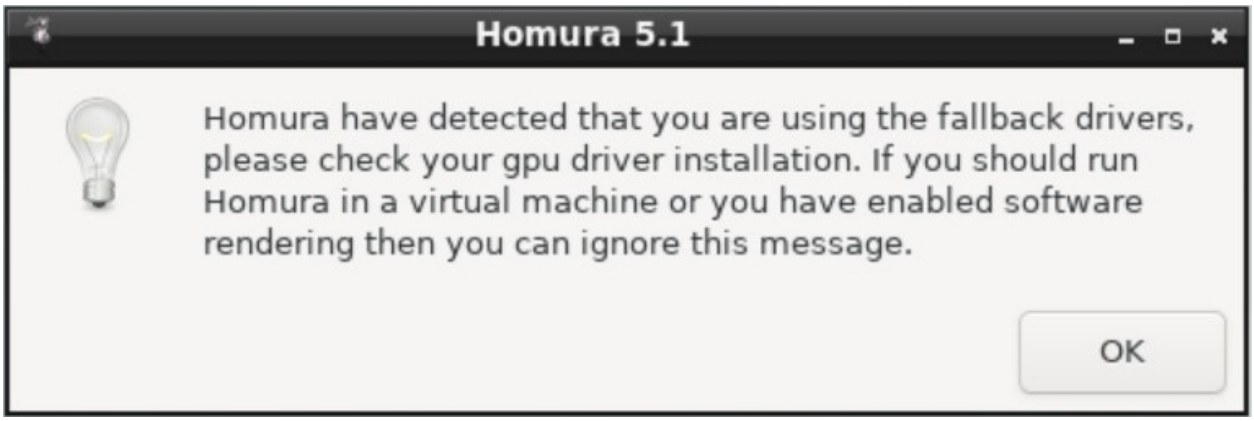
这应该会产生一个友好的欢迎信息。点击 *OK* 继续。



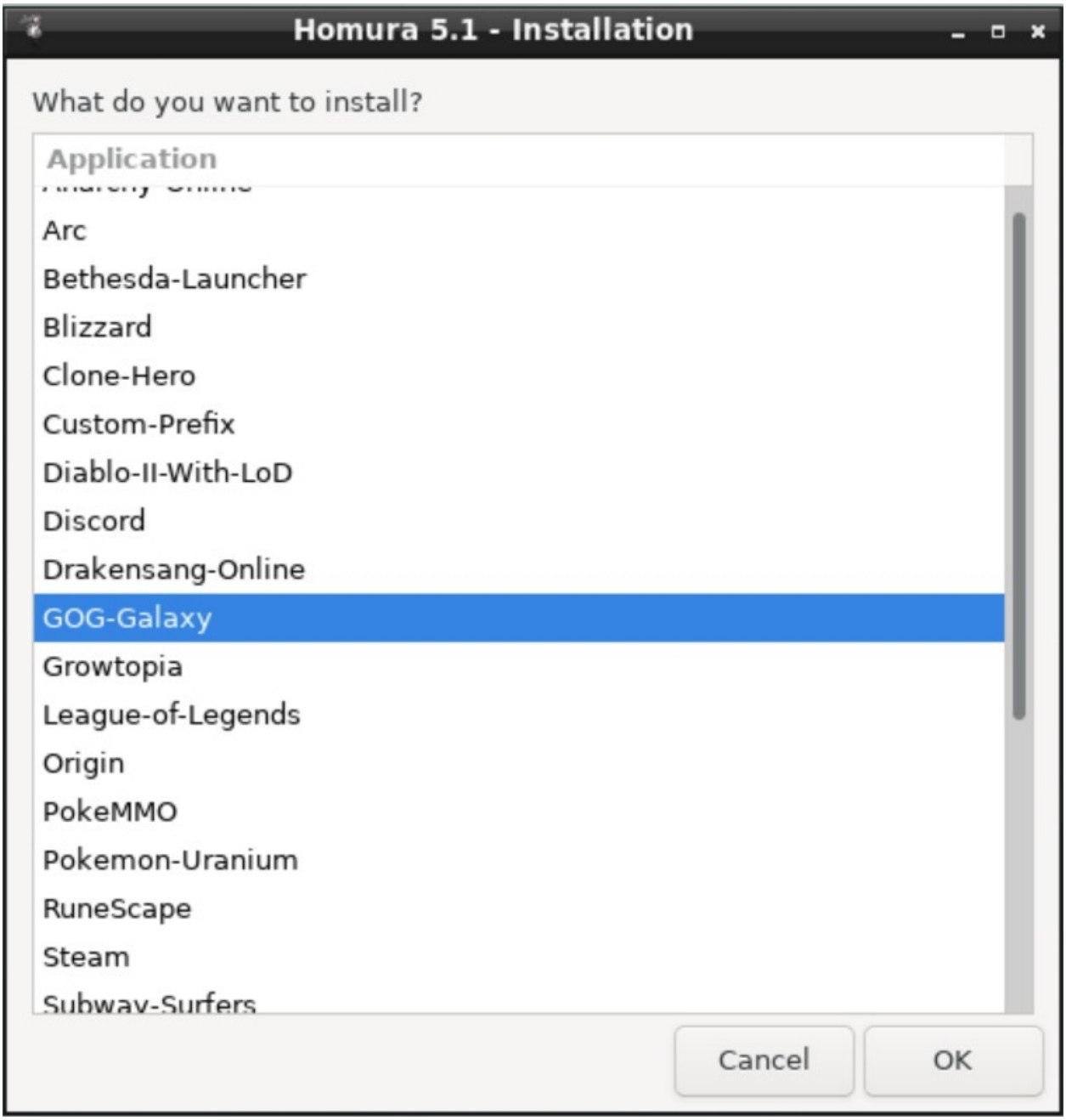
该程序还将提供在兼容环境的应用菜单中放置一个链接：



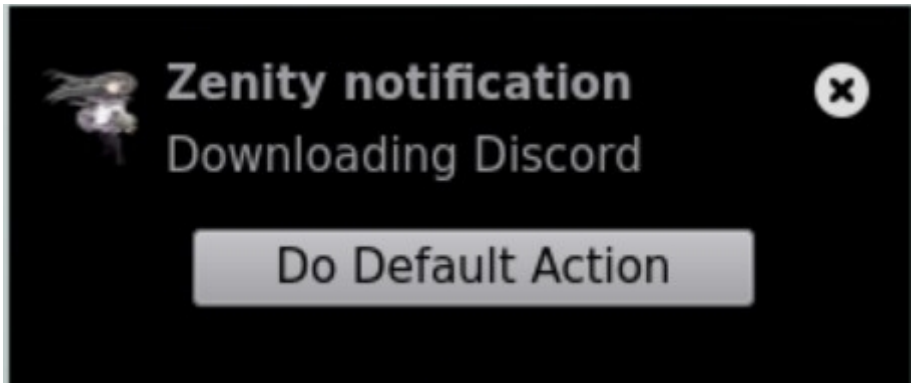
根据 FreeBSD 机器的设置，Homura 可能会显示一条敦促安装本地图形驱动程序的信息：



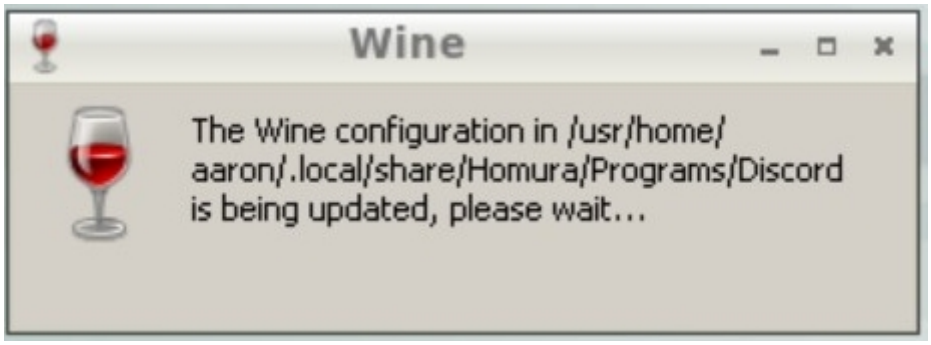
然后，应用程序的窗口应该出现，这相当于一个带有所有选项的“主菜单”。许多项目与 `winetricks` 相同，尽管 Homura 提供了一些额外的、有用的选项，如打开其数据文件夹（*Open Homura Folder*）或运行一个指定的程序（*Run a executable in prefix*）：



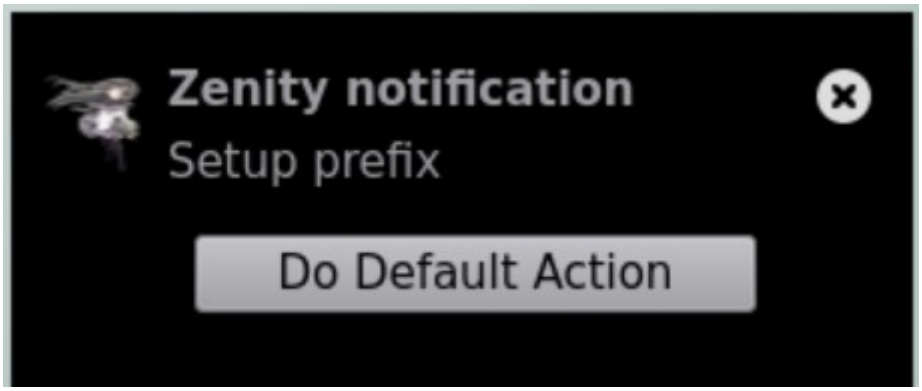
要选择 Homura 支持的应用程序之一进行安装，选择安装，然后点击确定。这将显示一个 Homura 可以自动安装的应用程序的列表。选择一个，然后单击 *OK* 以开始该过程。



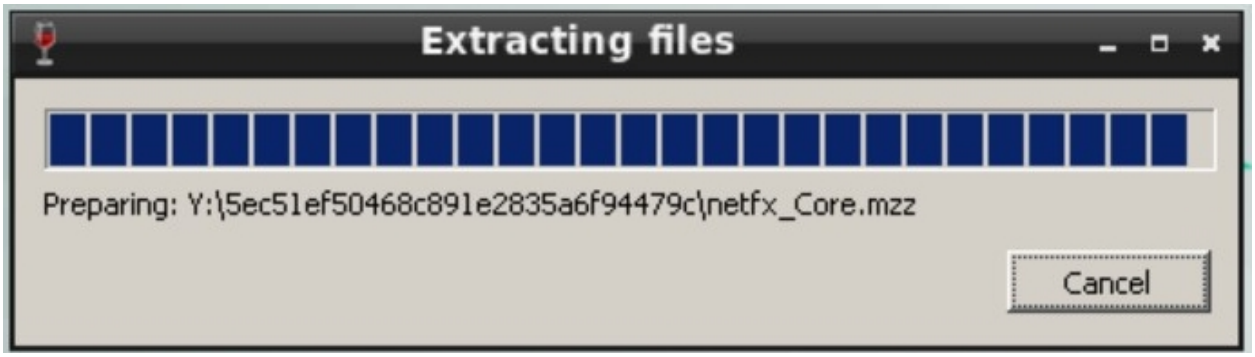
第一步，Homura 将下载选定的程序。在支持的桌面环境中可能会出现一个通知。



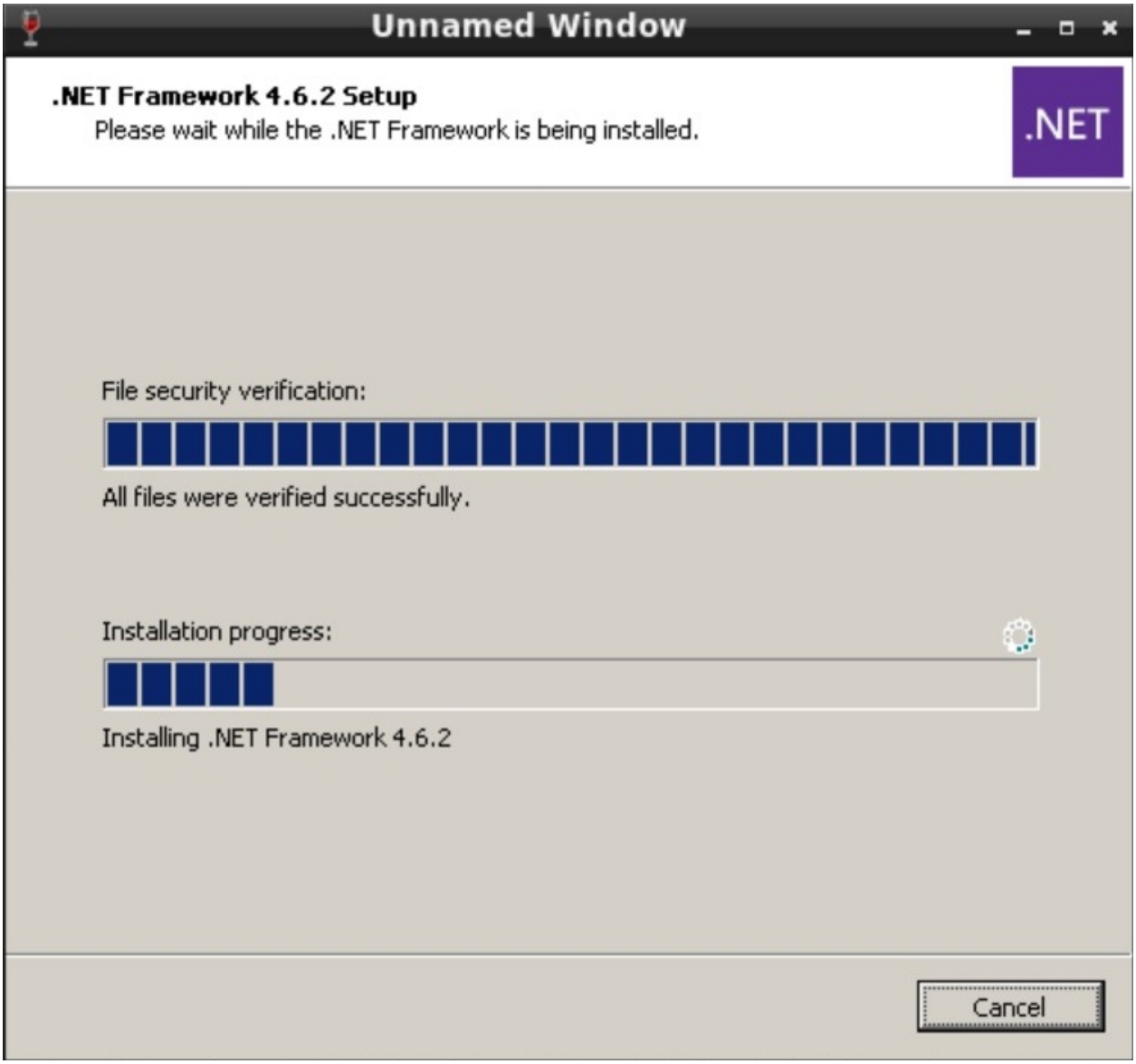
该程序还将为应用程序创建一个新的 WINE Prefixes。一个标准的 WINE 对话框将显示这个信息。



接下来，Homura 将为选定的程序安装先决条件。这可能涉及下载和提取相当数量的文件，其细节将显示在对话框中。

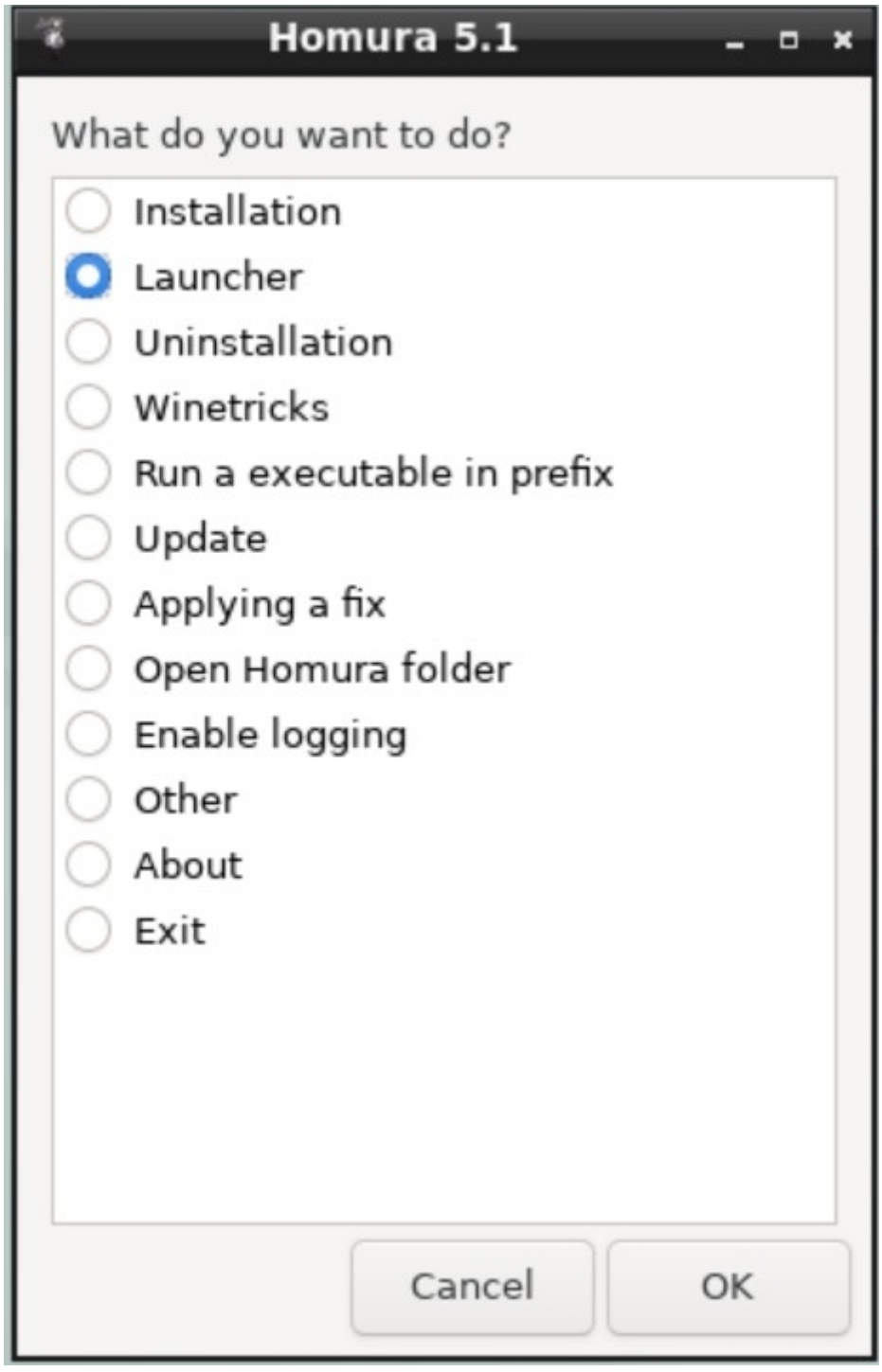


下载的软件包会根据需要自动打开并运行。

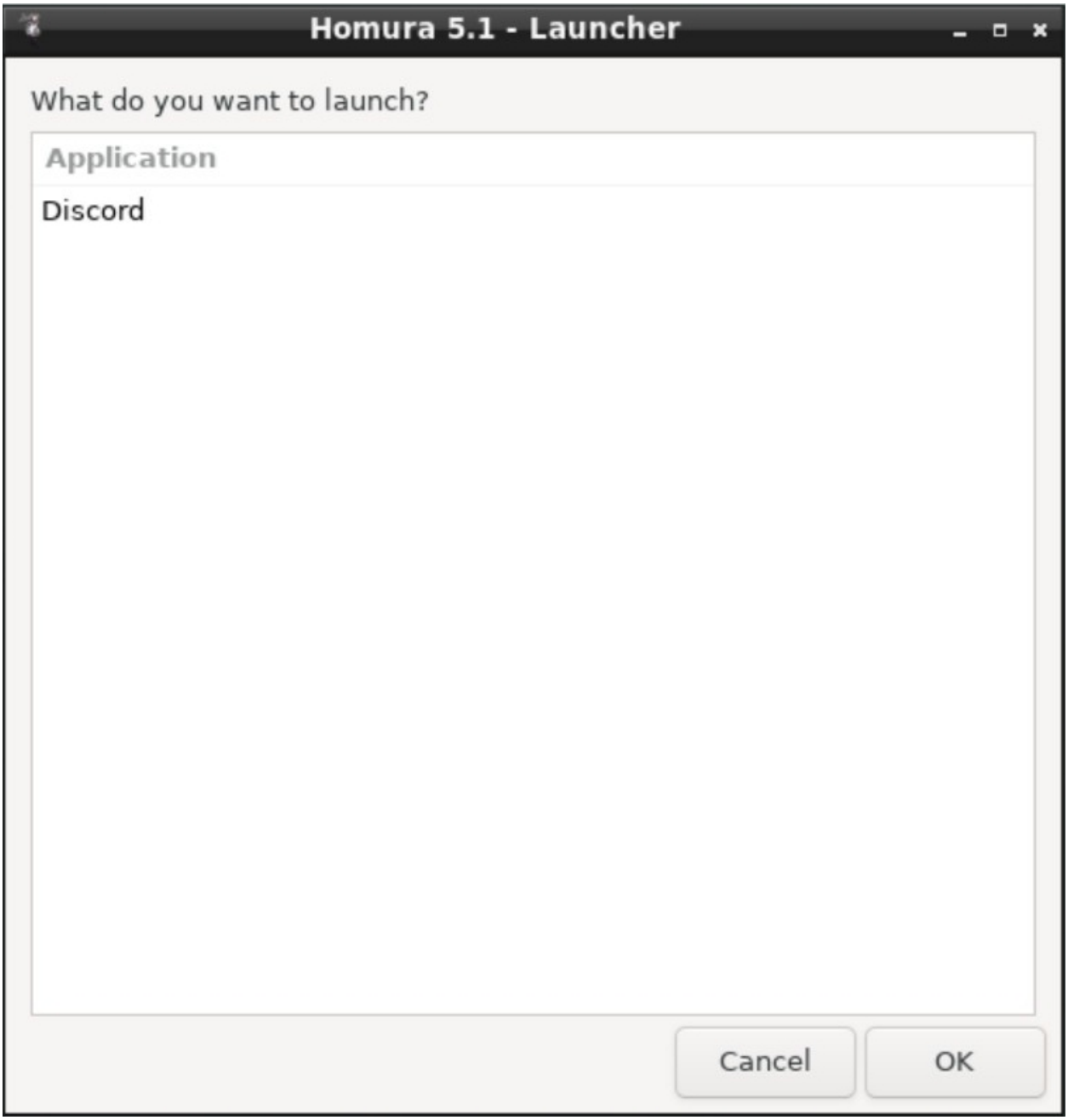


安装可能以一个简单的桌面通知或终端信息结束，这取决于 Homura 的启动方式。但无论哪种情况，Ho-

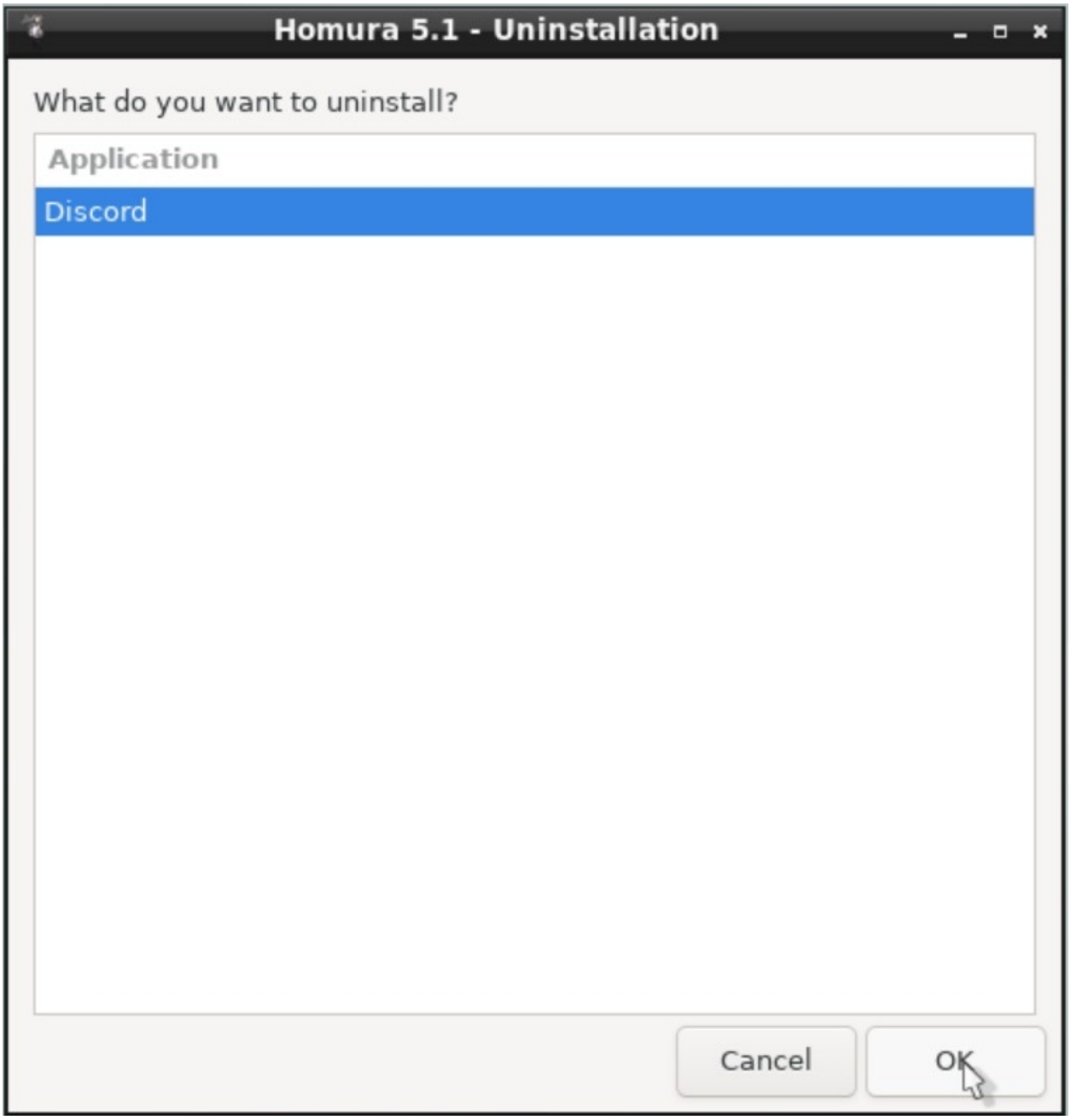
mura 都应该回到主屏幕。要确认安装成功，选择 *Launcher*，然后点击 *OK*。



这将显示一个已安装的应用程序的列表。



要运行新程序，从列表中选择它，然后点击 *OK*。要卸载该程序，从主屏幕上选择 *Uninstallation*，它将显示一个类似的列表。选择要删除的程序，然后点击 *OK*。



13.6.3.运行多个管理图形用户界面

值得注意的是，上述解决方案并不是相互排斥的。完全可以同时接受，这甚至是有利的，可同时安装这两个软件，因为它们支持不同的程序集。

然而，明智的做法是，确保它们不访问任何相同的 WINE Prefixe。这些解决方案中的每一个都应用了变通方法，并根据现有 WINE 问题的已知变通方法对注册表进行了修改，以使特定的应用程序顺利运行。允许 winetricks 和 Homura 访问相同的 WINE Prefixe 可能会导致其中一些被覆盖，其结果是一些或所有的应用程序不能按预期运行。

13.7.FreeBSD 多用户与 WINE

13.7.1.使用通用 WINE Prefixe 的问题

与大多数类 UNIX® 的操作系统一样，FreeBSD 是为多个用户同时登录和工作而设计的。另一方面，Windows® 是多用户的，即在一个系统上可以设置多个用户账户。但是，人们期望在任何时候都只有一个人在使用物理机器（台式电脑或笔记本电脑）。

最近的消费者版本的 Windows® 已经采取了一些措施来改善多用户情况下的操作系统。但是，它在很大程度上仍然是围绕着单用户体验而设计的。此外，WINE 项目为创建一个兼容的环境所采取的措施意味着，与 FreeBSD 应用程序（包括 WINE 本身）不同，它将类似于这种单用户环境。

因此，每个用户将不得不维护他们自己的配置，这有可能是好事。然而，安装应用程序，特别是像办公套件或游戏这样的大型应用程序，只安装一次是很有利的。这样做的原因有两个例子：维护（软件更新只需要应用一次）和存储效率（没有重复的文件）。

有两种策略可以尽量减少系统中多个 WINE 用户的影响。

13.7.2.将应用程序安装到一个公共驱动器上

如 WINE 配置一节所示，WINE 提供了将额外的驱动器附加到一个给定 WINE Prefixe 的能力。通过这种方式，应用程序可以被安装到一个共同的位置，而每个用户仍然有一个 WINE Prefixe，可以保留个人设置（取决于程序）。如果有相对较少的应用程序需要在用户之间共享，并且这些程序需要对 WINE Prefixe 进行少量的自定义调整才能运行，那么这是一个很好的设置。

以这种方式安装应用程序的步骤如下：

1. 首先，在系统上设置一个共享位置，将文件存放在那里，例如 **/mnt/windows-drive_d/**。创建新的目录在 mkdir[716] 命令的手册页中有说明。
2. 接下来，为这个新目录设置权限，只允许需要的用户访问它。一种方法是创建一个新的组，例如 “windows”，将所需的用户添加到该组中（参见用户和基本账户管理[717]部分中关于组的小节），并将该目录的权限设置为 `770`（权限[718]一节说明了这个过程。）。
3. 最后，按照本章中关于 WINE 配置一节的说明，使用 winecfg 将该位置作为一个驱动器添加到用户的 WINE Prefixe 中。

完成后，应用程序可以被安装到这个位置，随后使用指定的盘符（或标准的 UNIX® 式目录路径）运行。然而，如上所述，应该只有一个用户同时运行这些应用程序（可能正在访问其安装目录内的文件）。一些应用程序在由非所有者的用户运行时也可能表现出意想不到的行为，尽管该用户是一个组的成员，应该对整个目录有完全的 “读/写/执行” 权限。

[716] https://www.freebsd.org/cgi/man.cgi?query=mkdir&sektion=1&format=html

[717] https://docs.freebsd.org/en/books/handbook/basics/index.html#users-groups

[718] https://docs.freebsd.org/en/books/handbook/basics/index.html#permissions

13.7.3.使用 WINE 的普通安装

另一方面，如果有许多应用程序需要共享，或者它们需要特定的调整才能正常工作，则可能需要一个不同的方法。在这种方法中，要创建一个完全独立的用户，专门用于存储 WINE Prefixe 和所有安装的应用程序。然后，个别用户被授予使用 sudo(8)[719] 命令作为该用户运行程序的权限。其结果是，这些用户可以像平时一样启动 WINE 程序，只是它将像由新创建的用户启动一样，因此使用集中维护的包含设置和程序的 WINE Prefixe。为了达到这个目的，请采取以下步骤：

用下面的命令创建一个新的用户（以 root 身份），这将逐步完成所需的细节：

```
# adduser
```

输入用户名（例如，*windows*）和全名（“Microsoft Windows”）。然后接受其余问题的默认值。接下来，使用软件包安装工具 sudo，方法如下：

```
# pkg install sudo
```

安装后，编辑 **/etc/sudoers** 如下：

```
# User alias specification

# define which users can run the wine/windows programs
User_Alias WINDOWS_USERS = user1,user2

# define which users can administrate (become root)
User_Alias ADMIN = user1

# Cmnd alias specification

# define which commands the WINDOWS_USERS may run
Cmnd_Alias WINDOWS = /usr/bin/wine,/usr/bin/winecfg

# Defaults
Defaults:WINDOWS_USERS env_reset
Defaults:WINDOWS_USERS env_keep += DISPLAY
Defaults:WINDOWS_USERS env_keep += XAUTHORITY
Defaults    !lecture,tty_tickets,!fqdn

# User privilege specification
root    ALL=(ALL) ALL

# Members of the admin user_alias, defined above, may gain root privileges
```

(continues on next page)

[719] https://www.freebsd.org/cgi/man.cgi?query=sudo&sektion=8&format=html

(continued from previous page)

```
ADMIN ALL=(ALL) ALL

# The WINDOWS_USERS may run WINDOWS programs as user windows without a password
WINDOWS_USERS ALL = (windows) NOPASSWD: WINDOWS
```

这些变化的结果是，在 *User_Alias* 部分命名的用户被允许使用 *Defaults* 部分列出的资源（当前显示）运行 *Cmnd Alias* 部分列出的程序，就好像他们是文件最后一行列出的用户一样。换句话说，被指定为 *WINDOWS_USERS* 的用户可以作为用户 *windows* 运行 WINE 和 `winecfg` 程序。因此，这里的配置意味着他们将不需要输入 *windows* 用户的密码。

接下来将显示的权限还给 *windows* 用户，WINE 程序将以其身份运行：

```
% xhost +local:windows
```

这应该被添加到登录时或默认图形环境启动时的命令列表中。完成上述所有工作以后，在 **sudoers** 中被配置为 `WINDOW_USERS` 之一的用户就可以使用以下命令运行共享 WINE Prefixe 的程序。

值得注意的是，多个用户同时访问这个共享环境仍然是有风险的。然而，还要考虑到共享环境本身可以包含多个 WINE Prefixe。通过这种方式，管理员可以创建一个经过测试和验证的程序集，每个程序都有自己的 WINE Prefixe。同时，一个用户可以玩一个游戏，而另一个用户可以使用办公程序，而无需安装多余的软件。

13.8.WINE 与 FreeBSD FAQ

以下部分介绍了一些在 FreeBSD 上运行 WINE 时经常被问到的问题、技巧/窍门或常见问题，以及它们各自的答案。

13.8.1.基本安装和使用

13.8.1.1.如何在同一个系统上安装 32 位和 64 位 WINE？

如本节前面所述，软件包 wine 和 i386-wine 相互冲突，因此不能以正常方式在同一系统上进行安装。然而，可以通过 *chroot/jail* 等机制，或通过从源码编译 WINE（注意这并 不意味着编译 port）来实现多重安装。

13.8.1.2.可以在 WINE 上运行 DOS 程序吗？

它们可以运行，正如本节前面提到的“控制台用户界面”应用程序。然而，可以说有一种更好的方法来运行 DOS 软件：emulators/dosbox[720]。另一方面，没有理由不尝试一下。简单地创建一个新的 WINE Prefixe，安装软件，如果不工作，就删除这个 WINE Prefixe。

13.8.1.3.是否应该通过 pkg 或 ports 安装 emulators/wine-devel[Page 298, 721] 来使用 WINE 的开发版而非稳定版？

是的，安装这个版本将安装 WINE 的“开发”版本。与 32 位和 64 位版本一样，除非采取额外措施，否则它们不能与稳定版一起安装。

请注意，WINE 也有一个“暂存”版本，它包含最新的更新。这个版本曾一度作为 FreeBSD 的移植版本；然而，它后来被删除了。不过，可以直接从源代码中编译它。

13.8.2.安装优化

13.8.2.1.应该如何处理 Windows® 硬件（例如，图形）驱动程序？

操作系统驱动程序在应用程序和硬件之间传输命令。WINE 模拟了一个 Windows® 环境，包括驱动程序，反过来又使用 FreeBSD 的本地驱动程序进行传输。不建议安装 Windows® 驱动程序，因为 WINE 系统被设计为使用主机系统驱动程序。例如，如果有一块得益于专用驱动程序的显卡，请使用标准的 FreeBSD 方法来安装，而不是 Windows® 安装程序。

13.8.2.2.是否有办法让 Windows® 的字体看起来更好？

在 FreeBSD 论坛上有个用户建议用这个配置来修复 WINE 字体的开箱即用的外观，因为它可能会有轻微的像素化。

根据 FreeBSD 论坛上的一个帖子[722]，在 **.config/fontconfig/fonts.conf** 中加入以下内容可以增加抗锯齿，使文本更易读。

```
<?xml version="1.0"?>
<!DOCTYPE fontconfig SYSTEM "fonts.dtd>"

<fontconfig>

  <!-- antialias all fonts -->
```

(continues on next page)

[720] https://cgit.freebsd.org/ports/tree/emulators/dosbox/pkg-descr

[721] https://cgit.freebsd.org/ports/tree/emulators/wine-devel/pkg-descr

[722] https://forums.freebsd.org/threads/make-wine-ui-fonts-look-good.68273/

(continued from previous page)

```
  <match target="font">
    <edit name="antialias" mode="assign"><bool>true</bool></edit>>
    <edit name="hinting" mode="assign"><bool>true</bool></edit>>
    <edit name="hintstyle" mode="assign"><const>hintslight</const></edit>>
    <edit name="rgba" mode="assign"><const>rgb</const></edit>>
  </match>
</fontconfig>
```

13.8.2.3.在系统的其他地方安装 Windows® 有助于 WINE 的运行吗？

有可能，这取决于正在运行的应用程序。正如在说明 `winecfg` 一节中提到的，一些内置的 WINE DLL 和其他库可以通过提供一个替代版本的路径来覆盖。只要 Windows® 分区或驱动器被挂载到 FreeBSD 系统中，并且用户可以访问，配置一些这样的覆盖将使用本地 Windows® 库，并可能减少发生意外行为的机会。

13.8.3.特定的应用程序

13.8.3.1.哪里是查看应用程序 X 是否在 WINE 上工作的最佳位置？

确定兼容性的第一步应该是 WINE AppDB[723]。这是在所有支持的平台上工作（或不工作）的程序报告的汇编，尽管（如前所述），一个平台的解决方案往往适用于其他平台。

13.8.3.2.有什么东西可以帮助游戏更好地运行吗？

也许吧。许多 Windows® 游戏依赖于 DirectX，这是微软专有的图形层。然而，在开放源码社区中有一些项目试图实现对这项技术的支持。

dxvk 就是这样一个项目，它试图使用与 FreeBSD 兼容的 Vulkan 图形子系统来实现 DirectX。尽管它的主要目标是 Linux 上的 WINE，但一些 FreeBSD 用户也报告[724]说正在编译和使用 dxvk。

此外，正在进行 port WINE-proton[725] 移植的工作。这将把 Valve，Steam 游戏平台的开发者的工作带到 FreeBSD。Proton 是一个 WINE 的发行版，旨在让许多 Windows® 游戏以最小的设置在其他操作系统上运行。

[723] https://appdb.winehq.org/

[724] https://forums.freebsd.org/threads/what-about-gaming-on-freebsd.723/page-9

[725] https://www.freshports.org/emulators/wine-proton/

13.8.3.3.有什么地方可以让 FreeBSD WINE 用户聚集在一起交流心得和技巧吗？

有很多 FreeBSD 用户讨论与 WINE 有关的问题的地方，可以搜索到解决方案。

- FreeBSD 论坛[726]，特别是 *安装和维护 ports 和软件包或仿真和虚拟化论坛*。
- FreeBSD IRC 频道[727]，包括 #freebsd (用于一般支持), #freebsd-games, 和其他。
- BSD World Discord 服务器的频道[728]，包括 *bsd-desktop*、*bsd-gaming*、*bsd-wine* 等。

13.8.4.其他操作系统资源

有许多专注于其他操作系统的资源可能对 FreeBSD 用户有用：

- WINE Wiki[729] 有大量关于使用 WINE 的信息，其中大部分适用于 WINE 支持的许多操作系统。
- 同样地，其他操作系统项目提供的文档也很有价值。Arch Linux Wiki 上的 WINE 页面[730]是一个特别好的例子，尽管一些“第三方应用程序”（即“配套应用程序”）显然不能在 FreeBSD 上使用。
- 最后，Codeweavers（WINE 商业版本的开发者）是一个活跃的上游贡献者。在他们的支持论坛[731]上，对问题的回答往往可以帮助解决 WINE 开源版本的问题。

[726] https://forums.freebsd.org/
[727] https://wiki.freebsd.org/IRC/Channels
[728] https://discord.gg/2CCuhCt
[729] https://wiki.winehq.org/
[730] https://wiki.archlinux.org/index.php/wine
[731] https://www.codeweavers.com/support/forums

第三部分：系统管理

本手册其余各章涵盖了 FreeBSD 系统管理的各个方面。每一章的开头都介绍了阅读本章后将会学到的东西，并详细说明了读者在阅读这些材料之前应该具有的背景知识。

这些章节的设计是为了在需要时查阅这些信息。它们无需按照任何特定的顺序来阅读，也不需要在开始使用 FreeBSD 之前阅读所有的章节。

第十四章配置与优化

14.1.概述

FreeBSD 的一个重要方面就是进行正确的系统配置，这一章解释了许多 FreeBSD 的配置过程，包括使用一些可以调整的参数来优化 FreeBSD 系统。

在本章开始前，你应该：

- 理解 UNIX® 和 FreeBSD 的基础内容（FreeBSD 基础[732]）。
- 熟悉内核配置和编译的基础内容（配置 FreeBSD 内核[733]）。

读完本章后，你将知道：

- 关于配置 **rc.conf** 的基础内容以及启动脚本 **/usr/local/etc/rc.d** 的基础配置。
- 如何在 */etc* 中使用多种配置文件。
- 如何使用 sysctl(8)[734] 变量来对 FreeBSD 进行优化。
- 如何优化磁盘性能及调整内核参数。

14.2.启动服务

许多用户在 FreeBSD 上安装了来自 ports 的第三方软件，并要求在系统初始化时启动所安装的服务。诸如 mail/postfix[735] 或 www/apache24[736] 等服务只是众多可能在系统初始化时被启动的软件包中的两个。本节将解释如何自启动第三方软件。

[732] https://docs.freebsd.org/en/books/handbook/basics/index.html#basics

[733] https://docs.freebsd.org/en/books/handbook/kernelconfig/index.html#kernelconfig

[734] https://www.freebsd.org/cgi/man.cgi?query=sysctl&sektion=8&format=html

[735] https://cgit.freebsd.org/ports/tree/mail/postfix/pkg-descr

[736] https://cgit.freebsd.org/ports/tree/www/apache24/pkg-descr

在 FreeBSD 中，大多数包含的服务，例如 cron(8)[737]，都是通过系统启动脚本启动的。

14.2.1.扩展应用配置

在现在 FreeBSD 包含了 **rc.d**，使得应用程序的启动配置变得更加容易，并提供了更多的功能。通过使用本手册在管理 FreeBSD 中的服务[738]中包含的命令，应用程序可以被设置为在启动某些其他服务之后再启动，并且可以通过 **/etc/rc.conf** 传递额外的标志，以取代启动脚本中的硬编码标志。一个基本的启动脚本可能类似于以下内容：

```
#!/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"
```

这个脚本将确保所提供的 `utility` 将在 `DAEMON` 伪服务之后启动，同时还提供了一种设置和追踪进程 ID（PID）的方法。

然后，这个应用程序可以在 **/etc/rc.conf** 中添加如下一行：

```
utility_enable="YES"
```

这种方法允许更容易地操作命令行参数，包含 **/etc/rc.subr** 中默认提供的函数，与 rcorder(8)[739] 兼容，并

[737] https://www.freebsd.org/cgi/man.cgi?query=cron&sektion=8&format=html

[738] https://docs.freebsd.org/en/books/handbook/book/#configtuning-rcd

[739] https://www.freebsd.org/cgi/man.cgi?query=rcorder&sektion=8&format=html

通过 **rc.conf** 提供更容易的配置。

14.2.2.使用服务来启动服务

其他服务可以使用 inetd(8)[740] 来启动。使用 inetd(8)[741] 及其配置在 inetd 超级服务器[742]中有深入描述。

在某些情况下，使用 cron(8)[743] 来启动系统服务可能更有意义，这种方法有很多优点，因为 cron(8)[744] 作为 crontab(5)[745] 的所有者运行这些进程，这使得普通用户可以启动和维护他们自己的应用程序。

可以用 cron(8)[746] 的 `@reboot` 功能来代替时间规范，这可以使任务在 cron(8)[747] 启动时运行，通常是在系统初始化期间。

14.3.配置 cron(8)[Page 305, 748]

cron 是 FreeBSD 中最有用的工具之一。这个工具在后台运行，并定期检查 **/etc/crontab** 中要执行的任务，并搜索 **/var/cron/tabs** 中的自定义 crontab 文件。这些文件被用来安排 cron 在指定时间运行的任务。crontab 中的每个条目都定义了一个要运行的任务，被称为 *cron 作业*（job）。

FreeBSD 使用了两种不同类型的配置文件：系统级 crontab 和用户级 crontab，其中不应该对前者进行修改，但后者可以根据需要来创建和编辑，这些文件使用的格式在 crontab(5)[749] 的手册中有说明。需要注意的是，系统级 crontab 的某些格式，即 **/etc/crontab** 包括的 `who` 列，在用户级 crontab 中不存在。在系统级 crontab 中，cron 以该列中指定的用户身份运行命令。在用户级 crontab 中，所有命令都以创建 crontab 的用户的身份运行。

用户级 crontab 允许单个用户安排他们自己的任务，root 用户也可以有一个用户级 **crontab**，用以安排在系统级 **crontab** 中不存在的任务。

如下是一个系统级 crontab，即 **/etc/crontab** 的示例：

```
# /etc/crontab - root's crontab for FreeBSD
#
# $FreeBSD$
①
SHELL=/bin/sh
PATH=/etc:/bin:/sbin:/usr/bin:/usr/sbin    ②
#
```

(continues on next page)

[740] https://www.freebsd.org/cgi/man.cgi?query=inetd&sektion=8&format=html
[741] https://www.freebsd.org/cgi/man.cgi?query=inetd&sektion=8&format=html
[742] https://docs.freebsd.org/en/books/handbook/network-servers/index.html#network-inetd
[743] https://www.freebsd.org/cgi/man.cgi?query=cron&sektion=8&format=html
[744] https://www.freebsd.org/cgi/man.cgi?query=cron&sektion=8&format=html
[745] https://www.freebsd.org/cgi/man.cgi?query=crontab&sektion=5&format=html
[746] https://www.freebsd.org/cgi/man.cgi?query=cron&sektion=8&format=html
[747] https://www.freebsd.org/cgi/man.cgi?query=cron&sektion=8&format=html
[748] https://www.freebsd.org/cgi/man.cgi?query=cron&sektion=8&format=html
[749] https://www.freebsd.org/cgi/man.cgi?query=crontab&sektion=5&format=html

(continued from previous page)

```
#minute hour    mday    month   wday    who command    ③
#
*/5 *   *   *   *   root    /usr/libexec/atrun    ④
```

对上述配置文件的解释如下：

① 以 # 开头的部分是注释，可以起到提醒的作用。注释不能和命令放在同一行，否则会被当成命令的一部分，因此注释必须另起一行。空行在执行过程中会被忽略。

② 等于号（=）符号是被用来定义环境设置的。在这个实例中，它被用来定义 SHELL 和 PATH。如果 SHELL 变量被省略的话，cron 将会使用默认的 Bourne Shell；如果 PATH 被省略的话，当你运行某个程序或脚本时，将必须给定其完整路径。

③ 这一行定义了系统级 crontab 所使用的七个字段（或参数）：minute、hour、mday、month、wday、who 以及 command。其中，minute 字段运行指定命令的时间（以分钟为单位），hour 是代表在一日中的第几个小时，mday 是指在一月中的某日，month 是指月份，wday 则是指在一周中的哪一天。这些字段的值必须是数字类型的，且使用 24 小时制。若使用 * 的话，则是指那个字段所包含的所有值，即对该字段下的条件不做限定。who 字段仅在系统级 crontab 中使用，是用来指定运行该命令的用户的。最后一个字段的对应值是你想要运行的程序。

④ 这一行指定了这个 cron 任务的具体细节，即各个参数的对应值。比如，*/5，以及它后面的那四个 *，代表了这个由 root 用户执行的 /usr/libexec/atrun 命令，需要在每月、每周的每天内，每隔五分钟执行一次。命令可以包含任何数目的开关。扩展到多行的命令需要用反斜杠“\”字符断开。

14.3.1.创建用户级的 crontab

要创建一个用户级 crontab，在编辑模式下调用 crontab：

```
% crontab -e
```

这将使用默认的文本编辑器打开用户的 crontab。用户第一次运行这个命令时，它将打开一个空文件，若用户曾经创建过 crontab，该命令将打开该文件进行编辑。

将这些行添加到 crontab 文件的顶部是很有用的，以便设置环境变量和记录 crontab 中各字段的含义：

```
SHELL=/bin/sh
PATH=/etc:/bin:/sbin:/usr/bin:/usr/sbin
# Order of crontab fields
# minute    hour    mday    month   wday    command
```

然后为每个要运行的命令或脚本添加一行，并在此指定运行该命令的时间。这个例子是每天下午两点运行指定的自定义 Bourne shell 脚本。由于脚本的路径没有在 PATH 中指定，所以我们给出了脚本的完整路径：

```
0   14  *   *   *   /usr/home/dru/bin/mycustomscript.sh
```

技巧

在使用自定义脚本之前，请确保它是可执行的，并且用 cron 在设置的有限环境变量下来进行测试。要复制用于运行上述 cron 条目的环境，请使用：

```
env -i SHELL=/bin/sh PATH=/etc:/bin:/sbin:/usr/bin:/usr/sbin HOME=/home/dru␣
↪LOGNAME=dru /usr/home/dru/bin/mycustomscript.sh
```

有关 cron 环境的设置可以在 crontab(5)[750] 手册中找到。如果脚本中包含任何使用通配符删除文件的命令，检查脚本是否在 cron 环境中正确运行就显得尤为重要。

当完成编辑 crontab 后，保存该文件，之后它将被自动读取，cron 会读取 crontab 并在其指定的时间运行其 cron 作业。要列出 crontab 中的 cron 作业，使用这个命令：

```
% crontab -l
0   14  *   *   *   /usr/home/dru/bin/mycustomscript.sh
```

要删除这个用户级 crontab 中的所有 cron 作业，使用这个命令：

```
% crontab -r
remove crontab for dru? y
```

14.4.管理 FreeBSD 中的服务

FreeBSD 在系统初始化和管理服务时使用 rc(8)[751] 系统的启动脚本。在 **/etc/rc.d** 中列出的脚本提供了基本的服务，可以用 service(8)[752] 的 `start` 、`stop` 和 `restart` 选项来控制。例如，可以用下面的命令重启 sshd(8)[753]：

```
# service sshd restart
```

这个程序可以用来在一个正在运行中的系统上启动服务。服务将在系统启动时按照 rc.conf(5)[754] 中的指定的那样自动启动。例如，要在系统启动时启用 natd(8)[755]，请在 **/etc/rc.conf** 中添加以下一行：

```
natd_enable="YES"
```

[750] https://www.freebsd.org/cgi/man.cgi?query=crontab&sektion=5&format=html
[751] https://www.freebsd.org/cgi/man.cgi?query=rc&sektion=8&format=html
[752] https://www.freebsd.org/cgi/man.cgi?query=service&sektion=8&format=html
[753] https://www.freebsd.org/cgi/man.cgi?query=sshd&sektion=8&format=html
[754] https://www.freebsd.org/cgi/man.cgi?query=rc.conf&sektion=5&format=html
[755] https://www.freebsd.org/cgi/man.cgi?query=natd&sektion=8&format=html

如果 natd_enable="NO" 一行已经存在，请把 NO 改为 YES。rc(8)[756] 脚本会在下一次启动时自动加载其依赖的服务，如下所述。

由于 rc(8)[757] 系统的主要目的是在系统启动和关闭时启动和停止服务，start 、stop 和 restart 选项只有在 **/etc/rc.conf** 中的变量设置适当时才会执行它们的操作。例如，只有在 **/etc/rc.conf** 中把 sshd_enable 设置为 YES 时，sshd restart 才会生效。要 start 、stop 和 restart 一个服务，而无需考虑 **/etc/rc.conf** 中的设置，需要在这些命令的前面添加前缀 “one”。例如，要重启 sshd(8)[758] 而不考虑当前 **/etc/rc.conf** 的设置，执行以下命令:

```
# service sshd onerestart
```

为检查某个服务是否在 **/etc/rc.conf** 中被启用，请用 rcvar 运行相应的 rc(8)[759] 脚本。这个例子将检查 sshd(8)[760] 是否在 **/etc/rc.conf** 中被启用:

```
# service sshd rcvar
# sshd
#
sshd_enable="YES"
#   (default: "")
```

> 提示
> # sshd 这一行是上面那个命令的输出，不是以 root 身份执行的命令。

要确定一个服务是否在运行，使用 status。例如，要验证 sshd(8)[761] 是否正在运行:

```
# service sshd status
sshd is running as pid 433.
```

在某些情况下，也可以 reload 一个服务。该操作试图向一个单独的服务发送信号，迫使服务重新加载其配置文件。在大多数情况下，这意味着向该服务发送一个 SIGHUP 信号。并非每个服务都支持这个功能。

rc(8)[762] 系统用于网络服务，它也为大部分的系统初始化做出了贡献。例如，当 **/etc/rc.d/bgfsck** 脚本被执行时，它会打印出以下信息:

```
Starting background file system checks in 60 seconds.
```

这个脚本被用来在系统初始化的时候在后台进行文件系统检测。

[756] https://www.freebsd.org/cgi/man.cgi?query=rc&sektion=8&format=html
[757] https://www.freebsd.org/cgi/man.cgi?query=rc&sektion=8&format=html
[758] https://www.freebsd.org/cgi/man.cgi?query=sshd&sektion=8&format=html
[759] https://www.freebsd.org/cgi/man.cgi?query=rc&sektion=8&format=html
[760] https://www.freebsd.org/cgi/man.cgi?query=sshd&sektion=8&format=html
[761] https://www.freebsd.org/cgi/man.cgi?query=sshd&sektion=8&format=html
[762] https://www.freebsd.org/cgi/man.cgi?query=rc&sektion=8&format=html

许多系统服务依赖于其他服务才能正常运行。例如，yp(8)[763] 和其它基于 RPC 的服务可能会在 rpcbind(8)[764] 服务启动后才会启动。为了解决这个问题，关于依赖关系和其他元数据的信息被包含在每个启动脚本顶部的注释中。rcorder(8)[765] 程序被用来在系统初始化过程中解析这些注释，以确定系统服务应该被调用的顺序，以满足依赖关系：

所有的启动脚本都必须包含以下关键词，因为 rc.subr(8)[766] 需要它来“enable”启动脚本：

- `PROVIDE`：指定这个文件提供的服务.

下列关键词可以放在每个启动脚本的顶部，他们并不是严格需要的，但是作为对 rcorder(8)[767] 的提示来说是很有用的：

- `REQUIRE`：列出此服务所需的服务，包含这个关键词的脚本将在指定的服务之 后运行。
- `BEFORE`：列出依赖这个的服务，包含这个关键词的脚本将在指定服务之 前运行。

通过为每个启动脚本仔细设置这些关键字，管理员可以对脚本的启动顺序进行精细的控制，而不需要某些 UNIX® 操作系统所使用的“运行级别”（runlevels）。

其他信息可以在 rc(8)[768] 和 rc.subr(8)[769] 的用户手册中中找到。关于创建自定义 rc(8)[770] 脚本的说明，请参考这篇文章[771]。

14.4.1.管理特定的系统配置

系统配置信息的主要位置是 **/etc/rc.conf**。该文件包含广泛的配置信息，在系统启动时被读取以配置系统，它提供了 **rc*** 文件的配置信息。

/etc/rc.conf 中的条目会覆盖 **/etc/defaults/rc.conf** 中的默认设置。包含默认设置的文件不应该被编辑。相反，所有针对系统所做的修改都应该在 **/etc/rc.conf** 中进行。

在集群应用中可以采用一些策略，将整个站点的配置与系统特定的配置分开，以减少管理开销。推荐的方法是将系统特定的配置放入 **/etc/rc.conf.local**。例如，**/etc/rc.conf** 中的这些条目适用于所有系统：

```
sshd_enable="YES"
keyrate="fast"
defaultrouter="10.1.1.254"
```

而 **/etc/rc.conf.local** 中的这些条目只适用于这个系统：

[763] https://www.freebsd.org/cgi/man.cgi?query=yp&sektion=8&format=html
[764] https://www.freebsd.org/cgi/man.cgi?query=rpcbind&sektion=8&format=html
[765] https://www.freebsd.org/cgi/man.cgi?query=rcorder&sektion=8&format=html
[766] https://www.freebsd.org/cgi/man.cgi?query=rc.subr&sektion=8&format=html
[767] https://www.freebsd.org/cgi/man.cgi?query=rcorder&sektion=8&format=html
[768] https://www.freebsd.org/cgi/man.cgi?query=rc&sektion=8&format=html
[769] https://www.freebsd.org/cgi/man.cgi?query=rc.subr&sektion=8&format=html
[770] https://www.freebsd.org/cgi/man.cgi?query=rc&sektion=8&format=html
[771] https://docs.freebsd.org/en/articles/rc-scripting/

```
hostname="node1.example.org"
ifconfig_fxp0="inet 10.1.1.1/8"
```

使用 `rsync` 或 `puppet` 等应用程序将 **/etc/rc.conf** 分发到每个系统，而保留各自的 **/etc/rc.conf.local**。

升级系统不会覆盖 **/etc/rc.conf**，所以系统配置信息不会丢失。

提示

/etc/rc.conf 和 **/etc/rc.conf.local** 都是由 sh(1)[772] 解析的，这使得系统操作者可以创建复杂的配置方案。参考 rc.conf(5)[773] 的命令手册以了解更多有关信息。

14.5.配置系统日志

生成和阅读系统日志是系统管理的一个重要内容。可以用系统日志中的信息来检测硬件和软件问题，以及应用程序和系统配置错误。这些信息在安全审计和事件响应中也起着重要作用。大多数系统守护程序和应用程序都会产生日志。

FreeBSD 提供了一个系统日志记录器 syslogd 来管理日志记录。默认情况下，syslogd 在系统启动时被启动。这是由 **/etc/rc.conf** 中的变量 `syslogd_enable` 所控制的。有许多应用参数可以使用 **/etc/rc.conf** 中的 `syslogd_flags` 来设置。有关可用参数的更多信息，请参考 syslogd(8)[774] 的手册页面。

这一节说明了如何为本地和远程 FreeBSD 系统配置日志，以及如何进行日志轮替和日志管理。

14.5.1.配置本地日志

配置文件 **/etc/syslog.conf** 可控制 syslogd 在收到日志条目时的处理方式。有几个参数可以控制对传入事件的处理。设施（*facility*）用来说明哪个子系统产生了消息，如内核或守护进程，等级（*level*）用来描述所发生事件的严重程度。这使得我们可以根据设施和等级来配置是否和在哪里记录日志信息。也可以根据发送消息的应用程序采取相应的操作，在远程日志的情况下，还可以根据产生日志事件的机器的主机名采取操作。

这个配置文件中每行代表一个动作，每行的语法是一个选择器字段，然后接着是一个动作字段。选择器字段的语法是 *facility.level*，它将匹配来自该 等级或更高等级设施的日志信息。也可以在等级前添加一个可选的比较标志，以更精确地指定记录的内容。多个选择器字段可以用于同一个动作，并以分号（;）分开。使用 * 将匹配所有的内容。动作字段表示要将日志信息发送到哪里，比如发送到文件或远程日志主机。作为一个例子，这里是 FreeBSD 的默认的 **syslog.conf**：

```
# $FreeBSD$
#
#       Spaces ARE valid field separators in this file. However,
```

(continues on next page)

[772] https://www.freebsd.org/cgi/man.cgi?query=sh&sektion=1&format=html

[773] https://www.freebsd.org/cgi/man.cgi?query=rc.conf&sektion=5&format=html

[774] https://www.freebsd.org/cgi/man.cgi?query=syslogd&sektion=8&format=html

(continued from previous page)

```
#       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
*.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
lpr.info                                                /var/log/lpd-errs
ftp.info                                                /var/log/xferlog
cron.*                                                  /var/log/cron
!-devd
*.=debug                                                /var/log/debug.log
*.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
# Uncomment this if you wish to see messages produced by devd
# !devd
# *.>=info
!ppp
*.*                                                     /var/log/ppp.log
!*
```

在这个例子中：

- 第 8 行匹配所有等级为 `err` 或更高等级的消息，以及 `kern.warning`、`auth.notice` 和 `mail.crit`，并将这些日志消息发送到控制台（**/dev/console**）。
- 第 12 行匹配来自邮件设施的所有信息，等级为 `info` 或以上，并将这些信息记录到 **/var/log/maillog**。
- 第 17 行使用一个比较标志（=），只匹配 `debug` 等级的消息，并将它们记录到 **/var/log/debug.log**。
- 第 33 行是指定特定程序的例子。这使得它后面的规则只对指定的程序有效。在这种情况下，只有 ppp 产生的消息被记录到 **/var/log/ppp.log** 中。

可用的等级中，从最严重到最不严重的顺序是：`emerg`，`alert`，`crit`，`err`，`warning`，`notice`，`info`

和 debug。

设施无特定顺序，有 auth, authpriv, console, cron, daemon, ftp, kern, lpr, mail, mark, news, security, syslog, user, uucp 和 local0 到 local7。请注意，其他操作系统可能有不同的设施。

要把所有 notice 等级及以上的内容记录到 **/var/log/daemon.log**，请添加以下条目：

```
daemon.notice                                   /var/log/daemon.log
```

更多关于不同等级和设施的信息，请参考 syslog(3)[775] 和 syslogd(8)[776] 的命令手册。更多关于 **/etc/syslog.conf** 的信息、语法和高级使用示例，请参见 syslog.conf(5)[777] 的命令手册。

14.5.2. 日志管理和轮替

日志文件的迅速增长会占用磁盘空间，使查找有用信息更加困难。日志管理试图缓解这种情况。在 FreeBSD 中，newsyslog 被用来管理日志文件。这个内置的程序周期性地轮替和压缩日志文件，并且可以选择创建缺失的日志文件，在日志文件被移动时向程序发出信号。日志文件可以由 syslogd 或任何其他生成日志文件的程序生成。虽然 newsyslog 通常由 cron(8)[778] 调用，但它不是一个系统守护程序。在默认配置中，它每小时运行一次。

为了知道要采取哪些操作，newsyslog 会读取其配置文件 **/etc/newsyslog.conf**。每个由 newsyslog 所管理的日志文件会在此文件中占用一行。每一行都说明了其文件的所有者、权限、何时轮替该文件、影响日志轮替的可选标志（如压缩）以及日志轮替时的信号程序。下面是 FreeBSD 中的默认配置:

```
# configuration file for newsyslog
# $FreeBSD$
#
# 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
```

(continues on next page)

[775] https://www.freebsd.org/cgi/man.cgi?query=syslog&sektion=3&format=html
[776] https://www.freebsd.org/cgi/man.cgi?query=syslogd&sektion=8&format=html
[777] https://www.freebsd.org/cgi/man.cgi?query=syslog.conf&sektion=5&format=html
[778] https://www.freebsd.org/cgi/man.cgi?query=cron&sektion=8&format=html

(continued from previous page)

```
# 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/devd.log                       644  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
```

每一行都以要轮替的日志名称开始，后面可以选择轮替的和新创建的文件的所有者和组。`mode` 字段设置日志文件的权限，`count` 表示应该保留多少个轮替的日志文件。`size` 和 `when` 字段告诉 newsyslog 何时轮替文件。当一个日志文件的大小大于 `size` 字段或 `when` 字段中的时间已过，它就会被轮替。星号（`*`）意味着这个字段被忽略。*flags* 字段给出了进一步的指示，比如如何压缩轮替的文件，或者在缺少日志文件的情况下如何创建。最后两个字段是可选的，指定一个进程的进程 ID（PID）文件的名称，以及当文件被轮替时要发送给该进程的信号号。

关于所有字段、有效标志以及如何指定轮替时间的更多信息，请参考 newsyslog.conf(5)[779]。由于 newsyslog 是通过 cron(8)[780] 调用的，它不能比从 cron(8)[781] 中指定的计划更频繁地轮替文件。

[779] https://www.freebsd.org/cgi/man.cgi?query=newsyslog.conf&sektion=5&format=html

[780] https://www.freebsd.org/cgi/man.cgi?query=cron&sektion=8&format=html

[781] https://www.freebsd.org/cgi/man.cgi?query=cron&sektion=8&format=html

14.5.3.配置远程日志

随着系统数量的增加，监控多个主机的日志文件可能会变得很麻烦。配置集中式日志可以减少日志文件管理的一些管理负担。

在 FreeBSD 中，可以使用 syslogd 和 newsyslog 来配置集中的日志文件聚合、合并和轮替。本节演示了一个配置的例子，主机 `A`，名为 `logserv.example.com`，将收集本地网络的日志信息。主机 `B`，名为 `logclient.example.com`，将被配置为将日志信息传递给日志服务器。

14.5.3.1.日志服务器的配置

日志服务器是一个已被配置为接受来自其他主机的日志信息的系统。在配置日志服务器之前，请检查以下内容：

- 如果在日志服务器和任何日志客户端之间有防火墙，确保防火墙规则集允许客户端和服务器使用 UDP 514 端口。
- 记录服务器和所有客户机必须在本地 DNS 中拥有正向和反向条目。如果网络没有 DNS 服务器，请在每个系统的 **/etc/hosts** 中创建条目。正确的域名解析是必需的，这样日志条目才不会被日志服务器拒绝。

在日志服务器上，编辑 **/etc/syslog.conf** 以指定接收日志条目的客户端名称、要使用的日志设施、以及存储主机日志条目的日志名称。这个例子添加了 `B` 的主机名，记录了所有设施，并将日志条目存储在 **/var/log/logclient.log** 中。

例 **24.** 日志服务器配置的示例

```
+logclient.example.com
*.*     /var/log/logclient.log
```

当添加多个日志客户端时，为每个客户端添加一个类似的双行条目。关于可用设施的更多信息可以在 syslog.conf(5)[782] 中找到。

接下来配置 **/etc/rc.conf**：

```
syslogd_enable="YES"
syslogd_flags="-a logclient.example.com -v -v"
```

第一个条目是在系统启动时启动 syslogd。第二个条目允许来自指定客户端的日志条目。`-v -v` 增加了记录的消息的详细程度。这对于调整设施是很有用的，因为管理员能够看到每个设施下记录的消息的类型。

可以指定多个 `-a` 选项以允许从多个客户端记录。也可以指定 IP 地址和整个网段。请参考 syslogd(8)[783] 以获得可能的选项的完整列表。

最后，创建日志文件：

[782] https://www.freebsd.org/cgi/man.cgi?query=syslog.conf&sektion=5&format=html

[783] https://www.freebsd.org/cgi/man.cgi?query=syslogd&sektion=8&format=html

```
# touch /var/log/logclient.log
```

在这一点上，应该重新启动 syslogd 并进行验证：

```
# service syslogd restart
# pgrep syslog
```

如果返回一个 PID，说明服务器重启成功，可以开始客户端配置。如果服务器没有重启，请查阅 **/var/log/messages** 中的错误。

14.5.3.2. 日志客户端的配置

日志客户端将日志条目发送到网络上的一个日志服务器。客户端还保留一份自己的日志的本地副本。

配置了日志服务器后，就可以在日志客户端上编辑 **/etc/rc.conf**：

```
syslogd_enable="YES"
syslogd_flags="-s -v -v"
```

第一个条目在开机时启用 syslogd。第二个条目防止该客户端接受来自其他主机的日志（-s），并增加了日志信息的详细程度。

接下来，在客户端的 **/etc/syslog.conf** 中定义日志服务器。在这个例子中，所有记录的设施都被发送到一个远程系统，用 @ 符号表示，并指定了主机名：

```
*.*     @logserv.example.com
```

保存之后，重启 syslogd 来使其生效：

```
# service syslogd restart
```

要测试日志信息是否在网络上被发送，请在客户端使用 logger(1)[784] 向 syslogd 发送一条信息：

```
# logger "Test message from logclient"
```

这条信息现在应该同时存在于客户端的 **/var/log/messages** 和日志服务器的 **/var/log/logclient.log**。

[784] https://www.freebsd.org/cgi/man.cgi?query=logger&sektion=1&format=html

14.5.3.3.调试日志客户端

如果在日志服务器上没有收到任何消息，那么原因很可能是网络连接问题、主机名解析问题或配置文件中的一个错字。为了隔离原因，确保日志服务器和日志客户端都能够使用它们在 **/etc/rc.conf** 中指定的主机名来 `ping` 对方。如果失败了，请检查网络布线、防火墙规则集，以及日志服务器和客户端的 DNS 服务器或 **/etc/hosts** 中的主机名条目。重复进行，直到从两台主机上 `ping` 成功。

如果在两台主机上 `ping` 成功，但仍然没有收到日志信息，请暂时增加日志的详细程度，以缩小配置问题。在下面的例子中，日志服务器上的 **/var/log/logclient.log** 是空的，而日志客户端上的 **/var/log/messages** 也没有指出失败的原因。为了增加调试输出，编辑日志服务器上的 `syslogd_flags` 条目，并发出重启信号：

```
syslogd_flags="-d -a logclient.example.com -v -v"
```

重启后，类似以下的调试数据将立即在控制台出现：

```
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/kernel
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.
```

在这个例子中，日志信息被拒绝是由于一个错别字，导致主机名不匹配。客户端的主机名应该是 `log-client`，而不是 `logclien`。修正错别字，执行重启命令，并验证结果：

```
# 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/kernel
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
```

现在，信息被正确地接收并放在正确的文件中。

14.5.3.4.安全事项

与任何网络服务一样，在实施日志服务器之前应考虑安全事项。日志文件可能包含关于本地主机上启用的服务、用户账户和配置数据的敏感数据。从客户端发送到服务器的网络数据将不会被加密或密码保护。如果存在加密的需要，考虑使用 security/stunnel，它将通过加密的隧道传输日志数据。

本地安全也是一个问题。日志文件在使用过程中或在日志轮替后都没有加密。本地用户可能会访问日志文件，以获得对系统配置的额外洞察力。在日志文件上设置适当的权限是至关重要的。内置的日志轮替器 newsyslog 支持对新创建和轮替的日志文件设置权限。将日志文件设置为模式 `600` 应能防止本地用户的不必要的访问。请参阅 newsyslog.conf(5)[785] 的手册以了解更多信息。

14.6.配置文件

14.6.1. /etc 的配置

有许多目录保存着配置信息。这些目录包括：

目录	用途
/etc	一般的系统特定配置信息。
/etc/defaults	系统配置文件的默认版本。
/etc/mail	额外的 sendmail(8)[786] 配置和其他 MTA 配置文件。
/etc/ppp	用户或系统 ppp 程序的配置
/usr/local/etc	已安装的应用程序的配置文件。可能包含有应用名的子目录。
/usr/local/etc/rc.d	已安装程序的 rc(8)[787] 脚本
/var/db	自动生成的系统特定数据库文件，例如软件包数据库和 locate(1)[788] 数据库。

14.7.使用 sysctl(8) 进行优化

sysctl(8)[789] 是用来对正在运行的 FreeBSD 系统进行修改的命令。这包括许多 TCP/IP 协议栈和虚拟内存系统的高级选项，对于有经验的系统管理员来说，可以极大地提高性能。使用 sysctl(8)[790] 可以对超过五百个系统变量进行读取和设置。

[785] https://www.freebsd.org/cgi/man.cgi?query=newsyslog.conf&sektion=5&format=html
[786] https://www.freebsd.org/cgi/man.cgi?query=sendmail&sektion=8&format=html
[787] https://www.freebsd.org/cgi/man.cgi?query=rc&sektion=8&format=html
[788] https://www.freebsd.org/cgi/man.cgi?query=locate&sektion=1&format=html
[789] https://www.freebsd.org/cgi/man.cgi?query=sysctl&sektion=8&format=html
[790] https://www.freebsd.org/cgi/man.cgi?query=sysctl&sektion=8&format=html

在其核心部分，sysctl(8)[791] 有两个功能：读取和修改系统设置。

要查看所有可读变量，请输入：

```
% sysctl -a
```

要查看某个特定的变量，请指定其名称：

```
% sysctl kern.maxproc
kern.maxproc: 1044
```

要设置一个特定的变量，使用语法 变量=值：

```
# sysctl kern.maxfiles=5000
kern.maxfiles: 2088 -> 5000
```

sysctl 变量值通常是字符串、数字或布尔值，其中布尔值为 1 时表示 yes，为 0 表示 no。

要在机器每次启动时自动设置一些变量，可以将其添加到 **/etc/sysctl.conf** 中。更多信息，请参考 sysctl.conf(5)[792] 手册页面和 sysctl.conf[793]。

14.7.1. sysctl.conf

sysctl(8)[794] 的配置文件 **/etc/sysctl.conf**，看起来很像 `/etc/rc.conf`。值是以 变量=值的形式设置的，指定的值在系统进入多用户模式后被设置。并非所有的变量都可以在该模式下进行设置。

例如，为了关闭对 fatal 信号退出的记录，并防止用户看到其他用户启动的进程，可以在 **/etc/sysctl.conf** 中设置以下参数：

```
# 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
```

[791] https://www.freebsd.org/cgi/man.cgi?query=sysctl&sektion=8&format=html
[792] https://www.freebsd.org/cgi/man.cgi?query=sysctl.conf&sektion=5&format=html
[793] https://docs.freebsd.org/en/books/handbook/book/#configtuning-sysctlconf
[794] https://www.freebsd.org/cgi/man.cgi?query=sysctl&sektion=8&format=html

14.7.2.只读 sysctl(8)[795]

在某些情况下，可能需要修改只读 sysctl(8)[796] 值，这将需要重启系统。

例如，在某些型号的笔记本电脑上，cardbus(4)[797] 设备不能探测内存范围，并会出现类似的错误：

```
cbb0: Could not map register memory
device_probe_and_attach: cbb0 attach returned 12
```

修复这个问题时，需要更改 sysctl(8)[798] 的只读设定，为此，可向 **/boot/loader.conf** 中添加 `hw.pci.allow_unsupported_io_range=1` 并重启。然后 cardbus(4)[799] 应该就能够正常运行了。

14.8.磁盘优化

以下章节将讨论可用于磁盘设备的各种优化机制和选项。在许多情况下，带有机械部件的磁盘，如 SCSI 硬盘，将成为降低整个系统性能的瓶颈。虽然可以通过安装一个没有机械部件的硬盘来解决该问题，如固态硬盘，但机械硬盘在短期内不会消失。在磁盘优化时，建议利用 `iostat(8)` 命令的功能来测试系统的各种变化。这个命令可使用户获得关于系统 IO 的宝贵信息。

14.8.1. sysctl 变量

14.8.1.1. `vfs.vmiodirenable`

sysctl(8)[800] 变量 `vfs.vmiodirenable` 可以被设置为 `0`（关闭）或 `1`（开启）。默认情况下，它被设置为 `1`。该变量控制系统对目录的缓存方式。大多数目录都很小，在文件系统中只使用一个片段（通常为 1K），在缓冲区缓存中通常为 512 字节。如果关闭这个变量，缓冲区缓存将只缓存固定数量的目录，即使系统有大量的内存。当打开时，这个 sysctl(8)[801] 允许缓冲区缓存使用虚拟内存页面缓存来缓存目录，使所有的内存都可以用于缓存目录。然而，用于缓存目录的最小核心内存是物理页大小（通常是 4K）而非 512 字节。如果系统正在运行某些操作大量文件的服务，建议保持该选项的启用。这些服务可能包括网络缓存、大型邮件系统和新闻系统。开启这个选项通常不会降低性能，即使会浪费内存，但人们应该通过实验来发现问题。

[795] https://www.freebsd.org/cgi/man.cgi?query=sysctl&sektion=8&format=html
[796] https://www.freebsd.org/cgi/man.cgi?query=sysctl&sektion=8&format=html
[797] https://www.freebsd.org/cgi/man.cgi?query=cardbus&sektion=4&format=html
[798] https://www.freebsd.org/cgi/man.cgi?query=sysctl&sektion=8&format=html
[799] https://www.freebsd.org/cgi/man.cgi?query=cardbus&sektion=4&format=html
[800] https://www.freebsd.org/cgi/man.cgi?query=sysctl&sektion=8&format=html
[801] https://www.freebsd.org/cgi/man.cgi?query=sysctl&sektion=8&format=html

14.8.1.2. vfs.write_behind

sysctl(8)[802] 变量 vfs.write_behind 的默认值是 1（开启）。这告诉文件系统在收集完整的族时进行媒体写入，这通常发生在写入大的连续文件时。这可以避免在 I/O 性能无益的情况下用脏缓冲区使缓冲区缓存饱和。然而，这可能会拖延进程，在某些情况下应该关闭。

14.8.1.3. vfs.hirunningspace

sysctl(8)[803] 变量 vfs.hirunningspace 决定了在任何给定的实例中，有多少未完成的写 I/O 可以被排到磁盘控制器的系统中。默认值通常是足够的，但在有许多磁盘的机器上，可以尝试把它提高到 4 或 5 兆字节。设置太高的值，超过缓冲区缓存的写入阈值，会导致糟糕的族性能。不要随便设置这个值，因为更高的写入值可能会增加同时进行的读取的延迟。

还有其他各种与缓冲区高速缓存和虚拟页面高速缓存相关的 sysctl(8)[804] 值。不建议修改这些值，因为虚拟系统在自动调整自己方面做得很好。

14.8.1.4. vm.swap_idle_enabled

sysctl(8)[805] 变量 vm.swap_idle_enabled 在有许多活跃的登录用户和大量空闲进程的大型多用户系统中非常有用。这样的系统往往会对空闲的保留内存持续产生压力。通过 vm.swap_idle_threshold1 和 vm.swap_idle_threshold2 打开这个功能并调整交换滞后（以空闲秒为单位），比正常的分页算法更快地降低与空闲进程相关的内存页的优先级。这将给分页守护程序带来帮助。只有在需要时才打开这个选项，因为这样做的代价是尽早对内存进行预分页，而不是晚分页，这会消耗更多的交换空间和磁盘带宽。在小系统中，这个选项的影响是可确定的，但是在一个已经在进行适度分页的大系统中，这个选项允许虚拟系统轻松地将整个进程放入和取出内存。

14.8.1.5. hw.ata.wc

关闭 IDE 写缓存可以减少对 IDE 磁盘的写入带宽，但由于硬盘供应商引入的数据一致性问题，有时可能是必要的。问题是，有些 IDE 硬盘会在写入完成时撒谎：在打开 IDE 写缓存的情况下，IDE 硬盘会不按顺序地向磁盘写入数据，在磁盘负载较重的情况下，有时会无限期地延迟写入一些块。崩溃或断电可能导致严重的文件系统损坏。可通过观察 sysctl(8)[806] 变量 hw.ata.wc 来检查系统上的默认情况。如果 IDE 写缓存被关闭，可以在 /boot/loader.conf 中把这个只读变量设置为 1，以便在启动时启用它。

更多信息，请参考 ata(4)[807] 的手册。

[802] https://www.freebsd.org/cgi/man.cgi?query=sysctl&sektion=8&format=html
[803] https://www.freebsd.org/cgi/man.cgi?query=sysctl&sektion=8&format=html
[804] https://www.freebsd.org/cgi/man.cgi?query=sysctl&sektion=8&format=html
[805] https://www.freebsd.org/cgi/man.cgi?query=sysctl&sektion=8&format=html
[806] https://www.freebsd.org/cgi/man.cgi?query=sysctl&sektion=8&format=html
[807] https://www.freebsd.org/cgi/man.cgi?query=ata&sektion=4&format=html

14.8.1.6. SCSI_DELAY (kern.cam.scsi_delay)

内核配置选项 SCSI_DELAY 可以用来减少系统启动时间。默认值是相当高的，可能会在启动过程中造成 15 秒的延迟。把它减少到 5 秒通常对现代硬盘有效。应该使用 kern.cam.scsi_delay 调整启动时间。可优化的内核配置选项接受值的单位是毫秒而非秒。

14.8.2.软更新

要对文件系统进行微调，可以使用 tunefs(8)[808]。这个程序有许多不同的选项。要打开或关闭软更新，请使用：

```
# tunefs -n enable /filesystem
# tunefs -n disable /filesystem
```

文件系统在挂载时不能用 tunefs(8)[809] 来修改。在单用户模式下，启用软更新的好时机是在任何分区被挂载之前。

软更新被推荐用于 UFS 文件系统，因为它通过使用内存缓存极大地提高了元数据的性能，主要是文件的创建和删除。软更新有两个需要注意的缺点：首先，软更新保证了文件系统在崩溃情况下的一致性，但可能很容易比物理磁盘的更新晚几秒甚至一分钟。如果系统崩溃了，尚未写入的数据可能会丢失。其次，软更新延迟了文件系统块的释放。如果根文件系统几乎满了，执行一个大的更新，如 make installworld，会导致文件系统的空间耗尽，更新失败。

14.8.2.1.关于软更新的更多细节

元数据更新是对非内容数据的更新，如节点或目录。有两种传统的方法来将文件系统的元数据写回磁盘。

过去，默认行为是同步写出元数据更新。如果一个目录发生了变化，系统会等待，直到该变化被实际写入磁盘。文件数据缓冲区（文件内容）通过缓冲区缓存，随后异步备份到磁盘。这种实现方式的优点是操作安全。如果在更新过程中出现了故障，元数据总是处于一致的状态。一个文件要么被完全创建，要么根本就没有。如果一个文件的数据块在崩溃时还没有从缓冲区缓存中找到它们的位置，fsck(8)[810] 会识别出这一点，并通过将文件长度设置为 0 来修复文件系统。此外，这个实现是清晰和简单的。缺点是，元数据的改变很慢。例如，rm -r 按顺序触及一个目录中的所有文件，但每个目录的变化都会同步写入磁盘。这包括对目录本身的更新，对节点表的更新，以及可能对文件分配的间接块的更新。类似的考虑也适用于使用 tar -x 解开大型层次结构。

第二种方法是使用异步的元数据更新。这是用 mount -o async 挂载的 UFS 文件系统时的默认做法。由于所有的元数据更新也要通过缓冲区缓存，它们将与文件内容数据的更新混在一起。这个实现的好处是不需要等待每个元数据的更新被写入磁盘，所以所有引起大量元数据更新的操作都比同步情况下快得多。这个实现仍然是清晰和简单的，所以代码中出现错误的风险很低。缺点是不能保证文件系统的一致状态如果在更

[808] https://www.freebsd.org/cgi/man.cgi?query=tunefs&sektion=8&format=html
[809] https://www.freebsd.org/cgi/man.cgi?query=tunefs&sektion=8&format=html
[810] https://www.freebsd.org/cgi/man.cgi?query=fsck&sektion=8&format=html

新大量元数据的操作中出现故障，比如断电或有人按下复位按钮，文件系统将被留在不可预测的状态中。当系统再次启动时，没有机会检查文件系统的状态，因为文件的数据块可能已经被写入磁盘，而节点表或相关目录的更新却没有。因为磁盘上没有必要的信息，所以不可能使用 fsck(8)[811] 来清理由此产生的混乱。如果文件系统已经被破坏得无法修复，唯一的选择就是重新格式化并从备份中恢复。

这个问题的通常解决方案是实现 脏区记录，这也被称为 日志。元数据的更新仍然是同步写入的，但只写入磁盘的一个小区域。后来，它们被移到适当的位置。由于日志区是磁盘上的一个小的、连续的区域，即使在繁重的操作中，磁盘磁头也不会有长距离的移动，所以这些操作比同步更新要快。此外，实现的复杂性是有限的，所以出现错误的风险很低。一个缺点是，所有的元数据都要写两次，一次写到日志区域，一次写到适当的位置，所以可能导致性能“下降”。另一方面，在崩溃的情况下，所有悬而未决的元数据操作可以迅速回滚，或者在系统再次启动后从日志区域完成，从而导致文件系统的快速启动。

Berkeley FFS 的开发者 Kirk McKusick 用软更新解决了这个问题。所有悬而未决的元数据更新被保存在内存中，并以排序的顺序写入磁盘（“有序元数据更新”）。这样做的效果是，在大量元数据操作的情况下，一个项目的后期更新会“抓住”仍在内存中且尚未被写入磁盘的早期更新。在更新被写入磁盘之前，所有的操作一般都是在内存中进行的，而且数据块会根据它们的位置进行排序，这样它们就不会在磁盘上领先于它们的元数据。如果系统崩溃，一个隐含的“日志回退”会使所有没有被写入磁盘的操作看起来就像它们从未发生一样。一个一致的文件系统状态被维持，看起来是 30 到 60 秒之前的状态。所用的算法保证了所有正在使用的资源在它们的块和节点中都被标记成这样。在崩溃之后，唯一发生的资源分配错误是资源被标记为“已使用”（used），而实际上是“已释放”（free）。fsck(8)[812] 识别这种情况，并释放不再使用的资源。在文件系统崩溃后，用 `mount -f` 强行挂载文件系统的脏状态是安全的。为了释放可能未使用的资源，fsck(8)[813] 需要在稍晚的时间运行。这就是 后台 fsck(8)[814] 的想法：在系统启动时，只记录文件系统的 快照，之后再运行 fsck(8)[815]。然后，所有的文件系统都可以“dirty”挂载，所以系统的启动在多用户模式下进行。然后，后台 fsck(8)[816] 被安排在所有需要这样做的文件系统上，以释放可能未使用的资源。不使用软更新的文件系统仍然需要常规的前台 fsck(8)[817]。

其优点是元数据操作的速度几乎与异步更新一样快，而且比 记录要快，因为记录要写两次元数据。缺点是代码的复杂性，较高的内存消耗，以及一些特异性的问题。在崩溃之后，文件系统的状态显得有些“旧”。在标准的同步方法会导致一些零长度的文件在 fsck(8)[818] 之后仍然存在的情况下，这些文件在软更新中根本就不存在，因为元数据和文件内容都没有被写入磁盘。在更新被写入磁盘之前，磁盘空间不会被释放，这可能发生在运行 rm(1)[819] 之后的一段时间。当在一个没有足够空闲空间容纳所有文件的文件系统上安放大量数据时，这可能会导致问题。

[811] https://www.freebsd.org/cgi/man.cgi?query=fsck&sektion=8&format=html
[812] https://www.freebsd.org/cgi/man.cgi?query=fsck&sektion=8&format=html
[813] https://www.freebsd.org/cgi/man.cgi?query=fsck&sektion=8&format=html
[814] https://www.freebsd.org/cgi/man.cgi?query=fsck&sektion=8&format=html
[815] https://www.freebsd.org/cgi/man.cgi?query=fsck&sektion=8&format=html
[816] https://www.freebsd.org/cgi/man.cgi?query=fsck&sektion=8&format=html
[817] https://www.freebsd.org/cgi/man.cgi?query=fsck&sektion=8&format=html
[818] https://www.freebsd.org/cgi/man.cgi?query=fsck&sektion=8&format=html
[819] https://www.freebsd.org/cgi/man.cgi?query=rm&sektion=1&format=html

14.9.内核参数优化

14.9.1.文件/进程优化

14.9.1.1. kern.maxfiles

可根据系统的要求来提高或降低 sysctl(8)[820] 变量 kern.maxfiles。这个变量表示系统中文件描述符的最大数量。当文件描述符表已满时，file: table is full 会在系统消息缓冲区中反复显示，可以用 dmesg(8)[821] 查看。

每个打开的文件、套接字或 fifo 都占用一个文件描述符。大规模的生产型服务器可能很容易需要成千上万的文件描述符，这取决于同时运行的服务的种类和数量。

在较早的 FreeBSD 版本中，kern.maxfiles 的默认值来自于内核配置文件中的 maxusers。kern.maxfiles 的增长与 maxusers 的值成比例。当编译一个定制内核时，应根据系统的使用情况来设置这个内核配置选项。从这个数字来看，内核被赋予了大部分的预定义优化。即使一台生产机器可能没有 256 个并发用户，但所需要的资源可能与一个高规模的网络服务器相似。

sysctl(8)[822] 只读变量 kern.maxusers 在启动时根据系统中可用的内存量自动确定大小，也可以在运行时通过检查 kern.maxusers 的值来确定。有些系统需要更大或更小的 kern.maxusers 值，64、128 和 256 的值多为常见。除非需要大量的文件描述符，否则不建议超过 256。许多被 kern.maxusers 设置为默认值的可调整值可以在启动时或运行时在 /boot/loader.conf 中被单独重新设置。请参考 loader.conf(5)[823] 和 **/boot/defaults/loader.conf** 以了解更多细节和一些提示。

在较早的版本中，如果 maxusers 被设置为 0（注 2），系统会自动进行调整。在设置这个选项时，至少要把 maxusers 设置为 4，特别是在系统运行 Xorg 或用来编译软件的情况下。maxusers 设置的最重要的表是最大进程数，它被设置为 20 + 16 * maxusers。如果 maxusers 被设置为 1，则只能有 36 个同时进行的进程，包括系统在启动时启动的 18 个左右的进程和 Xorg 使用的 15 个左右的进程。即使是一个简单的任务，比如阅读一个手册页面，也会启动 9 个进程来过滤、解压和查看它。将 maxusers 设置为 64，最多允许 1044 个同时进行的进程，这应该足以满足几乎所有的用途。但是，如果在试图启动另一个程序时显示错误，或者服务器在运行时有大量的并发用户，请增加数量并重新编译。

> 注意
>
> maxusers 并不限制可以登录机器的用户数量。相反，考虑到系统中的最大用户数和每个用户将运行的进程数，它将各种表的大小设置为合理值。

> 注 **2**
>
> 自动调整算法将 maxusers 设定为等同系统中的内存量，最小为 32，最大为 384。

[820] https://www.freebsd.org/cgi/man.cgi?query=sysctl&sektion=8&format=html

[821] https://www.freebsd.org/cgi/man.cgi?query=dmesg&sektion=8&format=html

[822] https://www.freebsd.org/cgi/man.cgi?query=sysctl&sektion=8&format=html

[823] https://www.freebsd.org/cgi/man.cgi?query=loader.conf&sektion=5&format=html

14.9.1.2. kern.ipc.soacceptqueue

sysctl(8)[824] 变量 kern.ipc.soacceptqueue 限制了用于接受新 TCP 连接的监听队列的大小。默认值 128 通常太低，不利于稳健地处理高负荷的 web 服务器上的新连接。对于这样的环境，建议将这个值增加到 1024 或更高。像 sendmail(8)[825] 或 Apache 这样的服务可能本身就限制了监听队列的大小，但通常在其配置文件中会有一个指令来调整队列大小。大的监听队列在避免拒绝服务（DoS）攻击方面更具优势。

14.9.2.网络优化

内核配置选项 NMBCLUSTERS 决定了系统可用的网络 Mbufs 的数量。一个大量使用的服务器，如果 Mbufs 的数量很低，会妨碍性能。每个簇代表大约 2K 的内存，所以 1024 的值代表 2 兆字节的内核内存保留给网络缓冲区。可以做一个简单的计算来计算出需要多少个。一个网络服务器的最大同时连接数为 1000，每个连接使用 6K 的接收和 16K 的发送缓冲区，需要大约 32MB 的网络缓冲区来覆盖网络服务器。一个好的经验法则是乘以 2，所以 2 x 32 MB / 2 KB = 64 MB / 2 kB = 32768。对于拥有更大内存的机器，建议使用 4096 和 32768 之间的值。千万不要为这个参数随意指定一个高的值，因为它可能导致启动时崩溃。要观察网络缓冲族的使用情况，可以使用 netstat(1)[826] -m。

应该使用可调参数 kern.ipc.nmbclusters 来在启动时进行优化。只有老版本的 FreeBSD 需要使用 NMBCLUSTERS 内核选项 config(8)[827]。

对于大量使用 sendfile(2) 系统调用的繁忙服务器，可能需要通过内核配置选项 NSFBUFS 或在 **boot/loader.conf** 中设置其值来增加 sendfile(2)[828]冲区的数量 (详见 oader(8)[829])。这个参数需要优化的一个常见指标是看到当进程处于 sfbufa 状态时。sysctl(8)[830] 变量 kern.ipc.nsfbufs 是只读的。这个参数在名义上与 kern.maxusers 呈比例关系，但可能需要进行相应的调整。

> 重要
>
> 即使一个套接字被标记为非阻塞，在其上调用 sendfile(2)[831] 时仍可能会导致 sendfile(2)[832] 的调用阻塞，除非有足够的 struct sf_buf 可用。

[824] https://www.freebsd.org/cgi/man.cgi?query=sysctl&sektion=8&format=html
[825] https://www.freebsd.org/cgi/man.cgi?query=sendmail&sektion=8&format=html
[826] https://www.freebsd.org/cgi/man.cgi?query=netstat&sektion=1&format=html
[827] https://www.freebsd.org/cgi/man.cgi?query=config&sektion=8&format=html
[828] https://www.freebsd.org/cgi/man.cgi?query=sendfile&sektion=2&format=html
[829] https://www.freebsd.org/cgi/man.cgi?query=loader&sektion=8&format=html
[830] https://www.freebsd.org/cgi/man.cgi?query=sysctl&sektion=8&format=html
[831] https://www.freebsd.org/cgi/man.cgi?query=sendfile&sektion=2&format=html
[832] https://www.freebsd.org/cgi/man.cgi?query=sendfile&sektion=2&format=html

14.9.2.1. net.inet.ip.portrange.*

sysctl(8)[833] 变量 net.inet.ip.portrange.* 控制自动绑定到 TCP 和 UDP 套接字的端口号范围，其中有三个范围：低范围，默认范围和高范围。大多数网络程序使用默认范围，它由 net.inet.ip.portrange.first 和 net.inet.ip.portrange.last 控制，它们的默认值分别为 1024 和 5000。绑定的端口范围用于出站连接，在某些情况下，系统有可能运行到端口之外。这种情况最常发生在运行一个高负荷的网络代理时。当运行一个主要处理传入连接的服务器，如 Web 服务器，或有有限数量的传出连接，如邮件中转时，端口范围不是问题。对于端口不足的情况，建议适度增加 net.inet.ip.portrange.last。10000、20000 或 30000 的数值可能是合理的。在改变端口范围时要考虑到防火墙的影响。一些防火墙可能会阻止大范围的端口，通常是低编号的端口，并希望系统使用较高范围的端口进行外发连接。出于这个原因，不建议降低 net.inet.ip.portrange.first 的值。

14.9.3.虚拟内存

14.9.3.1. kern.maxvnodes

虚拟节点（vnode）是一个文件或目录的内部存在形式。增加操作系统可用的虚拟节点的数量可以减少磁盘 I/O。通常情况下，这是由操作系统处理的，不需要改变。在某些情况下，磁盘 I/O 是一个瓶颈，系统正在耗尽虚拟节点，这时需要增加虚拟节点。非活跃和空闲的内存的数量需要被考虑在内。

要查看当前正在使用的虚拟节点的数量：

```
# sysctl vfs.numvnodes
vfs.numvnodes: 91349
```

查看虚拟节点的最大数量：

```
# sysctl kern.maxvnodes
kern.maxvnodes: 100000
```

如果当前的虚拟节点使用量接近最大值，可以尝试将 kern.maxvnodes 的数值增加到 1000。应密切关注 vfs.numvnodes 的数量，如果它再次攀升到最大值，将需要进一步增加 kern.maxvnodes。除此之外，top(1)[834] 报告的内存的使用变化是可见的，并且应该有更多的内存处于活动状态。

[833] https://www.freebsd.org/cgi/man.cgi?query=sysctl&sektion=8&format=html

[834] https://www.freebsd.org/cgi/man.cgi?query=top&sektion=1&format=html

14.10.添加交换空间

有时，系统需要更多的交换空间。本节介绍了两种增加交换空间的方法：将交换空间添加到现有分区或新的硬盘上、在现有分区上创建一个交换文件。

关于如何加密交换空间、存在哪些选项以及为什么要这样做的有关信息，请参考“加密交换空间”[835]。

14.10.1.将交换空间添加到现有分区或新的硬盘上

添加一个新的硬盘用于交换，比使用现有硬盘上的分区有更好的性能。设置分区和硬盘在“添加磁盘”[836]中解释，而“设计分区布局”[837]讨论了分区布局和交换分区大小的考虑。

使用 `swapon` 向系统添加一个交换分区，比如说：

```
# swapon /dev/ada1s1b
```

> 警告
>
> 可以使用任何当前未挂载的分区，即使它已经包含数据，但在包含数据的分区上使用 `swapon` 会覆盖并破坏这些数据。在运行 `swapon` 之前，请确保被添加为交换空间的分区确实是预定的分区。

要在启动时自动使用这个交换分区，请在 **/etc/fstab** 中添加一行：

```
/dev/ada1s1b    none    swap    sw  0   0
```

关于 `/etc/fstab` 中的条目的解释，参见 fstab(5)[838] 的手册。关于 `swapon` 的更多信息可以在 swapon(8)[839] 的手册页中找到。

14.10.2.创建 Swap 文件

这个例子创建了一个 512M 的交换文件，叫做 **/usr/swap0**，而没有使用文件系统分区。

使用 swap 文件需要确保 md(4)[840] 所需要的模块已经编译在内核中，或者在启用 swap 之前已经加载。请参阅配置 FreeBSD 内核[841]以了解关于编译定制内核的信息。

例 **25.** 创建 **Swap** 文件

1. 创建 swap 文件：

[835] https://docs.freebsd.org/en/books/handbook/disks/index.html#swap-encrypting
[836] https://docs.freebsd.org/en/books/handbook/disks/index.html#disks-adding
[837] https://docs.freebsd.org/en/books/handbook/bsdinstall/index.html#configtuning-initial
[838] https://www.freebsd.org/cgi/man.cgi?query=fstab&sektion=5&format=html
[839] https://www.freebsd.org/cgi/man.cgi?query=swapon&sektion=8&format=html
[840] https://www.freebsd.org/cgi/man.cgi?query=md&sektion=4&format=html
[841] https://docs.freebsd.org/en/books/handbook/kernelconfig/index.html#kernelconfig

```
# dd if=/dev/zero of=/usr/swap0 bs=1m count=512
```

2. 给这个新文件设置合适的权限：

```
# chmod 0600 /usr/swap0
```

3. 通过向 **/etc/fstab** 添加以下一行来告知系统 swap 文件的存在：

```
md    none    swap    sw,file=/usr/swap0,late 0    0
```

4. 在系统重新启动时，交换空间将会被自动加载。要使其立即生效，使用如下 swapon(8)[842] 命令：

```
# swapon -aL
```

14.11.电源和资源管理

以一种有效的方式利用硬件资源是很重要的。电源和资源管理允许操作系统监控系统极限，并可在系统温度意外升高时提供警报。提供电源管理的一个早期规范是高级电源管理（APM）设施。APM 根据系统的活动来控制其电源的使用。然而，对于操作系统来说，管理系统的电源使用和温度属性是困难的，也不灵活。硬件是由 BIOS 管理的，用户对电源管理设置的可配置性和可见性有限。APMBIOS 是由供应商提供的，是针对特定硬件平台的。操作系统中的 APM 驱动程序提供了对 APM 的软件访问接口，通过该接口可对功率水平进行管理。

APM 有四个主要的问题：首先，电源管理是由供应商特定的 BIOS 来完成，与操作系统分离。例如，用户可以在 APMBIOS 中为硬盘设置空闲时间值，这样一来，如果超过了空闲时间，BIOS 就会在未经操作系统同意的情况下将硬盘转速降低。第二，APM 逻辑被嵌入到 BIOS 中，它在操作系统的范围之外运行。这意味着用户只能通过将新的 APMBIOS 刷入 ROM 来解决 APMBIOS 的问题，这是一个危险的程序，一旦失败，有可能使系统处于无法恢复的状态。第三，APM 是一项供应商的特有技术，这意味着存在很多重复的工作，在一个供应商的 BIOS 中发现的错误可能无法在其他供应商中得到解决。最后，APMBIOS 没有足够的空间来实现复杂的电源策略或能够很好地适应机器的用途。

在大多数情况下即插即用 BIOS（PNPBIOS）都不可靠。PNPBIOS 是 16 位的技术，所以操作系统必须使用 16 位的仿真技术才能与 PNPBIOS 方法对接。FreeBSD 提供了 APM 驱动程序，因为在 2000 年或之前制造的系统仍会使用 APM。这个驱动在 apm(4)[843] 中做了说明。

APM 的后续版本是高级配置与电源接口（ACPI）。ACPI 是一套由供应商联盟编写的标准，为硬件资源和电源管理提供一个接口。它是 在操作系统指导下的配置和电源管理的一个关键因素，因为它为操作系统提供了更多的控制方式和灵活性。

[842] https://www.freebsd.org/cgi/man.cgi?query=swapon&sektion=8&format=html

[843] https://www.freebsd.org/cgi/man.cgi?query=apm&sektion=4&format=html

本章演示了如何在 FreeBSD 上配置 ACPI。然后，它提供了一些关于如何调试 ACPI 的提示，以及如何提交包含调试信息的问题报告，以便开发人员能够诊断和修复 ACPI 问题。

14.11.1.配置 ACPI

在 FreeBSD 中，驱动 acpi(4)[844] 在系统启动时默认加载，不应该被编译到内核中。这个驱动在启动后不能被卸载，因为系统总线使用它进行各种硬件交互。然而，如果系统遇到问题，可以通过在 **/boot/loader.conf** 中设置 `hint.acpi.0.disabled="1"` 后重启，或者通过在引导器提示下设置此变量来完全禁用 ACPI，如“第三阶段”[845]所述。

> 注意
>
> ACPI 和 APM 不能共存，应该分开使用。如果驱动程序注意到另一个正在运行，最后加载的那个将停止运行。

ACPI 可以用 `acpiconf`、`-s` 标志和 1 到 5 的数字来使系统进入睡眠模式。大多数用户只需要 1（快速挂起到内存）或 3（挂起到内存）。选项 5 执行软关机，与运行 `halt -p` 相同。

驱动程序 acpi_video(4)[846] 使用 ACPI 视频扩展[847]来控制屏幕切换和背光亮度。它必须在任何 DRM 内核模块之后加载。在加载该驱动后，使用 `Fn` 亮度键可改变屏幕的亮度。可以通过检查 **/var/run/devd.pipe** 来查看 ACPI 事件：

```
...
# cat /var/run/devd.pipe
!system=ACPI subsystem=Video type=brightness notify=62
!system=ACPI subsystem=Video type=brightness notify=63
!system=ACPI subsystem=Video type=brightness notify=64
...
```

其他选项可以使用 `sysctl` 查看。更多信息请参考 acpi(4)[848] 和 acpiconf(8)[849] 的手册。

14.11.2.常见问题

ACPI 存在于所有符合 ia32（x86）和 amd64（AMD）架构的现代计算机中。完整的标准有许多功能，包括 CPU 性能管理、电源层控制、热管理、各种电池系统、嵌入式控制器和总线枚举。大多数系统实现的功能比完整标准要少。例如，台式机系统通常只实现总线枚举，而笔记本电脑可能也有冷却和电池管理支持。笔记本电脑还有挂起和恢复功能，并有其相关的复杂性。

[844] https://www.freebsd.org/cgi/man.cgi?query=acpi&sektion=4&format=html
[845] https://docs.freebsd.org/en/books/handbook/boot/index.html#boot-loader
[846] https://www.freebsd.org/cgi/man.cgi?query=acpi_video&sektion=4&format=html
[847] https://uefi.org/specs/ACPI/6.4/Apx_B_Video_Extensions/Apx_B_Video_Extensions.html
[848] https://www.freebsd.org/cgi/man.cgi?query=acpi&sektion=4&format=html
[849] https://www.freebsd.org/cgi/man.cgi?query=acpiconf&sektion=8&format=html

一个符合 ACPI 标准的系统有各种组件。BIOS 和芯片组供应商在内存中提供各种固定表，如 FADT，指定 APIC 映射（用于 SMP）、配置寄存器和简单的配置值等。此外，字节码表，即差异化系统描述表 DSDT，指定了一个树状的设备和方法的名称空间。

ACPI 驱动程序必须解析这些固定的表格，实现字节码的解释器，并修改设备驱动程序和内核以接受来自 ACPI 子系统的信息。对于 FreeBSD，Intel® 提供了一个解释器（ACPI-CA），与 Linux® 和 NetBSD 共享。ACPI-CA 源代码的路径是 **src/sys/contrib/dev/acpica**。允许 ACPI-CA 在 FreeBSD 上工作的胶水代码在 **src/sys/dev/acpica/Osd**。最后，实现各种 ACPI 设备的驱动可在 **src/sys/dev/acpica** 中找到。

要使 ACPI 正确工作，所有的部分都必须正确工作。下面是一些常见的问题，按照出现的频率排列，以及一些可能的解决方法或修复方法。如果修复方法不能解决问题，请参阅 “获取和提交调试信息”[850]以了解如何提交错误报告的说明。

14.11.2.1.鼠标问题

在某些情况下，从休眠操作中恢复会导致鼠标失败。一个已知的解决方法是在 **/boot/loader.conf** 中添加 `hint.psm.0.flags="0x3000"`。

14.11.2.2.休眠/恢复

ACPI 有三种休眠到内存（STR）的状态，`S1-S3`，和一种休眠到磁盘的状态（STD），称为 `S4`。STD 可以用两种不同的方式实现。`S4`BIOS 是一个由 BIOS 协助的挂起到磁盘的状态，`S4`OS 则完全由操作系统实现。当插上电源但不开机时，系统所处的正常状态是 “软关机”（`S5`）。

使用 `sysctl hw.acpi` 可检查与挂起有关的项目。这些例子的结果来自 Thinkpad：

```
hw.acpi.supported_sleep_state: S3 S4 S5
hw.acpi.s4bios: 0
```

使用 `acpiconf -s` 来测试 `S3`、`S4` 和 `S5`。`s4bios` 为一（1）表示 `S4`BIOS 的支持而不是 `S4` 操作系统的支持。

当测试休眠/恢复时，如果支持，从 `S1` 开始。这个状态最有可能工作，因为它不需要太多的驱动支持。没有人实现 `S2`，它与 `S1` 类似。接下来，尝试 `S3`。这是最深的 STR 状态，需要大量的驱动支持来正确地重新初始化硬件。

休眠/恢复的一个常见问题是，许多设备驱动程序不能正确地保存、恢复或重新初始化其固件、寄存器或设备内存。作为调试问题的第一次尝试，请尝试：

```
# sysctl debug.bootverbose=1
# sysctl debug.acpi.suspend_bounce=1
# acpiconf -s 3
```

[850] https://docs.freebsd.org/en/books/handbook/book/#ACPI-submitdebug

这个测试模拟了所有设备驱动程序的休眠/恢复周期，而没有实际进入 S3 状态。在某些情况下，诸如丢失固件状态、设备看门狗超时、永远重试等问题，都可以用这种方法捕捉到。注意，系统不会真正进入 S3 状态，这意味着设备可能不会失去电源，即使休眠/恢复方法完全消失，许多设备也能正常工作，这与真正的 S3 状态不同。

如果前面的测试有效，在笔记本电脑上可以配置系统在关闭盖子时休眠进入 S3，并在再次打开时恢复：

```
# sysctl hw.acpi.lid_switch_state=S3
```

这一变化可以在重启后持续进行：

```
# echo 'hw.acpi.lid_switch_state=S3' >> /etc/sysctl.conf
```

更困难的情况需要额外的硬件，例如通过串行控制台进行调试的串行端口和电缆，使用 dcons(4)[851] 的火线端口和电缆，以及内核调试技能。

为了帮助分离故障，应尽可能多地卸载驱动程序。如果它能工作，通过加载驱动程序来缩小问题所在，直到它再次失败。通常，像 **nvidia.ko** 这样的二进制驱动、显卡驱动和 USB 最有可能出现问题，而以太网接口通常工作良好。如果驱动程序可以被正确加载和卸载，可以通过在 **/etc/rc.suspend** 和 **/etc/rc.resume** 中添加适当的命令来实现自动化。如果恢复后显示混乱，试着将 hw.acpi.reset_video 设置为 1。尝试为 hw.acpi.sleep_delay 设置更长或更短的值，看看是否有帮助。

试着使用一个最新的 Linux® 发行版，看看休眠/恢复在相同的硬件上是否工作。如果它在 Linux® 上有效，那就很可能是 FreeBSD 的驱动问题。缩小哪个驱动导致问题的范围将有助于开发人员解决这个问题。由于 ACPI 的维护者很少维护其他的驱动，例如声音或 ATA，任何驱动问题都应该发布到 FreeBSD-CURRENT 邮件列表[852]中并邮寄给驱动维护者。专业用户可以在有问题的驱动程序中加入调试 printf(3)[853] 的功能，以追踪它在恢复功能的哪个地方挂掉。

最后，尝试禁用 ACPI，启用 APM。如果休眠/恢复功能在 APM 下有效，那就坚持使用 APM，特别是在旧硬件上（2000 年以前）。供应商需要花时间来纠正对 ACPI 的支持，而且旧硬件更有可能在 ACPI 上出现 BIOS 问题。

14.11.2.3.系统挂起

大多数系统挂起是由于中断丢失或中断风暴造成的。芯片组可能存在问题，具体取决于引导、BIOS 如何在 APIC (MADT) 表正确之前配置中断以及系统控制中断 (SCI) 的路由。

中断风暴可以通过检查 vmstat -i 的输出并查看有 acpi0 的那一行来区别于丢失的中断。如果计数器的增加速度超过每秒几次，就有一个中断风暴。如果系统出现挂起的情况，尝试中断到 DDB（在控制台键入 CTRL + ALT + ESC）并输入 show interrupts。

当处理中断问题时，尝试在 **/boot/loader.conf** 中用 hint.apic.0.disabled="1" 禁用 APIC 支持。

[851] https://www.freebsd.org/cgi/man.cgi?query=dcons&sektion=4&format=html

[852] https://lists.freebsd.org/subscription/freebsd-current

[853] https://www.freebsd.org/cgi/man.cgi?query=printf&sektion=3&format=html

14.11.2.4. Panic

Panic 对于 ACPI 来说是比较少见的，而且是最优先修复的。第一步是隔离再现 panic 的步骤，如果可能的话，并得到回溯信息。按照 “从串行线进入 DDB 调试器” [854]选项 DDB 和设置串行控制台的建议，或者设置一个转储分区。要在 DDB 中获得回溯信息，使用 `tr`。当手动记录回溯信息时，至少要获得回溯信息中的前五行和最后五行。

然后，尝试通过禁用 ACPI 启动来分离故障。如果成功了，通过使用 `debug.acpi.disable` 的不同值来隔离 ACPI 子系统。参见 acpi(4)[855] 的命令手册以了解一些例子。

14.11.2.5.系统在暂停和关机之后启动

首先，尝试在 **/boot/loader.conf** 中设置 `hw.acpi.disable_on_poweroff="0"`。这可以防止 ACPI 在关机过程中禁用各种事件，一些系统出于同样的原因需要将这个值设置为 1（默认值）。这通常可以解决系统在挂起或关机后自发开机的问题。

14.11.2.6. BIOS 含有带 bug 的字节码

一些 BIOS 供应商提供了不正确或有错误的字节码。这通常表现为像这样的内核控制台信息：

```
ACPI-1287: *** Error: Method execution failed [\\_SB_.PCI0.LPC0.FIGD._STA] \\
(Node 0xc3f6d160), AE_NOT_FOUND
```

通常，这些问题可以通过把 BIOS 更新到最新版本来解决。大多数控制台信息是无害的，但如果有其他问题，如电池状态不工作，这些信息是开始寻找问题的一个好地方。

14.11.3.覆盖默认的 AML

被称为 ACPI 机器语言（AML）的 BIOS 字节码是由称为 ACPI 源语言（ASL）的源语言编译而成。AML 可以在被称为差异化系统描述表（DSDT）的表格中找到。

FreeBSD 的目标是让每个人都有正常运行的 ACPI，而不需要任何用户干预。对于 BIOS 供应商所犯的常见错误，目前仍在开发解决方法。Microsoft® 解释器（**acpi.sys** 和 **acpiec.sys**）并不严格检查 AML 是否符合标准，因此许多只在 Windows® 下测试 ACPI 的 BIOS 供应商从未修复他们的 ASL。FreeBSD 的开发人员继续识别并记录哪些非标准的行为是 Microsoft® 的解释器所允许的，并对其进行复制，这样 FreeBSD 就可以在不强迫用户修正 ASL 的情况下工作。

为了帮助识别有缺陷的行为，并有可能进行手动修复，可以对系统的 ASL 进行复制。要把系统的 ASL 复制到一个指定的文件名下，使用 `acpidump -t`，显示固定表的内容，和`-d`，来反汇编 AML：

[854] https://docs.freebsd.org/en/books/handbook/serialcomms/index.html#serialconsole-ddb

[855] https://www.freebsd.org/cgi/man.cgi?query=acpi&sektion=4&format=html

```
# acpidump -td > my.asl
```

一些 AML 版本假设用户运行的是 Windows®。要覆盖这一点，在 **/boot/loader.conf** 中设置 `hw.acpi.osname="Windows 2009"`，可使用 ASL 中列出的最新的 Windows® 版本。

其他解决方法可能需要定制 **my.asl**。如果这个文件被编辑了，用下面的命令编译新的 ASL。警告通常可以被忽略，但错误是 bug，通常会妨碍 ACPI 的正常工作。

```
# iasl -f my.asl
```

包含的 `-f` 将强制创建 AML，即使在编译过程中出现了错误。一些错误，例如缺少返回语句，会由 FreeBSD 解释器自动解决。

`iasl` 的默认输出文件名是 **DSDT.aml**。应通过编辑 **/boot/loader.conf** 来加载这个文件，而非 BIOS 的错误副本，后者仍然存在于存储器中，如下所示：

```
acpi_dsdt_load="YES"
acpi_dsdt_name="/boot/DSDT.aml"
```

请确保将 **DSDT.aml** 复制到 **/boot**，然后重新启动系统。如果这能解决这个问题，请将新旧 ASL 的 diff(1) 发送到 FreeBSD ACPI 邮件列表[856]，以便开发人员能够解决 **acpica** 中的错误行为。

14.11.4.获得并提交调试信息

ACPI 驱动程序有一个灵活的调试工具。可以指定一组子系统和粗略的程度。调试的子系统被指定为层，并被分解为组件（`ACPI_ALL_COMPONENTS`）和 ACPI 硬件支持（`ACPI_ALL_DRIVERS`）。调试输出的粗略程度被指定为级别，范围从仅仅报告错误（`ACPI_LV_ERROR`）到全部（`ACPI_LV_VERBOSE`）。级别是一个位掩码，所以可以同时设置多个选项，用空格隔开。在实践中，应该用一个串行控制台来记录输出，这样它就不会因为控制台消息缓冲区的刷新而丢失。在 acpi(4)[857] 中可以找到各个层和级别的完整列表。

调试输出在默认情况下是不启用的。要启用它，如果 ACPI 被编译到内核中，请在定制内核配置文件中添加 `options ACPI_DEBUG`。在 **/etc/make.conf** 中添加 `ACPI_DEBUG=1` 来全局启用它。如果使用模块而不是定制内核，只需重新编译 **acpi.ko** 模块，如下所示：

```
# cd /sys/modules/acpi/acpi && make clean && make ACPI_DEBUG=1
```

将编译好的 **acpi.ko** 复制到 **/boot/kernel**，并在 **/boot/loader.conf** 中添加所需的级别和层次。本例中的条目启用了所有 ACPI 组件和硬件驱动的调试信息，并以最简洁的级别输出错误信息：

```
debug.acpi.layer="ACPI_ALL_COMPONENTS ACPI_ALL_DRIVERS"
debug.acpi.level="ACPI_LV_ERROR"
```

[856] https://lists.freebsd.org/subscription/freebsd-acpi

[857] https://www.freebsd.org/cgi/man.cgi?query=acpi&sektion=4&format=html

如果所需的信息是由特定的事件触发的，比如挂起后再恢复，不要修改 **/boot/loader.conf**。而应该在启动并为特定事件准备好系统后，使用 `sysctl` 来指定层和级别。使用 `sysctl` 设置的变量与 **/boot/loader.conf** 中的参数项命名相同。

如果收集到了调试信息，就可以将其发送到 FreeBSD ACPI 邮件列表[858]中，以便 FreeBSD ACPI 维护者使用这些信息来确定问题的根本原因并制定解决方案。

> 注意
>
> 在向该邮件列表提交调试信息之前，请确保安装了最新版本的 BIOS，如果有的话，也请更新嵌入式控制器固件版本。

当提交错误报告时，请包含以下信息：

- 对错误行为的摘要，包括系统类型、型号以及导致错误出现的任何东西。如果是新发生的，尽可能准确地记下该 bug 开始出现的时间。
- 运行 `boot -v` 后 `dmesg` 的输出，包括由错误产生的任何错误信息。
- 如果禁用 ACPI 有助于解决这个问题的话，那么请包含在禁用 ACPI 的情况下，得到 `boot -v` 后 `dmesg` 的输出。
- `sysctl hw.acpi` 的输出，它列出了系统提供的功能。
- 系统 ASL 的复制版本的网址链接。不要直接将 ASL 发送到列表中，因为它可能非常大。通过运行这个命令生成一份 ASL 的副本：

```
# acpidump -dt > name-system.asl
```

用登录名代替 `姓名`，用制造商/型号代替 `系统`。例如，使用 `njl-FooCo6000.asl`。

大多数 FreeBSD 开发者会关注 FreeBSD-CURRENT 邮件列表[859]，但也应该把问题提交到 FreeBSD ACPI 邮件列表[860]，以确保它被看到。在等待回复时要有耐心。如果问题没有立即显现出来，请提交一份错误报告。当输入 PR 时，请包括与上述要求相同的信息。这有助于开发人员跟踪问题并解决它。不要在没有给 FreeBSD ACPI 邮件列表[861]发邮件的情况下发送 PR，因为这个问题很可能已经被报告过了。

14.11.5.参考资料

你可以在以下地点找到更多关于 ACPI 的信息：

- 归档在 https://lists.freebsd.org/pipermail/freebsd-acpi/ 和最近的列表在 https://lists.freebsd.org/archives/freebsd-acpi/
- ACPI 规范[862]

[858] https://lists.freebsd.org/subscription/freebsd-acpi
[859] https://lists.freebsd.org/subscription/freebsd-current
[860] https://lists.freebsd.org/subscription/freebsd-acpi
[861] https://lists.freebsd.org/subscription/freebsd-acpi
[862] https://uefi.org/specifications#ACPI

- acpi(4)[863], acpi_thermal(4)[864], acpidump(8)[865], iasl(8)[866] 和 acpidb(8)[867] 的手册页面

[863] https://www.freebsd.org/cgi/man.cgi?query=acpi&sektion=4&format=html

[864] https://www.freebsd.org/cgi/man.cgi?query=acpi_thermal&sektion=4&format=html

[865] https://www.freebsd.org/cgi/man.cgi?query=acpidump&sektion=8&format=html

[866] https://www.freebsd.org/cgi/man.cgi?query=iasl&sektion=8&format=html

[867] https://www.freebsd.org/cgi/man.cgi?query=acpidb&sektion=8&format=html

第十五章 FreeBSD 的引导过程

15.1.概述

启动计算机并加载操作系统的过程被称为“引导过程”，或“引导”。FreeBSD 的引导程序对于自定义系统启动时所发生的事情提供了很大的灵活性，提供了可以选择安装在同一台计算机上的不同操作系统，同一操作系统的不同版本，或不同内核的能力。

本章详细介绍了可以配置的选项。它演示了如何定制 FreeBSD 的引导程序，包括在 FreeBSD 内核启动、探测设备和启动 init(8)[868] 之前发生的一切。这一切发生在当启动信息从明亮的白色到灰色之间。

读完本章后，你将了解：

- FreeBSD 引导程序的组件以及它们如何相互作用。
- 可以传递给 FreeBSD 引导程序中组件的选项，以控制启动过程。
- 设置 Device Hints 的基础知识。
- 如何引导到单用户和多用户模式，以及如何正确关闭 FreeBSD 系统。

> 注意
> 本章只说明 FreeBSD 在 x86 和 amd64 系统上的引导过程。

15.2.FreeBSD 的引导过程

开启计算机和启动操作系统带来了一个有趣的难题。根据定义，在操作系统启动之前，计算机不知道如何做任何事情，包括从磁盘上运行程序。如果计算机在没有操作系统的情况下那么他就不能从磁盘上运行程序，而操作系统的程序又在磁盘上，那么操作系统是如何启动的？

这个问题与《吹牛大王历险记》一书中的情节类似：一个人掉入了窖井里，通过提起鞋带（Bootstrap）就把自己拉了出来。因此在早期的计算机中，*bootstrap* 这个词就被应用于用于加载操作系统的机制。此后，它被缩短为 `booting`。

[868] https://www.freebsd.org/cgi/man.cgi?query=init&sektion=8&format=html

在 X86 硬件上，基本输入/输出系统（BIOS）负责加载操作系统。BIOS 在硬盘上寻找主引导记录（MBR），该记录必须位于磁盘上的一个特定位置。BIOS 有足够的能力来加载和运行 MBR，并假设 MBR 随后可以执行加载操作系统所涉及的其余任务，并有可能要在 BIOS 的帮助下进行。

注意

FreeBSD 提供了从旧的 MBR 标准和新的 GUID 分区表（GPT）启动的功能。GPT 分区通常出现在使用统一可扩展固件接口（UEFI）的计算机上。然而，FreeBSD 可以使用 gptboot(8)[869] 以从只有传统 BIOS 的机器上的 GPT 分区启动。目前正在努力提供直接的 UEFI 启动。

MBR 中的代码通常被称为 *boot manager*，尤其是当它与用户进行交互时。引导管理器通常在磁盘的第一个磁道或文件系统内有更多的代码。引导管理器的例子包括标准的 FreeBSD 引导管理器 `boot0`（也叫 Boot Easy），以及被许多 Linux® 发行版所使用的 GNU GRUB。

注意

GRUB 用户请参考 GNU 提供的文档[870]。

如果只安装了一个操作系统，MBR 会在磁盘上搜索第一个可引导的（活动的）分区，然后运行这个分区上的代码来加载操作系统的剩余部分。当有多个操作系统时，可以安装不同的引导管理器来显示操作系统的列表，这样用户就可以选择一个系统来启动。

FreeBSD 引导系统的其余部分被分为三个阶段：第一阶段只知道让计算机进入一个特定的状态并运行第二阶段。第二阶段可以多做一些事，然后运行第三阶段。第三阶段完成加载操作系统的任务。这项工作被分成三个阶段是因为 MBR 对第一和第二阶段可以运行的程序的大小进行了限制。将这些任务串联起来，就可以使 FreeBSD 提供一个更灵活的加载器。

内核随后被启动，同时开始探测设备并初始化它们以便使用。内核启动过程结束以后，内核将控制权移交给用户进程 init(8)[871]。它将确保磁盘处于可用状态，启动用户级资源配置，挂载文件系统，设置网卡以便在网络上通信，并启动已经配置好的启动时运行的进程。

这一节更详细地介绍了这些阶段，并演示了如何与 FreeBSD 引导程序进行交互。

15.2.1.引导管理器

MBR 中的引导管理器代码有时被称为引导过程的 第零阶段。默认情况下，FreeBSD 使用 boot0 引导管理器。

由 FreeBSD 安装程序安装的 MBR 是基于 **/boot/boot0** 的。由于分区表和 MBR 末尾的 `0x55AA` 标识符，`boot0` 的大小被限制在 446 字节。如果 `boot0` 和多个操作系统被安装，在启动时将会显示类似这个例子的信息。

例 **1. boot0** 屏幕截图

[869] https://www.freebsd.org/cgi/man.cgi?query=gptboot&sektion=8&format=html

[870] https://www.gnu.org/software/grub/grub-documentation.html

[871] https://www.freebsd.org/cgi/man.cgi?query=init&sektion=8&format=html

```
F1 Win
F2 FreeBSD

Default: F2
```

如果在 FreeBSD 之后安装其他操作系统，它们就会覆盖现有的 MBR。如果发生这种情况，或者想用 FreeBSD 的 MBR 替换现有的 MBR，请使用下面的命令：

```
# fdisk -B -b /boot/boot0 device
```

其中 *device* 代表着启动盘，比如 **ad0** 代表第一个 IDE 磁盘，**ad2** 代表第二个 IDE 控制器上的第一个 IDE 磁盘，或者 **da0** 代表第一个 SCSI 磁盘。要创建 MBR 的自定义配置，请参考 boot0cfg(8)[872]。

15.2.2.第一阶段和第二阶段

从概念上讲，第一阶段和第二阶段是同一程序的两个部分，在磁盘的同一区域。由于空间的限制，它们被分成了两个，但总是被安装在一起。它们是由 FreeBSD 安装程序或 `bsdlabel` 从合并的 **/boot/boot** 中复制出来的。

这两个阶段位于文件系统之外，在引导 slice 的第一个磁道上，从第一个扇区开始。这是 boot0 或任何其他引导管理器期望找到一个可以继续引导过程的程序的地方。

第一阶段，**boot1**，非常简单，因为它只能有 512 字节大小。它对 FreeBSD *bsdlabel* 的了解只够找到并执行 **boot2**，bsdlabel 存储了关于分片的信息。

第二阶段，**boot2**，稍微复杂一些，它对 FreeBSD 文件系统的理解足以找到文件。它可以提供一个简单的界面来选择要运行的内核或加载器。它运行更复杂的 loader，并提供一个引导配置文件。如果启动过程在第二阶段被打断，将显示以下交互式屏幕：

例 **2. boot2** 屏幕截图

```
>> FreeBSD/i386 BOOT
Default: 0:ad(0,a)/boot/loader
boot:
```

使用 `bsdlabel` 可以替换已安装的 **boot1** 和 **boot2**。其中 *diskslice* 是要启动的磁盘和分区，例如 **ad0s1** 表示第一个 IDE 磁盘上的第一个分区。

```
# bsdlabel -B diskslice
```

警告

如果只使用磁盘名称，如 **ad0**，`bsdlabel` 将在“危险专用模式”下创建磁盘（没有分区）。这可能不是我们想要的结果，所以在按下回车键之前，请仔细检查 *diskslice*。

[872] https://www.freebsd.org/cgi/man.cgi?query=boot0cfg&sektion=8&format=html

15.2.3.第三阶段

loader 是三阶段引导过程的最后阶段。它位于文件系统中，通常为 **/boot/loader**。

loader 的目的是作为一种交互式的配置方法，使用内置的命令集，由一个更强大的解释器来支持，该解释器有一个更复杂的命令集。

在初始化过程中，loader 将探测终端和磁盘，并检测出它是从哪个磁盘启动的。它将相应地设置变量，并启动一个解释器，用户命令可以用脚本或交互式的方式传递。

然后，加载器将读取 **/boot/loader.rc**。默认情况下，它会读取 **/boot/defaults/loader.conf**，并为变量设置合理的默认值。然后读取 **/boot/loader.conf** 对这些变量进行本地修改。**loader.rc** 会根据这些变量进行处理，加载所选择的模块和内核。

最后，默认情况下，loader 会用 10 秒的时间等待按键。如果没有被打断，就启动内核。如果被打断，用户会看到一个可理解的命令集的提示，在这里用户可以调整变量，卸载所有模块，加载模块，最后启动或重启。loader 内置命令[873]列出了最常用的 loader 命令。对于所有可用命令的完整讨论，请参考 loader(8)[874]。

表 1. Loader 的内置命令

[873] https://docs.freebsd.org/en/books/handbook/boot/#boot-loader-commands

[874] https://www.freebsd.org/cgi/man.cgi?query=loader&sektion=8&format=html

变量	说明
autoboot *seconds*	如果在给定的时间跨度内没有被打断，就继续启动内核，单位是秒。它显示一个倒计时，默认的时间跨度是 10 秒。
boot [-options] [kernelname]	立即启动内核，并使用任何指定的选项或内核名称。在命令行上提供内核名称只适用于已经声明过 unload 的情况。否则，将使用先前加载的内核。如果 *kernelname* 没有限定，它将在 */boot/kernel* 和 */boot/modules* 下搜索。
boot-conf	根据指定的变量（最常见的是 kernel），经历同样的自动配置模块的过程。这只有在先使用 unload，然后再改变一些变量时才有意义。
help [topic]	显示从 **/boot/loader.help** 读取的帮助信息。如果给出的 topic 是 index，则显示可用的 topic 列表。
include filename …	读取指定的文件并逐行进行解释。如果出现错误，立即停止 include。
load [-t type] filename	加载内核、内核模块或所给类型的文件，并指定文件名（filename）。在 *filename* 后面的任何参数都被传递给文件。如果 *filename* 没有限定，它将在 */boot/kernel* 和 */boot/modules* 下搜索。
ls [-l] [path]	显示给定路径中的文件清单，如果没有指定路径，则显示根目录。如果指定了 -l，文件的大小也将被显示。
lsdev [-v]	列出所有可能加载模块的设备。如果指定了 -v，就会打印更多的细节。
lsmod [-v]	显示已加载的模块。如果指定了 -v，会显示更多的细节。
more filename	显示指定的文件，每显示一行就暂停一次。
reboot	立即重启系统。
set variable, set variable=value	设置指定的环境变量。
unload	移除所有已加载的模块。

下面是一些使用加载器的实际例子。在单用户模式下启动通常使用的内核：

```
boot -s
```

卸载通常使用的内核和模块，然后加载以前使用的的或另一个指定的内核：

```
unload
load /path/to/kernelfile
```

使用限定的 **/boot/GENERIC/kernel** 来指代安装时附带的默认内核，或者使用 **/boot/kernel.old/kernel** 来指代系统升级前或配置定制内核前安装的内核。

使用下面的方法来加载其他内核的常用模块。注意，在这种情况下，不需要限定名称：

```
unload
set kernel="mykernel"
boot-conf
```

加载一个自动化的内核配置脚本：

```
load -t userconfig_script /boot/kernel.conf
```

15.2.4.最后阶段

一旦内核被 loader 或绕过 loader 直接被 boot2 加载，它就会检查所有引导标志，并根据需要调整其行为。Kernel Interaction During Boot[875] 列出了常用的引导标志。更多关于其他引导标志的信息请参考 boot(8)[876]。

表 2. 内核在启动过程中的相互作用

参数	说明
`-a`	在内核初始化过程中，要求将该设备作为根文件系统挂载。
`-C`	从 CDROM 引导根文件系统。
`-s`	引导到单用户模式。
`-v`	在内核启动过程中更加详细地说明。

内核完成了启动以后，它就把控制权交给用户进程 init(8)[877]，它位于 **/sbin/init**，或者在 `loader` 的 `init_path` 变量中指定的程序路径。这是启动过程的最后阶段。

启动顺序确保系统中可用的文件系统是一致的。如果 UFS 文件系统不一致，并且 `fsck` 不能修复不一致的地方，init 将系统降到单用户模式，这样系统管理员可以直接解决这个问题。若无以上问题，系统会启动到多用户模式。

15.2.4.1.单用户模式

用户可以通过使用 `-s` 启动或在加载器中设置 `boot_single` 变量来指定这种模式。它也可以通过在多用户模式下运行 `shutdown now` 来达到。单用户模式以这个信息开始：

```
Enter full pathname of shell or RETURN for /bin/sh:
```

如果用户按回车键，系统将进入默认的 sh。要指定其他的 shell，请输入 shell 的完整路径。

[875] https://docs.freebsd.org/en/books/handbook/boot/#boot-kernel
[876] https://www.freebsd.org/cgi/man.cgi?query=boot&sektion=8&format=html
[877] https://www.freebsd.org/cgi/man.cgi?query=init&sektion=8&format=html

单用户模式通常用于修复由于文件系统不一致或启动配置文件错误而无法启动的系统。当根密码未知时，它也可以用来重置 root 密码。这些操作是可能的，因为单用户模式的提示给了系统和它的配置文件完全的、本地的访问权。在这种模式下无网络链接。

虽然单用户模式对修复系统很有用，但它会带来安全风险，除非系统在物理上处于一个安全的位置。默认情况下，任何能够获得系统物理访问权的用户，在启动到单用户模式后，将完全控制该系统。

如果在 **/etc/ttys** 中把系统控制台改为不安全的，那么在启动单用户模式之前，系统会首先提示 root 密码。此操作增加了一种安全措施，同时取消了在 root 密码未知时对其进行重置的能力。

例 **3.** 在 **/etc/ttys** 中设置不安全的控制台

```
# 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
```

不安全的控制台意味着控制台的物理安全性被认为是不安全的，所以只有知道 root 密码的人可以使用单用户模式。

15.2.4.2.多用户模式

如果 init 发现文件系统没有问题，或者假设用户在单用户模式下完成了他们的命令并键入 exit 离开单用户模式，系统就会进入多用户模式，在这个模式下，它开始对系统进行资源配置。

资源配置系统从 **/etc/defaults/rc.conf** 中读取配置默认值，从 **/etc/rc.conf** 中读取系统的具体细节。然后，它开始装载 **/etc/fstab** 中列出的系统文件系统。它启动网络服务、各种系统守护程序，然后启动本地安装的软件包的启动脚本。

要了解更多关于资源配置系统的信息，请参考 rc(8)[878] 并检查位于 **/etc/rc.d** 中的脚本。

15.3. Device Hints

在系统初始启动时，boot loader(8)[879] 会读取 device.hints(5)[880]。这个文件存储了被称为变量的内核启动信息，有时被称为“device hints”。这些“device hints”被设备驱动用于对设备进行配置。

如 第三阶段[881] 所演示的，device hints 也可以在第三阶段引导 loader 提示下指定。变量可以用 set 来添加，用 unset 来删除，并且可以用 show 来查看。在 **/boot/device.hints** 中设置的变量也可以被覆盖。在引导 loader 中键入的 device hints 不是永久性的，不会在下次重启时应用。

[878] https://www.freebsd.org/cgi/man.cgi?query=rc&sektion=8&format=html

[879] https://www.freebsd.org/cgi/man.cgi?query=loader&sektion=8&format=html

[880] https://www.freebsd.org/cgi/man.cgi?query=device.hints&sektion=5&format=html

[881] https://docs.freebsd.org/en/books/handbook/boot/#boot-loader

系统启动之后，可以用 kenv(1)[882] 来转储所有的变量。

/boot/device.hints 的语法是每行一个变量，使用井号“#”作为注释标记。每行的格式如下：

```
hint.driver.unit.keyword="value"
```

第三阶段引导 loader 的语法是：

```
set hint.driver.unit.keyword=value
```

其中 `driver` 是设备驱动名称，`unit` 是设备驱动单元号，`keyword` 是提示关键字。关键字可以由以下选项组成：

- `at`：指定设备所连接的总线。
- `port`: 指定要使用的 I/O 的起始地址。
- `irq`：指定要使用的中断请求编号。
- `drq`：指定 DMA 通道的编号。
- `maddr`：指定设备占用的物理内存地址。
- `flags`：为设备设置各种标志位。
- `disabled`: 如果设置为 1，则设备被禁用。

由于设备驱动程序可能接受或需要更多这里没有列出的 hints，建议查看驱动程序的手册页。更多信息请参考 device.hints(5)[883]，kenv(1)[884]，loader.conf(5)[885]，以及 loader(8)[886]。

15.4.关机流程

在使用 shutdown(8)[887] 控制关机时，init(8)[888] 会尝试运行 **/etc/rc.shutdown** 脚本，然后向所有进程发送 `TERM` 信号，并随后向没有及时终止的进程发送 `KILL` 信号。

在支持电源管理的架构和系统上关闭 FreeBSD 机器的电源，只需使用`shutdown -p now` 即可立即关闭电源。要重新启动 FreeBSD 系统，使用 `shutdown -r now`。必须是 `root` 或 `operator` 成员才能运行 shutdown(8)[889]。也可以使用 halt(8)[890] 和 reboot(8)[891]。更多信息请参考它们的手册页面和 shutdown(8)[892]。

[882] https://www.freebsd.org/cgi/man.cgi?query=kenv&sektion=1&format=html
[883] https://www.freebsd.org/cgi/man.cgi?query=device.hints&sektion=5&format=html
[884] https://www.freebsd.org/cgi/man.cgi?query=kenv&sektion=1&format=html
[885] https://www.freebsd.org/cgi/man.cgi?query=loader.conf&sektion=5&format=html
[886] https://www.freebsd.org/cgi/man.cgi?query=loader&sektion=8&format=html
[887] https://www.freebsd.org/cgi/man.cgi?query=shutdown&sektion=8&format=html
[888] https://www.freebsd.org/cgi/man.cgi?query=init&sektion=8&format=html
[889] https://www.freebsd.org/cgi/man.cgi?query=shutdown&sektion=8&format=html
[890] https://www.freebsd.org/cgi/man.cgi?query=halt&sektion=8&format=html
[891] https://www.freebsd.org/cgi/man.cgi?query=reboot&sektion=8&format=html
[892] https://www.freebsd.org/cgi/man.cgi?query=shutdown&sektion=8&format=html

通过参考 “用户和基本账户管理”[893]来修改组成员资格。

注意

电源管理需要 acpi(4)[894] 作为模块加载或静态编译到定制内核中。

[893] https://docs.freebsd.org/en/books/handbook/basics/index.html#users-synopsis

[894] https://www.freebsd.org/cgi/man.cgi?query=acpi&sektion=4&format=html

第十六章安全

16.1.概述

无论是物理机还是虚拟机，安全都是一个非常广泛的话题，以至于全行业都在围绕它而发展。目前已经有数以百计的关于如何保证系统和网络安全的标准实践被撰写出来，但了解如何防范各种计算机入侵和黑客攻击仍是作为 FreeBSD 的用户必须掌握的内容。

在本章中，将讨论几个基本原理和技术。FreeBSD 系统带有多层安全性。并且能够通过添加更多的第三方实用程序来增强安全性。

读完本章，你就会知道：

- 基本的 FreeBSD 系统安全概念。
- FreeBSD 中可用的各种加密机制。
- 如何设置一次性密码身份认证。
- 如何配置 TCP Wrapper 以便与 inetd(8)[895] 协同使用。
- 如何在 FreeBSD 上设置 Kerberos。
- 如何配置 IPsec 和创建 VPN。
- 如何在 FreeBSD 上配置和使用 OpenSSH。
- 如何使用文件系统访问控制列表（ACL）。
- 如何使用 pkg 来审计从 ports 安装的第三方软件包。
- 如何利用 FreeBSD 安全公告。
- 什么是进程审计以及如何在 FreeBSD 上启用它。
- 如何使用登录分级或资源限制数据库来控制用户资源。

在阅读本章之前，你应该：

[895] https://www.freebsd.org/cgi/man.cgi?query=inetd&sektion=8&format=html

- 了解基本的 FreeBSD 和互联网概念。

本手册的其他篇章介绍了其他安全主题。例如，强制访问控制中介绍了强制访问控制[896]，网络防火墙中讨论了防火墙[897]。

16.2.简介

安全是每个人的责任。任何系统中的弱入口点都可能允许入侵者并访问关键信息，并对整个网络造成严重破坏。信息安全的核心原则之一是 CIA 三位一体，它代表信息系统的机密性（Confidentiality），完整性（Integrity）和可用性（Availability）。

CIA 三位一体是计算机安全的基本概念，因为客户和用户希望他们的数据受到保护。例如，客户希望其信用卡信息被安全存储（机密性），他们的订单不会在幕后更改（完整性），并且他们可以随时访问其订单信息（可用性）。

为了保证 CIA，专业安全人员应用了纵深防御策略。纵深防御策略的思想是多增加几层安全层，以防止单一层失效导致整个安全系统崩溃。例如，系统管理员不能简单地打开防火墙就认为网络或系统是安全的。还必须审计帐户，检查二进制文件的完整性，并确保没有安装恶意工具。要实施有效的安全战略，必须了解威胁以及如何防范它们。

什么是与计算机安全有关的威胁？威胁不仅限于试图在未经许可的情况下从远程位置访问系统的远程攻击者。威胁还包括员工、恶意软件、未经授权的网络设备、自然灾害、安全漏洞，甚至是竞争对手。

系统和网络可能在未经许可的情况下被访问，有时是意外，有时则是被远程攻击者访问，在某些情况下，通过企业间谍活动或前雇员访问。作为用户，重要的是准备和承认当一个错误导致了安全漏洞时，并向安全团队报告可能存在的问题。作为管理员，重要的是了解威胁并准备好缓解它们。

在对系统进行安全防护时，建议从保护基本帐户和系统配置开始，然后保障网络层的安全，使其符合系统安全策略和组织的安全程序。许多组织已经制定了涵盖技术设备配置的安全策略。该策略应包括工作站、台式机、移动设备、电话、生产服务器和开发服务器的安全配置。在许多情况下，标准操作程序（SOP）已经存在。当有疑问时，请询问安全团队。

本节的其余部分介绍了如何在 FreeBSD 系统上执行这些基本安全配置。本章的其余部分介绍了在 FreeBSD 系统上实施安全策略时可以使用的一些具体工具。

16.2.1.阻止登录

在确保系统安全方面，对账户进行审计是一个好的开始。确保 `root` 账户有一个健壮的密码，并且这个密码未被共享。禁用任何不需要登录权限的账户。

拒绝账户的登录访问有两种方法。第一种是锁定账户。以下是锁定 `toor` 账户的示例：

[896] https://docs.freebsd.org/en/books/handbook/mac/index.html#mac

[897] https://docs.freebsd.org/en/books/handbook/firewalls/index.html#firewalls

```
# pw lock toor
```

第二种方法是通过将其 shell 更改为 **/usr/sbin/nologin** 来阻止登录访问。只有使用超级账户可以为其他用户更改 shell：

```
# chsh -s /usr/sbin/nologin toor
```

/usr/sbin/nologin 这个 shell 可以阻止系统在用户试图登录时为其分配 shell。

16.2.2.帐户升级权限

在某些情况下，需要与其他用户共享系统管理。FreeBSD 上有两种方法可以实现。第一种方法（不推荐使用）是采用 `wheel` 组成员共同使用 `root` 密码。使用此方法时，用户在需要超级用户访问时键入 `su` 并输入 `wheel` 的密码。在完成需要管理权限的命令后，用户应该键入 `exit` 来退出用户共享。如需将一个用户添加到这个组中，请编辑 **/etc/group** 并将该用户添加到 `wheel` 条目的末尾。该用户必须用逗号字符隔开，不能有空格。

第二种方法，也是推荐的方法是通过二进制包或 port 安装 security/sudo[898]。该软件能提供额外的审计，更细粒度的进行用户控制，并且可以为锁定用户配置只能执行某个特定需要权限的命令。

安装后，使用 `visudo` 来编辑 **/usr/local/etc/sudoers**。下面的示例创建了一个新的 `webadmin` 组，将 `trhodes` 账户加入该组，并配置该组的访问权限——可以重新启动 apache24[899]：

```
# pw groupadd webadmin -M trhodes -g 6000
# visudo
%webadmin ALL=(ALL) /usr/sbin/service apache24 *
```

16.2.3.哈希密码

密码是技术的必要之恶。当必须使用时，它们应该很复杂，并且应该使用强大的哈希机制来加密存储在密码数据库中的密码。FreeBSD 在其 `crypt()` 库中支持 DES、MD5、SHA256、SHA512 和 Blowfish 哈希算法。不应该将默认的 SHA512 改成安全性较低的哈希算法，但可以改成安全性更高的 Blowfish 算法。

> 注意
>
> **Blowfish** 算法不是 **AES** 的一部分，通常被认为不符合联邦信息处理标准（**FIPS**）。在某些环境中可能不允许使用它。

为了确定用来加密用户的密码使用了哪个哈希算法。超级用户可以在 FreeBSD 密码数据库中查看用户的哈希值。每个哈希值以一个符号开始，表示用于加密密码的哈希机制的类型。如果使用的是 DES，则没有开头符号。对于 MD5, 符号是 `$`。对于 SHA256 和 SHA512, 符号是 `6$`。对于 Blowfish，符号是 `2a$`。在下面的

[898] https://cgit.freebsd.org/ports/tree/security/sudo/pkg-descr

[899] https://cgit.freebsd.org/ports/tree/apache24/pkg-descr

例子中，dru 的密码是用默认的 SHA512 算法散列的，因为哈希值以 6 开始。请注意，密码是以加密的哈希值的形式存储在密码数据库中。

```
# grep dru /etc/master.passwd
dru:$6$pzIjSvCAn.PBYQBA$PXpSeWPx3g5kscj3IMiM7tUEUSPmGexxta.
↪8Lt9TGSi2lNQqYGKszsBPuGME0:1001:1001::0:0:dru:/usr/home/dru:/bin/csh
```

哈希机制是在用户的登录分级中设置的。在下面的例子中，用户是在 default 登录分级中，哈希算法是通过 **/etc/login.conf** 中的下面的配置进行配置的：

```
:passwd_format=sha512:\
```

假如要将加密算法更改为 Blowfish，请将该行修改为如下所示：

```
:passwd_format=blf:\
```

然后按照配置登录分级[900] 中所述运行 cap_mkdb /etc/login.conf。注意，此更改不会改变任何现有的哈希密码。这意味着所有密码都应通过要求用户运行 passwd 来更改他们的密码散列值。

对于远程登录，应使用双重认证（2FA）。一个双重认证的示例是“你拥有的东西”（如密钥）和“你知道的内容”（如该密钥的密码）。由于 OpenSSH 是 FreeBSD 基础系统的一部分，因此所有网络登录都应该通过加密连接，并使用基于密钥的身份验证而非密码。有关更多信息，请参阅 OpenSSH[901]。Kerberos 用户可能需要进行其他更改才能在其网络中实现 OpenSSH。在 Kerberos[902] 中对有关的内容进行了说明。

16.2.4.密码策略实施

对本地帐户强制实施强密码策略是系统安全的一个基本方面。在 FreeBSD 中，密码长度、密码强度和密码复杂性可以通过内置的可插拔认证模块（PAM）来实现。

本节演示了如何使用 **pam_passwdqc.so** 模块进行最小和最大密码长度以及混合字符强制执行的配置。当用户更改其密码时，将强制执行此模块。

要配置此模块，请以超级用户权限取消 **/etc/pam.d/passwd** 中包含 pam_passwdqc.so 所在行的注释。然后，编辑该行以匹配密码策略：

```
password    requisite    pam_passwdqc.so    min=disabled,disabled,disabled,12,10␣
↪similar=deny retry=3 enforce=users
```

本示例为新密码设置了多个要求。min 设置控制最小密码长度。它有五个值，因为此模块根据其复杂性定义了五种不同类型的密码。复杂性由密码中必须存在的字符类型（如字母、数字、符号和大小写）定义。密

[900] https://docs.freebsd.org/en/books/handbook/security/#users-limiting
[901] https://docs.freebsd.org/en/books/handbook/security/#openssh
[902] https://docs.freebsd.org/en/books/handbook/security/#kerberos5

码类型在 pam_passwdqc(8)[903] 中进行了说明。在此示例中，前三种类型的密码被禁用，这意味着满足这些复杂性要求的密码将不被接受，无论其长度如何。12 设置了一个最小的密码策略，即如果密码也包含有三种类型的复杂性的字符，则至少要有 12 个字符。10 设置了密码策略，如果密码包含有四种类型的复杂性的字符，则也允许至少十个字符的密码。

similar 设置拒绝与用户之前的密码相似的密码。retry 设置为用户提供了三次输入新密码的机会。

保存此文件后，更改密码的用户将看到类似于以下内容的信息：

```
% passwd
Changing local password for trhodes
Old Password:

You can now choose the new password.
A valid password should be a mix of upper and lower case letters,
digits and other characters.  You can use a 12 character long
password with characters from at least 3 of these 4 classes, or
a 10 character long password containing characters from all the
classes.  Characters that form a common pattern are discarded by
the check.
Alternatively, if no one else can see your terminal now, you can
pick this as your password: "trait-useful&knob".
Enter new password:
```

如果输入的密码与策略不匹配，将被拒绝并发出警告，用户可重试，重试次数不超过配置的次数。

大多数密码策略要求密码在多少天后失效。在 FreeBSD 中设置密码时间，在 **/etc/login.conf** 中为用户的登录分级设置 passwordtime。default 登录分级包含一个示例：

```
#       :passwordtime=90d:\
```

因此，要将此登录类的有效期设置为 90 天，请删除注释符号（#），保存编辑，然后运行 cap_mkdb /etc/login.conf。

要设置单个用户的过期时间，需要传递过期日期或过期天数以及用户名给 pw：

```
# pw usermod -p 30-apr-2015 -n trhodes
```

如此处所示，到期日期以日、月和年的形式设置。有关更多信息，请参见 pw(8)[904]。

[903] https://www.freebsd.org/cgi/man.cgi?query=pam_passwdqc&sektion=8&format=html

[904] https://www.freebsd.org/cgi/man.cgi?query=pw&sektion=8&format=html

16.2.5.Rootkit 检测

任何未经授权的试图获得系统 `root` 权限的软件都是 *rootkit*。在安装后，这种恶意软件通常会给攻击者开辟另一条进入途径。现实中，一旦系统被 rootkit 入侵并进行了踩点，该系统就应该从头开始重新安装。即使是最谨慎的安全或系统工程师，也会有极大的风险遗漏攻击者留下的东西。

rootkit 对管理员来说有一个作用：一旦发现，即表明某处已被渗透。但是，此类的应用程序往往隐藏得非常好。本节展示了一个可以用来检测 rootkit 的工具——security/rkhunter[905]。

在通过二进制包或 port 安装此软件后，可使用以下命令检查系统。它将产生大量信息，并且需要手动按下回车键：

```
# rkhunter -c
```

该过程完成后，一条状态信息将被打印到屏幕上。该信息将包括检查的文件数量、可疑文件、可能的 rootkit 等等。在检查过程中，可能会产生一些关于隐藏文件、OpenSSH 协议选择和已安装软件的已知脆弱版本的通用安全警告。可以立即处理这些问题，也可以在进行了更详细的分析后处理。

每个管理员都应该知道他们所负责的系统上正在运行什么。第三方工具（如 rkhunter 和 sysutils/lsof[906]）以及本地命令如 `netstat` 和 `ps`，可以显示大量的系统信息。记下什么是正常的，当有什么东西看起来不对劲的时候要问清楚，并且要多疑。虽然预防入侵是理想的，但检测入侵是必须的。

16.2.6.二进制验证

对系统文件和二进制文件进行验证非常重要，因为它为系统管理和安全团队提供了有关系统变化的信息。监控系统变化的软件应用程序被称为入侵检测系统（IDS）。

FreeBSD 为基本的 IDS 系统提供了原生支持。虽然每晚的安全邮件会通知管理员更改的信息。但这些信息是存储在本地的。恶意用户有可能会修改这些信息以隐藏他们对系统的更改。因此，建议创建一套单独的二进制签名，并将其存储在一个只读的、root 拥有的目录下，或者最好是存储在可移动的优盘或远程 rsync 服务器上。

内置的 `mtree` 工具可以用来生成一个目录内容的规范。使用一个种子，或一个数字常数来生成规范，并被要求检查规范是否已经改变。这使得确定一个文件或二进制文件是否被修改成为可能。由于种子值对攻击者来说是未知的，伪造或检查文件的校验值将很难甚至不可能。下面的例子生成了一组 SHA256 哈希值，为 **/bin** 中的每个系统二进制文件都会生成一个，并将这些值保存在 `root` 的主目录下的一个隐藏文件中，即 **/root/.bin_chksum_mtree**。

```
# mtree -s 3483151339707503 -c -K cksum,sha256digest -p /bin > /root/.bin_chksum_mtree
# mtree: /bin checksum: 3427012225
```

3483151339707503 代表种子。应记住此值，但不要共享此值。

[905] https://cgit.freebsd.org/ports/tree/security/rkhunter/pkg-descr

[906] https://cgit.freebsd.org/ports/tree/sysutils/lsof/pkg-descr

查看 **/root/.bin_cksum_mtree** 应生成类似于以下内容的输出：

```
#           user: root
#        machine: dreadnaught
#           tree: /bin
#           date: Mon Feb  3 10:19:53 2014

# .
/set type=file uid=0 gid=0 mode=0555 nlink=1 flags=none
.                type=dir mode=0755 nlink=2 size=1024 \
                 time=1380277977.000000000
    \133         nlink=2 size=11704 time=1380277977.000000000 \
                 cksum=484492447 \
                ␣
↪sha256digest=6207490fbdb5ed1904441fbfa941279055c3e24d3a4049aeb45094596400662a
    cat          size=12096 time=1380277975.000000000 cksum=3909216944 \
                ␣
↪sha256digest=65ea347b9418760b247ab10244f47a7ca2a569c9836d77f074e7a306900c1e69
    chflags      size=8168 time=1380277975.000000000 cksum=3949425175 \
                ␣
↪sha256digest=c99eb6fc1c92cac335c08be004a0a5b4c24a0c0ef3712017b12c89a978b2dac3
    chio         size=18520 time=1380277975.000000000 cksum=2208263309 \
                ␣
↪sha256digest=ddf7c8cb92a58750a675328345560d8cc7fe14fb3ccd3690c34954cbe69fc964
    chmod        size=8640 time=1380277975.000000000 cksum=2214429708 \
                ␣
↪sha256digest=a435972263bf814ad8df082c0752aa2a7bdd8b74ff01431ccbd52ed1e490bbe7
```

此报告中包括计算机的主机名、创建规范的日期和时间以及创建规范的用户的名称。目录中的每个二进制文件都有一个校验和、大小、时间和 SHA256 摘要。

为了验证二进制签名没有变化，将目录中的当前内容与先前生成的规范进行比较，并将结果保存到文件中。这个命令需要用来生成原始规范的种子：

```
# mtree -s 3483151339707503 -p /bin < /root/.bin_chksum_mtree >> /root/.bin_chksum_
↪output
# mtree: /bin checksum: 3427012225
```

这应产生与创建规范时相同的 **/bin** 的校验和。如果这个目录中的二进制文件没有发生变化，那么 **/root/.bin_chksum_output** 输出文件将是空的。为了模拟变化，用 touch 改变 **/bin/cat** 的日期，然后再次运行验证命令。

```
# touch /bin/cat
# mtree -s 3483151339707503 -p /bin < /root/.bin_chksum_mtree >> /root/.bin_chksum_
```

(continues on next page)

(continued from previous page)

```
↪output
# more /root/.bin_chksum_output
cat changed
    modification time expected Fri Sep 27 06:32:55 2013 found Mon Feb  3 10:28:43 2014
```

建议为包含二进制文件和配置文件的目录以及包含敏感数据的任何目录都创建规范。一般可为 **/bin**、**/sbin**、**/usr/bin**、**/usr/sbin**、**/usr/local/bin**、**/etc** 和 **/usr/local/etc** 创建规范。

还有更高级的 IDS 系统存在，例如 security/aide[907]。在大多数情况下，`mtree` 提供了管理员需要的功能。保持种子值和校验输出不被恶意用户发现是很重要的。关于 `mtree` 的更多信息可以在 mtree(8)[908] 中找到。

16.2.7.系统安全优化

在 FreeBSD 中，许多系统特性都可以使用 `sysctl` 进行调整。本节将介绍一些可以优化以阻止拒绝服务（DoS）攻击的安全功能。关于使用 `sysctl` 的更多信息，包括如何临时改变数值，以及如何在测试后使改变永久化，请参阅 “使用 sysctl(8) 进行优化”[909]。

> 注意
>
> 任何时候用 `sysctl` 改变一个设置，都会使造成不想要的伤害的可能性上升，影响系统的可用性。应该监控所有的改变，如果可能的话，在生产系统上使用之前，应该在测试系统上进行测试。

默认情况下。FreeBSD 内核启动时的安全级别为 `-1`。这被称为“不安全模式”。因为不可变的文件标志可以被关闭，所有设备都可以被读出或写入。除非通过 `sysctl` 或启动脚本的设置来改变安全级别，否则安全级别将保持为 `-1`。通过在 **/etc/rc.conf** 中设置 `kern_securelevel_enable` 为 `YES`，并将 `kern_securelevel` 的值设为所需的安全级别，可以在系统启动时提高安全级别。参见 security(7)[910] 和 init(8)[911] 以了解更多关于这些设置和可用安全级别的信息。

> 警告
>
> 提升 `securelevel` 可能会破坏 Xorg 并造成其他问题。应该准备好做一些调试工作。

`net.inet.tcp.blackhole` 和 `net.inet.udp.blackhole` 设置可以用来在关闭的端口上丢弃传入的 SYN 数据包，而不发送返回的 RST 响应。默认行为是返回 RST 以显示一个端口已关闭。更改默认值可提供针对端口扫描的某种级别的保护，用于确定系统上运行的应用程序。将 `net.inet.tcp.blackhole` 设置为 `2`，将 `net.inet.udp.blackhole` 设置为 `1`。请参考 blackhole(4)[912]以了解关于这些设置的更多信息。

`et.inet.icmp.drop_redirect` 和 `net.inet.ip.redirect` 设置有助于防止 重定向攻击。重定向攻击是 DoS 的一种类型，它发送大量的 ICMP 类型 5 数据包。由于不需要这些数据包，请将 `net.inet.icmp.drop_redirect` 设置为 `1`，将 `net.inet.ip.redirect` 设置为 `0`。

[907] https://cgit.freebsd.org/ports/tree/security/aide/pkg-descr
[908] https://www.freebsd.org/cgi/man.cgi?query=mtree&sektion=8&format=html
[909] https://docs.freebsd.org/en/books/handbook/config/index.html#configtuning-sysctl
[910] https://www.freebsd.org/cgi/man.cgi?query=security&sektion=7&format=html
[911] https://www.freebsd.org/cgi/man.cgi?query=init&sektion=8&format=html
[912] https://www.freebsd.org/cgi/man.cgi?query=blackhole&sektion=4&format=html

源路由是一种检测和访问内部网络中不可路由的地址的方法。应该禁用他，因为不可路由的地址通常都是故意不可路由的。要禁用这个功能，请将 `net.inet.ip.sourceroute` 和 `net.inet.ip.accept_sourceroute` 设置为 `0`。

当网络上的一台机器需要向子网中的所有主机发送消息时，会向广播地址发送一个 ICMP 回声请求消息。然而，外部主机没有理由进行这样的操作。要拒绝所有的外部广播请求，把 `net.inet.icmp.bmcastecho` 设置为 `0`。

另外 security(7)[913] 中记录了一些其他设置。

16.3.TCP Wrapper

TCP Wrapper 是一个基于主机的访问控制系统，它扩展了 “inetd 超级服务器” [914] 的功能。它可以被配置成给 inetd 控制下的服务器守护进程提供日志支持、返回信息和连接限制。有关 TCP Wrapper 及其功能的更多信息，请参阅 tcpd(8)[915]。

TCP Wrapper 不应被视为是正确配置的防火墙的替代品。相反，TCP Wrapper 应该与防火墙和其他安全改进措施一起使用，以便在实施安全策略时提供另一层保护。

16.3.1.初始配置

要在 FreeBSD 中启用 TCP Wrapper，请将以下行添加到 **/etc/rc.conf** 中：

```
inetd_enable="YES"
inetd_flags="-Ww"
```

然后，正确配置 **/etc/hosts.allow** 即可。

> 注意
>
> 与 TCP Wrapper 的其他实现不同，在 FreeBSD 中无法使用 **hosts.deny**。所有配置选项都应放在 **/etc/hosts.allow** 中。

在最简单的配置中，守护程序连接策略是设置为允许或阻止，具体取决于 **/etc/hosts.allow** 中的选项。FreeBSD 中的默认配置是允许所有与以 inetd 启动的守护程序的连接。

基本配置通常采用的形式为 `daemon : address : action`，其中 `daemon` 是 inetd 启动的守护程序，`address` 是有效的主机名、IP 地址或括在方括号（[ ]）中的 IPv6 地址，并且 `action` 是 `allow` 或 `deny`。TCP Wrapper 使用第一条规则匹配语义，即从头扫描配置文件以寻找匹配规则。当找到一个匹配规则时，该规则被应用，搜索过程停止。

例如，要允许通过 mail/qpopper[916] 守护程序进行 POP3 连接，应将以下行追加到 **hosts.allow**：

[913] https://www.freebsd.org/cgi/man.cgi?query=security&sektion=7&format=html

[914] https://docs.freebsd.org/en/books/handbook/network-servers/index.html#network-inetd

[915] https://www.freebsd.org/cgi/man.cgi?query=tcpd&sektion=8&format=html

[916] https://cgit.freebsd.org/ports/tree/mail/qpopper/pkg-descr

```
# This line is required for POP3 connections:
qpopper : ALL : allow
```

每次编辑此文件后，请重新启动 inetd：

```
# service inetd restart
```

16.3.2.高级配置

TCP Wrapper 提供了高级选项，允许更好地控制连接的处理方式。在某些情况下，向某些主机或守护程序连接返回信息可能是合适的。在其他情况下，应记录日志条目或向管理员发送电子邮件。其他情况可能仅需要将服务用于本地连接。这一切都可以通过使用称为通配符、扩展字符和外部命令执行的配置选项来实现。

假设出现了这样一种情况：拒绝连接，但应向试图建立该连接的主机发送原因。`twist` 是可能的。当尝试连接时，`twist` 执行一个 shell 命令或脚本。**hosts.allow** 中有一个示例：

```
# The rest of the daemons are protected.
ALL : ALL \
    : severity auth.info \
    : twist /bin/echo "You are not welcome to use %d from %h."
```

在此示例中，对于未在 **hosts.allow** 中配置的任何守护程序，将返回消息 “You are not allowed to use daemon name from hostname”。这对于在建立的连接断开后立即将答复发送回连接发起方非常有用。返回的任何消息都必须用引号（"）字符括起来。

> 警告
> 如果攻击者用连接请求淹没这些守护程序，则可能会在服务器上发起拒绝服务攻击。

另一种可能性是使用 `spawn`。和 `twist` 一样，`spawn` 隐式地拒绝连接，可用于运行外部 shell 命令或脚本。与 `twist` 不同，`spawn` 不会向建立连接的主机发送回复。例如，思考以下配置：

```
# 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
```

这将拒绝来自 `*.example.com` 的所有连接尝试，并将主机名、IP 地址和试图访问的守护进程记录到 **/var/log/connections.log**。这个例子使用了替换字符 `%a` 和 `%h`。有关完整列表，请参阅 hosts_access(5)[917]。

要匹配守护进程、域或 IP 地址的每个实例，请使用 `ALL`。另一个通配符是 `PARANOID`，它可能用于匹配任何提供 IP 地址的主机，这些 IP 地址可能是伪造的，因为 IP 地址与其解析的主机名不同。在这个例子中，

[917] https://www.freebsd.org/cgi/man.cgi?query=hosts_access&sektion=5&format=html

所有发送到 sendmail 的连接请求，如果其 IP 地址与主机名不同，都将被拒绝:

```
# Block possibly spoofed requests to sendmail:
sendmail : PARANOID : deny
```

当心

如果客户或服务器的 DNS 设置有问题，使用通配符将导致拒绝连接。

要了解有关通配符及其关联功能的更多信息，请参阅 hosts_access(5)[918]。

注意

当添加新的配置行时，确保在 **hosts.allow** 中注释掉该守护进程的任何不需要的条目。

16.4.Kerberos

Kerberos 是一个网络认证协议，最初由麻省理工学院（MIT）创建，作为一种在潜在的敌对网络中安全地提供认证的方式。Kerberos 协议使用了强大的加密技术，因此客户端和服务器都可以证明他们的身份，而无需在网络上发送任何未加密的秘密。Kerberos 可以被当成一个身份验证代理系统或一个可信的第三方认证系统。在用户通过 Kerberos 认证后，他们的通信会被加密以保证隐私和数据的完整性。

Kerberos 唯一的功能是为网络上的用户和服务器提供安全认证。它并不提供授权或审计功能。建议将 Kerberos 与其他提供授权和审计服务的安全方法一起使用。

如 RFC 4120 中所述，该协议的当前版本是第 5 版。在大部分的操作系统上都有该协议的几种免费实现可用。麻省理工学院持续开发他们的 Kerberos 软件。它在美国通常用作加密产品，并且历来受美国出口法规的约束。在 FreeBSD 中，MITKerberos 可通过二进制包或 port security/krb5[919] 获取。Heimdal Kerberos 的实现是明确的在美国境外开发，以避免出口管制。Heimdal Kerberos 软件包含在 FreeBSD 基本系统中。而另一个具有更多可配置选项的软件在 ports 中以 security/heimdal[920] 提供。

在 Kerberos 中，用户和服务被识别为“主体”（principal），它们被包含在一个管理组中，称为“领域”（realm）。典型的用户的形式是 **user@REALM**（传统上领域是大写的）。

本节提供了如何使用 FreeBSD 中包含的 Heimdal 发行版来设置 Kerberos 的指南。

为了演示 Kerberos 安装，命名空间将如下所示：

- DNS 域名（区域）是 `example.org`。
- Kerberos 领域是 `EXAMPLE.ORG`。

注意

设置 **Kerberos** 时应使用真实域名，即使它将在内部运行。这样可以避免出现 **DNS** 问题，并确保与其他 **Kerberos** 域进行交互。

[918] https://www.freebsd.org/cgi/man.cgi?query=hosts_access&sektion=5&format=html

[919] https://cgit.freebsd.org/ports/tree/security/krb5/pkg-descr

[920] https://cgit.freebsd.org/ports/tree/security/heimdal/pkg-descr

16.4.1.建立 Heimdal KDC

密钥分发中心（KDC）是 Kerberos 提供的集中式身份验证服务，是系统的“受信的第三方”。是计算机发出 Kerberos 凭据，这些凭据用于客户端向服务器进行身份验证。由于 KDC 被 Kerberos 域的所有其他计算机视为可信，因此它加剧了安全问题。应限制对 KDC 的直接访问。

虽然运行 KDC 只需要很少的计算资源。但出于安全考虑。建议使用一台专用机器充当 KDC。

首先，请按如下方式安装软件包 security/heimdal[921] ：

```
# pkg install heimdal
```

接下来，使用 `sysrc` 更新 **/etc/rc.conf**，如下所示:

```
# sysrc kdc_enable=yes
# sysrc kadmind_enable=yes
```

接下来，编辑 **/etc/krb5.conf**，如下所示：

```
[libdefaults]
    default_realm = EXAMPLE.ORG
[realms]
    EXAMPLE.ORG = {
    kdc = kerberos.example.org
    admin_server = kerberos.example.org
    }
[domain_realm]
    .example.org = EXAMPLE.ORG
```

在此示例中，KDC 将使用完全限定主机名 `kerberos.example.org`。KDC 的主机名必须在 DNS 中可解析。

Kerberos 也可以使用 DNS 来定位 KDC。而不是在 **/etc/krb5.conf** 中的 [realms] 部分。对于拥有自己的 DNS 服务器的大型组织来说。上面的例子可以修改为：

```
[libdefaults]
      default_realm = EXAMPLE.ORG
[domain_realm]
    .example.org = EXAMPLE.ORG
```

在 `example.org` 区域文件中加入以下几行：

[921] https://cgit.freebsd.org/ports/tree/security/heimdal/pkg-descr

```
_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
```

注意

为了使客户端能够找到 Kerberos 服务，它们必须具有完整配置的 **/etc/krb5.conf** 或最简配置的 **/etc/krb5.conf**，并且正确的配置 DNS 服务器。

接下来，创建 Kerberos 数据库，它包含了所有主体（用户和主机）的密钥，并以主密码加密。不需要记住这个密码，因为它将被保存在 **/var/heimdal/m-key** 中；为此，使用一个 45 个字符的随机密码比较合适。要创建主密钥，请运行 `kstash` 并输入一个密码。

```
# kstash
Master key: xxxxxxxxxxxxxxxxxxxxxx
Verifying password - Master key: xxxxxxxxxxxxxxxxxxxxxx
```

主密钥被创建后，就应该初始化数据库。Kerberos 管理工具 kadmin(8)[922] 可以在 KDC 上以 `kadmin -l` 的模式直接操作数据库。而不必使用 kadmind(8)[923] 网网络服务。这就解决了在数据库创建之前试图连接到数据库的鸡生蛋蛋生鸡的问题。在 `kadmin` 提示下，使用 `init` 来创建领域的初始数据库：

```
# kadmin -l
kadmin> init EXAMPLE.ORG
Realm max ticket life [unlimited]:
```

最后，在 `kadmin` 中，使用 `add` 创建第一个主体。现在坚持使用主体的默认选项，因为这些选项可以在以后用 `modify` 修改。在提示符下输入 `?` 来查看可用的选项。

```
kadmin> add tillman
Max ticket life [unlimited]:
Max renewable life [unlimited]:
Principal expiration time [never]:
Password expiration time [never]:
Attributes []:
Password: xxxxxxxx
Verifying password - Password: xxxxxxxx
```

接下来，通过运行以下命令启动 KDC 服务：

[922] https://www.freebsd.org/cgi/man.cgi?query=kadmin&sektion=8&format=html

[923] https://www.freebsd.org/cgi/man.cgi?query=kadmind&sektion=8&format=html

```
# service kdc start
# service kadmind start
```

虽然此时不会运行任何 kerberized 守护程序，但可以通过获取刚刚创建的主体的凭据来确认 KDC 是否正在运行：

```
% kinit tillman
tillman@EXAMPLE.ORG's Password:
```

使用 `kilist` 命令确认已成功获取凭据：

```
% klist
Credentials cache: FILE:/tmp/krb5cc_1001
    Principal: tillman@EXAMPLE.ORG

  Issued                Expires               Principal
Aug 27 15:37:58 2013  Aug 28 01:37:58 2013  krbtgt/EXAMPLE.ORG@EXAMPLE.ORG
```

测试完成后，可以销毁临时凭据：

```
% kdestroy
```

16.4.2.配置服务器使用 Kerberos

将服务器配置为使用 Kerberos 身份验证的第一步是确保它在 **/etc/krb5.conf** 中具有正确的配置。KDC 中的版本可以按原样使用，也可以在新系统上重新生成。

接下来。在服务器上创建 **/etc/krb5.keytab**。这是“Kerberizing”服务的主要部分——它相当于生成一个在服务和 KDC 之间共享的秘密。这个秘密是存储在“keytab”中的一个加密密钥。keytab 包含了服务器的主机密钥，它允许服务器和 KDC 互相验证对方的身份。它必须以安全的方式传送给服务器，因为如果钥匙被公开，服务器的安全就会被破坏。通常，**keytab** 是在管理员的信任机器上使用 `kadmin` 生成的，然后安全传输（例如使用 scp(1)[924]）到服务器；如果符合所需的安全策略，也可以直接在服务器上创建。以安全的方式将密钥表传输到服务器是非常重要的：如果密钥被其他方知道，该方可以冒充任何用户到服务器上! 在服务器上直接使用 `kadmin` 是很方便的。因为 KDC 数据库中的主机负责人条目也是用 `kadmin` 创建的。

当然，`kadmin` 是一个 `kerberized` 服务；需要 `Kerberos` 凭据来验证网络服务。但为了确保运行 `kadmin` 的用户确实存在 (并且他们的会话没有被劫持)。`kadmin` 会提示密码以获得一个新的凭据。正如在 **/var/heimdal/kadmind.acl** 中指定的那样，验证 kadmin 服务的主体必须被允许使用 `kadmin` 接口。关于设计访问控制列表的细节，请参见 `info heimdal` 中题为“远程管理”的章节。管理员可以通过本地控制台或 ssh(1)[925] 安全地连接到 KDC。并使用 `kadmin -l` 进行本地管理，而不是启用远程 `kadmin` 访问。

[924] https://www.freebsd.org/cgi/man.cgi?query=scp&sektion=1&format=html
[925] https://www.freebsd.org/cgi/man.cgi?query=ssh&sektion=1&format=html

安装 **/etc/krb5.conf** 后，在 `kadmin` 中使用 `add --random-key`。这将把服务器的主机主密钥添加到数据库中。但不会将主机主密钥的副本提取到钥匙串中。要生成钥匙串，请使用 `ext` 来提取服务器的主机主密钥到它自己的钥匙串中：

```
# kadmin
kadmin> add --random-key host/myserver.example.org
Max ticket life [unlimited]:
Max renewable life [unlimited]:
Principal expiration time [never]:
Password expiration time [never]:
Attributes []:
kadmin> ext_keytab host/myserver.example.org
kadmin> exit
```

注意，`ext_keytab` 默认将提取的密钥存储在 **/etc/krb5.keytab** 中。当运行在支持 kerberos 的服务器上时，这很好，但是当在其他地方提取 keytab 时，应该使用 `--keytab` *path/to/file* 参数：

```
# kadmin
kadmin> ext_keytab --keytab=/tmp/example.keytab host/myserver.example.org
kadmin> exit
```

然后，可以使用 scp(1)[926] 或可移动设备将密钥表安全地复制到服务器。请务必指定非默认键表名称，以避免将不需要的键插入到系统的键表中。

此时，服务器可以使用存储在 **krb5.keytab** 中的共享密钥从 KDC 读取加密邮件。现在，它已准备好启用使用 Kerberos 的服务。最常见的此类服务之一是 sshd(8)[927]，它通过 GSS-API 支持 Kerberos。在 **/etc/ssh/sshd_config** 中，添加以下行：

```
GSSAPIAuthentication yes
```

进行此更改后，必须重新启动 sshd(8)[928] 才能使新配置生效：`service sshd restart`。

16.4.3.配置客户端使用 Kerberos

与服务器的情况一样，客户端也需要在 **/etc/krb5.conf** 中进行配置。将该文件复制到原地 (安全地) 或根据需要重新输入。

通过在客户机上使用 `kinit`、`klist` 和 `kdestroy` 来获取、显示并删除现有主体的凭据来测试客户机。Kerberos 应用程序也应该能够连接到支持 Kerberos 的服务器。如果这不起作用，而获取 ticket 却起作用。那么问题可能出在服务器上，而不是在客户机或 KDC 上。在使用 Kerberized ssh(1)[929] 的情况下，GSS-API 是默

[926] https://www.freebsd.org/cgi/man.cgi?query=scp&sektion=1&format=html
[927] https://www.freebsd.org/cgi/man.cgi?query=sshd&sektion=8&format=html
[928] https://www.freebsd.org/cgi/man.cgi?query=sshd&sektion=8&format=html
[929] https://www.freebsd.org/cgi/man.cgi?query=ssh&sektion=1&format=html

认禁用的。所以请使用 `ssh -o GSSAPIAuthentication=yes hostname` 进行测试。

在测试 Kerberized 应用程序时。请尝试使用诸如 `tcpdump` 这样的数据包嗅探器来确认没有敏感信息是以明文方式发送的。

有各种 Kerberos 客户端应用程序可用。随着桥的出现，使用 SASL 进行认证的应用程序也可以使用 GSS-API 机制。大类客户应用程序可以使用 Kerberos 进行认证——从 Jabber 客户端到 IMAP 客户端。

领域内的用户通常将他们的 Kerberos 主体映射到一个本地用户帐户。偶尔，我们需要把对本地用户帐户的访问权授予那些没有相应 Kerberos 主体的人。例如，`tillman@EXAMPLE.ORG` 可能需要访问本地用户帐户 `webdevelopers`，其他主体也可能需要访问该本地帐户。

放在用户主目录下的 **.k5login** 和 **.k5users** 文件可以用来解决这个问题。例如，如果将下面的 **.k5login** 放在 `webdevelopers` 的主目录中，那么列出的两个主体都可以访问该帐户，而无需共享密码：

```
tillman@example.org
jdoe@example.org
```

有关 **.k5users** 的更多信息，请参阅 ksu(1)[930]。

16.4.4.与 MIT 实现的差异

MIT 和 Heimdal 实现之间的主要区别在于 `kadmin` 有一套不同的、但又等价的命令，并使用不同的协议。如果 KDC 是 MIT 的，则 `Heimdal` 版本的 `kadmin` 就不能用来远程管理 KDC，反之亦然。

客户端应用程序也可能使用略有不同的命令行选项来完成同样的任务。建议遵循 http://web.mit.edu/Kerberos/www/ 上的说明。要注意路径问题：port 中的 MIT 默认会安装到 **/usr/local/**。如果 PATH 中将系统目录列在前面，FreeBSD 版本的系统应用程序会代替 MIT 版本运行。

当在 FreeBSD 上使用 MIT Kerberos 作为 KDC 时。还应该对 **rc.conf** 进行以下编辑：

```
kdc_program="/usr/local/sbin/kdc"
kadmind_program="/usr/local/sbin/kadmind"
kdc_flags=""
kdc_enable="YES"
kadmind_enable="YES"
```

[930] https://www.freebsd.org/cgi/man.cgi?query=ksu&sektion=1&format=html

16.4.5.Kerberos 提示、技巧和故障排除

在配置 Kerberos 并进行故障排除时，请记住以下几点：

- 当通过 ports 使用 Heimdal 或 MITKerberos 时，确保 `PATH` 在系统版本之前列出 port 的客户应用程序的版本。
- 如果领域中的所有计算机都没有同步时间设置，则身份验证可能会失败。 "NTP 的时钟同步" [931] 介绍了如何使用 NTP 同步时钟。
- 如果更改了主机名，则必须更改 `host/` 主体并更新 keytab。这也适用于特殊的 keytab 条目，如 `HTTP/` 主体用于 Apache 的 www/mod_auth_kerb[932]。
- 领域中的所有主机必须在 DNS 中可以正向和反向解析，或者至少在 **/etc/hosts** 中存在。CNAME 可以工作，但 A 和 PTR 记录必须是正确的。对于无法解析的主机，错误消息并不直观：`Kerberos5 refuses authentication because Read req failed: Key table entry not found`。
- 一些作为 KDC 客户的操作系统没有将 `ksu` 的权限设置为 setuid `root`。这意味着 `ksu` 不能工作。这是权限问题，而非 KDC 的错误。
- 在 MITKerberos 中。要让一个主体的凭证寿命超过默认的 10 小时。可以在 kadmin(8)[933] 提示下使用 `modify_principal` 来改变相关主体和 `krbtgt` 主体的最大寿命。然后。主体可以使用 `kinit -l` 来请求一个寿命更长的凭据。
- 在工作站上运行 `kinit` 时，在 KDC 上运行数据包嗅探器以帮助排除故障时，即使在输入密码之前，也会立即发送凭据授予凭据（Ticket Granting Ticket，TGT）。这是因为 Kerberos 服务器会对任何未经授权的请求自由地发送 TGT。然而，每一个 TGT 都被加密在一个从用户的密码衍生出来的密钥中。当用户输入他们的密码时，它不会被发送到 KDC，而是被用来解密 `kinit` 已经获得的 TGT。如果解密过程的结果是带有有效时间戳的有效凭据，那么该用户就拥有有效的 Kerberos 凭证。这些凭证包括一个会话密钥，用于在将来与 Kerberos 服务器建立安全通信，以及实际的 TGT，它是用 Kerberos 服务器自己的密钥加密的。这第二层加密允许 Kerberos 服务器验证每个 TGT 的真实性。
- 主机主体可以具有更长的凭据生存期。如果用户主体的生存期为一周，但要连接到的主机的生存期为 9 小时，则用户缓存将具有过期的主机主体，并且凭据缓存将无法按预期工作。
- 在设置 **krb5.dict** 以防止像 kadmind(8)[934] 中所述使用特定的错误密码时，请记住，它仅适用于分配了密码策略的主体。**krb5.dict** 中使用的格式是每行一个字符串，创建指向 **/usr/share/dict/words** 的符号链接可能很有用。

[931] https://docs.freebsd.org/en/books/handbook/network-servers/index.html#network-ntp

[932] https://cgit.freebsd.org/ports/tree/www/mod_auth_kerb/pkg-descr

[933] https://www.freebsd.org/cgi/man.cgi?query=kadmin&sektion=8&format=html

[934] https://www.freebsd.org/cgi/man.cgi?query=kadmind&sektion=8&format=html

16.4.6.缓解 Kerberos 限制

由于 Kerberos 是一种全有或全无的方法，因此必须修改网络上启用的每个服务才能与 Kerberos 配合使用，否则必须保护其免受网络攻击。这是为了防止用户凭据被盗和重复使用。例如，在所有远程 shell 上启用了 Kerberos，但非 Kerberized POP3 邮件服务器以纯文本形式发送密码。

KDC 是单点故障。根据设计，KDC 必须与其主密码数据库一样安全。KDC 上绝对不应运行其他服务，并且应该在物理上是安全的。其重大危险在于 Kerberos 存储了使用相同主密钥加密的所有密码，该主密钥作为文件存储在 KDC 上。

主密钥的泄露并不像人们担心的那么糟糕：主密钥仅用于加密 Kerberos 数据库，并用作随机数生成器的种子。只要对 KDC 的访问是安全的，攻击者就无法使用主密钥执行太多操作。

如果 KDC 不可用，则网络服务不可用，因为无法执行身份验证。这可以通过单个主 KDC 和一个或多个从站以及使用 PAM 谨慎小心实现辅助或回退身份验证来缓解。

Kerberos 允许用户、主机和服务在它们之间进行认证。它没有一个机制可以让 KDC 对用户、主机或服务进行认证。这意味着一个被木马化的 `kinit` 可以记录所有的用户名和密码。像 security/tripwire[935] 可以缓解这种情况。

16.4.7.资源和更多信息

- Kerberos 常见问题解答[936]
- 设计身份验证系统：四个场景中的对话[937]
- RFC 4120，Kerberos 网络身份验证服务（V5）[938]
- MIT Kerberos 主页[939]
- Heimdal Kerberos 项目维基页面[940]

16.5.OpenSSL

OpenSSL 是 SSL 与 TLS 协议的开源实现。它在普通通信层之上提供了一个加密传输层，可与许多网络应用程序和服务交织在一起。

FreeBSD 中包含的 OpenSSL 版本支持传输层安全性协议 1.0/1.1/1.2/1.3（TLSv1/TLSv1.1/TLSv1.2/TLSv1.3）网络安全协议，可以用作通用加密库。

[935] https://cgit.freebsd.org/ports/tree/security/tripwire/pkg-descr
[936] http://www.faqs.org/faqs/Kerberos-faq/general/preamble.html
[937] http://web.mit.edu/Kerberos/www/dialogue.html
[938] https://www.ietf.org/rfc/rfc4120.txt
[939] http://web.mit.edu/Kerberos/www/
[940] https://github.com/heimdal/heimdal/wiki

OpenSSL 通常用于加密邮件客户端的身份验证，并保护基于 Web 的交易，如信用卡支付。一些 port，如 www/apache24[941] 和 databases/postgresql11-server[942]，包含一个用于使用 OpenSSL 进行编译的编译选项。如果选定该 port，该 port 将从基本系统添加使用对 OpenSSL 的支持。要让 port 从 security/openssl[943] 获取 OpenSSL 并进行编译，请将以下内容添加到 **/etc/make.conf**：

```
DEFAULT_VERSIONS+= ssl=openssl
```

OpenSSL 的另一个常见用途是提供用于软件应用的证书。证书可以用来验证一个公司或个人的资质。如果一个证书没有经过外部 证书授权机构（CA）（如 http://www.verisign.com）的签名，使用该证书的应用程序将产生一个警告。获得签名证书是有成本的，但是使用签名证书并不是强制性的，因为证书可以是自签的。然而，使用外部权威机构可防止警告，可以让用户放心。

本节将演示如何在 FreeBSD 系统上创建和使用证书。有关如何创建用于签署自己的证书的 CA 的示例，请参阅 “配置 LDAP 服务器” [944]。

有关 SSL 的更多信息，请阅读免费的 OpenSSL 说明书[945]。

16.5.1.生成证书

若要生成将由外部 CA 签名的证书，请执行以下命令并输入在提示符下请求的信息。此输入信息将写入证书。在 `Common Name` 提示符下，输入将使用该证书的系统的完全限定域名。如果此域名与服务器不匹配，则验证证书的应用程序将向用户发出警告，使证书提供的验证变得无用。

```
# openssl req -new -nodes -out req.pem -keyout cert.key -sha256 -newkey rsa:2048
Generating a 2048 bit RSA private key
..................+++
.............................................................+++
writing new private key to 'cert.key'
-----
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
```

(continues on next page)

[941] https://cgit.freebsd.org/ports/tree/www/apache24/pkg-descr
[942] https://cgit.freebsd.org/ports/tree/databases/postgresql11-server/pkg-descr
[943] https://cgit.freebsd.org/ports/tree/security/openssl/pkg-descr
[944] https://docs.freebsd.org/en/books/handbook/network-servers/index.html#ldap-config
[945] https://www.feistyduck.com/books/openssl-cookbook/

(continued from previous page)

```
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 []:
An optional company name []:Another Name
```

创建证书时，可以使用其他选项，例如过期时间和备用加密算法。完整的选项列表在 openssl(1)[946] 中进行了说明。

此命令将在当前目录中创建两个文件。证书请求 **req.pem** 可以发送到 CA，CA 将验证输入的凭据，对请求进行签名，并返回签名的证书。第二个文件 **cert.key** 是证书的私钥，应存储在安全的位置。如果落入他人手中，则可以使用它来模拟用户或服务器。

或者，如果不需要来自 CA 的签名，则可以创建自签名证书。首先，生成 RSA 密钥：

```
# openssl genrsa -rand -genkey -out cert.key 2048
0 semi-random bytes loaded
Generating RSA private key, 2048 bit long modulus
.........................................+++
.....................................................................................
↪.........................+++
e is 65537 (0x10001)
```

使用此密钥创建自签名证书。按照创建证书的常规提示进行操作：

```
# openssl req -new -x509 -days 365 -key cert.key -out cert.crt -sha256
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
```

(continues on next page)

[946] https://www.freebsd.org/cgi/man.cgi?query=openssl&sektion=1&format=html

(continued from previous page)

```
Common Name (e.g. server FQDN or YOUR name) []:localhost.example.org
Email Address []:trhodes@FreeBSD.org
```

这将在当前目录中创建两个新文件：私钥文件 **cert.key** 证书和证书本身 **cert.crt**。这些应该放在同一个目录中，最好是在 **/etc/ssl/** 下，只有 root 才可读。可以使用 chmod 设置权限为 0700，这比较适合于这些文件。

16.5.2.使用证书

证书的一种用途是加密与 sendmail 邮件服务器的连接，以防止使用明文身份验证。

> 注意
> 如果用户尚未安装证书的本地副本，某些邮件客户端将显示错误。有关证书安装的详细信息，请参阅软件附带的文档。

在 FreeBSD 10.0-RELEASE 及更高版本中。可以自动为 sendmail 创建一个自签名证书。若要启用此功能，请将以下行添加到 **/etc/rc.conf**：

```
sendmail_enable="YES"
sendmail_cert_create="YES"
sendmail_cert_cn="localhost.example.org"
```

这将自动创建自签名证书，**/etc/mail/certs/host.cert**、签名密钥、**/etc/mail/certs/host.key** 和一个 CA 证书 **/etc/mail/certs/cacert.pem**。该证书将使用 sendmail_cert_cn 中指定的 Common Name。保存编辑内容后，重新启动 sendmailL：

```
# service sendmail restart
```

如果一切顺利，**/var/log/maillog** 中将不会出现错误消息。一个简单的测试：请使用 telnet 命令连接到邮件服务器的侦听端口：

```
# telnet example.com 25
Trying 192.0.34.166...
Connected to example.com.
Escape character is '^]'.
220 example.com ESMTP Sendmail 8.14.7/8.14.7; Fri, 18 Apr 2014 11:50:32 -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
```

(continues on next page)

(continued from previous page)

```
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.
```

如果 `STARTTLS` 出现在输出中，则一切正常。

16.6.IPsec VPN

互联网安全协议（IPsec）是一组位于互联网协议（IP）层之上的协议。它允许两个或多个主机通过对通信会话的每个 IP 数据包进行身份验证和加密，以安全的方式进行通信。FreeBSD IPsec 网络堆栈基于 http://www.kame.net/ 实现，同时支持 IPv4 和 IPv6 会话。

IPsec 由以下子协议组成：

- *封装安全载荷*（*ESP*）：此协议通过使用对称加密算法（如 Blowfish 和 3DES）加密内容来保护 IP 数据包数据免受第三方干扰。
- *身份验证标头*（*AH*）：此协议通过计算加密校验和并使用安全加密方式对 IP 数据包标头字段进行哈希处理来保护 IP 数据包标头免受第三方干扰和欺骗。然后，后面是包含哈希的附加标头，以允许对数据包中的信息进行身份验证。
- *IP 有效载荷压缩协议*（*IPComp*）：此协议尝试通过压缩 IP 有效载荷来提高通信性能，以减少发送的数据量。

这些协议可以一起使用，也可以单独使用，具体取决于环境。

IPsec 支持两种操作模式：第一种模式，*传输模式*用于保护两台主机之间的通信。第二种模式，*隧道模式*用于构建虚拟隧道，通常称为虚拟专用网络（VPN）。请查阅 ipsec(4)[947] 以获取有关 FreeBSD 中 IPsec 子系统的详细信息。

一般情况下，IPsec 支持在 FreeBSD 11 及更高版本上是默认启用的。对于旧的 FreeBSD 版本，应将这些选项添加到定制内核配置文件中，然后按照配置 FreeBSD 内核[948] 中的说明重建内核：

```
options   IPSEC        IP security
device    crypto
```

如果需要 IPsec 调试支持，还应添加以下内核选项：

[947] https://www.freebsd.org/cgi/man.cgi?query=ipsec&sektion=4&format=html

[948] https://docs.freebsd.org/en/books/handbook/kernelconfig/index.html#kernelconfig

```
options   IPSEC_DEBUG  debug for IP security
```

本章的其余部分演示了在家庭网络和企业网络之间设置 IPsecVPN 的过程。在示例场景中：

- 这两个站点都通过运行 FreeBSD 的网关连接到互联网。
- 每个网络上的网关至少有一个外部 IP 地址。在这个例子中，公司局域网的外部 IP 地址是 `172.16.5.4`，家庭局域网的外部 IP 地址是 `192.168.1.12`。
- 两个网络的内部地址可以是公用或私有 IP 地址。但是，地址空间不得重叠。在此示例中，公司 LAN 的内部 IP 地址为 `10.246.38.1`，而家庭 LAN 的内部 IP 地址为 `10.0.0.5`。

```
           corporate                          home
10.246.38.1/24 -- 172.16.5.4 <--> 192.168.1.12 -- 10.0.0.5/24
```

16.6.1.在 FreeBSD 上配置 VPN

首先，必须从 ports 安装 security/ipsec-tools[949]。该软件提供了许多支持该配置的应用程序。

下一个要求是创建两个 gif(4)[950] 伪设备，它们将用于隧道数据包并允许两个网络正确通信。使用 `root` 账户，在每个网关上运行以下命令：

```
corp-gw# ifconfig gif0 create
corp-gw# ifconfig gif0 10.246.38.1 10.0.0.5
corp-gw# ifconfig gif0 tunnel 172.16.5.4 192.168.1.12
```

```
home-gw# ifconfig gif0 create
home-gw# ifconfig gif0 10.0.0.5 10.246.38.1
home-gw# ifconfig gif0 tunnel 192.168.1.12 172.16.5.4
```

使用 `ifconfig gif0` 验证每个网关上的设置。以下是家庭网关的输出：

```
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
```

下面是企业网关的输出：

[949] https://cgit.freebsd.org/ports/tree/security/ipsec-tools/pkg-descr

[950] https://www.freebsd.org/cgi/man.cgi?query=gif&sektion=4&format=html

```
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
```

完成后，两个内部 IP 地址都应该可以使用 ping(8)[951] 访问：

```
home-gw# 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-gw# 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
```

正如预期的那样，双方都能够从配置的私有地址发送和接收 ICMP 数据包。接下来，必须告知两个网关如何路由数据包，以便网关能够准确的转发来自网关之后的网络流量。使用下面的命令实现这一配置：

```
corp-gw# route add 10.0.0.0 10.0.0.5 255.255.255.0
corp-gw# route add net 10.0.0.0: gateway 10.0.0.5
home-gw# route add 10.246.38.0 10.246.38.1 255.255.255.0
home-gw# route add host 10.246.38.0: gateway 10.246.38.1
```

内部计算机应可从每个网关以及网关后面的计算机访问。同样，使用 ping(8)[952] 确认：

```
corp-gw# ping -c 3 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
```

(continues on next page)

[951] https://www.freebsd.org/cgi/man.cgi?query=ping&sektion=8&format=html

[952] https://www.freebsd.org/cgi/man.cgi?query=ping&sektion=8&format=html

(continued from previous page)

```
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
--- 10.0.0.8 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max/stddev = 21.870/101.846/198.022/74.001 ms

home-gw# ping -c 3 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
--- 10.246.38.107 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max/stddev = 21.145/31.721/53.491/12.179 ms
```

此时，流量在封装在 gif 隧道中的网络之间传输，但没有任何加密。接下来，使用 IPSec 使用预共享密钥（PSK）加密流量。除了 IP 地址之外，两个网关上的 **/usr/local/etc/racoon/racoon.conf** 将完全相同，并且看起来类似于：

```
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 listen on
```

(continues on next page)

(continued from previous page)

```
{
        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 address $network/$netmask $type ( $type being any or esp)
{                               # $network must be the two internal networks you are␣
↪joining.
        pfs_group       1;
        lifetime        time    36000 sec;
        encryption_algorithm    blowfish,3des;
        authentication_algorithm        hmac_md5,hmac_sha1;
        compression_algorithm   deflate;
}
```

有关每个可用选项的说明，请参阅 **racoon.conf** 的手册页。

安全策略数据库（SPD）需要配置，以便 FreeBSD 和 racoon 能够加密和解密主机之间的网络流量。

这可以通过企业网关上的 shell 脚本（类似于以下内容）来实现。此文件将在系统初始化期间使用，并应另存为 **/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;
```

应用后，可以使用以下命令在两个网关上启动 racoon：

```
# /usr/local/sbin/racoon -F -f /usr/local/etc/racoon/racoon.conf -l /var/log/racoon.
↪log
```

输出应类似于以下内容：

```
corp-gw# /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:623b9b3bd2492452:7deab82d54ff704a
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=28496098(0x1b2d0e2)
2006-01-30 01:36:09: INFO: IPsec-SA established: ESP/Tunnel 172.16.5.4[0]->192.168.1.
↪12[0] spi=47784998(0x2d92426)
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=124397467(0x76a279b)
2006-01-30 01:36:18: INFO: IPsec-SA established: ESP/Tunnel 172.16.5.4[0]->192.168.1.
↪12[0] spi=175852902(0xa7b4d66)
```

要确保隧道正常工作，请切换到另一个控制台，并使用 tcpdump(1)[953] 通过以下命令查看网络流量。根据需要更换为网络接口卡 em0：

```
corp-gw# tcpdump -i em0 host 172.16.5.4 and dst 192.168.1.12
```

类似于以下内容的数据应显示在控制台上。如果不是，则存在问题，需要调试返回的数据。

[953] https://www.freebsd.org/cgi/man.cgi?query=tcpdump&sektion=1&format=html

```
01:47:32.021683 IP corporatenetwork.com > 192.168.1.12.privatenetwork.com:␣
↪ESP(spi=0x02acbf9f,seq=0xa)
01:47:33.022442 IP corporatenetwork.com > 192.168.1.12.privatenetwork.com:␣
↪ESP(spi=0x02acbf9f,seq=0xb)
01:47:34.024218 IP corporatenetwork.com > 192.168.1.12.privatenetwork.com:␣
↪ESP(spi=0x02acbf9f,seq=0xc)
```

此时，两个网络都应该可用，并且似乎是同一网络的一部分。很可能两个网络都受到防火墙的保护。若要允许流量在它们之间传输，需要添加规则来传递数据包。对于 ipfw(8)[954] 防火墙。请将以下行添加到防火墙配置文件中：

```
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
```

> 注意
> 规则编号可能需要更改，具体取决于当前的主机配置。

对于 pf(4)[955] 或 ipf(8)[956] 的用户。以下规则应该可以解决问题：

```
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
```

最后，要允许计算机在系统初始化期间启动对 VPN 的支持，请将以下行添加到 **/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"
```

[954] https://www.freebsd.org/cgi/man.cgi?query=ipfw&sektion=8&format=html
[955] https://www.freebsd.org/cgi/man.cgi?query=pf&sektion=4&format=html
[956] https://www.freebsd.org/cgi/man.cgi?query=ipf&sektion=8&format=html

16.7.OpenSSH

OpenSSH 是一组网络连接工具，用于提供对远程计算机的安全访问。此外，TCP/IP 连接可以通过 SSH 连接安全地进行隧道或转发。OpenSSH 对所有流量进行加密，以有效消除窃听、连接劫持和其他网络级攻击。

OpenSSH 由 OpenBSD 项目维护，并且被预装在 FreeBSD 中。

当数据以未加密的形式通过网络发送时，客户端和服务器之间任何位置的网络嗅探器都可以窃取用户/密码信息或在会话期间传输的数据。OpenSSH 提供了多种身份验证和加密方法来防止这种情况发生。有关 OpenSSH 的更多信息，请访问 http://www.openssh.com/。

本节概述了内置的客户端实用程序，以安全地访问其他系统，并从 FreeBSD 系统安全地传输文件。然后它说明了如何在 FreeBSD 系统上配置 SSH 服务器。更多信息可在本章提及的手册页中找到。

16.7.1.使用 SSH 客户端工具

要登录到 SSH 服务器，请使用 `ssh` 并指定该服务器上存在的用户名以及服务器的 IP 地址或主机名。如果这是第一次与指定的服务器建立连接，系统将提示用户首先验证服务器的指纹：

```
# ssh user@example.com
The authenticity of host 'example.com (10.0.0.1)' can't be established.
ECDSA key fingerprint is 25:cc:73:b5:b3:96:75:3d:56:19:49:d2:5c:1f:91:3b.
Are you sure you want to continue connecting (yes/no)? yes
Permanently added 'example.com' (ECDSA) to the list of known hosts.
Password for user@example.com: user_password
```

SSH 利用密钥指纹系统在客户端连接时验证服务器的真实性。当用户在首次连接时通过键入 `yes` 来接受密钥的指纹时，密钥的副本将保存到用户主目录中的 **.ssh/known_hosts**。以后的登录尝试将根据保存的密钥进行验证，如果服务器的密钥与保存的密钥不匹配，`ssh` 则会显示警告信息。如果发生这种情况，用户应先验证密钥更改的原因，然后再继续连接。

最新版本的 OpenSSH 仅接受 SSHv2 连接。SSH 协议版本 1 已过时。

使用 scp(1)[957] 可安全地将文件复制到远程计算机或从远程计算机复制文件。本示例将远程系统上的 **COPYRIGHT** 复制到本地系统的当前目录中：

```
# scp user@example.com:/COPYRIGHT COPYRIGHT
Password for user@example.com: *******
COPYRIGHT            100% |*****************************|  4735
00:00
#
```

由于已针对此主机验证了指纹，因此在提示输入用户密码之前会自动检查服务器的密钥。

[957] https://www.freebsd.org/cgi/man.cgi?query=scp&sektion=1&format=html

传递给 scp 的参数类似于 cp 命令。要复制的一个或多个文件是第一个参数，要复制到的目标变量是第二个参数。由于文件是通过网络获取的，因此一个或多个文件参数采用的形式如下：用户名@主机:<远程文件路径>。在以递归方式复制目录时异于 cp 使用的 -R，scp 使用 -r。

要打开用于复制文件的交互式会话，请使用 sftp。请参阅 sftp(1)[958] 获取会话中 sftp 可用命令的列表。

16.7.1.1.基于密钥的身份验证

可以将客户端配置为使用密钥连接到远程计算机，而非使用密码。要生成 RSA 身份验证密钥，请使用 ssh-keygen 来生成公钥和私钥对，请指定密钥类型并按照提示操作。建议使用不易忘记但难以被别人猜测出来的密码来保护密钥。

```
% ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/user/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):  ①
Enter same passphrase again:                 ②
Your identification has been saved in /home/user/.ssh/id_rsa.
Your public key has been saved in /home/user/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:54Xm9Uvtv6H4NOo6yjP/YCfODryvUU7yWHzMqeXwhq8 user@host.example.com
The key's randomart image is:
+---[RSA 2048]----+
|                 |
|                 |
|                 |
|        . o..    |
|       .S*+*o    |
|      . O=Oo . . |
|       = Oo= oo..|
|      .oB.* +.oo.|
|       =OE**.o..=|
+----[SHA256]-----+
```

① 在此处键入密码。它可以包含空格和符号。

② 重新键入密码以进行验证。

私钥存储在 **~/.ssh/id_rsa** 中，公钥存储在 **~/.ssh/id_rsa.pub**。必须将公钥复制到远程计算机上的 **~/.ssh/authorized_keys**，才能确保基于密钥的身份验证正常工作。

> 警告
>
> 许多用户认为密钥在设计上是安全的，并且会使用没有密码的密钥，这是很危险的行为。管理员可以通过手动查看私钥来验证一个密钥对是否受到密码的保护。如果私钥文件包含 EN-

[958] https://www.freebsd.org/cgi/man.cgi?query=sftp&sektion=1&format=html

CRYPTED 字样，则说明密钥所有者使用了密码。此外，为了更好地保护最终用户，可以在公钥文件中加入 from。例如，在 ssh-rsa 前缀前面加上 from="192.168.10.5" 将只允许该特定用户从该 IP 地址登录。

选项和文件因不同版本的 OpenSSH 而异。为避免出现问题，请查阅 ssh-keygen(1)[959]。

如果使用了认证密码，则每次与服务器建立连接时，系统都会提示用户输入认证密码。要将 SSH 密钥加载到内存中，并且每次都无需键入密码，请使用 ssh-agent(1)[960] 和 ssh-add(1)[961]。

认证由 ssh-agent 处理，使用加载到其中的私钥。ssh-agent 可用于启动另一个应用程序，如 shell 或窗口管理器。

若要在 shell 中使用 ssh-agent，请以 shell 作为参数开始，通过运行 ssh-add 并输入私钥的密码来添加标识。然后，用户就能够用 ssh 访问安装了相应公钥的任何主机。例如：

```
% ssh-agent csh
% ssh-add
Enter passphrase for key '/usr/home/user/.ssh/id_rsa':  ①
Identity added: /usr/home/user/.ssh/id_rsa (/usr/home/user/.ssh/id_rsa)
%
```

① 输入密钥的密码。

要在 Xorg 中使用 ssh-agent，请在 **~/.xinitrc** 中为其添加一个条目。这为在 Xorg 中启动 ssh-agent 的所有程序提供服务。示例 **~/.xinitrc** 如下所示：

```
exec ssh-agent startxfce4
```

这会启动 ssh-agent，而 Xorg 在每次启动时都会启动 XFCE。重新启动 Xorg 以使更改生效后，请运行 ssh-add 以加载所有 SSH 密钥。

16.7.1.2.SSH 隧道

OpenSSH 能够创建一个隧道，将另一个协议封装在加密会话中。

以下命令让 ssh 为 telnet 创建隧道：

```
% ssh -N -f -L 5023:localhost:23 user@foo.example.com
%
```

此示例使用以下选项：

- **-N**

[959] https://www.freebsd.org/cgi/man.cgi?query=ssh-keygen&sektion=1&format=html
[960] https://www.freebsd.org/cgi/man.cgi?query=ssh-agent&sektion=1&format=html
[961] https://www.freebsd.org/cgi/man.cgi?query=ssh-add&sektion=1&format=html

代表不执行命令，只建立隧道。如果省略，则 ssh 将会初始化一个正常的会话。

- **-f**

  强制 ssh 在后台运行。

- **-L**

  代表 本地端口:远程主机:远程端口格式的本地隧道。

- **user@foo.example.com**

  要在指定的远程 SSH 服务器上使用的登录名。

SSH 隧道的工作原理是在指定的 本地主机的 本地端口上创建一个监听套接字。然后，它把在 本地端口上收到的任何连接通过 SSH 连接转发到指定的 远程主机:远程端口。在这个例子中，客户端的 5023 端口被转发到远程机器的 23 端口。由于端口 23 是由 telnet 使用的，这就通过 SSH 隧道创建了一个加密的 telnet 会话。

此方法可用于封装任意数量的不安全 TCP 协议，如 SMTP、POP3 和 FTP，如以下示例所示。

例 **29.**为 **SMTP** 创建安全隧道

```
% ssh -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
```

可以与其他用户帐户结合使用 ssh-keygen，以创建更加无缝的 SSH 隧道环境。可以使用密钥代替键入密码，并且可以以单独的用户运行隧道。

例 **30.POP3** 服务器的安全访问

在此示例中，有一个 SSH 服务器接受来自外部的连接。在同一网络上驻留着运行 POP3 服务器的邮件服务器。要以安全的方式检查电子邮件，请创建与 SSH 服务器的 SSH 连接，并通过隧道连接到邮件服务器:

```
% ssh -N -f -L 2110:mail.example.com:110 user@ssh-server.example.com
user@ssh-server.example.com's password: ******
```

一旦隧道启动并运行，将电子邮件客户端指向 2110 端口的本地主机发送 POP3 请求。这个连接将被安全地通过隧道转发到 mail.example.com。

例 **31.**绕过防火墙

某些防火墙会过滤传入和传出连接。例如，防火墙可能会将从远程计算机到端口 22 和 80 的访问限制为仅允许 SSH 和浏览网页。这将阻止任何访问使用 22 或 80 以外的端口的其他服务。

解决方案是创建与网络防火墙外部的计算机的 SSH 连接，并使用它来通过隧道连接到所需的服务:

```
% ssh -N -f -L 8888:music.example.com:8000 user@unfirewalled-system.example.
↪org
user@unfirewalled-system.example.org's password: *******
```

在此示例中，Ogg Vorbis 流媒体客户端现在可以指向本地主机端口 8888，该端口将被转发到 `music.example.com` 端口 8000 上，成功绕过防火墙。

16.7.2.启用 SSH 服务器

除了提供内置的 SSH 客户端之外，FreeBSD 系统还可以配置为 SSH 服务器，接受来自其他 SSH 客户端的连接。

要查看 sshd 是否正在运行，请使用 service(8)[962] 命令：

```
# service sshd status
```

如果服务未运行，请将以下行添加到 **/etc/rc.conf**：

```
sshd_enable="YES"
```

这将在下次系统启动时启动 sshd，即 OpenSSH 的守护程序。要立即启动：

```
# service sshd start
```

当首次在 FreeBSD 系统上启动 sshd 时，将自动创建系统的主机密钥，指纹将显示在控制台上。可为用户提供指纹，以便他们可以在首次连接到服务器时进行验证。

有关启动 sshd 时可用选项的列表，以及有关身份验证、登录过程和各种配置文件的更完整讨论，请参阅 sshd(8)[963]。

此时，sshd 应可供系统上具有用户名和密码的所有用户使用。

16.7.3.SSH 服务器安全性

虽然 sshd 是 FreeBSD 中使用最广泛的远程管理工具。但暴力破解和路过式攻击对于任何暴露在公共网络中的系统都是常见的。有几个其他参数可用于防止这些攻击的成功，本节将对此进行介绍。

使用 OpenSSH 服务器配置文件中的 `AllowUsers` 关键字来限制哪些用户可以登录 SSH 服务器以及从哪里登录是一个好主意。例如，要想只允许 `root` 从 `192.168.1.32` 登录，请在 **/etc/ssh/sshd_config** 中添加这一行：

[962] https://www.freebsd.org/cgi/man.cgi?query=service&sektion=8&format=html
[963] https://www.freebsd.org/cgi/man.cgi?query=sshd&sektion=8&format=html

```
AllowUsers root@192.168.1.32
```

要允许 admin 从任何位置登录，请列出该用户而无需指定 IP 地址：

```
AllowUsers admin
```

多个用户应列在同一行中，如下所示：

```
AllowUsers root@192.168.1.32 admin
```

对 **/etc/ssh/sshd_config** 进行更改后，请执行以下内容以使 sshd 重新加载其配置文件：

```
# service sshd reload
```

> 注意
>
> 使用此关键字时，列出所有需要登录此计算机的用户非常重要。任何未在该行中指定的用户都将被锁定。此外，OpenSSH 服务器配置文件中使用的关键字区分大小写。如果关键字拼写不正确（包括其大小写），则将忽略该关键字。要一直测试对此文件所做的更改，以确保编辑工作按预期进行。请参阅 sshd_config(5)[964] 以验证可用关键字的拼写和用法。

此外，用户可能会被强制要求通过使用公钥和私钥使用双重认证。当需要时，用户可以通过使用 ssh-keygen(1)[965] 生成密钥对，并向管理员发送公钥。如上面的客户端部分所述，此密钥文件将放置在 **authorized_keys**。若要强制用户仅使用密钥，可以配置以下选项：

```
AuthenticationMethods publickey
```

> 技巧
>
> 不要将 /etc/ssh/sshd_config 与 /etc/ssh/ssh_config 相混淆（请注意第一个文件名中多出的 d）。第一个文件用于配置服务器，第二个文件用于配置客户端。请参阅 ssh_config(5)[966] 了解可用于客户端设置的详单。

16.8.文件系统访问控制列表

访问控制列表（ACL）以兼容 POSIX®.1e 的方式扩展了标准 UNIX® 权限模型。这使得管理员可以使用更细化的权限模型。

FreeBSD **GENERIC** 内核为 UFS 文件系统提供了 ACL 支持。偏好编译定制内核的用户必须在他们的定制内核配置文件中加入下列选项：

[964] https://www.freebsd.org/cgi/man.cgi?query=sshd_config&sektion=5&format=html

[965] https://www.freebsd.org/cgi/man.cgi?query=ssh-keygen&sektion=1&format=html

[966] https://www.freebsd.org/cgi/man.cgi?query=ssh_config&sektion=5&format=html

```
options UFS_ACL
```

如果未编译此选项，则在尝试挂载支持 ACL 的文件系统时将显示一条警告消息——ACLs rely on extended attributes which are natively supported in UFS2.（ACL 依赖于 UFS2 中原生支持的扩展属性）

本章介绍了如何启用 ACL 支持，并提供了一些使用示例。

16.8.1.启用 ACL 支持

ACL 是由挂载时的管理标签 `acls` 启用的，可以将其添加至 **/etc/fstab**。也可以用 tunefs(8)[967] 来修改文件系统标签中的超级块 ACL 标签，以实现用持久化的方式自动设置挂载时的参数。一般来说，出于几个原因，最好使用超级块标签：

- 超级块标签不会被 `mount -u` 的重新挂载所改变，因为它需要完整的 `umount` 和新的 `mount`。这意味着启动后不能在根文件系统上启用 ACL。这也意味着文件系统上的 ACL 支持在系统使用时不能被改变。
- 设置超级块标签会强制始终在启用 ACL 的情况下挂载文件系统，即使没有 **fstab** 参数，设备顺序发生变化也如此。这可以防止在没有启用 ACL 的情况下意外挂载文件系统。

  注意

  阻止在没有启用 ACL 的情况下意外挂载是可取的，因为如果 ACL 被启用，然后被禁用，紧接着在没有刷新扩展属性的情况下重新启用，就会发生令人讨厌的事情。一般来说，一旦在文件系统上启用了 ACL，就不应该被禁用，因为所产生的文件保护措施可能与系统用户所期望的不一致，而且重新启用 ACL 可能会将以前的 ACL 重新附加到权限已经改变的文件上，导致不可预测的行为。

启用了 ACL 的文件系统将在其权限设置中显示加号（+）号：

```
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
```

在此示例中，**directory1**、**directory2** 和 **directory3** 都使用了 ACL，而 **private** 和 **public_html** 则没有。

[967] https://www.freebsd.org/cgi/man.cgi?query=tunefs&sektion=8&format=html

16.8.2.使用 ACL

可以用 `getfacl` 查看文件系统的 ACL。例如，要查看 **test** 中的 ACL 设置：

```
% getfacl test
    #file:test
    #owner:1001
    #group:1001
    user::rw-
    group::r--
    other::r--
```

要改变这个文件的 ACL 设置，须使用 `setfacl`。要从一个文件或文件系统中删除所有当前定义的 ACL，可使用 `-k`。然而，首选方法是使用 `-b`，因为它保留了 ACL 工作所需的基本参数。

```
% setfacl -k test
```

要修改默认的 ACL 条目，请使用 `-m`：

```
% setfacl -m u:trhodes:rwx,group:web:r--,o::--- test
```

在这个例子中，没有预定义的条目，因为它们已经被前一条命令删除了。这条命令恢复了默认选项，并分配了所列的选项。如果添加的用户或组在系统中不存在，将显示 `Invalid argument` 错误。

有关这些命令的更多可用选项的信息，请参阅 getfacl(1)[968] 和 setfacl(1)[969]。

16.9.监测第三方安全问题

近年来，安全领域对漏洞评估的处理方式进行了许多改进。现在几乎任何可用的操作系统都会安装和配置第三方软件，系统被入侵的威胁也会增加。

漏洞评估是安全性的关键因素。虽然 FreeBSD 为基本系统发布安全公告，但为每个第三方软件发布安全公告超出了 FreeBSD 项目的能力范围。有一种方法可以缓解第三方漏洞并警告管理员已知的安全问题。FreeBSD 的一个附加软件 pkg，可明确用于此目的。

pkg 将轮询数据库是否存在安全问题。数据库由 FreeBSD 安全团队和 ports 开发人员更新和维护。

请参考 pkg 的安装说明[970]。

在安装过程中，periodic(8)[971] 提供了配置文件来维护 pkg 审计数据库，并提供了一种保持数据库更新的

[968] https://www.freebsd.org/cgi/man.cgi?query=getfacl&sektion=1&format=html
[969] https://www.freebsd.org/cgi/man.cgi?query=setfacl&sektion=1&format=html
[970] https://docs.freebsd.org/en/books/handbook/ports/index.html#pkgng-intro
[971] https://www.freebsd.org/cgi/man.cgi?query=periodic&sektion=8&format=html

计划任务。如果在 periodic.conf（5）[972] 中把 daily_status_security_pkgaudit_enable 设置为 YES，就可以启用这个功能。确保每日的安全运行邮件被读取，这些邮件会被发送到 root 的电子邮件账户。

安装后，为了随时审计作为 ports 中的第三方软件，管理员可以选择更新数据库并查看已安装软件包的已知漏洞，方法是执行：

```
# pkg audit -F
```

pkg 将显示已安装软件中任何已发布的漏洞的消息：

```
Affected package: cups-base-1.1.22.0_1
Type of problem: cups-base -- HPGL buffer overflow vulnerability.
Reference: <https://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.
```

通过点击显示的链接打开网络浏览器，管理员可以获得有关漏洞的更多信息。这包括受到影响的 FreeBSD port 版本，以及其他可能包含安全公告的网站。

pkg 是一个功能强大的软件，当与 ports-mgmt/portmaster[973] 结合使用时非常有用。

16.10.FreeBSD 安全公告

像许多高质量操作系统的开发者一样，FreeBSD 项目有一个安全团队，负责确定每个 FreeBSD 发行版的生命周期结束（EoL）日期，并为尚未达到其生命周期的受支持发行版本提供安全更新。有关更多 FreeBSD 安全团队和支持的版本的信息，请访问 FreeBSD 安全页面[974]。

安全团队的一项任务是响应 FreeBSD 操作系统中报告的安全漏洞。确认漏洞后，安全团队将验证修复漏洞所需的步骤，并使用修复程序更新源代码。然后，它将详细信息发布为“安全公告”。安全公告发布在 FreeBSD 网站上[975]，并邮寄到 FreeBSD 安全通知邮件列表[976]、FreeBSD 安全邮件列表[977] 和 FreeBSD 公告邮件列表[978] 邮件列表。

本节讨论了 FreeBSD 安全公告的格式。

[972] https://www.freebsd.org/cgi/man.cgi?query=periodic.conf&sektion=5&format=html
[973] https://cgit.freebsd.org/ports/tree/ports-mgmt/portmaster/pkg-descr
[974] https://www.freebsd.org/security
[975] https://www.freebsd.org/security/advisories/
[976] https://lists.freebsd.org/subscription/freebsd-security-notifications
[977] https://lists.freebsd.org/subscription/freebsd-security
[978] https://lists.freebsd.org/subscription/freebsd-announce

16.10.1.安全公告的格式

下面是一个 FreeBSD 安全公告的例子:

```
=============================================================================
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA512

=============================================================================
FreeBSD-SA-14:04.bind                                        Security Advisory
                                                          The FreeBSD Project

Topic:          BIND remote denial of service vulnerability

Category:       contrib
Module:         bind
Announced:      2014-01-14
Credits:        ISC
Affects:        FreeBSD 8.x and FreeBSD 9.x
Corrected:      2014-01-14 19:38:37 UTC (stable/9, 9.2-STABLE)
                2014-01-14 19:42:28 UTC (releng/9.2, 9.2-RELEASE-p3)
                2014-01-14 19:42:28 UTC (releng/9.1, 9.1-RELEASE-p10)
                2014-01-14 19:38:37 UTC (stable/8, 8.4-STABLE)
                2014-01-14 19:42:28 UTC (releng/8.4, 8.4-RELEASE-p7)
                2014-01-14 19:42:28 UTC (releng/8.3, 8.3-RELEASE-p14)
CVE Name:       CVE-2014-0591

For general information regarding FreeBSD Security Advisories,
including descriptions of the fields above, security branches, and the
following sections, please visit <URL:http://security.FreeBSD.org/>.

I.   Background

BIND 9 is an implementation of the Domain Name System (DNS) protocols.
The named(8) daemon is an Internet Domain Name Server.

II.  Problem Description

Because of a defect in handling queries for NSEC3-signed zones, BIND can
crash with an "INSIST" failure in name.c when processing queries possessing
certain properties.  This issue only affects authoritative nameservers with
at least one NSEC3-signed zone.  Recursive-only servers are not at risk.

III. Impact
```

(continues on next page)

(continued from previous page)

```
An attacker who can send a specially crafted query could cause named(8)
to crash, resulting in a denial of service.

IV.  Workaround

No workaround is available, but systems not running authoritative DNS service
with at least one NSEC3-signed zone using named(8) are not vulnerable.

V.   Solution

Perform one of the following:

1) Upgrade your vulnerable system to a supported FreeBSD stable or
release / security branch (releng) dated after the correction date.

2) To update your vulnerable system via a source code patch:

The following patches have been verified to apply to the applicable
FreeBSD release branches.

a) Download the relevant patch from the location below, and verify the
detached PGP signature using your PGP utility.

[FreeBSD 8.3, 8.4, 9.1, 9.2-RELEASE and 8.4-STABLE]
# fetch http://security.FreeBSD.org/patches/SA-14:04/bind-release.patch
# fetch http://security.FreeBSD.org/patches/SA-14:04/bind-release.patch.asc
# gpg --verify bind-release.patch.asc

[FreeBSD 9.2-STABLE]
# fetch http://security.FreeBSD.org/patches/SA-14:04/bind-stable-9.patch
# fetch http://security.FreeBSD.org/patches/SA-14:04/bind-stable-9.patch.asc
# gpg --verify bind-stable-9.patch.asc

b) Execute the following commands as root:

# cd /usr/src
# patch < /path/to/patch

Recompile the operating system using buildworld and installworld as
described in <URL:https://www.FreeBSD.org/handbook/makeworld.html>.

Restart the applicable daemons, or reboot the system.
```

(continues on next page)

(continued from previous page)

```
3) To update your vulnerable system via a binary patch:

Systems running a RELEASE version of FreeBSD on the i386 or amd64
platforms can be updated via the man:freebsd-update[8] utility:

# freebsd-update fetch
# freebsd-update install

VI.  Correction details

The following list contains the correction revision numbers for each
affected branch.

Branch/path                                                      Revision
- -------------------------------------------------------------------------
stable/8/                                                        r260646
releng/8.3/                                                      r260647
releng/8.4/                                                      r260647
stable/9/                                                        r260646
releng/9.1/                                                      r260647
releng/9.2/                                                      r260647
- -------------------------------------------------------------------------

To see which files were modified by a particular revision, run the
following command, replacing NNNNNN with the revision number, on a
machine with Subversion installed:

# svn diff -cNNNNNN --summarize svn://svn.freebsd.org/base

Or visit the following URL, replacing NNNNNN with the revision number:

<URL:https://svnweb.freebsd.org/base?view=revision&revision=NNNNNN>

VII. References

<URL:https://kb.isc.org/article/AA-01078>

<URL:http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-0591>

The latest revision of this advisory is available at
<URL:http://security.FreeBSD.org/advisories/FreeBSD-SA-14:04.bind.asc>
-----BEGIN PGP SIGNATURE-----
```

(continues on next page)

(continued from previous page)

```
iQIcBAEBCgAGBQJS1ZTYAAoJEO1n7NZdz2rnOvQP/2/68/s9Cu35PmqNtSZVVxVG
ZSQP5EGWx/lramNf9566iKxOrLRMq/h3XWcC4goVd+gZFrvITJSVOWSa7ntDQ7TO
XcinfRZ/iyiJbs/Rg2wLHc/t5oVSyeouyccqODYFbOwOlk35JjOTMUG1YcX+Zasg
ax8RV+7Zt1QSBkMlOz/myBLXUjlTZ3Xg2FXVsfFQW5/g2CjuHpRSFx1bVNX6ysoG
9DT58EQcYxIS8WfkHRbbXKh9I1nSfZ7/Hky/kTafRdRMrjAgbqFgHkYTYsBZeav5
fYWKGQRJulYfeZQ90yMTvlpF42DjCC3uJYamJnwDIu8OhS1WRBI8fQfr9DRzmRua
OK3BK9hUiScDZOJB6OqeVzUTfe7MAA4/UwrDtTYQ+PqAenv1PK8DZqwXyxA9ThHb
zKO3OwuKOVHJnKvpOcr+eNwo7jbnHlis0oBksj/mrq2P9m2ueF9gzCiq5Ri5Syag
Wssb1HUoMGwqU0roS8+pRpNC8YgsWpsttvUWSZ8u6Vj/FLeHpiV3mYXPVMaKRhVm
067BA2uj4Th1JKtGleox+Em0R7OFbCc/9aWC67wiqI6KRyit9pYiF3npph+7D5Eq
7zPsUdDd+qc+UTiLp3liCRp5w6484wWdhZO6wRtmUgxGjNkxFoNnX8CitzF8AaqO
UWWemqWuz3lAZuORQ9KX
=OQzQ
-----END PGP SIGNATURE-----
```

每个安全公告都使用以下格式：

- 每个安全公告都由安全官的 PGP 密钥签名。可以在 OpenPGP 密钥[979] 中验证安全官的公钥。
- 安全公告的名字总是以 `FreeBSD-SA-` 开头 (代表 FreeBSD 安全公告)，后面是两位数格式的年份 (`14:`)，然后是该年的公告编号 (`04.`)，再后面是受影响的软件或子系统的名称 (`bind`)。这里显示的安全公告是 2014 年的第四个安全公告，它影响到了 BIND。
- `Topic` 字段概述了该漏洞。
- `Category` 指系统中受影响的部分，可以是 `core` 、`contrib` 或 `ports` 之一。`core` 类别意味着该漏洞影响了 FreeBSD 操作系统的核心组件。`contrib` 类别意味着该漏洞会影响 FreeBSD 附带的软件，例如 BIND。`ports` 类别表示该漏洞影响通过 ports 提供的软件。
- `module` 字段指的是组件的位置。在这个例子中，`bind` 模块受到了影响。因此，这个漏洞影响了随操作系统安装的软件。
- `Announced` 字段反映了安全公告的发布日期。这意味着安全团队已经验证了问题的存在，并且已经将补丁提交到 FreeBSD 源代码存储库。
- `Credits` 字段表示发现该漏洞并报告它的个人或组织。
- `Affects` 字段解释了哪些 FreeBSD 版本受此漏洞影响。
- `Corrected` 指出了被纠正的日期、时间、时间偏移和版本。括号中的部分显示了已经合并了修正的每个分支，以及该分支对应的版本号，版本标识符本身包括版本号，如果合适，还包括补丁级别。补丁级别是字母 `p` 后面的数字，表示补丁的序列号，使用户追踪哪些补丁已经应用到系统中。
- `CVE Name` 字段列出了通用漏洞披露数据库 cve.mitre.org[980] 中的公告编号（如果存在）。

[979] https://docs.freebsd.org/en/books/handbook/pgpkeys/index.html#pgpkeys

[980] http://cve.mitre.org/

- `Background` 字段提供受影响模块的说明。
- `Problem Description` 字段说明了此漏洞技术细节。这可能包括有关有缺陷的代码以及如何恶意该利用软件的信息。
- `Impact` 字段说明了问题可能对系统产生的影响类型。
- `Workaround` 字段指示无法立即修补系统的系统管理员是否可以使用变通的办法解决。
- `Solution` 字段提供了有关修补受影响系统的说明。这是一种经过逐步测试和验证的方法，用于修补系统让其安全运行。
- `Correction Details` 字段显示每个受影响的 Subversion 分支，其中包含更正代码的修订版号。
- `References` 字段提供了有关此漏洞的其他信息来源。

16.11.进程审计

进程审计是一种安全方法，管理员可以在其中审计所使用的系统资源及其在用户之间的分配，提供系统监控，并最低限度地审计用户的命令。

进程审计既有正面也有负面。其中一个积极因素是，入侵可能会缩小到入口点。负面影响是指进程审计生成的日志量以及它们可能需要的磁盘空间。本节将引导管理员了解进程审计的基础知识。

> 注意
> 如果需要更细粒度的审计，请参阅安全事件审计[981]。

16.11.1.启用和利用进程审计

在使用进程审计之前，必须使用以下命令启用它：

```
# sysrc accounting_enable=yes
# service accounting start
```

审计信息存储在位于 **/var/account** 的文件中，如有必要，审计服务将在首次启动时自动创建该文件。这些文件包含敏感信息，包括所有用户执行的所有命令。此文件的写入权限限制为 `root` 账户，读取权限仅限于 `root` 和 `wheel` 组的成员。要同时防止 `wheel` 的成员读取文件，请将 **/var/account** 目录的权限更改为仅允许让 `root` 访问。

启用后，审计将开始追踪 CPU 统计信息和执行的命令等信息。所有审计日志均采用非人类可读格式，但可使用 `sa` 查看。如果执行时没有任何选项，`sa` 则打印与每用户调用数、总运行时间（以分钟为单位）、总 CPU 和用户时间（以分钟为单位）以及平均 I/O 操作数相关的信息。请参阅 sa(8)[982] 以获取控制输出的可用选项列表。

[981] https://docs.freebsd.org/en/books/handbook/audit/index.html#audit
[982] https://www.freebsd.org/cgi/man.cgi?query=sa&sektion=8&format=html

要显示用户执行的命令，请使用 `lastcomm`。例如，使用此命令打印出 `ttyp1` 终端上 `trhodes` 所有的 `ls` 操作：

```
# lastcomm ls trhodes ttyp1
```

还存在许多其他有用的选项，并在 lastcomm(1)[983]、acct(5)[984] 和 sa(8)[985] 中进行了解释。

16.12.资源配额

FreeBSD 为管理员提供了几种方法来限制个人可以使用的系统资源量。磁盘配额限制用户可用的磁盘空间量。磁盘配额[986] 中讨论了配额。

可以使用平面文件或命令来配置资源限制数据库实现对其他资源（如 CPU 和内存）的限制。传统方法通过编辑 **/etc/login.conf** 来定义登录分级。虽然仍然支持此方法，但任何更改都需要多个步骤过程：编辑此文件、重建资源数据库、对 **/etc/master.passwd** 进行必要的更改以及重建密码数据库。这可能会变得非常耗时，具体取决于要配置的用户数。

`rctl` 可用于提供更细粒度的方法来控制资源限制。此命令支持的不仅仅是用户限制，因为它还可用于设置进程和 jail 的资源限制。

本节演示了控制资源的两种方法（从传统方法开始）。

16.12.1.配置登录分级

在传统方法中，登录分级和应用于登录分级的资源限制在 **/etc/login.conf** 中定义。每个用户账户都可以被分配到一个登录分级，其中 `default` 是默认的登录分级。每个登录分级都有一组与之相关的登录能力。一个登录能力是一个 `name=value` 对，其中 *name* 是一个众所周知的标识符，*value* 是一个任意字符串，会根据名称进行相应的处理。

> 注意
> 每当编辑 **/etc/login.conf** 后，必须通过执行以下命令更新 **/etc/login.conf.db**：

```
# cap_mkdb /etc/login.conf
```

资源限制在两个方面与默认登录功能不同。首先，对于每个限制，都有一个 *软限制*和 *硬限制*。软限制可以由用户或应用程序调整，但不能设置为高于硬限制。硬限制可以由用户降低，但只能由超级用户提高。其次，大多数资源限制适用于指定用户的每个进程。

登录分级资源限制[987] 列出了最常用的资源限制。所有可用的资源限制和功能都在 login.conf(5)[988] 中有详

[983] https://www.freebsd.org/cgi/man.cgi?query=lastcomm&sektion=1&format=html
[984] https://www.freebsd.org/cgi/man.cgi?query=acct&sektion=5&format=html
[985] https://www.freebsd.org/cgi/man.cgi?query=sa&sektion=8&format=html
[986] https://docs.freebsd.org/en/books/handbook/disks/index.html#quotas
[987] https://docs.freebsd.org/en/books/handbook/security/#resource-limits
[988] https://www.freebsd.org/cgi/man.cgi?query=login.conf&sektion=5&format=html

细介绍。

资源配额	说明
coredumpsize	程序生成的核心转储大小限制从属于磁盘使用的其他限制，例如 `filesize` 或磁盘配额。此限制通常用作控制磁盘空间消耗的不太严格的方法。由于用户不生成核心转储，并且通常不会删除它们，因此在大型程序崩溃时，此设置可能会使它们免于磁盘空间不足。
cputime	用户进程可能消耗的最大 CPU 时间量。有问题的进程将被内核杀死。这是对所用 CPU 时间的限制，而非 `top` 和 `ps` 生成的某些字段中显示的 CPU 百分比。
filesize	用户可能拥有的文件的最大大小。与磁盘配额（磁盘配额[989]）不同，此限制是对单个文件强制执行的，而不是用户拥有的所有文件。
maxproc	用户可以运行的前台和后台进程的最大数目。此限制不得大于 `kern.maxproc` 指定的系统限制。将此限制设置得太小可能会妨碍用户的工作效率，因为某些任务（如编译大型程序）会启动许多进程。
memorylocked	使用 mlock(2)[990] 可以请求将进程锁定到主内存中的最大内存量。一些系统关键型程序,如 amd(8)[991]，锁定到主内存中。这样，如果系统开始交换空间，它们不会导致磁盘抖动。
memoryuse	进程在任何给定时间可能消耗的最大内存量。它包括核心内存和交换空间的使用。这不是限制内存消耗的包罗万象的限制，但这是一个良好的开端。
openfiles	进程可能打开的最大文件数。在 FreeBSD 中，文件用于表示套接字和 IPC 通道。所以要小心不要设置得太低。系统范围的限制由 `kern.maxfiles` 定义。
sbsize	对用户可能消耗的网络内存量的限制。这通常可用于限制网络通信。
stacksize	进程堆栈的最大大小。仅凭这一点不足以限制程序可能使用的内存量，因此应将其与其他限制结合使用。

设置资源限制时，还需要记住其他一些事项：

- 在系统启动时由 **/etc/rc** 启动的进程将分配给 `daemon` 登录分级。
- 尽管对于大多数限制，默认的 **/etc/login.conf** 是合理值的良好来源，但它们可能并不适合每个系统。将限制设置得太高可能会使系统容易被滥用，而将其设置得太低可能会给生产力带来压力。
- Xorg 占用大量资源，并鼓励用户同时运行更多程序。
- 许多限制适用于单个进程，而不是整个用户。例如，设置 `openfiles` 为 `50` 意味着用户运行的每个进程最多可以打开 `50` 个文件。一个用户可以打开的文件总量是 `openfiles` 的值乘以 `maxproc` 的值。这也适用于内存消耗。

[989] https://docs.freebsd.org/en/books/handbook/disks/index.html#quotas

[990] https://www.freebsd.org/cgi/man.cgi?query=mlock&sektion=2&format=html

[991] https://www.freebsd.org/cgi/man.cgi?query=amd&sektion=8&format=html

有关资源限制以及登录分级和功能的更多信息，请参阅 cap.mkdb(1)[992]、getrlimit(2)[993] 和 login.conf(5)[994]。

16.12.2.启用和配置资源限制

可调参数 `kern.racct.enable` 必须设置为非零值。定制内核需要特定配置：

```
options         RACCT
options         RCTL
```

系统重新引导到新内核中以后，`rctl` 可用于为系统设置规则。

规则的语法是通过使用主体（subject）、主体-ID（subject-id）、资源（resource）和动作（action）来控制的，如本例规则所见：

```
user:trhodes:maxproc:deny=10/user
```

在此规则中，主体是 `user`，主体-ID 是 `trhodes`，资源 `maxproc` 是最大进程数，动作是 `deny`，它阻止任何新进程的创建。这意味着，用户 `trhodes` 将被限制在不超过 10 个进程的范围内。其他可能的动作包括向控制台记录，向 devd(8)[995] 传递通知，或者向进程发送一个 sigterm。

在添加规则时必须注意一些问题。由于这个用户被限制为 10 个进程，这个例子将阻止用户在登录和执行 `screen` 会话后执行其他任务。一旦达到资源限制，将打印出一个错误，如本例：

```
% man test
    /usr/bin/man: Cannot fork: Resource temporarily unavailable
eval: Cannot fork: Resource temporarily unavailable
```

作为另一个示例，它防止 jail 超过内存限制。此规则可以写为：

```
# rctl -a jail:httpd:memoryuse:deny=2G/jail
```

如果规则已添加到 **/etc/rctl.conf** 中，则这些规则将在重新启动后持续存在。格式是规则，没有前面的命令。例如，可以将前面的规则添加为：

```
# Block jail from using more than 2G memory:
jail:httpd:memoryuse:deny=2G/jail
```

要删除规则，请使用 `rctl` 将其从列表中删除：

[992] https://www.freebsd.org/cgi/man.cgi?query=cap.mkdb&sektion=1&format=html
[993] https://www.freebsd.org/cgi/man.cgi?query=getrlimit&sektion=2&format=html
[994] https://www.freebsd.org/cgi/man.cgi?query=login.conf&sektion=5&format=html
[995] https://www.freebsd.org/cgi/man.cgi?query=devd&sektion=8&format=html

```
# rctl -r user:trhodes:maxproc:deny=10/user
```

删除所有规则的方法记录在 rctl(8)[996] 中。但是，如果需要删除单个用户的所有规则，则可能会执行以下命令：

```
# rctl -r user:trhodes
```

还有许多其他的资源，可以用来对各种主体进行额外的控制。参见 rctl(8)[997] 来了解它们。

16.13.使用 sudo 管理权限

系统管理员通常需要向用户授予额外的权限，以便他们可以执行特权任务。团队成员需要访问 FreeBSD 系统来执行他们的特定任务，这给每个管理员带来了独特的挑战。这些团队成员只需要超出正常用户级别的访问权限子集。但是，他们几乎总是告诉管理层，如果没有超级用户访问权限，他们就无法执行任务。值得庆幸的是，不必向一般用户提供此类访问权限，因为存在用于管理此类要求的工具。

到目前为止，安全章节已包含了允许授权用户访问以及尝试阻止未经授权的访问。授权用户有权访问系统资源后，会出现另一个问题：在许多情况下，某些用户可能需要访问应用程序启动脚本，或者管理员团队需要维护系统。传统上，标准用户和组、文件权限，甚至 su(1)[998] 命令都可管理这种权限。但是由于应用程序需要更多的访问权限，并且由于更多的用户需要使用系统资源，因此需要更好的解决方案——目前最常用的软件是 sudo。

sudo 允许管理员配置对系统命令的更严格的访问，并提供一些高级日志记录功能。作为一个工具。它可通过 ports 中的 security/sudo[999] 使用，也可以通过使用 pkg(8)[1000] 工具获得。要使用 pkg(8)[1001] 工具：

```
# pkg install sudo
```

安装完成后，安装的 `visudo` 将使用文本编辑器打开配置文件。强烈建议使用 `visudo`，因为它带有内置的语法检查器，用于在保存文件之前验证是否有误。

配置文件由几个小节组成，允许进行详细的设置。在下面的例子中，web 软件维护者 `user1` 需要启动、停止和重新启动 web 软件，即 *webservice*。要授予此用户执行这些任务的权限，请将以下行添加到 **/usr/local/etc/sudoers** 的末尾：

```
user1   ALL=(ALL)       /usr/sbin/service webservice *
```

用户现在可以使用以下命令启动 *webservice*：

[996] https://www.freebsd.org/cgi/man.cgi?query=rctl&sektion=8&format=html
[997] https://www.freebsd.org/cgi/man.cgi?query=rctl&sektion=8&format=html
[998] https://www.freebsd.org/cgi/man.cgi?query=su&sektion=1&format=html
[999] https://cgit.freebsd.org/ports/tree/security/sudo/pkg-descr
[1000] https://www.freebsd.org/cgi/man.cgi?query=pkg&sektion=8&format=html
[1001] https://www.freebsd.org/cgi/man.cgi?query=pkg&sektion=8&format=html

```
% sudo /usr/sbin/service webservice start
```

虽然此配置允许单个用户访问 webservice 服务。但是，在大多数组织中，有一个完整的 Web 团队负责管理服务。单行还可以授予对整个组的访问权限。以下步骤将创建一个 Web 组，将用户添加到此组，并允许该组的所有成员管理服务：

```
# pw groupadd -g 6001 -n webteam
```

还可以使用相同作用的 pw(8)[1002] 命令将添加用户到 webteam 组：

```
# pw groupmod -m user1 -n webteam
```

最后，**/usr/local/etc/sudoers** 中的这一行允许 webteam 组的任何成员都对 *webservice* 进行管理：

```
%webteam   ALL=(ALL)       /usr/sbin/service webservice *
```

与 su(1)[1003] 不同，sudo 只需要一般用户密码。这增加了一个优势，用户无需共享密码，这是大多数安全审计中的发现——共用密码这种问题普遍存在且一直很糟糕。

允许使用 sudo 运行应用程序的用户只需输入自己的密码。这比 su(1)更安全，并能提供更好的控制，在 su（1）[1004] 中，用户输入 root 密码，就能获得所有 root 权限。

技巧

大多数组织正在或已经转向双重认证。在这些情况下，用户可能不需要再输入密码。sudo 为这些情况提供了 NOPASSWD 变量。将它添加到上述配置中，将允许 *webteam* 组的所有成员在没有密码要求的情况下管理该服务：

```
%webteam   ALL=(ALL)       NOPASSWD: /usr/sbin/service webservice *
```

16.13.1.日志记录

采用 sudo 的一个优点是能够启用会话日志记录。使用内置的日志机制和内置的 sudoreplay 命令，将记录通过 sudo 启动的所有命令以供以后验证。若要启用此功能，请添加默认日志目录条目，此示例使用用户变量。还存在其他几种日志文件名约定，有关其他信息，请参阅 sudoreplay 的手册页。

```
Defaults iolog_dir=/var/log/sudo-io/%{user}
```

技巧

[1002] https://www.freebsd.org/cgi/man.cgi?query=pw&sektion=8&format=html

[1003] https://www.freebsd.org/cgi/man.cgi?query=su&sektion=1&format=html

[1004] https://www.freebsd.org/cgi/man.cgi?query=su&sektion=1&format=html

这个目录将在配置好日志后自动创建。为了安全起见，最好让系统以默认权限创建此目录。此外，这个条目还将记录使用 sudoreplay 命令的管理员。要改变这种行为，请阅读并取消对 **sudoers** 里面的日志选项的注释。

这个指令被添加到 **sudoers** 文件后，所有用户的配置都可以加入日志访问这个操作。在所示的例子中，更新后的 *webteam* 条目将有以下额外的变化：

```
%webteam ALL=(ALL) NOPASSWD: LOG_INPUT: LOG_OUTPUT: /usr/sbin/service webservice *
```

从此时起，将记录所有更改 *webservice* 软件状态的 *webteam* 成员。可以通过以下方式显示以前和当前会话的列表：

```
# sudoreplay -l
```

在输出中，要回放特定会话，请搜索 `TSID=` 条目，然后将其传递给 sudoreplay，无其他选项将以正常速度回放会话。例如：

```
# sudoreplay user1/00/00/02
```

警告

当会话被记录下来时，任何管理员都能够删除会话，使得没人知道它们做了什么事情。值得通过入侵检测系统（IDS）或类似的软件增加一个日常检查，以便在有人为修改时通知其他管理人员。

`sudoreplay` 是非常具有可扩展性的。有关详细信息，请参阅文档。

16.14.使用 doas 来代替 sudo

作为 security/sudo[1005] 的替代品，security/doas[1006] 可赋予用户增强权限的能力。

doas 软件可通过 ports 中的 security/doas[1007] 或 pkg(8)[1008] 工具获得。

安装后，必须将 `/usr/local/etc/doas.conf` 配置为对用户授予访问特定命令或身份的权限。

最简单的条目可能是以下内容，它在 local_user 执行 doas 命令时授予其 root 权限，而无需询问其密码：

```
permit nopass local_user as root
```

有关更多配置示例，请阅读 doas.conf(5)[1009]。

安装和配置 `doas` 软件后，现在可以使用增强的权限执行命令，例如：

[1005] https://cgit.freebsd.org/ports/tree/security/sudo/pkg-descr
[1006] https://cgit.freebsd.org/ports/tree/security/doas/pkg-descr
[1007] https://cgit.freebsd.org/ports/tree/security/doas/pkg-descr
[1008] https://www.freebsd.org/cgi/man.cgi?query=pkg&sektion=8&format=html
[1009] https://www.freebsd.org/cgi/man.cgi?query=doas.conf&sektion=5&format=html

```
$ doas vi /etc/rc.conf
```

第十七章 Jail

17.1.概述

由于系统管理是一项艰巨的任务，因此已经开发出许多工具来让管理员的生活更轻松。这些工具通常可以拓展系统的安装、配置和维护方式。jail 就是可以用来拓展 FreeBSD 系统安全性的工具之一，*jail* 从 FreeBSD 4.X 开始可用，并且在实用性、性能、可靠性和安全性方面持续增强。

jail 建立在 chroot(2)[1010] 概念的基础上，该概念用于更改一组进程的根目录。这将创建一个与系统其余部分分开的安全环境。在 chroot 环境中创建的进程无法访问其外部的文件或资源。因此，破坏在 chroot 环境中运行的服务不会让攻击者破坏整个系统。但是，chroot 存在几个限制。它适用于不需要太多灵活性或复杂高级功能的简单任务。随着时间的流逝，已经发现了许多方法可以逃逸出 chroot 环境，使其不再是确保服务安全的理想解决方案。

jail 在几个方面改进了传统 chroot 环境的模型。在传统的 chroot 环境中，进程仅受其可以访问的文件系统部分的限制。其余的系统资源、系统用户、正在运行的进程和网络子系统由 chroot 进程和主机系统的进程共享。jail 通过虚拟化对文件系统、用户集和网络子系统的访问来扩展此模型。更细粒度的控件可用于调整对 jail 环境的访问。jail 可以被视为一种系统级的虚拟化。

jail 有四个要素：

- 目录子树：进入 jail 的起点目录。一旦进入 jail，任何一个进程都不能逃出这个子树。
- 主机名：将由 jail 使用。
- IP 地址：分配给 jail。jail 的 IP 地址通常是现有网络接口的别名地址。
- 命令：要在 jail 内运行的可执行文件的路径名。该路径相对于 jail 环境的根目录。

jail 有自己的 `root` 账号和自己的用户，这些 `root` 账号仅限于 jail 环境。jail 的用户在关联的 jail 环境之外对系统执行操作是被禁止的。

本章概述了用于管理 FreeBSD jail 的术语和命令。对于系统管理员和高级用户来说 Jail 都是一个强大的工具。

[1010] https://www.freebsd.org/cgi/man.cgi?query=chroot&sektion=2&format=html

读完本章，你就会知道：

- 什么是 jail，它在 FreeBSD 中的可能性用处。
- 如何建立、启动和停止 jail。
- jail 管理的基础知识，包括 Jail 内部与外部。

重要

jail 是一个强大的工具，但它不是安全问题的万金油。虽然 jail 内的进程不可能自行溢出，但有几种方法可以让 jail 外的非特权用户与 jail 内的特权用户串通，以获得主机环境中的高权限。

通过阻止主机环境中的非特权用户访问 jail 根目录，可以缓解大多数此类攻击。一般地，不应向具有 jail 访问权限的不受信的用户授予主机环境的访问权限。

17.2.与 Jail 有关的术语

为了便于更好地理解 FreeBSD 系统中与 jail 相关的部分、它们的内部结构以及它们与 FreeBSD 其他部分的交互方式，本章将进一步使用以下术语：

- chroot(8)[1011]（**命令**）

  使用 FreeBSD 系统调用 chroot(2)[1012] 来更改进程及其所有子进程的根目录的工具。

- chroot(2)[1013]（**环境**）

  在“chroot”中运行的进程的环境。它包括资源，例如文件系统中可见的部分，可用的用户和组 ID，网络接口和其他 IPC 机制等。

- jail(8)[1014]（**命令**）

  系统管理工具，允许在 jail 环境中启动进程。

- **主机（系统、进程、用户等）**

  jail 环境的控制系统。主机系统可以访问所有可用的硬件资源，并且可以控制 jail 环境内外的进程。主机系统与 jail 的一个重要区别是，对 jail 内的超级用户进程的限制不会对主机系统的进程加以实行。

- **hosted（系统、进程、用户等）**

  进程，用户或其他实体，其对资源的访问受到 FreeBSD jail 的限制。

[1011] https://www.freebsd.org/cgi/man.cgi?query=chroot&sektion=8&format=html
[1012] https://www.freebsd.org/cgi/man.cgi?query=chroot&sektion=2&format=html
[1013] https://www.freebsd.org/cgi/man.cgi?query=chroot&sektion=2&format=html
[1014] https://www.freebsd.org/cgi/man.cgi?query=jail&sektion=8&format=html

17.3.建立和控制 Jail

一些管理员将 jail 分为以下两种类型：“完整” jail，类似于真正的 FreeBSD 系统; 和 “服务” jail，专用于一个程序或服务，可能以特权运行。这只是一个概念上的划分，不影响建立 jail 的过程。在建立 “完整” jail 时，用户空间的源代码有两个选项：使用预构建的可执行文件（例如在安装镜像上提供的可执行文件）或从源代码建立。

17.3.1.安装 Jail

17.3.1.1.从网络安装 Jail

bsdinstall(8)[1015] 工具可用于获取和安装 jail 所需的可执行文件。这演示了如何选择镜像，将哪个发行版安装到目标目录中，以及 jail 的一些基本配置：

```
# bsdinstall jail /here/is/the/jail
```

命令完成后，下一步是配置主机以运行 jail。

17.3.1.2.从 ISO 安装 Jail

要从安装镜像安装用户空间，请先为 jail 创建根目录。可以通过将变量 `DESTDIR` 设置到正确的位置来完成。

启动 shell 并定义 `DESTDIR`：

```
# sh
# export DESTDIR=/here/is/the/jail
```

使用 ISO 安装时，按照 mdconfig(8)[1016] 中所述安装镜像：

```
# mount -t cd9660 /dev/`mdconfig -f cdimage.iso` /mnt
# cd /mnt/usr/freebsd-dist/
```

将可执行文件从安装镜像上的压缩包中提取到指定的目标中。最简单的是只需要提取基本系统，但也可执行完整安装。

要仅安装基本系统：

```
# tar -xf base.txz -C $DESTDIR
```

[1015] https://www.freebsd.org/cgi/man.cgi?query=bsdinstall&sektion=8&format=html
[1016] https://www.freebsd.org/cgi/man.cgi?query=mdconfig&sektion=8&format=html

要安装除内核之外的所有内容：

```
# for set in base ports; do tar -xf $set.txz -C $DESTDIR ; done
```

17.3.1.3.从源代码构建和安装 Jail

jail(8)[1017] 手册页说明了建立 jail 的过程：

```
# setenv D /here/is/the/jail
# mkdir -p $D    ①
# cd /usr/src
# make buildworld    ②
# make installworld DESTDIR=$D    ③
# make distribution DESTDIR=$D    ④
# mount -t devfs devfs $D/dev    ⑤
```

① 第一步是为 jail 选择一个位置。这是 jail 在 jail 主机的文件系统中的物理位置。一个常用选择是 **/usr/jail/jailname**，此处 *jailname* 是 jail 的主机名。**/usr/** 在通常情况下有足够的空间容纳 jail 文件系统，对于“完整” jail 来说，它基本上是 FreeBSD 基本系统默认安装的每个文件的副本。

② 如果你已经使用 `make world` 或 `make buildworld` 重新编译了你的用户空间，你可以跳过这一步并将现有的用户空间安装到新的 jail 中。

③ 这条命令将在文件系统中的 jail 物理位置的目录子树上安装必要的可执行文件、库、手册页等。

④`distribution` 这个 `make` 目标会安装所有需要的配置文件。简单地说，它把 **/usr/src/etc/** 的每个可安装文件都安装到 jail 环境的 **/etc** 目录：**$D/etc/** 下。

⑤ 在 jail 中挂载文件系统 devfs(8)[1018] 不是必须的。另一方面，所有或几乎所有软件都需要访问至少一个设备，这主要取决于给定应用程序的用途。控制对 jail 内部的设备访问是非常重要的，因为不恰当的设置可能允许攻击者在 jail 中做一些令人讨厌的事情。对 devfs(8)[1019] 的控制是通过 devfs(8)[1020] 和 devfs.conf(5)[1021] 手册页中描述的规则集进行管理的。

[1017] https://www.freebsd.org/cgi/man.cgi?query=jail&sektion=8&format=html
[1018] https://www.freebsd.org/cgi/man.cgi?query=devfs&sektion=8&format=html
[1019] https://www.freebsd.org/cgi/man.cgi?query=devfs&sektion=8&format=html
[1020] https://www.freebsd.org/cgi/man.cgi?query=devfs&sektion=8&format=html
[1021] https://www.freebsd.org/cgi/man.cgi?query=devfs.conf&sektion=5&format=html

17.3.2.配置主机

安装了 jail 后，就可以使用 jail(8)[1022] 工具启动它。jail(8)[1023] 工具采用四个必要参数，这些参数在 jail 概述[1024]中进行了说明。也可以指定其他参数，例如，使用特定用户的身份运行 jail 进程。命令参数取决于 jail 的类型：对于 虚拟系统来说，**/etc/rc** 是一个不错的选择，因为它将复制真正的 FreeBSD 系统的启动顺序；对于 服务 jail 来说，它取决于将在 jail 内运行的服务或应用程序。

jail 通常在系统引导时启动，FreeBSD 的 **rc** 机制提供了用简单的方法来做到这一点。

- 在 **jail.conf** 中配置 jail 参数：

```
www {
   host.hostname = www.example.org;           # 主机名
   ip4.addr = 192.168.0.10;                   # jail 的 IP 地址
   path = "/usr/jail/www";                    # jail 所在路径
   mount.devfs;                               # 在 jail 内部挂载 devfs
   exec.start = "/bin/sh /etc/rc";            # 启动命令
   exec.stop = "/bin/sh /etc/rc.shutdown";    # 停止命令
}
```

在 **rc.conf** 中将 jail 配置为在系统引导时启动：

```
jail_enable="YES" # 如设置为 NO，会禁止启动所有的 jail。
```

在 jail.conf(5)[1025] 中配置的 jail 在默认启动时将运行其中的 **/etc/rc** 脚本，该脚本默认 jail 是一个完整的虚拟系统。对于服务 jail，应通过对 `exec.start` 这个选项的适当设置来更改 jail 的默认启动命令。

> 注意
>
> 有关可用选项的完整列表，请参阅 jail.conf(5)[1026] 手册页。

service(8)[1027] 可用于手动启动或停止 jail，如果它的条目存在于 jail.conf 中：

```
# service jail start www
# service jail stop www
```

还可以用 jexec(8)[1028] 停止 jail。先使用 jls(8)[1029] 来获得 jail 的 `JID`，然后使用 jexec(8)[1030] 在该 jail 中运行关机脚本。

[1022] https://www.freebsd.org/cgi/man.cgi?query=jail&sektion=8&format=html
[1023] https://www.freebsd.org/cgi/man.cgi?query=jail&sektion=8&format=html
[1024] https://docs.freebsd.org/en/books/handbook/book/#jails-synopsis
[1025] https://www.freebsd.org/cgi/man.cgi?query=jail.conf&sektion=5&format=html
[1026] https://www.freebsd.org/cgi/man.cgi?query=jail.conf&sektion=5&format=html
[1027] https://www.freebsd.org/cgi/man.cgi?query=service&sektion=8&format=html
[1028] https://www.freebsd.org/cgi/man.cgi?query=jexec&sektion=8&format=html
[1029] https://www.freebsd.org/cgi/man.cgi?query=jls&sektion=8&format=html
[1030] https://www.freebsd.org/cgi/man.cgi?query=jexec&sektion=8&format=html

```
# jls
   JID  IP Address      Hostname                      Path
     3  192.168.0.10    www                           /usr/jail/www
# jexec 3 /etc/rc.shutdown
```

有关这方面的更多信息，请参阅 jail(8)[1031] 手册页。

17.4.微调和管理

你可以为 jail 设置许多不同的选项，并让 FreeBSD 主机系统与 jail 相交互，以支持更高级别的应用程序。本节介绍了：

- 一些可用于微调 jail 行为和安全限制的选项。
- 一些可以通过 FreeBSD ports 获得的用于 jail 管理的高级应用程序，他们可用于实现基于 jail 的整体解决方案。

17.4.1.FreeBSD 中用于调整 Jail 的系统工具

对 jail 的配置微调主要通过设置 sysctl(8)[1032] 变量来完成。系统提供了一个特殊的 sysctl 子树，全部相关的选项均在这棵子树中；=这就是 FreeBSD 内核的 `security.jail.*` 选项子树。以下是与 jail 有关的主要 sysctl，以及这些变量的默认值。这些名称的意思不言自明，但有关它们的更多信息，请参阅 jail(8)[1033] 和 sysctl(8)[1034] 手册页。

- `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`

这些变量可以由 *主机系统*的系统管理员用来增加或删除默认施加在 `root` 用户身上的一些限制。注意，某些限制是不能被移除的。不允许 jail(8)[1035] 内部的 `root` 挂载或卸载文件系统。在 jail 中的 `root` 用户也不

[1031] https://www.freebsd.org/cgi/man.cgi?query=jail&sektion=8&format=html
[1032] https://www.freebsd.org/cgi/man.cgi?query=sysctl&sektion=8&format=html
[1033] https://www.freebsd.org/cgi/man.cgi?query=jail&sektion=8&format=html
[1034] https://www.freebsd.org/cgi/man.cgi?query=sysctl&sektion=8&format=html
[1035] https://www.freebsd.org/cgi/man.cgi?query=jail&sektion=8&format=html

能加载或卸载 devfs(8)[1036] 规则集，不能设置防火墙规则，也不能做许多其他需要修改内核内数据的管理操作，比如设置内核的 `securelevel`。

FreeBSD 的基本系统包含一组基本的工具，用于查看有关活动 jail 的信息，以及连接 jail 并执行管理。jls(8)[1037] 和 jexec(8)[1038] 命令都是 FreeBSD 基本系统的一部分，可以用来执行以下简单的任务：

- 列出使用中的 jail 及其相应的 jail 标识（JID）、IP 地址、主机名和路径。
- 从主机系统连接到正在运行的 jail，并在 jail 内部运行命令或在 jail 内部执行管理任务。这在 `root` 用户想要干净利落地关闭 jail 时尤其有用。jexec(8)[1039] 工具也可用于在 jail 中启动 shell，以便在其中进行管理。例如：

```
# jexec 1 tcsh
```

17.4.2.FreeBSD Ports 中的高级管理工具

在用于 jail 管理的众多第三方工具中，最完整和最有用的软件之一是 sysutils/ezjail[1040]。它是一组有助于 jail(8)[1041] 管理的脚本。有关详细信息，请参阅手册中有关 ezjail 的部分[1042]。

17.4.3.给 Jail 打补丁与更新

应该在主机操作系统中对 jail 进行更新，因为在 jail 内部给用户空间打补丁的尝试可能会失败，因为 FreeBSD 中默认禁止在 jail 中使用 chflags(1)[1043]，而这会阻止某些文件的替换。可以改变这种机制，但建议使用 freebsd-update(8)[1044] 来维护 jail。用 `-b` 指定要更新的 jail 的路径。

要将 jail 更新到它正在运行的 FreeBSD 版本的最新补丁版本，需要在主机上执行以下命令：

```
# freebsd-update -b /here/is/the/jail fetch
# freebsd-update -b /here/is/the/jail install
```

要将 jail 升级到新的主要或次要版本，请首先按照 “执行主要和次要版本升级” [1045]中所述升级主机系统。升级并重新引导主机后，再升级 jail。例如，要从 12.2-RELEASE 升级到 12.3-RELEASE，请在主机上运行：

[1036] https://www.freebsd.org/cgi/man.cgi?query=devfs&sektion=8&format=html
[1037] https://www.freebsd.org/cgi/man.cgi?query=jls&sektion=8&format=html
[1038] https://www.freebsd.org/cgi/man.cgi?query=jexec&sektion=8&format=html
[1039] https://www.freebsd.org/cgi/man.cgi?query=jexec&sektion=8&format=html
[1040] https://cgit.freebsd.org/ports/tree/sysutils/ezjail/pkg-descr
[1041] https://www.freebsd.org/cgi/man.cgi?query=jail&sektion=8&format=html
[1042] https://docs.freebsd.org/en/books/handbook/Jail/#Jail-ezjail
[1043] https://www.freebsd.org/cgi/man.cgi?query=chflags&sektion=1&format=html
[1044] https://www.freebsd.org/cgi/man.cgi?query=freebsd-update&sektion=8&format=html
[1045] https://docs.freebsd.org/en/books/handbook/cutting-edge/index.html#freebsdupdate-upgrade

```
# freebsd-update -b /here/is/the/jail --currently-running 12.2-RELEASE -r 12.3-
↪RELEASE upgrade
# freebsd-update -b /here/is/the/jail install
# service jail restart myjail
# freebsd-update -b /here/is/the/jail install
```

然后，如果是主要版本升级，请重新安装所有已安装的软件包并再次重新启动 jail。这是必需的，因为在 FreeBSD 的主要版本之间升级时 ABI 版本会发生变化。在主机上执行：

```
# pkg -j myjail upgrade -f
# service jail restart myjail
```

17.5.更新多个 Jail

对多个 jail 的管理可能会成为问题，因为每个 jail 在升级时都必须从头开始重新编译。如果要创建并手动更新许多 jail，这可能会十分耗时且无聊。

本节演示了一种解决此问题的方案，其方法是使用 mount_nullfs(8)[1046] 只读挂载在 jail 之间尽可能多的以安全的方式共享内容，以便让更新更简单。从而使得将单个服务（如 HTTP，DNS 和 SMTP）放入不同的 jail 方案更具吸引力。此外，它还提供了一种增加、删除和升级 jail 的简单方法。

> 注意
>
> 有更简单的解决方案，例如 ezjail，它提供了一种更简单的 FreeBSD jail 管理方法，但其通用性不如此方法。在用 ezjail 管理 jail[1047] 中有更详细的介绍。

本节中介绍的配置的目标是：

- 建立简单易懂的 jail 结构，不必在每个 jail 上执行完整的 installworld 操作。
- 轻松增加或删除 jail。
- 轻松更新或升级已有 jail。
- 使运行定制的 FreeBSD 分支成为可能。
- 对安全问题持偏执态度，尽可能地减少妥协的可能性。
- 尽可能节省空间和节点。

这种设计依赖于一份单一的、只读的主模板，它被安装到每个 jail 中，每个 jail 有一个读写设备。这个设备可以是单独的物理磁盘，分区，或者虚拟节点支持的内存盘。在这个例子中使用了可读写的 nullfs 挂载。

文件系统布局如下：

- 每个 jail 都位于 **/home** 分区下。

[1046] https://www.freebsd.org/cgi/man.cgi?query=mount_nullfs&sektion=8&format=html

[1047] https://docs.freebsd.org/en/books/handbook/Jail/#Jail-ezjail

- 每个 jail 都挂载在 **/home/j** 目录下。
- **/home/j/mroot** 是每个 jail 共用的模板，对所有 jail 而言都是只读分区。
- 每个 jail 都有一个对应的空目录在 **/home/j** 中。
- 每个 jail 都有一个 **/s** 目录，该目录将链接到系统的可读写部分。
- 每个 jail 都有自己的可读写系统，该系统基于 **/home/j/skel**。
- 每个 jail 的可读写空间创建在 **/home/js** 中。

17.5.1.创建模板

本节介绍创建主模板所需的步骤。

建议首先使用 “从源代码更新 FreeBSD” [1048]中的说明将 FreeBSD 主机系统更新到最新的 -RELEASE 分支。此外，此模板还依赖 sysutils/cpdup[1049]，可从二进制包或 ports 安装。将使用 Git[1050] 下载 FreeBSD Ports。

1. 首先，为只读文件系统建立一个目录，它将包含 jail 的 FreeBSD 可执行文件。然后，进入 FreeBSD 源代码所在目录，并将只读文件系统安装到 jail 模板中：

```
# mkdir /home/j /home/j/mroot
# cd /usr/src
# make installworld DESTDIR=/home/j/mroot
```

1. 接下来，为 jail 准备 FreeBSD ports 以及 FreeBSD 源代码，这是 mergemaster 所必需的：

```
# cd /home/j/mroot
# mkdir usr/ports
# git clone -o freebsd https://git.FreeBSD.org/ports.git /home/j/mroot/usr/
↪ports
# cpdup /usr/src /home/j/mroot/usr/src
```

1. 为系统的可读写部分创建框架：

```
# 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
```

1. 使用 mergemaster 安装缺少的配置文件。然后，删除 mergemaster 创建的多余目录：

[1048] https://docs.freebsd.org/en/books/handbook/cutting-edge/index.html#makeworld

[1049] https://cgit.freebsd.org/ports/tree/sysutils/cpdup/pkg-descr

[1050] https://docs.freebsd.org/en/books/handbook/mirrors/#git

```
# 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
```

1. 现在，将可读写文件系统符号链接到只读文件系统。确保在正确的 **s/** 位置创建符号链接，因为在错误的位置创建目录将导致安装失败。

```
# 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
```

1. 最后一步，创建一个包含以下行的通用配置文件 **/home/j/skel/etc/make.conf**：

```
WRKDIRPREFIX?=  /s/portbuild
```

这使得在每个 jail 中分别编译 FreeBSD ports 成为可能。请记住，ports 目录是只读系统的一部分。而 WRKDIRPREFIX 则使得编译过程得以在 jail 中的可读写部分完成。

17.5.2.创建 Jail

jail 模板现在可用于在 **/etc/rc.conf** 中设置和配置 jail。此示例了演示如何创建 3 个 jail：NS、MAIL 和 WWW。

1. 将以下行添加到 **/etc/fstab**，以便 jail 的只读模板和可读写空间在各自的 jail 中可用：

```
/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
```

为了防止 fsck 在引导期间检查 nullfs 挂载，并防止转储备份 jail 中的只读 nullfs 挂载，将最后两列都设置为 0。

1. 在 **/etc/rc.conf** 中配置 jail：

```
jail_enable="YES"
jail_set_hostname_allow="NO"
```

(continues on next page)

(continued from previous page)

```
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"
```

应该把 *jailnamerootdir* 变量设置为 **/usr/home** 而不是 **/home**，因为在默认安装的 FreeBSD 上，**/home** 的物理路径是 **/usr/home**。*jailnamerootdir* 变量必须是一个不包含符号连接的路径，否则 jail 将拒绝启动。

1. 为每个 jail 创建只读文件系统所需的挂载点：

```
# mkdir /home/j/ns /home/j/mail /home/j/www
```

1. 使用 sysutils/cpdup[1051] 将可读写模板安装到每个 jail 中：

```
# mkdir /home/js
# cpdup /home/j/skel /home/js/ns
# cpdup /home/j/skel /home/js/mail
# cpdup /home/j/skel /home/js/www
```

1. 在这个阶段，jail 已经建成并准备运行。首先，为每个 jail 挂载所需的文件系统，然后启动它们：

```
# mount -a
# service jail start
```

现在 jail 应该已经运行了。要检查它们是否正常运行，请使用 `jls`。其输出应类似于以下内容：

```
# 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
```

[1051] https://cgit.freebsd.org/ports/tree/sysutils/cpdup/pkg-descr

此时，应该可以登录到每一个 jail，以添加新的用户，或者配置守护程序了。JID 列表示每个运行中的 jail 的标识。例如使用下面的命令可执行 JID 为 3 的 jail 的管理任务：

```
# jexec 3 tcsh
```

17.5.3.升级

此设置的设计提供了一种简单的方法来升级现有的 jail，同时最大限度地减少其停机时间。此外，它还提供了一种在出现问题时回滚到旧版本的方法。

1. 第一步是升级主机系统。然后，在 **/home/j/mroot2** 中创建新的临时只读模板。

```
# mkdir /home/j/mroot2
# cd /usr/src
# make installworld DESTDIR=/home/j/mroot2
# cd /home/j/mroot2
# cpdup /usr/src usr/src
# mkdir s
```

installworld 将创建一些多余的目录，应将其删除：

```
# chflags -R 0 var
# rm -R etc var root usr/local tmp
```

1. 为主文件系统重新创建可读写符号链接：

```
# 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
```

1. 接下来，关闭 jail：

```
# service jail stop
```

1. 卸载原先的文件系统，因为可读写系统需要连接到只读系统（**/s**）：

```
# umount /home/j/ns/s
# umount /home/j/ns
# umount /home/j/mail/s
# umount /home/j/mail
```

(continues on next page)

(continued from previous page)

```
# umount /home/j/www/s
# umount /home/j/www
```

1. 移动旧的只读文件系统，并将其替换为新的文件系统。如果出现问题，这将用作旧的只读文件系统的备份和存档。此处使用的文件命名对应于创建新的只读文件系统。此外将原来的 FreeBSD ports 移到新的文件系统上以节省一些空间和节点：

```
# cd /home/j
# mv mroot mroot.20060601
# mv mroot2 mroot
# mv mroot.20060601/usr/ports mroot/usr
```

1. 此时，新的只读模板已准备就绪，因此唯一剩下的任务是重新挂载文件系统并启动 jail：

```
# mount -a
# service jail start
```

最后使用 `jls` 来检查 jail 是否正确启动。在每个 jail 中运行 `mergemaster` 来更新配置文件。

17.6.使用 ezjail 管理 Jail

创建和管理多个 jail 可能很快会变得繁琐且容易出错。Dirk Engling 开发的 ezjail 自动化软件大大的简化了许多 jail 任务。*basejail* 被创建为模板。其他 jail 使用 mount_nullfs(8)[1052] 来共享许多 basejail 目录，而无需使用多余的磁盘空间。在安装应用程序之前，每个额外的 jail 仅占用几兆字节的磁盘空间。升级 *basejail* 中用户空间的副本会自动升级所有其他 jail。

其他优点和功能在 ezjail 网站上有详细描介绍：https://erdgeist.org/arts/software/ezjail/。

17.6.1.安装 ezjail

ezjail 的安装包括增加一个环回接口以在 jail 安装 ports 或软件包以及启用服务时使用。

1. 为了保持 jail 环回流量离开主机的环回网络接口 `lo0`，通过向 **/etc/rc.conf** 添加一个条目来创建第二个环回接口：

```
cloned_interfaces="lo1"
```

第二个环回接口 `lo1` 将在系统启动时创建。也可以手动创建它，而无需重启系统：

```
# service netif cloneup
Created clone interfaces: lo1.
```

[1052] https://www.freebsd.org/cgi/man.cgi?query=mount_nullfs&sektion=8&format=html

可以允许 jail 使用这个二级环回接口的别名而不干扰主机。

在 jail 内，对环回地址 127.0.0.1 的访问被重定向到分配给 jail 的第一个 IP 地址。为了使 jail 的环回地址与新的 lo1 接口相对应，在创建新 jail 时，必须首先在接口和 IP 地址列表中指定该接口。

在 127.0.0.0/8 网块中给每个分配一个 jail 唯一的环回地址。

1. 安装 sysutils/ezjail[1053]：

```
# cd /usr/ports/sysutils/ezjail
# make install clean
```

1. 通过将这行添加到 **/etc/rc.conf** 来启用 ezjail：

```
ezjail_enable="YES"
```

1. 该服务将在系统启动时自动启动。它可以立即为当前会话启动：

```
# service ezjail start
```

17.6.2.初始设置

安装 ezjail 后，可以创建和补充 basejail 目录。此步骤仅需要在 jail 主机上执行一次。

在这两个示例中。-p 让用 portsnap(8)[1054] 下载 ports 到 basejail。所有的 jail 都将共享这个 ports 目录的副本。为 jail 使用单独的 ports 目录副本可以将它们与主机隔离。在 ezjailFAQ 中有更详细的解释：http://erdgeist.org/arts/software/ezjail/#FAQ。

1. 用 FreeBSD-RELEASE 来建立 jail

对于基于与主机相匹配的 FreeBSD RELEASE 的 basejail，使用 install。例如，在一台运行 FreeBSD 13-STABLE 的主机上，最新的 FreeBSD-13 RELEASE 版本将被安装在 jail 中：

```
# ezjail-admin install -p
```

1. 用 installworld 建立 jail

basejail 可以用 ezjail-admin update 从 buildworld 在主机上创建的可执行文件安装。

在这个例子中，FreeBSD 10-STABLE 是从源代码构建的。jail 目录被创建。然后执行 installworld，将主机的 **/usr/obj** 安装到 basejail 中：

```
# ezjail-admin update -i -p
```

默认情况下使用主机的 **/usr/src**。可以用 -s 和路径指定主机上不同的源目录，或者用 **/usr/local/etc/ezjail.conf** 中的 ezjail_sourcetree 设置。

[1053] https://cgit.freebsd.org/ports/tree/sysutils/ezjail/pkg-descr

[1054] https://www.freebsd.org/cgi/man.cgi?query=portsnap&sektion=8&format=html

技巧 >basejail ports 由其他 jail 共享。但是，下载的 distfile 存储在它们各自的 jail 中。默认情况下，这些文件存储在每个 jail 内的 **/var/ports/distfile** 中。在编译 ports 时，每个 jail 中的 **/var/ports** 也用作工作目录。

技巧

默认情况下，使用 FTP 协议下载用于安装 basejail 的软件包。防火墙或代理配置可以阻止或干扰 FTP 传输。HTTP 协议的工作方式不同，可规避这些问题。可以通过在 **/usr/local/etc/ezjail.conf** 中为特定下载镜像站指定完整链接来选择 HTTP：

```
ezjail_ftphost=http://ftp.FreeBSD.org
```

有关站点列表，请参阅镜像站[1055] 部分。

17.6.3.建立和启动新 Jail

使用 `ezjail-admin create` 建立新的 jail。在这些示例中，环回接口 `lo1` 按上述方式使用。

示例：建立和启动新 jail

1. 创建 jail，指定名称、要使用的环回和网络接口及其 IP 地址。在此示例中，jail 被命名为 `dnsjail`。

```
# ezjail-admin create dnsjail 'lo1|127.0.1.1,em0|192.168.1.50'
```

技巧

大多数网络服务可以 jail 中正常运行。某些网络服务，最明显的是 ping(8)[1056]会使用原始网络套接字。在 jail 中，出于安全考虑，默认情况下禁用原始网络套接字。依赖它们的服务将不起作用。

有时，jail 中确实需要原始套接字。例如，网络监控工具经常使用 ping(8)[1057] 来检查其他计算机的可用性。当 jail 中实际需要原始网络套接字时，可以通过编辑各个 jail 的 ezjail 配置文件 **/usr/local/etc/ezjail/jailname** 来启用它们。修改 `parameters` 条目：

```
export jail_jailname_parameters="allow.raw_sockets=1"
```

请勿启用原始网络套接字，除非 jail 中的服务确实需要它们。

1. 启动 jail：

```
# ezjail-admin start dnsjail
```

1. 在 jail 中使用控制台：

```
# ezjail-admin console dnsjail
```

[1055] https://docs.freebsd.org/en/books/handbook/mirrors/index.html#mirrors

[1056] https://www.freebsd.org/cgi/man.cgi?query=ping&sektion=8&format=html

[1057] https://www.freebsd.org/cgi/man.cgi?query=ping&sektion=8&format=html

jail 正在运行，可以完成其他配置。此时可进行的一般设置包括：

1. 设置 root 密码

连接到 jail 并设置 root 用户的密码：

```
# ezjail-admin console dnsjail
# passwd
Changing local password for root
New Password:
Retype New Password:
```

1. 时区配置

可以用 tzsetup(8)[1058] 来设置 jail 的时区。为避免虚假的错误消息，可以注释或删除 **/etc/crontab** 中的 adjkerntz(8)[1059] 条目。此任务使用时区更改来更新计算机的硬件时钟，但不允许 jail 访问该硬件。

1. 域名服务器

在 **/etc/resolv.conf** 中输入域名服务器一行，以便让 DNS 在 jail 中工作。

1. 编辑 **/etc/hosts**

更改地址并将 jail 名称添加到 **/etc/hosts** 中的 `localhost` 条目中。

1. 配置 **/etc/rc.conf**

在 **/etc/rc.conf** 中输入配置设置。这很像配置一台完整的计算机。此处未设置主机名和 IP 地址。这些值已由 jail 配置提供。

在配置 jail 后，可以安装为其创建 jail 的应用程序。

技巧

某些 port 必须使用特殊选项编译才能在 jail 中使用。例如，网络监控插件软件 net-mgmt/nagios-plugins[1060] 和 net-mgmt/monitoring[1061] 插件都有一个 `JAIL` 选项，必须启用该选项才能在 jail 中正常工作。

17.6.4.更新 Jail

17.6.4.1.更新操作系统

由于 basejail 的用户空间副本由其他 jail 共享，因此更新 basejail 会自动更新所有其他 jail。可以使用软件源更新或二进制更新。

要从主机上的源代码编译 world，然后将其安装在 basejail 中，请使用：

```
# ezjail-admin update -b
```

[1058] https://www.freebsd.org/cgi/man.cgi?query=tzsetup&sektion=8&format=html

[1059] https://www.freebsd.org/cgi/man.cgi?query=adjkerntz&sektion=8&format=html

[1060] https://cgit.freebsd.org/ports/tree/net-mgmt/nagios-plugins/pkg-descr

[1061] https://cgit.freebsd.org/ports/tree/net-mgmt/monitoring-plugins/pkg-descr

如果 world 已经在主机上编译，请使用以下命令将其安装在 basejail 中：

```
# ezjail-admin update -i
```

二进制更新使用 freebsd-update(8)[1062]。这些更新与直接运行 freebsd-update(8)[1063] 具有相同的限制。最重要的一点是，只有 -RELEASE 版本的 FreeBSD 可以使用此方法。

用 basejail 将主机的 FreeBSD 版本更新到最新补丁版本。例如，从 RELEASE-p1 更新到 RELEASE-p2。

```
# ezjail-admin update -u
```

要将 basejail 升级到新版本，请首先按照“执行主要和次要版本升级”[1064]中所述升级主机系统。升级并重新引导主机后，即可升级 basejail。freebsd-update(8)[1065] 无法确定 basejail 中当前安装了哪个版本。因此必须指定原始版本。使用 file(1)[1066] 确定 basejail 中的原始版本：

```
# file /usr/jail/basejail/bin/sh
/usr/jail/basejail/bin/sh: ELF 64-bit LSB executable, x86-64, version 1 (FreeBSD),␣
↪dynamically linked, interpreter /libexec/ld-elf.so.1, for FreeBSD 13.0, FreeBSD-
↪style, stripped
```

现在，使用此命令执行从当前的主机系统版本到 13.0-RELEASE 的升级：

```
# ezjail-admin update -U -s 13.0-RELEASE
```

更新 basejail 后。必须运行 mergemaster(8)[1067] 来更新每个 jail 的配置文件。

如何使用 mergemaster(8)[1068] 取决于 jail 的目的和可信度。如果 jail 的服务或用户不可信，则 mergemaster(8)[1069] 只能在该 jail 内部运行：

例 **32.** 在不可信的 **Jail** 上运行 mergemaster(8)[1070]

删除从 jail 的 **/usr/src** 到 basejail 的链接，并在 jail 中创建一个新的 **/usr/src** 作为挂载点。将主机的 **/usr/src** 只读挂载到 jail 的新 **/usr/src** 挂载点上：

```
# rm /usr/jail/jailname/usr/src
# mkdir /usr/jail/jailname/usr/src
# mount -t nullfs -o ro /usr/src /usr/jail/jailname/usr/src
```

[1062] https://www.freebsd.org/cgi/man.cgi?query=freebsd-update&sektion=8&format=html
[1063] https://www.freebsd.org/cgi/man.cgi?query=freebsd-update&sektion=8&format=html
[1064] https://docs.freebsd.org/en/books/handbook/cutting-edge/index.html#freebsdupdate-upgrade
[1065] https://www.freebsd.org/cgi/man.cgi?query=freebsd-update&sektion=8&format=html
[1066] https://www.freebsd.org/cgi/man.cgi?query=file&sektion=1&format=html
[1067] https://www.freebsd.org/cgi/man.cgi?query=mergemaster&sektion=8&format=html
[1068] https://www.freebsd.org/cgi/man.cgi?query=mergemaster&sektion=8&format=html
[1069] https://www.freebsd.org/cgi/man.cgi?query=mergemaster&sektion=8&format=html
[1070] https://www.freebsd.org/cgi/man.cgi?query=mergemaster&sektion=8&format=html

在 jail 中获取控制台：

```
# ezjail-admin console jailname
```

在 jail 里运行 mergemaster。然后退出 jail 控制台：

```
# cd /usr/src
# mergemaster -U
# exit
```

最后，卸载 jail 的 **/usr/src**：

```
# umount /usr/jail/jailname/usr/src
```

例 **33.**可信 **Jail** 与 mergemaster(8)[1071]

如果 jail 中的用户和服务是可信的。mergemaster(8)[1072] 可以在主机运行：

```
# mergemaster -U -D /usr/jail/jailname
```

技巧

在主要版本更新后，建议使用 sysutils/ezjail[1073]，以确保你的版本正确。因此，请输入：

```
# pkg-static upgrade -f pkg`
```

以升级或降级到相应的版本。

17.6.4.2.更新 ports

基本系统 jail 中的 port 由其他 Jail 共享。更新 port 的该副本也会为其他 jail 提供更新的版本。

basejail ports 可使用 portsnap(8)[1074] 进行更新：

```
# ezjail-admin update -P
```

[1071] https://www.freebsd.org/cgi/man.cgi?query=mergemaster&sektion=8&format=html

[1072] https://www.freebsd.org/cgi/man.cgi?query=mergemaster&sektion=8&format=html

[1073] https://cgit.freebsd.org/ports/tree/sysutils/ezjail/pkg-descr

[1074] https://www.freebsd.org/cgi/man.cgi?query=portsnap&sektion=8&format=html

17.6.5.控制 Jail

17.6.5.1.停止和启动 Jail

ezjail 在计算机启动时会自动启动 jail。可以使用 `stop` 和 `start` 手动停止和重新启动 jail：

```
# ezjail-admin stop sambajail
Stopping jail: sambajail.
```

默认情况下，jail 在主机启动时自动启动。可以通过 `config` 禁用自动启动：

```
# ezjail-admin config -r norun seldomjail
```

这将在下次启动主机时生效。已经运行的 jail 不会停止。

启用自动启动非常相似：

```
# ezjail-admin config -r run oftenjail
```

17.6.5.2.归档和恢复 Jail

使用 `archive` 来创建一个 **.tar.gz** 的 jail 归档。文件名由 jail 的名称和当前日期组成。归档文件被写入归档目录——**/usr/jail/ezjail_archives**。可以通过在配置文件中设置 `ezjail_archivedir` 来选择其他的归档目录。

可以将归档文件作为备份复制到其他位置，也可以使用 `restore` 从中恢复现有 jail。可以从存档中创建新的 jail，从而为克隆现有 jail 提供便捷的方法。

关闭 jail 并归档到名为 `wwwserver`：

```
# ezjail-admin stop wwwserver
Stopping jail: wwwserver.
# ezjail-admin archive wwwserver
# ls /usr/jail/ezjail-archives/
wwwserver-201407271153.13.tar.gz
```

根据在上一步中创建的归档创建一个名为 `wwwserver-clone` 的新 jail。使用 `em1` 接口并分配新的 IP 地址以避免与原始 IP 地址冲突：

```
# ezjail-admin create -a /usr/jail/ezjail_archives/wwwserver-201407271153.13.tar.gz␣
↪wwwserver-clone 'lo1|127.0.3.1,em1|192.168.1.51'
```

17.6.6.完整示例：Jail 中的 BIND

将 BINDDNS 服务器放入 jail 可以通过隔离来提高安全性。此示例创建一个简单的仅缓存域名服务器。

- jail 将被称为 `dns1`。
- jail 将使用主机接口 `re0` 上的 IP 地址 `192.168.1.240`。
- 上游 ISP 的 DNS 服务器位于 `10.0.0.62` 和 `10.0.0.61`。
- 如初始设置[1075]中所示，已经创建 basejail，并且安装了 ports。

例 34.在 jail 中运行 BIND

通过向 **/etc/rc.conf** 添加一行来创建克隆的环回接口：

```
cloned_interfaces="lo1"
```

立即创建新的环回接口：

```
# service netif cloneup
Created clone interfaces: lo1.
```

创建 jail：

```
# ezjail-admin create dns1 'lo1|127.0.2.1,re0|192.168.1.240'
```

启动 jail，连接到在其上运行的控制台，然后执行一些基本配置：

```
# ezjail-admin start dns1
# ezjail-admin console dns1
# passwd
Changing local password for root
New Password:
Retype New Password:
# tzsetup
# sed -i .bak -e '/adjkerntz/ s/^/#/' /etc/crontab
# sed -i .bak -e 's/127.0.0.1/127.0.2.1/g; s/localhost.my.domain/dns1.my.
↪domain dns1/' /etc/hosts
```

在 **/etc/resolv.conf** 中临时设置上游 DNS 服务器，以便可以下载 ports：

```
nameserver 10.0.0.62
nameserver 10.0.0.61
```

仍使用 jail 控制台，安装 dns/bind99[1076]。

[1075] https://docs.freebsd.org/en/books/handbook/jail/#jail-ezjail-initialsetup

[1076] https://cgit.freebsd.org/ports/tree/dns/bind99/pkg-descr

```
# make -C /usr/ports/dns/bind99 install clean
```

通过编辑 **/usr/local/etc/namedb/named.conf** 来配置域名服务器。

创建允许向此域名服务器发送 DNS 查询的地址和网络的访问控制列表（ACL）。此部分将添加到文件中已有的 options 部分之前：

```
...
// or cause huge amounts of useless Internet traffic.
...
// or cause huge amounts of useless Internet traffic.

acl "trusted" {
  192.168.1.0/24;
  localhost;
  localnets;
};

options {
...
```

在 listen-on 设置中使用 jail 的 IP 地址来接受来自网络上其他计算机的 DNS 查询:

```
listen-on   { 192.168.1.240; };
```

通过更改 forwarders 该部分来创建一个简单的仅缓存 DNS 域名服务器。原始文件包含：

```
/*
  forwarders {
      127.0.0.1;
  };
*/
```

通过删除 /* 和 */ 行来取消注释该部分。输入上游 DNS 服务器的 IP 地址。紧跟在 forwarders 部分之后，添加对前面定义的 trusted ACL 的引用：

```
forwarders {
    10.0.0.62;
    10.0.0.61;
};

allow-query { any; };
allow-recursion{ trusted; };
allow-query-cache { trusted; };
```

在 **/etc/rc.conf** 中启用该服务：

```
named_enable="YES"
```

启动并测试域名服务器：

```
# service named start
wrote key file "/usr/local/etc/namedb/rndc.key"
Starting named.
# /usr/local/bin/dig @192.168.1.240 freebsd.org
```

响应包括

```
;; Got answer;
```

表明新的 DNS 服务器正在工作。长时间延迟后出现响应，例如

```
;; connection timed out; no servers could be reached
```

表明发生错误。请检查配置设置，并确保任何本地防火墙允许对上游 DNS 服务器进行新的 DNS 访问。

新的 DNS 服务器可以像其他本地计算机一样，将自身用于本地域名解析。在客户端计算机的 **/etc/resolv.conf** 中设置 DNS 服务器的地址：

```
nameserver 192.168.1.240
```

可以将本地 DHCP 服务器配置为为本地 DNS 服务器提供此地址，从而在 DHCP 客户端上提供自动配置。

第十八章强制访问控制

18.1.概述

FreeBSD 支持基于 POSIX.1e® 草案的安全扩展。这些安全机制包括文件系统访问控制列表（“访问控制列表”[1077]）和强制访问控制（MAC）。MAC 允许加载访问控制模块以实现安全策略。某些模块为系统的一小部分提供保护，从而加固特定服务。其他的则为所有主体和客体提供全面的标记安全性。定义的强制部分操作控制措施由管理员和操作系统执行。这与自主访问控制（DAC）的默认安全机制形成鲜明对比——在该机制中，强制措施由用户自行决定。

本章重点介绍 MAC 框架和 FreeBSD 提供的用于启用各种安全机制的一组可插拔的安全策略模块。

读完本章，你将知道：

- 与 MAC 框架相关的术语。
- MAC 安全策略模块的功能以及标记策略和非标记策略之间的区别。
- 在配置系统使用 MAC 框架之前要考虑的事项。
- 哪些 MAC 安全策略模块包含在 FreeBSD 中以及如何配置它们。
- 如何使用 MAC 框架实现更安全的环境。
- 如何测试 MAC 配置以确保框架已正确实施。

在阅读本章之前，你应该：

- 了解 UNIX® 和 FreeBSD 基础（FreeBSD 基础[1078]）。
- 熟悉安全性以及它与 FreeBSD（安全）[1079]的关系。

警告

[1077] https://docs.freebsd.org/en/books/handbook/security/index.html#fs-acl

[1078] https://docs.freebsd.org/en/books/handbook/basics/index.html#basics

[1079] https://docs.freebsd.org/en/books/handbook/security/index.html#security

错误的 MAC 配置可能会导致系统无法访问、引起用户的烦恼或无法使用 Xorg 提供的功能。更重要的是，不应完全依赖 MAC 来确保系统安全。MAC 框架仅增强现有的安全策略。若无健全的安全措施和定期的安全审核，系统将永远不会彻底安全。

本章中包含的示例仅用于演示目的，不应在生产系统上应用示例设置。实施任何安全策略都需要大量的思考、适当的设计和完全的测试。

虽然本章涵盖了与 MAC 框架相关的诸多安全问题，但新的 MAC 安全策略模块的开发成果则不会涉及。MAC 框架中包含的许多安全策略模块具有特定属性，这些属性用于测试和新模块开发。有关这些安全策略模块及其提供的众多机制的详细信息，请参阅 mac_test(4)[1080]、mac_stub(4)[1081] 和 mac_none(4)[1082]。

18.2.关键术语

在提及 MAC 框架时，使用以下关键术语：

- 区间（*compartment*）：指一组要被划分或分离的程序和数据，其中用户被明确授予对系统特定组件的访问权限。区间表示分组，如工作组、部门、项目或主题。区间使实现基于按需知密的安全策略成为可能。
- 完整性（*integrity*）：对数据的信任级别。数据完整性提升，该数据的可信度也随之提升。
- 级别（*level*）：安全属性的增加或减少设置。随着级别的增加，其安全性也被认为会提高。
- 标签（*label*）：一种安全属性，可应用于系统中的文件、目录或其他项目。它可以被视为加密印章。在文件上放置标签后，标签会说明该文件的安全属性，并且仅允许具有类似安全设置的文件、用户和资源进行访问。标签值的含义和解释取决于策略配置。某些策略将标签视为表示对象的完整性或机密性，而其他策略可能使用标签来保存访问规则。
- 多重标签（*multilabel*）：此属性是一个文件系统选项，可以在单用户模式下使用 tunefs(8)[1083] 设置，也可以在系统引导期间使用 fstab(5)[1084] 进行设置，还可以在创建文件系统期间进行设置。此选项允许管理员在不同的主体上应用不同的 MAC 标签。此选项仅适用于支持标记的安全策略模块。
- 单一标签（*single label*）：整个文件系统使用一个标签对数据流实施访问控制的策略。只要 多重标签未设置，所有文件都将符合相同的标签设置。
- 客体（*object*）：在 主体的导向下，信息流经的实体。这包括目录、文件、字段、屏幕、键盘、存储器、磁存储、打印机或任何其他数据存储或移动设备。客体是数据容器或系统资源。对一个客体的访问实际上意味着对其数据的访问。
- 主体（*subject*）：导致信息在 主体（如用户、用户进程或系统进程）之间流动的任何活动实体。在 FreeBSD 上，主体几乎总是代表用户在某个进程中运行的某一线程。
- 策略（*policy*）：一套定义如何实现目标的规则。策略通常记录如何处理某些项目。在本章中，策略被视为控制数据和信息流的规则集合，并定义谁有权访问该数据和信息。

[1080] https://www.freebsd.org/cgi/man.cgi?query=mac_stub&sektion=4&format=html
[1081] https://www.freebsd.org/cgi/man.cgi?query=mac_stub&sektion=4&format=html
[1082] https://www.freebsd.org/cgi/man.cgi?query=mac_none&sektion=4&format=html
[1083] https://www.freebsd.org/cgi/man.cgi?query=tunefs&sektion=8&format=html
[1084] https://www.freebsd.org/cgi/man.cgi?query=fstab&sektion=5&format=html

- 高水位线（*high water mark*）：高水位线策略允许提高安全级别，以便访问更高级别的信息。在大多数情况下，在过程完成后会恢复原始级别。目前，FreeBSD 的 MAC 框架不包括这种类型的策略。
- 低水位线（*low water mark*）：低水位线策略允许降低安全级别，以便访问安全性较差的信息。在大多数情况下，在该过程完成后，将还原用户的原始安全级别。在 FreeBSD 中唯一使用它的安全策略模块是 mac_lomac(4)[1085]。
- 敏感性（*sensitivity*）：通常在讨论多级别安全（MLS）时使用。敏感度级别代表了数据的重要性或机密性。敏感度提高，数据的保密性或机密性的重要性也随之提高。

18.3.了解 MAC 标签

MAC 标签是一种安全属性，可应用于整个系统的主体和客体。在设置标签时，管理员必须了解其含义，以防止系统出现意外或不愿意看到的行为。一个客体上的可用属性取决于加载的策略模块，因为策略模块以不同的方式解释它们的属性。

客体上的安全标签是作为策略的安全访问控制决策的一部分来使用的。对于某些策略，标签包含了做出决定所需的所有信息。在其他策略中，标签可以作为更大的一套规则的一部分来处理。

有两种类型的标签策略：单一标签和多重标签。默认情况下，系统将使用单一标签。管理员应该了解每一种的优点和缺点，以便实施符合系统安全模型要求的策略。

单一标签安全策略只允许对每个主体或客体使用一个标签。由于单一标签策略在整个系统中执行一套访问权限，它提供了较低的管理开销，但减少支持标签的策略的灵活性。然而，在许多环境中，单一标签策略可能是所需的。

单一标签策略有点类似于 DAC，因为 `root` 配置策略，使用户被放在适当的类别和访问级别中。一个值得注意的区别是，许多策略模块也可以限制 `root`。然后，对客体的基本控制将被释放给组，但 `root` 可以在任何时候撤销或修改这些设置。

在适当的时候，可以通过向 tunefs(8)[1086] 传递多重标签来在 UFS 文件系统上设置多标签策略。多重标签策略允许每个主体或客体有自己独立的 MAC 标签。使用多重标签或单一标签策略的决定只需要用于实现标签功能的策略，如 `biba`、`lomac` 和 `mls`。一些策略，如 `seeotheruids`、`portacl` 和 `partition` 根本不使用标签。

在一个区间上使用多标签策略并建立一个多标签安全模型会增加管理开销，因为该文件系统中的所有东西（包括目录、文件，甚至是设备节点）都有一个标签。

下面的命令将在指定的 UFS 文件系统上设置多重标签。这只能在单用户模式下进行，对交换分区文件系统来说不是一个要求。

```
# tunefs -l enable /
```

注意

[1085] https://www.freebsd.org/cgi/man.cgi?query=mac_lomac&sektion=4&format=html

[1086] https://www.freebsd.org/cgi/man.cgi?query=tunefs&sektion=8&format=html

某些用户在 root 分区上设置多重标签标志时遇到了问题。如果是这种情况，请查看 MAC 框架疑难解答[1087]。

由于多重标签策略是在每个文件系统的基础上设置的，如果文件系统布局设计得好，可能就不需要多重标签策略。思考以下用于 FreeBSD web 服务器的安全 MAC 模型示例：这台机器对默认文件系统中的所有东西都使用单一标签，即 `biba/high`。如果网络服务器需要在 `biba/low` 下运行以防止写入功能，它可以被安装到一个单独的 UFS **/usr/local** 文件系统，设置为 `biba/low`。

18.3.1.标签配置

实际上，标签策略模块配置的所有方面都将使用基本的系统实用程序进行。这些命令为客体或主体的配置或对配置的操作和验证提供一个简单的界面。

所有的配置都可以使用 `setfmac` 和 `setpmac` 来完成，前者用于设置系统客体的 MAC 标签，后者用于设置系统主体的标签。例如，在 **test** 中把 biba 的 MAC 标签设置为 `high`。

```
# setfmac biba/high test
```

如果配置成功，将无报错地返回到命令行。一个常见的错误是 `Permission denied`，这通常发生在对一个受限制的客体设置或修改标签时。其他情况可能产生不同的问题。例如，文件可能不属于试图重新标记该客体的用户，该客体可能不存在，或者该客体可能是只读的。强制性的策略可能会因为文件属性，进程属性，或建议的新标签值属性而不允许进程重新标记文件。例如，如果一个以低完整性运行的用户试图改变一个高完整性文件的标签，或者一个以低完整性运行的用户试图将一个低完整性文件的标签改为高完整性标签，这些操作将失败。

系统管理员可以使用 `setpmac` 来覆盖策略模块的设置，给被调用的进程分配一个不同的标签：

```
# setfmac biba/high test
Permission denied
# setpmac biba/low setfmac biba/high test
# getfmac test
test: biba/high
```

对于当前正在运行的进程，如 `sendmail`，通常使用 `getpmac` 来代替。这个命令用一个进程 ID（PID）来代替命令名称。如果用户试图操作一个不在其访问范围内的文件，根据加载的策略模块的规则，将显示不允许操作的错误。

[1087] https://docs.freebsd.org/en/books/handbook/mac/#mac-troubleshoot

18.3.2.预定义标签

一些支持标签功能的 FreeBSD 策略模块提供了三个预定义的标签：`low`、`equal` 和 `high`，其中：

- `low` 被视为客体或主体可能具有的最低标签设置。在客体或主体上设置此项会阻止它们访问标记为“高”的客体或主体。
- `equal` 将主题或客体设置为禁用或不受影响，并且只应放置在被视为从策略中免除的客体上。
- `high` 为客体或主体授予 Biba 和 MLS 策略模块中可用的最高设置。

这些策略模块包括 mac_biba(4)[1088]、mac_mls（4）[1089] 和 mac_lomac(4)[1090]。每个预定义的标签都建立不同的信息流指令。请参阅模块的手册页，以确定通用标签配置的特征。

18.3.3.数字标签

Biba 和 MLS 策略模块支持数字标签，可以设置该标签以指示分层控制的精确级别。此数字级别用于将信息分区或分类到不同的分类组中，仅允许访问该组或更高级别的组。例如：

```
biba/10:2+3+6(5:2+3-20:2+3+4+5+6)
```

可能被解释为“Biba Policy Label/Grade 10:Compartments 2, 3 and 6: (grade 5 …)”

在此示例中，第一级将被视为具有有效区间的有效等级，第二级是低等级，最后一级是高等级。在大多数配置中，不需要这种细粒度设置，因为它们被认为是高级配置。

系统客体只有当前等级和区间。系统主体反映系统中可用权限的范围，以及用于访问控制的网络接口。

主体和客体对中的等级和区间被用来构建一种被称为 *支配*的关系，在这种关系中，主体支配客体，客体支配主体，两者都不支配对方，或者两者都支配对方。当两个标签相同时，就会出现“都支配”的情况。由于 Biba 的信息流性质，用户对一组可能对应于项目的区间有权利，但客体也有一组区间。用户可能不得不使用 `su` 或 `setpmac` 对他们的权力进行分组，以便访问他们不受限制的区间中的客体。

18.3.4.用户标签

用户需具有标签，以便其文件和进程与系统上定义的安全策略正确交互。这是在 **/etc/login.conf** 中使用登录分级配置的。每个使用标签的策略模块都将实现用户类设置。

要设置由 MAC 强制执行的用户类默认标签，请添加一个条目。下面显示包含每个策略模块的示例条目。请注意，在实际配置中，管理员永远不会启用每个策略模块。建议在实现任何配置之前查看本章的其余部分。

[1088] https://www.freebsd.org/cgi/man.cgi?query=mac_biba&sektion=4&format=html
[1089] https://www.freebsd.org/cgi/man.cgi?query=mac_mls&sektion=4&format=html
[1090] https://www.freebsd.org/cgi/man.cgi?query=mac_lomac&sektion=4&format=html

```
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]:
```

虽然用户不能修改默认值，不过他们可以在登录后改变他们的标签，但也要遵守策略的约束。上面的例子告诉 Biba 执行策略，一个进程的最小完整性是 `5`，最大是 `15`，而默认的有效标签是 `10`。该进程将以 `10` 运行，直到它选择改变标签，也许是由于用户使用 `setpmac`，这将被 Biba 限制在配置的范围内。

对 **login.conf** 进行任何更改后，必须使用 `cap_mkdb` 重新构建登录类功能数据库。

许多网站都有大量用户需要多个不同的用户分级。需要进行深入的规划，因为这可能变得难以管理。

18.3.5.网络接口标签

可以在网络接口上设置标签，以帮助控制网络上的数据流。使用网络接口标签的策略的工作方式与策略相对于客体的功能相同。例如，在 Biba 中设置较高的用户将不允许访问带有标签为 `low` 的网络接口。

设置网口的 MAC 标签时，`maclabel` 可以传递给 `ifconfig`:

```
# ifconfig bge0 maclabel biba/equal
```

这个例子将在 `bge0` 接口上设置 `biba/equal` 的 MAC 标签。当使用类似 于`biba/high(low-high)` 的设置时，应该引用整个标签以防止返回错误。

每一个支持标签的策略模块都有一个可调整的功能，可以用来禁用网络接口上的 MAC 标签。将标签设

置为等值会有类似的效果。请查看 `sysctl` 的输出，策略的手册页面，以及本章其余部分的信息，以了解关于这些可调选项的更多信息。

18.4.规划安全配置

在实施任何 MAC 策略之前，建议实行一个规划阶段。在规划阶段，管理员应考虑实现要求和目标，例如：

- 如何对目标系统上可用的信息和资源进行分类。
- 要限制访问的信息或资源以及应用的限制类型。
- 需要哪些 MAC 模块来实现此目标。

在生产系统上使用 MAC 实现 之前，应该对可信系统及其配置进行试运行。由于不同的环境有不同的需求和要求，因此在系统上线后，安全配置文件的建立和完善将会降低修改的必要性。

请考虑 MAC 框架如何增强整个系统的安全性。MAC 框架提供的各种安全策略模块可用于保护网络和文件系统，或阻止用户访问某些端口和套接字。或许策略模块的最佳用途是一次加载多个安全策略模块，以便提供 MLS 环境。此方法与加固策略不同，强化策略通常会加固系统中仅用于特定目的的元素。MLS 的缺点是增加管理开销。

与一个框架的持久效果相比，这种开销微乎其微，该框架提供了挑选特定配置所需政策的能力，并使性能开销降低。减少对不需要的策略的支持可以提高系统的整体性能，并提高选择的灵活性。一个好的实现将考虑总体安全要求，并有效地实现框架提供的各种安全策略模块。

使用 MAC 的系统保证不允许用户随意更改安全属性。所有用户实用程序、程序和脚本都必须在所选安全策略模块提供的访问规则的约束下工作，并且 MAC 访问规则的控制权掌握在系统管理员手中。

仔细选择正确的安全策略模块是系统管理员的职责。对于需要限制网络访问控制的环境，mac_portacl(4)[1091]、mac_ifoff(4)[1092] 和 mac_biba(4)[1093] 策略模块是很好的选择。对于需要文件系统客体严格保密的环境，请考虑 mac_bsdextended(4)[1094] 和 mac_mls(4)[1095] 策略模块。

可以根据网络配置做出策略决策。如果只允许某些用户访问 ssh(1)[1096]，那么 mac_portacl(4)[1097]策略模块是一个不错的选择。对于文件系统，对客体的访问对某些用户可能加密，但对其他用户则不然。例如，大型开发团队可能会拆分为较小的项目，其中可能不允许项目 A 中的开发人员访问项目 B 中开发人员编写的客体。然而，这两个项目都可能需要访问开发人员在项目 C 中创建的客体。使用 MAC 框架提供的不同安全策略模块，可以将用户分为这些组，然后获得对相应客体的访问权限。

每个安全策略模块都有处理系统整体安全性的独特方法。模块的选择应基于深思熟虑的安全策略，这可能需要修订和重新实现。了解 MAC 框架提供的不同安全策略模块将有助于管理员选择适合其情况的最佳策

[1091] https://www.freebsd.org/cgi/man.cgi?query=mac_portacl&sektion=4&format=html
[1092] https://www.freebsd.org/cgi/man.cgi?query=mac_ifoff&sektion=4&format=html
[1093] https://www.freebsd.org/cgi/man.cgi?query=mac_biba&sektion=4&format=html
[1094] https://www.freebsd.org/cgi/man.cgi?query=mac_bsdextended&sektion=4&format=html
[1095] https://www.freebsd.org/cgi/man.cgi?query=mac_mls&sektion=4&format=html
[1096] https://www.freebsd.org/cgi/man.cgi?query=ssh&sektion=1&format=html
[1097] https://www.freebsd.org/cgi/man.cgi?query=mac_portacl&sektion=4&format=html

略。

本章的其余部分介绍可用的模块，说明了它们的使用和配置，并在某些情况下提供有关适用情况的见解。

> 当心
>
> MAC 的实现与防火墙的实现非常相似，都必须注意防止完全被锁定在系统之外。应考虑还原到原先配置的能力，并且在通过远程连接实现 MAC 时应尤为慎重。

18.5.可用的 MAC 策略

FreeBSD 默认的内核选项包括 MAC。这意味着在 MAC 框架中的每个模块都可以用 `kldload` 作为运行时内核被加载。在测试完模块后，将模块名称添加到 **/boot/loader.conf** 中，这样它就会在启动时加载。每个模块都为那些选择自己编译内核的管理员提供了一个内核选项。

FreeBSD 包含一组策略，这些策略可满足大多数安全要求。每项策略总结如下。最后三个策略支持整数设置来代替三个默认标签。

18.5.1.MAC 查看其他 UID 策略

模块名称：**mac_seeotheruids.ko**

内核配置选项：`options MAC_SEEOTHERUIDS`

引导选项：`mac_seeotheruids_load="YES"`

mac_seeotheruids(4)[1098] 模块扩展了 `security.bsd.see_other_uids` 和 `security.bsd.see_other_gids` `sysctl` 可调控项。这个选项不需要在配置前设置任何标签，并且可以与其他模块透明地运行。

加载模块后，可以使用以下 `sysctl` 可调参数来控制其功能：

- `security.mac.seeotheruids.enabled` 启用该模块并实现默认设置，这些设置拒绝用户查看其他用户拥有的进程和套接字。
- `security.mac.seeotheruids.specificgid_enabled` 允许指定的组豁免于此策略。要豁免特定的组，使用 `security.mac.seeotheruids.specificgid=XXX sysctl tunable`，用要豁免的数字组 ID 代替 *XXX*。
- `security.mac.seeotheruids.primarygroup_enabled` 用于将特定的主群组从该策略中豁免。当使用这个调整时，`security.mac.seeotheruids.specificgid_enabled` 可能不会被设置。

[1098] https://www.freebsd.org/cgi/man.cgi?query=mac_seeotheruids&sektion=4&format=html

18.5.2.MAC BSD 扩展策略

模块名称：**mac_bsdextended.ko**

内核配置选项：`options MAC_BSDEXTENDED`

引导选项：`mac_bsdextended_load="YES"`

mac_bsextended(4)[1099] 模块强制执行文件系统防火墙。它提供标准文件系统权限模型的扩展，允许管理员创建类似防火墙的规则集来保护文件系统层次结构中的文件、实用程序和目录。当尝试访问文件系统对象时，将迭代规则列表，直到找到匹配的规则或到达结束。此行为可以使用 `security.mac.bsdextended.firstmatch_enabled` 来更改。类似于 FreeBSD 中的其他防火墙模块，可以被系统在启动时使用 rc.conf(5)[1100] 变量创建和读取包含访问控制规则的文件。

可以使用 ugidfw(8)[1101] 输入规则列表，其语法类似于 ipfw(8)[1102]。可以使用 libugidfw(3)[1103] 库中的函数编写更多工具。

加载 mac_bsdextended(4)[1104] 模块后，可以使用以下命令列出当前规则配置：

```
# ugidfw list
0 slots, 0 rules
```

默认情况下，没有定义规则，所有内容都是完全可访问的。创建一个禁止用户访问但不影响 `root` 用户访问的规则：

```
# ugidfw add subject not uid root new object not uid root mode n
```

虽然这个规则实现起来很简单，但却是一个非常糟糕的主意，因为它阻止所有用户发布任何命令。一个更现实的例子是阻止 `user1` 对 `user2` 的主目录的所有访问，包括目录列表：

```
# ugidfw set 2 subject uid user1 object uid user2 mode n
# ugidfw set 3 subject uid user1 object gid user2 mode n
```

为了对所有用户执行相同的访问限制，`not uid user2` 可以用来代替 `user1`。然而，`root` 用户不受这些规则的影响。

> 注意
> 使用此模块时应格外小心，因为错误使用可能会阻止对文件系统某些部分的访问。

[1099] https://www.freebsd.org/cgi/man.cgi?query=mac_bsdextended&sektion=4&format=html
[1100] https://www.freebsd.org/cgi/man.cgi?query=rc.conf&sektion=5&format=html
[1101] https://www.freebsd.org/cgi/man.cgi?query=ugidfw&sektion=8&format=html
[1102] https://www.freebsd.org/cgi/man.cgi?query=ipfw&sektion=8&format=html
[1103] https://www.freebsd.org/cgi/man.cgi?query=libugidfw&sektion=3&format=html
[1104] https://www.freebsd.org/cgi/man.cgi?query=mac_bsdextended&sektion=4&format=html

18.5.3.MAC 接口静默策略

模块名称：**mac_ifoff.ko**

内核配置选项：`options MAC_IFOFF`

引导选项：`mac_ifoff_load="YES"`

mac_ifoff(4)[1105] 模块用于动态禁用网络接口，并防止在系统引导期间启动网络接口。它不使用标签，也不依赖于任何其他 MAC 模块。

此模块的大部分控制都是通过 `sysctl` 的以下可调参数执行的：

- `security.mac.ifoff.lo_enabled` 启用或禁用环回 lo(4)[1106] 接口上的所有流量。
- `security.mac.ifoff.bpfrecv_enabled` 启用或禁用伯克利数据包过滤器接口 bpf(4)[1107] 上的所有流量。
- `security.mac.ifoff.other_enabled` 启用或禁用所有其他接口上的流量。

mac_ifoff(4)[1108] 最常见的用途之一是在引导过程中不应允许网络流量的环境中进行网络监控。另一个用途是编写一个脚本，该脚本使用诸如 security/aide[1109] 之类的应用程序，以便在受保护的目录中找到新的或更改的文件时自动阻止网络流量。

18.5.4.MAC 端口访问控制列表策略

模块名称：**mac_portacl.ko**

内核配置选项：`MAC_PORTACL`

引导选项：`mac_portacl_load="YES"`

mac_portacl(4)[1110] 模块用于限制绑定到本地 TCP 和 UDP 端口，从而允许非 `root` 用户绑定到低于 1024 的指定特权端口。

加载后，此模块将在所有套接字上启用 MAC 策略。以下可调功能可用：

- `security.mac.portacl.enabled` 完全启用或禁用策略。
- `security.mac.portacl.port_high` 设置 mac_portacl(4)[1111] 保护的最高端口号。
- `security.mac.portacl.suser_exempt`，如果设置为非零值，则使 `root` 用户免受此策略的约束。
- `security.mac.portacl.rules` 以 `rule[,rule,…]` 形式的文本字符串来指定策略，规则数量不限，每个规则的形式为 `idtype:id:protocol:port`。`idtype` 是 `uid` 或 `gid`。`protocol` 参数可

[1105] https://www.freebsd.org/cgi/man.cgi?query=mac_ifoff&sektion=4&format=html
[1106] https://www.freebsd.org/cgi/man.cgi?query=lo&sektion=4&format=html
[1107] https://www.freebsd.org/cgi/man.cgi?query=bpf&sektion=4&format=html
[1108] https://www.freebsd.org/cgi/man.cgi?query=mac_ifoff&sektion=4&format=html
[1109] https://cgit.freebsd.org/ports/tree/security/aide/pkg-descr
[1110] https://www.freebsd.org/cgi/man.cgi?query=mac_portacl&sektion=4&format=html
[1111] https://www.freebsd.org/cgi/man.cgi?query=mac_portacl&sektion=4&format=html

以是 tcp 或 udp。端口参数是允许指定的用户或组绑定的端口号。用户 ID、组 ID 和端口参数只能使用数字值。

默认情况下，小于 1024 的端口只能被以 root 运行的特权进程使用。要允许 mac_portacl(4)[1112] 以非 root 特权进程绑定到小于 1024 的端口，请按如下方式设置以下可调参数：

```
# sysctl security.mac.portacl.port_high=1023
# sysctl net.inet.ip.portrange.reservedlow=0
# sysctl net.inet.ip.portrange.reservedhigh=0
```

若要防止 root 用户受此策略的影响，请设置 security.mac.portacl.suser_exempt 为非零值：

```
# sysctl security.mac.portacl.suser_exempt=1
```

允许 UID 为 80 的 www 用户绑定到 80 端口，而无需 root 权限：

```
# sysctl security.mac.portacl.rules=uid:80:tcp:80
```

下一个示例允许 UID 为 1001 的用户绑定到 TCP 端口 110（POP3）和 995（POP3s）：

```
# sysctl security.mac.portacl.rules=uid:1001:tcp:110,uid:1001:tcp:995
```

18.5.5.MAC 分区策略

模块名称：**mac_partition.ko**

内核配置选项：options MAC_PARTITION

引导选项：mac_partition_load="YES"

mac_partition(4)[1113] 策略根据进程的 MAC 标签将其放入特定的“分区”。这个策略的大部分配置是通过 setpmac(8)[1114] 完成的。这个策略有一个 sysctl 可调参数：

- security.mac.partition.enabled 启用 MAC 进程分区的强制实施。

当启用这个策略时，用户将只能看到他们的进程，以及他们分区内的任何其他进程，将不能使用这个分区范围以外的实用程序。例如，一个 insecure 等级用户将不被允许访问 top 以及许多其他必须生成进程的命令。

这个例子在 insecure 级别的用户上增加 top 的标签设置。所有由不安全类中的用户产生的进程将停留在 partition/13 标签中。

[1112] https://www.freebsd.org/cgi/man.cgi?query=mac_portacl&sektion=4&format=html
[1113] https://www.freebsd.org/cgi/man.cgi?query=mac_partition&sektion=4&format=html
[1114] https://www.freebsd.org/cgi/man.cgi?query=setpmac&sektion=8&format=html

```
# setpmac partition/13 top
```

此命令显示分区标签和进程列表：

```
# ps Zax
```

此命令显示其他用户的进程分区标签以及该用户当前正在运行的进程：

```
# ps -ZU trhodes
```

> 注意
> 用户可以在 root 的标签中看到进程，除非加载 mac_seeotheruids(4)[1115] 策略。

18.5.6.MAC 多级安全模块

模块名称：**mac_mls.ko**

内核配置选项：`options MAC_MLS`

引导选项：`mac_mls_load="YES"`

mac_mls(4)[1116] 策略通过强制实施严格的信息流策略来控制系统中主体和对象之间的访问。

在 MLS 环境中，每个主体或物体的标签中都设置一个“清除”级别，同时还有隔间。由于这些清除级别可以达到超过几千的数字，对所有主体或客体都进行完全配置将是一项艰巨的任务。为了减轻这种管理上的开销，本政策中包括三个标签：`mls/low`、`mls/equal`、`mls/high`，其中:

- `mls/low` 任何标有的都具有较低的许可级别，并且不允许访问更高级别的信息。此标签还可以防止间隙级别较高的对象写入或将信息传递到较低级别。
- `mls/equal` 应放置在应从策略中免除的对象上。
- `mls/high` 是最高的许可级别。分配这个标签的对象将控制系统中所有其他对象;但是，不会允许向下层对象泄露信息。

MLS 提供：

- 具有一组非分层类别的分层安全级别。
- 固定的规则是 `no read up, no write down`。这意味着，一个主体可以对它自己那一层或下面的对象有阅读权限，但不能对上面的对象有阅读权限。同样地，一个主体可以对自己这一层或以上的对象有写入权限，但不能对下面的对象有写入权限。
- 保密，防止不当的数据披露。
- 设计系统的基础，该系统以多个敏感度级别同时处理数据，而不会在机密和机密之间泄漏信息。

[1115] https://www.freebsd.org/cgi/man.cgi?query=mac_seeotheruids&sektion=4&format=html
[1116] https://www.freebsd.org/cgi/man.cgi?query=mac_mls&sektion=4&format=html

sysctl 可调参数如下:

- security.mac.mls.enabled 用于启用或禁用 MLS 策略。
- security.mac.mls.ptys_equal 在创建时将所有 pty(4)[1117] 设备标记为 mls/equal。
- security.mac.mls.revocation_enabled 在对象的标签更改为较低等级的标签后，撤消对对象的访问权限。
- security.mac.mls.max_compartments 设置系统上允许的最大隔间级别数。

要操作 MLS 标签，请使用 setfmac(8)[1118]。为对象指定标签:

```
# setfmac mls/5 test
```

获取文件 **test** 的 MLS 标签:

```
# getfmac test
```

另一种方法是在 **/etc/** 中创建一个主策略文件，该文件指定 MLS 策略信息，并将该文件反馈到 setfmac。

使用 MLS 策略模块时，管理员计划控制敏感信息的流。block read up block write down 默认值将所有内容设置为低状态。一切都可以访问，管理员慢慢地增强信息的机密性。

除了三个基本标签选项之外，管理员还可以根据需要对用户和组进行分组，以阻止它们之间的信息流。使用描述性词（如 Confidential、Secret 和 Top Secret 的分类）查看清关级别中的信息可能会更容易。某些管理员会根据项目级别创建不同的组。无论分类方法如何，在实施限制性策略之前，都必须有一个经过深思熟虑的计划。

MLS 策略模块的一些示例情况包括电子商务 Web 服务器、保存关键公司信息的文件服务器和金融机构环境。

18.5.7.MAC Biba 模块

模块名称：**mac_biba.ko**

内核配置选项：options MAC_BIBA

引导选项：mac_biba_load="YES"

mac_biba(4)[1119] 模块加载 MAC Biba 策略。此策略类似于 MLS 策略，但信息流规则略有不同。这是为了防止敏感信息的向下流动，而 MLS 策略则防止敏感信息的向上流动。

在 Biba 环境中，每个主题或对象都设置“完整性”标签。这些标签由分层等级和非分层组件组成。随着等级的上升，其完整性也会上升。

支持的标签是 biba/low、biba/equal 和 biba/high，其中:

[1117] https://www.freebsd.org/cgi/man.cgi?query=pty&sektion=4&format=html
[1118] https://www.freebsd.org/cgi/man.cgi?query=setfmac&sektion=8&format=html
[1119] https://www.freebsd.org/cgi/man.cgi?query=mac_biba&sektion=4&format=html

- `biba/low` 被视为对象或主体可能具有的最低完整性。在对象或主题上设置此设置将阻止它们对标记为 `biba/high` 的对象或主题的写访问，但不会阻止读访问。
- `biba/equal` 应仅放置在被视为从策略中免除的对象上。
- `biba/high` 允许写入设置在较低标签处的对象，但不允许读取该对象。建议将此标签放在影响整个系统完整性的对象上。

Biba 提供：

- 具有一组非分层完整性类别的分层完整性级别。
- 固定规则是 `no write up, no read down`，与 MLS 相反。主体可以对其自身级别或以下级别的对象具有写访问权限，但不能对其更高级别的对象具有写访问权限。类似地，主体可以在自己的级别或更高级别上对对象具有读访问权限，但不能在更低级别上。
- 通过防止对数据进行不当修改来保持完整性。
- 完整性级别，而不是 MLS 敏感度级别。

以下可调参数可用于操作 Biba 策略：

- `security.mac.biba.enabled` 用于在目标计算机上启用或禁用 Biba 策略的实施。
- `security.mac.biba.ptys_equal` 用于在 pty(4)[1120] 设备上禁用 Biba 策略。
- `security.mac.biba.revocation_enabled` 如果标签更改为主体，则强制吊销对客体的访问。

要访问系统对象上的 Biba 策略设置，请使用 `setfmac` 和 `getfmac`：

```
# setfmac biba/low test
# getfmac test
test: biba/low
```

不同于敏感度，完整性用于保证信息不被不受信任的各方操纵。这包括在主体和客体之间传递的信息。它确保用户只能修改或访问他们已被授予显式访问权限的信息。mac_biba(4)[1121] 安全策略模块允许管理员配置用户可以看到和调用哪些文件和程序，同时确保系统为该用户信任这些程序和文件。

在初始规划阶段，管理员必须准备好将用户划分为等级、级别和区域。启用此策略模块后，系统将默认为高标签，并由管理员为用户配置不同的等级和级别。一个好的规划方法可以包括主题，而不是使用许可级别。例如，仅允许开发人员修改源代码存储库、源代码编译器和其他开发实用程序。其他用户将被分组到其他类别，如测试人员、设计人员或最终用户，并且只允许读取访问权限。

完整性较低的主体无法写入完整性较高的主体，并且完整性较高的主体无法列出或读取完整性较低的对象。将标签设置为尽可能低的等级可能会使受试者无法访问。此安全策略模块的一些预期环境将包括一个受约束的 Web 服务器、一个开发和测试计算机以及一个源代码存储库。不太有用的实现是个人工作站，用作路由器的计算机或网络防火墙。

[1120] https://www.freebsd.org/cgi/man.cgi?query=pty&sektion=4&format=html

[1121] https://www.freebsd.org/cgi/man.cgi?query=mac_biba&sektion=4&format=html

18.5.8.MAC 低安全级别模块

模块名称：**mac_lomac.ko**

内核配置选项：`options MAC_LOMAC`

引导选项：`mac_lomac_load="YES"`

与 MAC Biba 策略不同，mac_lomac(4)[1122] 策略仅在降低完整性级别后才允许访问完整性较低的对象，以免破坏任何完整性规则。

低位标志完整性策略的工作方式几乎与 Biba 相同，只是使用浮动标签来支持通过辅助等级隔间降级的受试者。此辅助隔间采用 `[auxgrade]` 的形式。为策略分配辅助等级时，请使用语法 `lomac/10[2]`，其中是辅助等级 `2`。

此策略依赖于所有系统对象的无处不在的完整性标签，允许受试者从低完整性对象读取，然后降级受试者的标签，以防止将来使用 `[auxgrade]` 写入高完整性对象。与 Biba 相比，该策略可能提供更高的兼容性，并且需要更少的初始配置。

与 Biba 和 MLS 策略类似，`setfmac` 和 `setpmac` 用于在系统对象上放置标签：

```
# setfmac /usr/home/trhodes lomac/high[low]
# getfmac /usr/home/trhodes lomac/high[low]
```

辅助等级 `low` 是仅由 MACLOMAC 策略提供的功能。

18.6.用户锁定

此示例为用户数少于 50 的相对较小的存储系统所设计。用户将具有登录功能，并被允许存储数据和访问资源。

对于此方案，mac_bsdextended(4)[1123] 和 mac_seeotheruids(4)[1124] 策略模块可以共存并阻止对系统客体的访问，同时隐藏用户进程。

首先将以下行添加到 **/boot/loader.conf**：

```
mac_seeotheruids_load="YES"
```

mac_bsdextended(4)[1125] 安全策略模块可以通过将此行添加到 **/etc/rc.conf** 来激活：

```
ugidfw_enable="YES"
```

[1122] https://www.freebsd.org/cgi/man.cgi?query=mac_lomac&sektion=4&format=html
[1123] https://www.freebsd.org/cgi/man.cgi?query=mac_bsdextended&sektion=4&format=html
[1124] https://www.freebsd.org/cgi/man.cgi?query=mac_seeotheruids&sektion=4&format=html
[1125] https://www.freebsd.org/cgi/man.cgi?query=mac_bsdextended&sektion=4&format=html

存储在 **/etc/rc.bsdextended** 中的默认规则将在系统初始化时加载。但是，有时可能需要修改默认条目。由于此计算机应仅用于为用户提供服务，因此除了最后两行之外，所有内容都可能被注释掉，以便默认情况下强制加载用户拥有的系统客体。

将所需用户添加到此计算机并重新启动。出于测试目的，请尝试在两个控制台上以其他用户身份登录。运行 `ps aux` 以查看其他用户的进程是否可见。并验证在其他用户的主目录上运行 ls(1)[1126] 是否失败。

不要尝试与 `root` 用户一起测试，除非已修改特定的 `sysctl` 以阻止超级用户访问。

> 注意
>
> 当添加一个新用户后，他们的 mac_bsdextended(4)[1127] 规则将不在规则集列表中。要快速更新规则集，请卸载安全策略模块，然后使用 kldunload(8)[1128] 和 kldload(8)[1129] 再次重新加载模块。

18.7.MAC Jail 中的 Nagios

本节演示了在 MAC 环境中实现 Nagios 网络监控系统所需的步骤。这是一个示例，该示例仍要求管理员在生产环境中使用之前测试已实现的策略是否满足网络的安全要求。

该示例需要在每个文件系统上都设置多重标签。它还假设 net-mgmt/nagios-plugins[1130]、net-mgmt/nagios[1131] 和 www/apache22[1132] 在尝试集成到 MAC 框架之前都已正确安装、配置和工作。

18.7.1.创建不安全的用户分级

通过将以下用户分级添加到 **/etc/login.conf** 来开始此过程：

```
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:\
```

(continues on next page)

[1126] https://www.freebsd.org/cgi/man.cgi?query=ls&sektion=1&format=html
[1127] https://www.freebsd.org/cgi/man.cgi?query=mac_bsdextended&sektion=4&format=html
[1128] https://www.freebsd.org/cgi/man.cgi?query=kldunload&sektion=8&format=html
[1129] https://www.freebsd.org/cgi/man.cgi?query=kldload&sektion=8&format=html
[1130] https://cgit.freebsd.org/ports/tree/net-mgmt/nagios-plugins/pkg-descr
[1131] https://cgit.freebsd.org/ports/tree/net-mgmt/nagios/pkg-descr
[1132] https://cgit.freebsd.org/ports/tree/www/apache22/pkg-descr

(continued from previous page)

```
:memoryuse=8M:\
:filesize=8M:\
:coredumpsize=8M:\
:openfiles=24:\
:maxproc=32:\
:priority=0:\
:requirehome:\
:passwordtime=91d:\
:umask=022:\
:ignoretime@:\
:label=biba/10(10-10):
```

然后，将以下行添加到默认用户分级部分：

```
:label=biba/high:
```

保存编辑内容并执行以下命令以重建数据库：

```
# cap_mkdb /etc/login.conf
```

18.7.2.配置用户

使用以下命令将用户 root 设置为默认分级：

```
# pw usermod root -L default
```

所有非 root 用户帐户现在都需要一个登录分级。登录分级是必需的，否则将拒绝用户访问常用命令。以下 sh 脚本应该可以解决问题：

```
# for x in `awk -F: '($3 >= 1001) && ($3 != 65534) { print $1 }' \
    /etc/passwd`; do pw usermod $x -L default; done;
```

接下来，将 nagios 和 www 帐户放入不安全分级中：

```
# pw usermod nagios -L insecure
# pw usermod www -L insecure
```

18.7.3.创建上下文文件

现在应创建上下文文件 **/etc/policy.contexts**：

```
# This is the default BIBA policy for this system.

# System:
/var/run(/.*)?          biba/equal

/dev/(/.*)?         biba/equal

/var                biba/equal
/var/spool(/.*)?        biba/equal

/var/log(/.*)?          biba/equal

/tmp(/.*)?          biba/equal
/var/tmp(/.*)?          biba/equal

/var/spool/mqueue       biba/equal
/var/spool/clientmqueue     biba/equal

# For Nagios:
/usr/local/etc/nagios(/.*)? biba/10

/var/spool/nagios(/.*)?     biba/10

# For apache
/usr/local/etc/apache(/.*)? biba/10
```

此策略通过设置信息流限制来强制实施安全性。在此特定配置中，绝不应允许用户（包括 root）访问 Nagios。作为 Nagios 一部分的配置文件和进程将完全独立或被隔离。

在每个文件系统上运行 setfsmac 后，将读取此文件。此示例在根文件系统上设置策略：

```
# setfsmac -ef /etc/policy.contexts /
```

接下来，将这些编辑添加到 **/etc/mac.conf** 的主要部分：

```
default_labels file ?biba
default_labels ifnet ?biba
default_labels process ?biba
default_labels socket ?biba
```

18.7.4.加载程序配置

要完成配置，请将以下行添加到 **/boot/loader.conf**：

```
mac_biba_load="YES"
mac_seeotheruids_load="YES"
security.mac.biba.trust_all_interfaces=1
```

并将以下行中的网卡配置存储在 **/etc/rc.conf** 中。如果主网络配置是通过 DHCP 完成的，则可能需要在每次系统引导后手动配置：

```
maclabel biba/equal
```

18.7.5.测试配置

首先，确保在系统初始化和重新启动时不会启动 Web 服务器和 Nagios。确保 `root` 无法访问 Nagios 配置目录中的任何文件。如果 `root` 可以列出 **/var/spool/nagios** 的内容，则存在问题。若无问题，应返回错误“permission denied”。

如果一切正常，现在可以启动 Nagios，Apache 和 Sendmail：

```
# 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
```

仔细检查以确保一切正常。如果有问题，请检查日志文件中的错误消息。如果需要，使用 sysctl(8)[1133] 禁用 mac_biba(4)[1134] 安全策略模块，然后像往常一样尝试重新启动所有内容。

> 注意 `root` 用户仍可以更改安全实施并编辑其配置文件。对于新生成的 shell，以下命令将允许将安全策略降级到较低的级别：
>
> ```
> # setpmac biba/10 csh
> ```
>
> 要阻止这种情况发生，请使用 login.conf(5)[1135] 强制用户进入某个范围。如果 setpmac(8)[1136] 尝试在分区范围之外运行命令，将返回错误，并且不会执行该命令。在这种情况下，请将 root 设置为 `biba/high(high-high)`。

[1133] https://www.freebsd.org/cgi/man.cgi?query=sysctl&sektion=8&format=html
[1134] https://www.freebsd.org/cgi/man.cgi?query=mac_biba&sektion=4&format=html
[1135] https://www.freebsd.org/cgi/man.cgi?query=login.conf&sektion=5&format=html
[1136] https://www.freebsd.org/cgi/man.cgi?query=setpmac&sektion=8&format=html

18.8.MAC 框架的故障排除

本节讨论常见的配置错误以及如何解决这些错误。

- **多重分区标志不在 root（/）分区上保持启用状态**

  以下步骤可以解决此临时性错误：

1. 编辑 **/etc/fstab** 并将根分区设置为 `ro`，即只读。
2. 重新启动到单用户模式。
3. 在 **/** 上运行 `tunefs -l enable`。
4. 重新启动系统。
5. 运行 `mount -urw /` 命令，将 **/etc/fstab** 中的 `ro` 修改回 `rw`，然后重新启动系统。
6. 仔细检查 `mount` 的输出，以确保已在根文件系统上正确设置多重分区。

- **使用 MAC 建立安全环境后，无法启动 Xorg**

  这可能是由 MAC 策略或其中一个 MAC `partition` 标记策略中的错误标记引起的。要进行调试，请尝试以下操作：

1. 检查错误消息。如果用户在 `insecure` 类中，则 `partition` 策略可能是罪魁祸首。尝试将用户 `default` 类设置回该分级，并使用 `cap_mkdb` 重建数据库。如果这不能解决问题，请转到步骤 2。
2. 仔细检查是否为用户、Xorg 和 **/dev** 条目正确设置标签策略。
3. 如果这些都不能解决问题，请将错误消息和环境简介发送到 FreeBSD 通用问题邮件列表[1137]。

- **出现错误 ``secure_path: unable to stat .login_conf``**

  当用户尝试在系统中从该用户切换到 `root` 用户时，可能会出现此错误。当用户的标签设置高于他们尝试成为的用户的标签设置时，通常会发生此消息。例如，假如`joe` 的默认标签为 `biba/low` 且 `root` 标签为 `biba/high`，则 `root` 无法查看 `joe` 的主目录。无论 `roo` 是否使用 `su` 成为 `joe`，这都会发生，因为 Biba 的完整性模型不允许 `root` 查看设置在较低完整性级别的对象。

- **系统不识别 ``root``**

  发生这种情况时，`whoami` 返回 `0`，`su` 返回 `who are you?`。

  如果标签策略已被 sysctl(8)[1138] 禁用或策略模块已卸载，则可能会发生这种情况。如果禁用该策略，则需要重新配置登录功能数据库。仔细检查 **/etc/login.conf** 确保已删除所有的 `label` 选项，并使用 `cap_mkdb` 重建数据库。

  如果策略限制对 **master.passwd** 的访问，也可能发生这种情况。这通常是由于管理员更改与系统使用的常规策略冲突的标签下的文件。在这些情况下，系统将读取用户信息，并且由于文件继承新标签，因此将阻止访问。使用 sysctl(8)[1139] 禁用策略，一切都应该恢复正常。

[1137] https://lists.freebsd.org/subscription/freebsd-questions

[1138] https://www.freebsd.org/cgi/man.cgi?query=sysctl&sektion=8&format=html

[1139] https://www.freebsd.org/cgi/man.cgi?query=sysctl&sektion=8&format=html

第十九章安全事件审计

19.1.概述

FreeBSD 操作系统包含对安全事件审计的支持。事件审计支持对各种与安全相关的系统事件（包括登录、配置变化、文件和网络访问）进行可靠的、精细的、可配置的日志记录。这些日志记录对于实时系统监控、入侵检测和事后分析都是非常宝贵的。FreeBSD 实现了 Sun™ 发布的基本安全模块（BSM）应用程序接口（API）和文件格式，并与 Solaris™ 和 Mac OS® X 的审计实现兼容。

本章重点介绍了事件审计的安装和配置。介绍了审计策略并提供一个审计配置的例子。

读完本章后，你会知道：

- 什么是事件审计，它是如何工作的。
- 如何在 FreeBSD 上为用户和进程配置事件审计。
- 如何使用审计还原和复核工具来复核审计跟踪。

在阅读本章之前，你应该：

- 了解 UNIX® 和 FreeBSD 的基础知识（FreeBSD 基础[1140]）。
- 熟悉内核配置/编译的基础知识（配置 FreeBSD 内核[1141]）。
- 对安全及其与 FreeBSD 的关系有一定的了解（安全[1142]）。

警告

审计机制有一些已知限制。并非所有与安全有关的系统事件都可以被审计，一些登录机制，如基于 Xorg 的显示管理器和第三方守护程序，并没有正确的配置用户登录会话的审计。

安全事件审计工具能够生成非常详细的系统活动日志。在一个繁忙的系统中，当配置为高度详细时，跟踪文件数据可能非常大，在某些配置中每周会超过几 GB。管理员应该考虑到与高频

[1140] https://docs.freebsd.org/en/books/handbook/basics/index.html#basics
[1141] https://docs.freebsd.org/en/books/handbook/kernelconfig/index.html#kernelconfig
[1142] https://docs.freebsd.org/en/books/handbook/security/index.html#security

审计配置有关的磁盘空间要求。例如，可能需要为 **/var/audit** 专设一个文件系统，这样，即使审计文件系统装满，也不会影响到其他文件系统。

19.2.关键术语

以下是有关安全事件审计的术语：

- 事件（*event*）：可审计事件是任何可以使用审计子系统来记录的事件。安全相关事件包括创建一个文件、建立一个网络连接或一个用户登录。事件要么是“归属”，意味着它们可以追踪到一个经过验证的用户，要么是“无归属”。无归属事件是在登录认证之前发生的任何事件，如错误的密码尝试。
- 类（*class*）：一组命名的在选择表达式中使用的相关事件。常用的事件类别包括“文件创建”（fc）、“执行”（ex）和“登录_退出”（lo）。
- 记录（*record*）：描述一个安全事件的审计日志条目。记录包含记录事件类型、执行行动的主体（用户）的信息、日期和时间信息、任何对象或参数的信息以及成功或失败状态。
- 跟踪（*trail*）：一个由一系列描述安全事件的审计记录组成的日志文件。跟踪大致是按照事件完成的时间顺序排列的。只有授权的进程才允许向审计跟踪提交记录。
- 选择表达式（*selection expression*）：包含用于匹配事件的一系列前缀和审计事件类别名称的字符串。
- 预选（*preselection*）：为系统识别哪些事件是管理员感兴趣的过程。预选配置使用一系列选择表达式来确定哪些用户的哪些事件类别需要审计，以及适用于已认证和未认证进程的全局设置。
- 精选（*reduction*）：从现有的审计跟踪中选择进行保存、打印或分析的记录的过程。同样，从审计跟踪中删除不需要的审计记录的过程也是如此。利用精选，管理员可以实施审计数据的保存策略。例如，详细的审计跟踪可能会被保留一个月，但此之后，跟踪可能会被减少，以便于存档而只保留登录信息。

19.3.审计配置

对事件审计的用户空间支持作为 FreeBSD 基本系统的一部分被预置。**GENERIC** 内核默认提供了内核的支持，并且可以通过在 **/etc/rc.conf** 中添加以下行来启用 auditd(8)[1143]：

```
auditd_enable="YES"
```

然后，启动审计守护进程：

```
# service auditd start
```

喜欢编译定制内核的用户必须在其定制内核配置文件中包含以下行：

[1143] https://www.freebsd.org/cgi/man.cgi?query=auditd&sektion=8&format=html

```
options AUDIT
```

19.3.1.事件选择表达式

审计配置中的许多地方都使用到了选择表达式，以确定应该审计哪些事件。表达式包含要匹配的事件类列表。选择表达式从左到右计算，两个表达式通过将一个附加到另一个来组合。

默认审计事件类[1144] 总结了默认审计事件类：

表 1. 默认审计事件类

类型名称	说明	操作
all	全部	匹配所有事件类。
aa	认证和授权	
ad	管理	对整个系统执行的管理操作。
ap	应用	应用程序定义的操作。
cl	文件关闭	审计对 `close` 系统调用的调用。
ex	执行	审计程序的执行。审计命令行参数和环境变量是通过使用 audit_control(5)[1145] 中 `policy` 的 `argv` 和 `envv` 参数来控制的。
fa	文件属性访问	审计对象属性的访问，如 stat(1)[1146] 和 pathconf(2)[1147]。
fc	文件创建	审计创建文件的事件。
fd	文件删除	审计文件删除的事件。
fm	文件属性修改	审计文件属性被修改的事件，例如 chown(8)[1148]、chflags(1)[1149] 和 flock(2)[1150]。
fr	文件读取	对读取数据或打开文件进行读取的审计事件。
fw	文件写入	对写入数据、写入文件或修改文件的事件进行审计。
io	ioctl	审计 `ioctl` 系统调用的使用。
ip	ipc	审计各种形式的进程间通信，包括 POSIX 管道和 System V IPC 操作。
lo	登录_注销	审计 login(1)[1151] 和 logout(1)[1152] 事件。
na	无归属	审计无归属的事件。
no	无效的类	不匹配审计事件。
nt	网络	审计与网络操作相关的事件，例如 connect(2)[1153] 和 accept(2)[1154]。
ot	其他	审计杂项事件。
pc	过程	审计进程操作，例如 exec(3)[1155] 和 exit(3)[1156]。

[1144] https://docs.freebsd.org/en/books/handbook/audit/#event-selection

这些审计事件类可以通过修改 `audit_class` 和 `audit_event` 配置文件来定制。

每个审计事件类可以与一个前缀组合，该前缀指示是否匹配成功/失败的操作，以及条目是添加还是删除类和类型的匹配。审计事件类[1157]的前缀总结了可用的前缀：

表 2. 审计事件类的前缀

前缀	操作
•	审计此类中的成功事件。
•	审计此类中的失败事件。
^	审计此类中既不成功也不失败的事件。
^+	不要审计此类中的成功事件。
^-	不要审计此类中的失败事件。

如果不存在前缀，则将审计事件的成功和失败实例。

以下示例选择字符串选择成功和失败的登录/注销事件，但仅选择成功的执行事件：

```
lo,+ex
```

19.3.2.配置文件

在 **/etc/security** 中可以找到以下用于安全事件审计的配置文件：

- **audit_class**： 包含审计类的定义。
- **audit_control**：控制审计子系统的各个方面，例如默认审计类、审计日志卷上保留的最小磁盘空间以及最大审计跟踪大小。
- **audit_event**： 系统审计事件的文本名称和描述以及每个事件所在类的列表。

[1145] https://www.freebsd.org/cgi/man.cgi?query=audit_control&sektion=5&format=html
[1146] https://www.freebsd.org/cgi/man.cgi?query=stat&sektion=1&format=html
[1147] https://www.freebsd.org/cgi/man.cgi?query=pathconf&sektion=2&format=html
[1148] https://www.freebsd.org/cgi/man.cgi?query=chown&sektion=8&format=html
[1149] https://www.freebsd.org/cgi/man.cgi?query=chown&sektion=8&format=html
[1150] https://www.freebsd.org/cgi/man.cgi?query=flock&sektion=2&format=html
[1151] https://www.freebsd.org/cgi/man.cgi?query=flock&sektion=2&format=html
[1152] https://www.freebsd.org/cgi/man.cgi?query=logout&sektion=1&format=html
[1153] https://www.freebsd.org/cgi/man.cgi?query=connect&sektion=2&format=html
[1154] https://www.freebsd.org/cgi/man.cgi?query=accept&sektion=2&format=html
[1155] https://www.freebsd.org/cgi/man.cgi?query=exec&sektion=3&format=html
[1156] https://www.freebsd.org/cgi/man.cgi?query=exit&sektion=3&format=html
[1157] https://docs.freebsd.org/en/books/handbook/audit/#event-prefixes

- **audit_user**： 用户特定的审计要求与登录时的全局默认值相结合。
- **audit_warn**： auditd(8)[1158] 使用的可定制的 shell 脚本，用于在异常情况下生成警告消息，例如当审计记录的空间不足或审计跟踪文件已轮替时。

  警告
  应仔细编辑和维护审计配置文件，因为错误配置可能导致不正确的事件记录。

在大多数情况下，管理员只需修改 **audit_control** 和 **audit_user**。第一个文件控制系统范围的审计属性和策略，第二个文件可用于用户的审计微调。

19.3.2.1. audit_control 文件

在 **audit_control** 中指定了一些审计子系统的默认值：

```
dist:off
flags:lo,aa
minfree:5
naflags:lo,aa
policy:cnt,argv
filesz:2M
expire-after:10M
```

`dir` 条目用于设置将存储审计日志的一个或多个目录。如果出现多个目录条目，它们将在存储时按顺序使用。通常配置审计，以便审计日志存储在专用文件系统上，以防止在文件系统用完时引起审计子系统与其他子系统之间的干扰。

如果将 `dist` 字段设置为 `on` 或 `yes`，将创建指向 **/var/audit/dist** 中所有跟踪文件的硬链接。

`flags` 字段为归属事件设置系统范围的默认预选掩码。在上面的示例中，对所有用户的成功和失败登录/注销事件以及身份验证和授权进行审计。

`minfree` 条目定义了存储审计跟踪的文件系统的最小可用空间百分比。

`naflags` 条目指定了要针对无归属事件进行审计的审计类，例如登录/注销过程以及身份验证和授权。

`policy` 条目指定了一个以逗号分隔的策略标志列表，用于控制审计行为的各个方面。`cnt` 表示尽管审计失败，系统仍应继续运行（强烈建议使用此标志）。另一个标志，`argv` 会让 execve(2)[1159] 系统调用的命令行参数作为命令执行的一部分被审计。

`filesz` 条目指定了在自动终止和轮替跟踪文件之前审计跟踪的最大大小。0 值禁用自动日志轮替。如果请求的文件大小低于 512k 的最小值，它将被忽略并生成一条日志消息。

`expire-after` 字段指定审计日志文件何时到期并被删除。

[1158] https://www.freebsd.org/cgi/man.cgi?query=auditd&sektion=8&format=html
[1159] https://www.freebsd.org/cgi/man.cgi?query=execve&sektion=2&format=html

19.3.2.2. audit_user 文件

管理员可以在 **audit_user** 中为特定用户进一步指定审计要求。每行通过两个字段为用户配置审计：`alwaysaudit` 字段指定应始终为用户审计的 `neveraudit` 一组事件，该字段指定一组不应为用户审计的事件。

以下示例条目审计登录/注销事件和成功的命令执行——`root` 以及文件创建和成功执行的 `www` 命令。如果与默认的 **audit_control** 一起使用，则 `lo` 条目 `root` 是多余的，并且对 `www` 的登录/注销事件也将被审计。

```
root:lo,+ex:no
www:fc,+ex:no
```

19.4.使用审计跟踪

由于审计跟踪以 BSM 二进制格式存储，因此可以使用几个内置工具来修改这些跟踪文件或将其转换为文本。要将跟踪文件转换为简单的文本格式，请使用 `praudit`。要精选用于分析、归档或打印目的的审计跟踪文件，请使用 `auditreduce`。此工具支持多种选择参数，包括事件类型、事件类别、用户、事件的日期或时间，以及作用于的文件路径或对象。

例如，以纯文本形式转储指定审计日志的全部内容：

```
# praudit /var/audit/AUDITFILE
```

其中 *AUDITFILE* 是要转储的审计日志。

审计跟踪由一系列由令牌组成的审计记录组成，`praudit` 按顺序打印令牌，每行打印一个。每个令牌都属于特定类型，例如 `header`（审计记录标题）或 `path`（来自名称查找的文件路径）。以下是一个 `execve` 事件的示例：

```
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
```

此审计代表一个成功的 `execve` 调用，其中命令 `finger doug` 已执行。`exec arg` 令牌包含由 shell 提供给内核的已处理命令行。`path` 令牌包含内核查找的可执行文件的路径。`attribute` 令牌说明了二进制文件并包括文件模式。`subject` 令牌存储了审计用户 ID、有效用户 ID 和组 ID、真实用户 ID 和组 ID、进程 ID、会话 ID、端口 ID 和登录地址。注意，审计用户 ID 和真实用户 ID 是不同的，因为在运行此命令之前，用户 `robert` 切换到了 `root` 帐户，但是使用原始的经过身份验证的用户进行审计。`return` 令牌表示成功执行，`trailer` 结束记录。

还可以通过包含 -x 来使用 XML 输出格式。

由于审计日志可能非常大，因此可以使用选择记录子集 auditreduce。这个例子选择了所有为存储在 **AUDITFILE** 中的用户 trhodes 产生的审计记录:

```
# auditreduce -u trhodes /var/audit/AUDITFILE | praudit
```

audit 组的成员有权读取 /var/audit 中的审计跟踪。默认情况下，该组为空，因此只有 root 用户可以读取审计跟踪。可以将用户添加到 audit 组中以授予审计复核权限。由于跟踪审计日志内容的能力提供了对用户和进程行为的重要洞察，因此建议谨慎执行审计审查权限的授权。

19.4.1.使用审计管道进行实时监控

审计管道是克隆的伪设备，它允许应用程序利用实时审计记录流。这主要是入侵检测和系统监控应用程序的作者感兴趣的。然而，审计管道设备是使管理员实时监控的便捷方式，而不会遇到审计跟踪文件所有权或日志轮替中断事件流的问题。要跟踪实时审计事件流:

```
# praudit /dev/auditpipe
```

默认情况下，审计管道设备节点只能由 root 用户访问。要使 audit 组成员可以访问，请将 devfs 规则添加到 **/etc/devfs.rules**：

```
add path 'auditpipe*' mode 0440 group audit
```

有关配置 devfs 文件系统的更多信息，请参阅 devfs.rules(5)[1160]。

> 警告
>
> 审计事件反馈周期很容易产生，其中查看每个审计事件会导致生成更多审计事件。例如，如果所有网络 I/O 都经过审计，并且 praudit 从 SSH 会话运行，则会以高速率生成连续的审计事件流，因为每个被打印的事件都会生成另一个事件。出于这个原因，建议在审计管道设备上从会话运行 praudit，而不进行细粒度的 I/O 审计。

19.4.2.轮替和压缩审计跟踪文件

审计跟踪由内核写入并由审计守护进程 auditd(8)[1161] 管理。管理员不应尝试使用 newsyslog.conf(5)[1162] 或其他工具直接轮替审核日志。相反，应该用 audit 关闭审计、重新配置审计系统和执行日志轮替。下面的命令使审计守护进程创建一个新的审计日志并通知内核切换到使用新的日志。旧日志将被终止并重命名，此时管理员可能会对其进行操作:

[1160] https://www.freebsd.org/cgi/man.cgi?query=devfs.rules&sektion=5&format=html

[1161] https://www.freebsd.org/cgi/man.cgi?query=auditd&sektion=8&format=html

[1162] https://www.freebsd.org/cgi/man.cgi?query=newsyslog.conf&sektion=5&format=html

```
# audit -n
```

如果 auditd(8)[1163] 当前未运行，此命令将失败并生成错误消息。

将以下行添加到 /etc/crontab 将安排每十二小时轮替一次：

```
0     */12       *       *       *       root    /usr/sbin/audit -n
```

更改将在 /etc/crontab 保存后生效。

如 The audit_control File[1164] 中所述，可以使用 audit_control 中的 filesz 根据文件大小自动轮替审计跟踪文件。

由于审计跟踪文件可能会变得非常大，因此通常希望在审计守护程序关闭跟踪后对其进行压缩或归档。audit_warn 脚本可用于对各种与审计相关的事件执行自定义操作，包括在轮替审计跟踪时彻底终止它们。例如，可以将以下内容添加到 /etc/security/audit_warn 以在关闭时压缩审计跟踪：

```
#
# Compress audit trail files on close.
#
if [ "$1" = closefile ]; then
        gzip -9 $2
fi
```

其他归档活动可能包括将跟踪文件复制到中央服务器、删除旧跟踪文件或精选审计跟踪以删除不需要的记录。仅当完全终止审计跟踪文件时才会运行此脚本。它不会在不正确关闭后未终止的路径上运行。

[1163] https://www.freebsd.org/cgi/man.cgi?query=auditd&sektion=8&format=html

[1164] https://docs.freebsd.org/en/books/handbook/audit/#audit-auditcontrol

第二十章存储

20.1.概述

这一章介绍了 FreeBSD 中磁盘和存储设备的使用方法，包括 SCSI 和 IDE 磁盘、CD 和 DVD 光盘、内存盘以及 USB 存储设备。

读完本章后，你会了解：

- 如何在 FreeBSD 系统中添加额外的硬盘。
- 如何在 FreeBSD 上增加磁盘的分区大小。
- 如何配置 FreeBSD 来使用 USB 存储设备。
- 如何在 FreeBSD 系统中使用 CD 和 DVD 光盘。
- 如何在 FreeBSD 下使用可用的备份程序。
- 如何设置闪存设备。
- 什么是文件系统快照以及如何有效地使用它们。
- 如何使用配额来限制磁盘空间的使用。
- 如何对磁盘和交换空间进行加密以防范攻击者。
- 如何配置高可用性存储网络。

在阅读本章之前，你应该：

- 知道如何配置和安装一个新的 FreeBSD 内核[1165]。

[1165] https://docs.freebsd.org/en/books/handbook/kernelconfig/index.html#kernelconfig

20.2.添加磁盘

本节介绍了如何在当前只有一块磁盘的机器上添加一块新的 SATA 磁盘。首先，关闭计算机，按照计算机、控制器和磁盘制造商的说明将磁盘安装到计算机中。重新启动系统并切换至 `root` 用户。

检查 **/var/run/dmesg.boot** 以确保新的磁盘被识别。在这个例子中，新增加的 SATA 磁盘显示为 **ada1**。

在这个例子中，将在新磁盘上创建一个唯一的大分区，并首选 GPT 分区方案，而非老的、功能较少的 MBR 方案。

> 注意
> 如果要添加的磁盘不是空的，可以用 `gpart delete` 来删除旧的分区信息。详见 gpart(8)[1166]。

分区方案创建之后，添加一个分区。为了提高在具有较大物理块的新磁盘上的性能，分区边界被对齐到一兆字节：

```
# gpart create -s GPT ada1
# gpart add -t freebsd-ufs -a 1M ada1
```

根据使用情况，可能需要几个较小的分区。参见 gpart(8)[1167] 以了解创建比整个磁盘小的分区的选项。

可以用 `gpart show` 来查看磁盘分区信息：

```
% gpart show ada1
=>        34  1465146988  ada1  GPT  (699G)
          34        2014        - free -  (1.0M)
        2048  1465143296     1  freebsd-ufs  (699G)
  1465145344        1678        - free -  (839K)
```

在新磁盘的新分区中创建一个文件系统：

```
# newfs -U /dev/ada1p1
```

创建一个空目录做 *挂载点*，这是一个用于在原先磁盘的文件系统中挂载新磁盘分区的位置：

```
# mkdir /newdisk
```

最后，在 **/etc/fstab** 中添加一个条目，这样将在启动时自动挂载新的磁盘分区：

```
/dev/ada1p1 /newdisk    ufs rw  2   2
```

也可以手动挂载新的磁盘分区，而无需重新启动系统。

[1166] https://www.freebsd.org/cgi/man.cgi?query=gpart&sektion=8&format=html
[1167] https://www.freebsd.org/cgi/man.cgi?query=gpart&sektion=8&format=html

```
# mount /newdisk
```

20.3.调整和增加磁盘大小

可以增加磁盘的容量，而无需对已经存在的数据做任何改变。这在虚拟机上经常使用，当虚拟机磁盘太小时可以增加容量。有时磁盘镜像被写入 U 盘，但没有使用全部容量。这里我们将介绍如何调整磁盘内容的大小来使用增加的容量。

通过查看 **/var/run/dmesg.boot** 来确定要调整大小的磁盘的设备名称。在这个例子中，系统中只有一个 SATA 磁盘，所以该磁盘将显示为 **ada0**。

列出磁盘上的分区来查看当前的配置：

```
# gpart show ada0
=>       34  83886013  ada0  GPT  (48G) [CORRUPT]
         34       128     1  freebsd-boot  (64k)
        162  79691648     2  freebsd-ufs  (38G)
   79691810   4194236     3  freebsd-swap  (2G)
   83886046         1        - free -  (512B)
```

> 注意
>
> 如果磁盘是用 GPT 分区方案格式化的，它可能显示为 corrupted，因为 GPT 备份分区表不再位于磁盘的末端。使用 gpart 可修正备份分区表：
>
> ```
> # gpart recover ada0
> ada0 recovered
> ```

现在，磁盘上的额外空间可供一个新的分区使用，或者可以扩展一个现有的分区：

```
# gpart show ada0
=>        34  102399933  ada0  GPT  (48G)
          34        128     1  freebsd-boot  (64k)
         162   79691648     2  freebsd-ufs  (38G)
    79691810    4194236     3  freebsd-swap  (2G)
    83886046   18513921        - free -  (8.8G)
```

只能在连续的未使用空间上调整分区。此处，磁盘上的最后一个分区是交换分区，但第二个分区是需要调整大小的分区。交换分区只包含临时数据，所以它可以安全地被卸载、删除，然后再调整第二个分区的大小后重建第三个分区。

禁用交换分区：

```
# swapoff /dev/ada0p3
```

从 *ada0* 磁盘中删除第三个分区，由 `-i` 参数来指定分区：

```
# gpart delete -i 3 ada0
ada0p3 deleted
# gpart show ada0
=>       34  102399933  ada0  GPT  (48G)
         34        128     1  freebsd-boot  (64k)
        162   79691648     2  freebsd-ufs  (38G)
   79691810   22708157        - free -  (10G)
```

> 警告
>
> 在修改已挂载文件系统的分区表时，存在数据丢失的风险。最好是在一个未挂载的文件系统上（从 live CD-ROM 或 USB 设备上运行）执行以下步骤。然而，如果绝对必要，可以在禁用 GEOM 安全功能后调整已挂载文件系统的大小。
>
> ```
> # sysctl kern.geom.debugflags=16
> ```

调整分区的大小，留出空间来重新创建一个所需大小的交换分区。用 `-i` 指定要调整的分区，用 `-s` 指定新的期望大小。可以用 `-a` 来控制分区的对齐。此处只是修改了分区的大小。将在另一个单独的步骤中扩展分区中的文件系统。

```
# gpart resize -i 2 -s 47G -a 4k ada0
ada0p2 resized
# gpart show ada0
=>       34  102399933  ada0  GPT  (48G)
         34        128     1  freebsd-boot  (64k)
        162   98566144     2  freebsd-ufs  (47G)
   98566306    3833661        - free -  (1.8G)
```

重新创建交换分区并启动它。如果没有用 `-s` 指定大小，则会使用所有剩余的空间：

```
# gpart add -t freebsd-swap -a 4k ada0
ada0p3 added
# gpart show ada0
=>       34  102399933  ada0  GPT  (48G)
         34        128     1  freebsd-boot  (64k)
        162   98566144     2  freebsd-ufs  (47G)
   98566306    3833661     3  freebsd-swap  (1.8G)
# swapon /dev/ada0p3
```

扩展 UFS 文件系统，以使用调整后的分区的新容量：

```
# growfs /dev/ada0p2
Device is mounted read-write; resizing will result in temporary write suspension for /
↪.
It's strongly recommended to make a backup before growing the file system.
OK to grow file system on /dev/ada0p2, mounted on /, from 38GB to 47GB? [Yes/No] Yes
super-block backups (for fsck -b #) at:
 80781312, 82063552, 83345792, 84628032, 85910272, 87192512, 88474752,
 89756992, 91039232, 92321472, 93603712, 94885952, 96168192, 97450432
```

如果文件系统是 ZFS，调整大小是通过运行带 `-e` 的 `online` 子命令来触发的：

```
# zpool online -e zroot /dev/ada0p2
```

分区和分区上的文件系统大小现在都已被调整，可以使用新的可用磁盘空间了。

20.4.USB 存储设备

许多外部存储解决方案，如硬盘、U 盘、CD 和 DVD 刻录机，都使用了通用串行总线（USB）。FreeBSD 提供了对 USB 1.x、2.0 和 3.0 设备的支持。

> 注意
>
> USB 3.0 支持与某些硬件不兼容，包括 Haswell（Lynx point）芯片组。如果 FreeBSD 启动时出现消息 `failed with error 19`，请在系统 BIOS 中禁用 xHCI/USB3。

对 USB 存储设备的支持是内置在 **GENERIC** 内核中的。对于定制内核，要确保在内核配置文件中存在以下几行：

```
device scbus    # SCSI bus (required for ATA/SCSI)
device da   # Direct Access (disks)
device pass # Passthrough device (direct ATA/SCSI access)
device uhci # provides USB 1.x support
device ohci # provides USB 1.x support
device ehci # provides USB 2.0 support
device xhci # provides USB 3.0 support
device usb  # USB Bus (required)
device umass    # Disks/Mass storage - Requires scbus and da
device cd   # needed for CD and DVD burners
```

FreeBSD 使用驱动程序 umass(4)[1168]，它使用 SCSI 子系统来访问 USB 存储设备。由于任何 USB 设备都会被系统视为 SCSI 设备，如果 USB 设备是 CD 或 DVD 刻录机，请不要在定制内核配置文件中包含 `device atapicam`。

[1168] https://www.freebsd.org/cgi/man.cgi?query=umass&sektion=4&format=html

本节的其余部分演示了如何验证 USB 存储设备是否被 FreeBSD 识别，以及如何配置使用该设备。

20.4.1.设备配置

要测试 USB 的配置，请插上 USB 设备。使用 `dmesg` 确认该驱动器出现在系统信息缓冲区中。它应该看起来像这样：

```
umass0: <STECH Simple Drive, class 0/0, rev 2.00/1.04, addr 3> on usbus0
umass0:  SCSI over Bulk-Only; quirks = 0x0100
umass0:4:0:-1: Attached to scbus4
da0 at umass-sim0 bus 0 scbus4 target 0 lun 0
da0: <STECH Simple Drive 1.04> Fixed Direct Access SCSI-4 device
da0: Serial Number WD-WXE508CAN263
da0: 40.000MB/s transfers
da0: 152627MB (312581808 512 byte sectors: 255H 63S/T 19457C)
da0: quirks=0x2<NO_6_BYTE>
```

品牌、设备节点（**da0**）、速率和大小将因设备而异。

由于 USB 设备被看作是 SCSI 设备，可以用 `camcontrol` 列出连接到系统的 USB 存储设备：

```
# camcontrol devlist
<STECH Simple Drive 1.04>          at scbus4 target 0 lun 0 (pass3,da0)
```

另外，可以用 `usbconfig` 列出设备。请参考 usbconfig(8)[1169] 以了解关于这个命令的更多信息。

```
# usbconfig
ugen0.3: <Simple Drive STECH> at usbus0, cfg=0 md=HOST spd=HIGH (480Mbps) pwr=ON (2mA)
```

如果设备还没有被格式化，请参阅“添加磁盘[1170]”，了解如何在 U 盘上格式化和创建分区。如果设备带有文件系统，可以使用“挂载和卸载文件系统[1171]”中的说明在 `root` 用户下挂载。

> 警告
>
> 通过启用下面介绍的 `vfs.usermount`，以允许不受信的用户挂载任意的设备是不安全的。大多数文件系统都不是为了防范恶意设备而建立的。

为了使普通用户可以挂载设备，可以使用 pw(8)[1172] 使设备的所有用户成为 `operator` 组的成员。接下来，通过在 **/etc/devfs.rules** 中添加这几行，确保 `operator` 能够读写设备：

[1169] https://www.freebsd.org/cgi/man.cgi?query=usbconfig&sektion=8&format=html
[1170] https://docs.freebsd.org/en/books/handbook/disks/#disks-adding
[1171] https://docs.freebsd.org/en/books/handbook/basics/index.html#mount-unmount
[1172] https://www.freebsd.org/cgi/man.cgi?query=pw&sektion=8&format=html

```
[localrules=5]
add path 'da*' mode 0660 group operator
```

注意

如果系统中也内置安装了 SCSI 磁盘，请将第二行修改如下：

```
add path 'da[3-9]*' mode 0660 group operator
```

这将使前三个 SCSI 磁盘（从 **da0** 到 **da2**）脱离 `operator` 组。用内置 SCSI 磁盘的数量来替换数字 *3*。参考 devfs.rules(5)[1173] 以了解更多关于这个文件的信息。

接下来，在 **/etc/rc.conf** 中启用该规则集：

```
devfs_system_ruleset="localrules"
```

然后，在 **/etc/sysctl.conf** 中添加以下一行，让系统允许普通用户挂载文件系统：

```
vfs.usermount=1
```

因为这只在下次重启后生效，可以使用 `sysctl` 来立即设置这个变量：

```
# sysctl vfs.usermount=1
vfs.usermount: 0 -> 1
```

最后一步是创建一个挂载文件系统的目录。这个目录需要由要挂载文件系统的用户拥有。一种方法是让 `root` 创建一个由该用户拥有的 **/mnt/username** 子目录。在下面的例子中，用用户的登录名代替 `username`，用用户的主组代替 `usergroup`：

```
# mkdir /mnt/username
# chown username:usergroup /mnt/username
```

假设一个 U 盘被插入，并且出现了一个设备 **/dev/da0s1**。如果该设备是用 FAT 文件系统格式化的，用户可以用以下方法挂载它：

```
% mount -t msdosfs -o -m=644,-M=755 /dev/da0s1 /mnt/username
```

在拔出设备之前，必须先将其卸载：

```
% umount /mnt/username
```

移除设备后，系统信息缓冲区将显示类似以下的信息：

[1173] https://www.freebsd.org/cgi/man.cgi?query=devfs.rules&sektion=5&format=html

```
umass0: at uhub3, port 2, addr 3 (disconnected)
da0 at umass-sim0 bus 0 scbus4 target 0 lun 0
da0: <STECH Simple Drive 1.04> s/n WD-WXE508CAN263          detached
(da0:umass-sim0:0:0:0): Periph destroyed
```

20.4.2.自动挂载可移动媒体

通过取消对 **/etc/auto_master** 中这一行的注释，可以自动挂载 USB 设备：

```
/media      -media      -nosuid
```

然后在 **/etc/devd.conf** 中添加这几行：

```
notify 100 {
    match "system" "GEOM";
    match "subsystem" "DEV";
    action "/usr/sbin/automount -c";
};
```

如果 autofs(5)[1174] 和 devd(8)[1175] 已经在运行，可以重新加载配置：

```
# service automount restart
# service devd restart
```

可以通过在 **/etc/rc.conf** 中添加这行来设置为在启动时启动 autofs(5)[1176]：

```
autofs_enable="YES"
```

autofs(5)[1177] 需要启用 devd(8)[1178]，它在默认情况下是启用的。

使用如下命令立即启动服务：

```
# service automount start
# service automountd start
# service autounmountd start
# service devd start
```

[1174] https://www.freebsd.org/cgi/man.cgi?query=autofs&sektion=5&format=html
[1175] https://www.freebsd.org/cgi/man.cgi?query=devd&sektion=8&format=html
[1176] https://www.freebsd.org/cgi/man.cgi?query=autofs&sektion=5&format=html
[1177] https://www.freebsd.org/cgi/man.cgi?query=autofs&sektion=5&format=html
[1178] https://www.freebsd.org/cgi/man.cgi?query=devd&sektion=8&format=html

每个可以自动挂载的文件系统在 **/media/** 中都显示为一个目录。该目录是以文件系统的标签命名的。如果标签缺失，该目录就以设备节点命名。

文件系统在第一次访问时被自动挂载，并在一段时间未使用后被取消挂载。也可以手动卸载自动挂载的设备：

```
# automount -fu
```

这种机制通常用于存储卡和 U 盘。它可以用于任何块设备，包括光驱或 iSCSILUN。

20.5.创建和使用 CD

紧凑型光盘（Compact Disc，CD）提供了一些有别于传统磁盘的特点。它们被设计成可以连续读取，而无需等待在轨道之间移动磁头的延迟。虽然 CD 确实有轨道，但这是指要连续读取的数据部分，而不是磁盘的物理属性。ISO 9660 文件系统就是为了处理这些差异而设计的。

FreeBSD ports 中提供了几个用于刻录和复制音频和数据 CD 的工具。这一章演示了几个命令行工具的使用。对于带有图形化的 CD 刻录软件，可以考虑通过软件包或 ports 安装 sysutils/xcdroast[1179] 或 sysutils/k3b[1180]。

20.5.1.支持的设备

GENERIC 内核提供了对 SCSI，USB 和 ATAPICD 读卡器和刻录机的支持。如果使用定制内核，需要在内核配置文件中出现的选项因设备的类型而异。

对于 SCSI 刻录机，确保这些选项是存在的：

```
device scbus    # SCSI bus (required for ATA/SCSI)
device da   # Direct Access (disks)
device pass # Passthrough device (direct ATA/SCSI access)
device cd   # needed for CD and DVD burners
```

对于 USB 刻录机，确保这些选项是存在的：

```
device scbus    # SCSI bus (required for ATA/SCSI)
device da   # Direct Access (disks)
device pass # Passthrough device (direct ATA/SCSI access)
device cd   # needed for CD and DVD burners
device uhci # provides USB 1.x support
device ohci # provides USB 1.x support
device ehci # provides USB 2.0 support
```

(continues on next page)

[1179] https://cgit.freebsd.org/ports/tree/sysutils/xcdroast/pkg-descr

[1180] https://cgit.freebsd.org/ports/tree/sysutils/k3b/pkg-descr

(continued from previous page)

```
device xhci # provides USB 3.0 support
device usb  # USB Bus (required)
device umass    # Disks/Mass storage - Requires scbus and da
```

对于 ATAPI 刻录机，要确保这些选项是存在的：

```
device ata  # Legacy ATA/SATA controllers
device scbus    # SCSI bus (required for ATA/SCSI)
device pass # Passthrough device (direct ATA/SCSI access)
device cd   # needed for CD and DVD burners
```

注意

在 10.x 之前的 FreeBSD 版本中，如果刻录机是 ATAPI 设备，在内核配置文件中还需要这一行：

```
device atapicam
```

另外，通过在 **/boot/loader.conf** 中添加以下一行，可以在启动时加载该驱动：

```
atapicam_load="YES"
```

需要重新启动系统，因为这个驱动程序只能在启动时被加载。

为了验证 FreeBSD 是否识别该设备，执行 `dmesg` 并查找该设备的条目。在 10.x 之前的系统中，输出的第一行中的设备名称将是 **acd0** 而非 **cd0**。

```
% dmesg | grep cd
cd0 at ahcich1 bus 0 scbus1 target 0 lun 0
cd0: <HL-DT-ST DVDRAM GU70N LT20> Removable CD-ROM SCSI-0 device
cd0: Serial Number M3OD3S34152
cd0: 150.000MB/s transfers (SATA 1.x, UDMA6, ATAPI 12bytes, PIO 8192bytes)
cd0: Attempt to query device size failed: NOT READY, Medium not present - tray closed
```

20.5.2.刻录 CD

在 FreeBSD 中，可以用 `cdrecord` 来刻录 CD。这个命令通过软件包或 port sysutils/cdrtools[1181] 来安装。

虽然 `cdrecord` 有许多选项，但基本用法很简单。指定要刻录的 ISO 文件的名称，如果系统有多个刻录机设备，则指定要使用的设备的名称：

```
# cdrecord dev=device imagefile.iso
```

为了确定刻录机的设备名称，使用 `-scanbus`，可能的输出如下：

[1181] https://cgit.freebsd.org/ports/tree/sysutils/cdrtools/pkg-descr

```
# cdrecord -scanbus
ProDVD-ProBD-Clone 3.00 (amd64-unknown-freebsd10.0) Copyright (C) 1995-2010 Jörg␣
↪Schilling
Using libscg version 'schily-0.9'
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) *
```

找到 CD 刻录机的条目，用逗号分隔的三个数字作为 `dev` 的值。在本例中，Yamaha 刻录机的设备是 `1,5,0`，所以指定该设备的适当输入是 `dev=1,5,0`。参考 `cdrecord` 的手册页，了解指定该值的其他方法，以及关于写入音轨和控制写入速度的信息。

另外，还可以执行下面的命令来获得刻录机的设备地址：

```
# camcontrol devlist
<MATSHITA CDRW/DVD UJDA740 1.00>   at scbus1 target 0 lun 0 (cd0,pass0)
```

使用 `scbus`、`target` 和 `lun` 的数字值。在这个例子中，`1,0,0` 是要使用的设备名。

20.5.3.将数据写入 ISO 文件系统

为了制作数据 CD，在刻录到 CD 上之前必须准备好构成 CD 轨道的数据文件。在 FreeBSD 中，sysutils/cdrtools[1182] 安装了 `mkisofs`，可以用它来制作一个 ISO 9660 文件系统，它是 UNIX® 文件系统中目录的镜像。最简单的用法是指定要创建的 ISO 文件的名称和要放入 ISO 9660 文件系统的文件的路径：

```
# mkisofs -o imagefile.iso /path/to/tree
```

[1182] https://cgit.freebsd.org/ports/tree/sysutils/cdrtools/pkg-descr

该命令将指定路径中的文件名映射为符合标准 ISO 9660 文件系统要求的名称，并将排除那些不符合 ISO 文件系统标准的文件。

有许多选项可以克服该标准所施加的限制。特别是，`-R` 将启用 UNIX® 系统常用的 Rock Ridge 扩展，`-J` 将启用 Microsoft® 系统使用的 Joliet 扩展。

对于只在 FreeBSD 系统上使用的 CD，可以用 `-U` 来禁用所有的文件名限制。当与 `-R` 一起使用时，它产生的文件系统镜像与指定的 FreeBSD 目录相同，即使它违反了 ISO 9660 标准。

最后一个常用选项是 `-b`。它被用来指定用于生产“El Torito”可引导 CD 的引导镜像的位置。这个选项需要一个参数，即被写入 CD 根目录的引导镜像的路径。默认情况下，`mkisofs` 在“软盘仿真”模式下创建 ISO 镜像，因此它希望引导镜像的大小正好是 1200、1440 或 2880 KB。一些像 FreeBSD 发行镜像使用的那种引导加载器，不使用仿真模式。在这种情况下，应该使用 `-no-emul-boot`。因此，如果 `/tmp/myboot` 包含一个可引导的 FreeBSD 系统，其启动镜像在 **/tmp/myboot/boot/cdboot** 中，这个命令将生成 **/tmp/bootable.iso**：

```
# mkisofs -R -no-emul-boot -b boot/cdboot -o /tmp/bootable.iso /tmp/myboot
```

由此产生的 ISO 镜像可以作为闪存设备被挂载：

```
# mdconfig -a -t vnode -f /tmp/bootable.iso -u 0
# mount -t cd9660 /dev/md0 /mnt
```

然后我们可以验证 **/mnt** 和 **/tmp/myboot** 是相同的。

`mkisofs` 还有许多其他选项可以用来微调其行为。详情请参考 mkisofs(8)[1183]。

> 注意
>
> 可以将数据 CD 复制到一个镜像文件中，该文件在功能上与用 `mkisofs` 创建的镜像文件相当。要做到这一点，使用 **dd**，将设备名称作为输入文件，将要创建的 ISO 的名称作为输出文件：
>
> ```
> # dd if=/dev/cd0 of=file.iso bs=2048
> ```

产生的镜像文件可以按照刻录 CD[1184] 中的说明刻录到 CD 上。

20.5.4.使用数据 CD

ISO 被刻录到 CD 上以后，就可以通过指定文件系统类型、包含 CD 的设备名称和一个现有的挂载点来挂载它：

```
# mount -t cd9660 /dev/cd0 /mnt
```

由于挂载时假定文件系统是 `ufs` 类型的，如果在挂载数据 CD 时不附带 `-t cd9660`，将产生错误 `Incorrect super block`。

[1183] https://www.freebsd.org/cgi/man.cgi?query=mkisofs&sektion=8&format=html

[1184] https://docs.freebsd.org/en/books/handbook/book/#cdrecord

虽然任何数据 CD 都可以这样挂载，但具有某些 ISO 9660 扩展名的磁盘可能表现得很奇怪。例如，Joliet 磁盘以两字节的 Unicode 字符存储所有文件名。如果一些非英语字符显示为问号，请用 `-C` 指定本地字符集。更多信息，请参考mount_cd9660(8)[1185]。

注意

为了使用 `-C` 进行这种字符转换，内核需要加载 **cd9660_iconv.ko** 模块。这可以通过在 **loader.conf** 中加入这一行来完成：

```
cd9660_iconv_load="YES"
```

然后重启机器，或者直接用 `kldload` 加载模块。

当试图挂载数据 CD 时，偶尔会显示 `Device not configured`。这通常意味着 CD 驱动器没有检测到托盘中的磁盘，或者驱动器在总线上不可见。光驱检测光盘可能需要几秒钟的时间，所以要有耐心。

有时，SCSI CD 驱动器可能会被遗漏，因为它没有足够的时间来应答总线复位。为了解决这个问题，可以创建一个定制内核，增加默认的 SCSI 延迟。在定制的内核配置文件中增加以下选项，并按照“编译与安装定制内核”[1186]的说明重建内核：

```
options SCSI_DELAY=15000
```

这让 SCSI 总线在启动过程中暂停 15 秒，以便给 CD 驱动器一切可能的机会来应答总线复位。

注意

可以直接将文件刻录到 CD，而不创建 ISO 9660 文件系统。这就是所谓的刻录原始数据 CD，有些人为了备份会这样做。

这种类型的磁盘不能像普通的数据 CD 一样被挂载。为了检索刻录到这种 CD 上的数据，必须从原始设备节点上读取数据。例如，这个命令将把位于第二个 CD 设备上的压缩 tar 文件提取到当前工作目录中：

```
# tar xzvf /dev/cd1
```

为了挂载数据 CD，必须使用 `mkisofs` 写入数据。

20.5.5.复制音频 CD

要复制一张音频 CD，把 CD 上的音频数据提取成一系列的文件，然后把这些文件写入一张空白 CD。

本节说明了如何复制和刻录一张音频 CD。如果 FreeBSD 的版本低于 10.0，并且设备是 ATAPI 总线，必须首先加载 `atapicam` 模块。

复制一张音频 CD 的过程

[1185] https://www.freebsd.org/cgi/man.cgi?query=mount_cd9660&sektion=8&format=html

[1186] https://docs.freebsd.org/en/books/handbook/kernelconfig/index.html#kernelconfig-building

1. 软件包或 port sysutils/cdrtools[1187] 安装了 cdda2wav。这个命令可以用来提取所有的音轨，每个音轨都写在当前工作目录下的一个单独的 WAV 文件中：

```
% cdda2wav -vall -B -Owav
```

如果系统上只有一个 CD 设备，则不需要指定设备名。参考 cdda2wav 手册中关于如何指定一个设备的说明，并了解更多关于这个命令的其他选项。

1. 用 cdrecord 来写入 **.wav** 文件。

```
% cdrecord -v dev=2,0 -dao -useinfo  *.wav
```

确保如刻录 CD[1188] 所述正确地设置了 2,0。

20.6.创建和使用 DVD

与 CD 相比，DVD 是下一代的光学媒体存储技术。相比于 CD，DVD 可以容纳更多的数据，是视频出版的标准。

可记录式 DVD 有五种物理格式：

- DVD-R：这是第一个可用的 DVD 可记录格式。DVD-R 标准是由 DVD 论坛[1189]定义的。这种格式仅支持一次性写入。
- DVD-RW: 这是 DVD-R 标准的可重写版本。一张 DVD-RW 可以被重写 1000 次左右。
- DVD-RAM: 这是一种可重写的格式，可以被看作是一种可移动的硬盘设备。然而，这种光盘与大多数 DVD-ROM 光驱和 DVD 视频播放器不兼容，只有少数 DVD 刻录机支持 DVD-RAM 格式。请参考 使用 DVD-RAM[1190]，了解更多关于 DVD-RAM 的使用信息。
- DVD+RW: 这是一种由 DVD+RW 联盟[1191]定义的可重写格式。一张 DVD+RW 可以被重写大约 1000 次。
- DVD+R: 这种格式是 DVD+RW 格式的一次性写入变体。

一张单层可刻录的 DVD 可以容纳 4,700,000,000 byte，实际上是 4.38 GB 或 4485 MB，因为 1 kb 是 1024 byte。

注意

必须对物理光盘和应用程序进行区分。例如，DVD-Video 是一种特定的文件系统，可以被刻录到任何 DVD 物理光盘上，如 DVD-R、DVD+R 或 DVD-RW。在选择光盘类型之前，要确保刻录机和 DVD 视频播放器都能与要选择的光盘兼容。

[1187] https://cgit.freebsd.org/ports/tree/sysutils/cdrtools/pkg-descr
[1188] https://docs.freebsd.org/en/books/handbook/book/#cdrecord
[1189] http://www.dvdforum.org/forum.shtml
[1190] https://docs.freebsd.org/en/books/handbook/disks/#creating-dvd-ram
[1191] https://en.wikipedia.org/wiki/DVD%2BRW_Alliance

20.6.1.配置

要执行 DVD 刻录，请使用 growisofs(1)[1192]。这个命令支持所有 DVD 光盘类型，它来自 sysutils/dvd+rw-tool[1193] 工具。

这些工具使用 SCSI 子系统来访问设备，因此必须加载或静态编译 ATAPI/CAM[1194] 到内核。如果刻录机使用 USB 接口，则不需要这种操作。关于 USB 设备配置的更多细节，请参考 USB 存储设备[1195]。

还必须为 ATAPI 设备启用 DMA 访问，在 **/boot/loader.conf** 中添加以下一行：

```
hw.ata.atapi_dma="1"
```

在尝试使用 `dvd+rw-tools` 之前，请查阅硬件兼容性说明[1196]。

> 注意
>
> 对于图形用户界面，可以考虑使用 sysutils/k3b[1197]，它为 growisofs(1)[1198] 和许多其他刻录工具提供了一个用户友好的界面。

20.6.2.刻录数据 DVD

由于 growisofs(1)[1199] 是 mkisofs[1200] 的前端，它将调用 mkisofs(8)[1201] 来创建文件系统并在 DVD 上执行写入。这意味着在刻录过程中不需要创建数据的镜像。

要将 **/path/to/data** 中的数据刻录到 DVD+R 或 DVD-R 上，请使用以下命令：

```
# growisofs -dvd-compat -Z /dev/cd0 -J -R /path/to/data
```

在这个例子中，`-J -R` 被传递给 mkisofs(8)[1202] 来创建一个带有 Joliet 和 Rock Ridge 扩展支持的 ISO 9660 文件系统。更多细节请参考 mkisofs(8)[1203]。

对于初始刻录，`-Z` 用于单次和多次会话。用 DVD 设备的名称替换 */dev/cd0*。使用 `-dvd-compat` 表示磁盘将被关闭，录制的内容将无法追加。这也应该提供与 DVD-ROM 驱动器更好的设备兼容性。

要刻录成一个预制的镜像，如 `imagefile.iso`，使用：

[1192] https://www.freebsd.org/cgi/man.cgi?query=growisofs&sektion=1&format=html
[1193] https://cgit.freebsd.org/ports/tree/sysutils/dvd+rw-tools/pkg-descr
[1194] https://docs.freebsd.org/en/books/handbook/disks/#atapicam
[1195] https://docs.freebsd.org/en/books/handbook/book/#usb-disks
[1196] http://fy.chalmers.se/~appro/linux/DVD+RW/hcn.html
[1197] https://cgit.freebsd.org/ports/tree/sysutils/k3b/pkg-descr
[1198] https://www.freebsd.org/cgi/man.cgi?query=growisofs&sektion=1&format=html
[1199] https://www.freebsd.org/cgi/man.cgi?query=growisofs&sektion=1&format=html
[1200] https://docs.freebsd.org/en/books/handbook/disks/#mkisofs
[1201] https://www.freebsd.org/cgi/man.cgi?query=mkisofs&sektion=8&format=html
[1202] https://www.freebsd.org/cgi/man.cgi?query=mkisofs&sektion=8&format=html
[1203] https://www.freebsd.org/cgi/man.cgi?query=mkisofs&sektion=8&format=html

```
# growisofs -dvd-compat -Z /dev/cd0=imagefile.iso
```

刻录速度根据设备和正在使用的驱动器自动设置。要强制设置刻录速度，请使用 `-speed=`。请参考 growisofs(1) 中例子的用法。

> 注意
>
> 为了支持大于 4.38GB 的工作文件，必须通过向 mkisofs(8)[1204] 和所有相关程序，如 growisofs(1)[1205] 传递 `-udf -iso-level 3` 来创建一个 UDF/ISO-9660 混合文件系统。只有在创建 ISO 镜像文件或直接向磁盘写入文件时才需要这样做。由于以这种方式创建的磁盘必须用 mount_udf(8)[1206] 挂载为 UDF 文件系统，所以它只能在支持 UDF 的操作系统上使用。否则，它看起来就像含有损坏的文件。
>
> 要创建这种类型的 ISO 文件:
>
> ```
> % mkisofs -R -J -udf -iso-level 3 -o imagefile.iso /path/to/data
> ```
>
> 将文件直接刻录到磁盘上:
>
> ```
> # growisofs -dvd-compat -udf -iso-level 3 -Z /dev/cd0 -J -R /path/to/data
> ```
>
> 当一个 ISO 映像已经包含大文件时，growisofs(1)[1207] 不需要额外的选项就可以将该镜像刻录到磁盘上。请确保使用包含 mkisofs(8)[1208] 的最新版本的 sysutils/cdrtools[1209]，因为旧版本可能不包含对大文件的支持。如果最新版本不能工作，请安装 sysutils/cdrtools-devel[1210]，并阅读其 mkisofs(8)[1211]。

20.6.3.刻录 DVD-Video

DVD-Video 是一种基于 ISO 9660 和 micro-UDF（M-UDF）规范的特定文件系统。由于 DVD-Video 提出了一个特定的数据结构层次，因此需要一个特定的程序，如 multimedia/dvdauthor[1212] 来编写 DVD。

如果 DVD-Video 文件系统的镜像已经存在，可以用与其他镜像相同的方式进行刻录。如果使用 `dvdauthor` 制作 DVD，并且目录在 **/path/to/video** 中，应该使用下面的命令来刻录 DVD-Video:

```
# growisofs -Z /dev/cd0 -dvd-video /path/to/video
```

`-dvd-video` 被传递给 mkisofs(8)[1213]，让它创建一个 DVD-Video 文件系统。这个选项意味着 grow-

[1204] https://www.freebsd.org/cgi/man.cgi?query=mkisofs&sektion=8&format=html
[1205] https://www.freebsd.org/cgi/man.cgi?query=growisofs&sektion=1&format=html
[1206] https://www.freebsd.org/cgi/man.cgi?query=mount_udf&sektion=8&format=html
[1207] https://www.freebsd.org/cgi/man.cgi?query=growisofs&sektion=1&format=html
[1208] https://www.freebsd.org/cgi/man.cgi?query=mkisofs&sektion=8&format=html
[1209] https://cgit.freebsd.org/ports/tree/sysutils/cdrtools/pkg-descr
[1210] https://cgit.freebsd.org/ports/tree/sysutils/cdrtools-devel/pkg-descr
[1211] https://www.freebsd.org/cgi/man.cgi?query=mkisofs&sektion=8&format=html
[1212] https://cgit.freebsd.org/ports/tree/multimedia/dvdauthor/pkg-descr
[1213] https://www.freebsd.org/cgi/man.cgi?query=mkisofs&sektion=8&format=html

isofs(1)[1214] 的 -dvd-compat 选项。

20.6.4.使用 DVD+RW

与 CD-RW 不同，需要在第一次使用前格式化空白的 DVD+RW。建议让 growisofs(1)[1215] 自动处理这个问题。也可以使用 dvd+rw-format 来格式化 DVD+RW：

```
# dvd+rw-format /dev/cd0
```

只需执行一次此操作，只有空白的 DVD+RW 需要被格式化。格式化之后，就可以像往常一样刻录 DVD+RW。

要刻录一个全新的文件系统，而不仅仅是在 DVD+RW 上追加一些数据，不需要先清空光盘。而是要这样清空数据：

```
# growisofs -Z /dev/cd0 -J -R /path/to/newdata
```

DVD+RW 格式支持将数据追加到以前的记录中。这个操作包括将一个新的片段合并到现有的片段上，因为它不被认为是多片段写入。growisofs(1)[1216] 将 扩展媒体上存在的 ISO 9660 文件系统。

例如，要向 DVD+RW 追加数据，请使用以下方法：

```
# growisofs -M /dev/cd0 -J -R /path/to/nextdata
```

在以后的写入过程中，应该使用与初次刻录时使用的相同的 mkisofs(8)[1217] 选项。

注意

使用 -dvd-compat 以获得与 DVD-ROM 驱动器更好的设备兼容性。当使用 DVD+RW 时，这个选项不会导致数据无法添加。

要清空光盘，请使用：

```
# growisofs -Z /dev/cd0=/dev/zero
```

[1214] https://www.freebsd.org/cgi/man.cgi?query=growisofs&sektion=1&format=html
[1215] https://www.freebsd.org/cgi/man.cgi?query=growisofs&sektion=1&format=html
[1216] https://www.freebsd.org/cgi/man.cgi?query=growisofs&sektion=1&format=html
[1217] https://www.freebsd.org/cgi/man.cgi?query=mkisofs&sektion=8&format=html

20.6.5.使用 DVD-RW

DVD-RW 接受两种光盘格式。递增顺序写入和受限覆写。默认情况下，DVD-RW 光盘使用递增顺序写入格式。

可以直接写入空白的 DVD-RW，不需要进行格式化。然而，在写入新的数据之前，需要以递增顺序写入格式处理非空白 DVD-RW。

要清空递增顺序写入 DVD-RW 请使用：

```
# dvd+rw-format -blank=full /dev/cd0
```

> 注意
>
> 在 1x 光盘上使用 `-blank=full` 进行完全清空，大约需要一个小时。如果 DVD-RW 将以整盘一次写入（DAO）模式刻录，可以使用 `-blank` 进行快速清空。要在 DAO 模式下刻录 DVD-RW，请使用以下命令：
>
> ```
> # growisofs -use-the-force-luke=dao -Z /dev/cd0=imagefile.iso
> ```
>
> 由于 growisofs(1)[1218] 会自动尝试检测快速的空白光盘并进行 DAO 写入，所以无需使用`-use-the-force-luke=dao`。

任何 DVD-RW 都应该使用受限覆写，因为这种格式比默认的递增顺序写入更灵活。

要在递增顺序写入格式的 DVD-RW 上写入数据，使用与其他 DVD 格式相同的指令：

```
# growisofs -Z /dev/cd0 -J -R /path/to/data
```

要将一些数据追加到以前的记录中，请使用 growisofs(1)[1219] 的 `-M` 选项。然而，如果在 DVD-RW 上以递增顺序写入模式追加数据，将在光盘上创建一个新的会话，结果将是一个多会话光盘。

在一个新的初始会话之前，不需要清空受限覆写格式的 DVD-RW。可以用 `-Z` 覆盖光盘。也可以用 `-M` 来扩展现有刻在光盘上的 ISO 9660 文件系统。最后会是一个单会话 DVD。

要使 DVD-RW 成为受限覆写格式，必须使用以下命令：

```
# dvd+rw-format /dev/cd0
```

要改回递增顺序写入格式，请使用:

```
# dvd+rw-format -blank=full /dev/cd0
```

[1218] https://www.freebsd.org/cgi/man.cgi?query=growisofs&sektion=1&format=html
[1219] https://www.freebsd.org/cgi/man.cgi?query=growisofs&sektion=1&format=html

20.6.6.多会话光盘

很少有 DVD-ROM 光驱支持多会话的 DVD，大多数时候只能读取第一个会话。DVD+R、DVD-R 和 DVD-RW 的递增顺序写入格式可以接受多个会话。对于 DVD+RW 和 DVD-RW 受限覆写格式，不存在多会话的概念。

在递增顺序写入格式的 DVD+R、DVD-R 或 DVD-RW 上的初始非封闭会话后，使用以下命令将在光盘上添加一个新的会话：

```
# growisofs -M /dev/cd0 -J -R /path/to/nextdata
```

在受限覆写模式下对 DVD+RW 或 DVD-RW 使用这个命令，会在将新的会话合并到现有会话的同时追加数据。其结果将是一张单节光盘。使用这种方法可以在这些类型的光盘上进行初始写入后添加数据。

> 注意
> 由于光盘上的一些空间在每个会话之间被用来标记会话的结束和开始，所以应该尽量使用大数据会话，减少会话数量以优化光盘空间。对于 DVD+R 来说，会话的数量被限制在 154 个，对于 DVD-R 来说大约是 2000 个，对于双层 DVD+R 来说是 127 个。

20.6.7.更多的信息

要获得关于 DVD 的更多信息，使用 `dvd+rw-mediainfo` */dev/cd0*，即指定驱动器中的光盘。

关于 `dvd+rw-tools` 的更多信息可以在 growisofs(1)[1220] 、dvd+rw-tools 网站[1221]以及 cdwrite 邮件列表[1222]存档中找到。

> 注意
> 当创建一个与 `dvd+rw-tools` 有关的问题报告时，请包括 `dvd+rw-mediainfo` 的输出。

20.6.8.使用 DVD-RAM

DVD-RAM 刻录机使用 SCSI 或 ATAPI 接口。对于 ATAPI 设备，必须通过在 **/boot/loader.conf** 中添加以下一行来启用 DMA 访问：

```
hw.ata.atapi_dma="1"
```

DVD-RAM 可以被看作是一个可移动的硬盘设备。像其他硬盘一样，在使用前必须被格式化 DVD-RAM。在这个例子中，整个磁盘空间将被格式化为标准的 UFS2 文件系统：

[1220] https://www.freebsd.org/cgi/man.cgi?query=growisofs&sektion=1&format=html
[1221] http://fy.chalmers.se/~appro/linux/DVD+RW/
[1222] http://lists.debian.org/cdwrite/

```
# dd if=/dev/zero of=/dev/acd0 bs=2k count=1
# bsdlabel -Bw acd0
# newfs /dev/acd0
```

必须根据配置改变 DVD 设备 **acd0** 的位置。

DVD-RAM 被格式化以后，它就可以像普通硬盘一样被挂载：

```
# mount /dev/acd0 /mnt
```

挂载后，DVD-RAM 将可读可写。

20.7.创建和使用软盘

这一节介绍了如何在 FreeBSD 中格式化 3.5 英寸软盘。

格式化软盘的步骤

软盘在使用前需要进行低级格式化。这通常是由供应商完成的，但格式化是检查软盘完整性的一个好方法。要在 FreeBSD 上对软盘进行低级格式化，可以使用 fdformat(1)[1223]。当使用这个工具时，请注意任何错误信息，因为这些信息可以帮助确定磁盘是好是坏。

1. 要格式化软盘，将一张新的 3.5 英寸软盘插入第一个软盘驱动器并使用：

   ```
   # /usr/sbin/fdformat -f 1440 /dev/fd0
   ```

2. 在低级格式化后，创建一个磁盘标签，因为系统需要它来确定磁盘的大小和它的 geometry。在 **/etc/disktab** 中列出了受支持的 geometry。
   要写入磁盘标签，请使用 bsdlabel(8)[1224]：

   ```
   # /sbin/bsdlabel -B -w /dev/fd0 fd1440
   ```

3. 现在，软盘已经准备好了，可以用文件系统进行高级格式化。软盘的文件系统可以是 UFS 或 FAT，一般选择 FAT。
   要用 FAT 格式化软盘，请执行：

   ```
   # /sbin/newfs_msdos /dev/fd0
   ```

现在，该磁盘已经可以使用了。要使用软盘，用 mount_msdosfs(8)[1225] 挂载它。我们还可以安装和使用 ports 中的emulators/mtools[1226]。

[1223] https://www.freebsd.org/cgi/man.cgi?query=fdformat&sektion=1&format=html
[1224] https://www.freebsd.org/cgi/man.cgi?query=bsdlabel&sektion=8&format=html
[1225] https://www.freebsd.org/cgi/man.cgi?query=mount_msdosfs&sektion=8&format=html
[1226] https://cgit.freebsd.org/ports/tree/emulators/mtools/pkg-descr

20.8.使用 NTFS 磁盘

这一节介绍了如何在 FreeBSD 中挂载 NTFS 磁盘。

NTFS（New Technology File System，新技术文件系统）是一个由 Microsoft® 开发的专有日志文件系统。多年来，它一直是 Microsoft Windows® 的默认文件系统。FreeBSD 可以使用 FUSE 文件系统挂载 NTFS 卷。这些文件系统是作为用户空间程序实现的，它通过一个定义好的接口与 fusefs(5)[1227] 内核模块交互。

挂载 **NTFS** 磁盘所需的步骤

1. 在使用 FUSE 文件系统之前，我们需要加载内核模块 fusefs(5)[1228]：

```
# kldload fusefs
```

使用 sysrc(8)[1229] 在开机时加载该模块：

```
# sysrc kld_list+=fusefs
```

2. 像示例那样从软件包中获得对 NTFS 文件系统的支持（参见 使用 pkg 进行二进制包管理[1230]）或 ports（参见使用 Ports[1231]）：

```
# pkg install fusefs-ntfs
```

3. 最后，我们需要创建一个用于挂载文件系统的目录：

```
# mkdir /mnt/usb
```

4. 假设插入了一个 USB 磁盘。可以用 gpart(8)[1232] 查看磁盘的分区信息：

```
# gpart show da0
=>       63  1953525105  da0 >MBR   (932G)
         63  1953525105    1 >ntfs  (932G)
```

5. 可以用下面的命令挂载该磁盘：

```
# ntfs-3g /dev/da0s1 /mnt/usb/
```

现在，已经可以使用该磁盘了。

6. 此外，可以在 **/etc/fstab** 中添加一个条目：

[1227] https://www.freebsd.org/cgi/man.cgi?query=fusefs&sektion=5&format=html
[1228] https://www.freebsd.org/cgi/man.cgi?query=fusefs&sektion=5&format=html
[1229] https://www.freebsd.org/cgi/man.cgi?query=sysrc&sektion=8&format=html
[1230] https://docs.freebsd.org/en/books/handbook/ports/index.html#pkgng-intro
[1231] https://docs.freebsd.org/en/books/handbook/ports/index.html#ports-using
[1232] https://www.freebsd.org/cgi/man.cgi?query=gpart&sektion=8&format=html

```
/dev/da0s1   /mnt/usb   ntfs mountprog=/usr/local/bin/ntfs-3g,noauto,rw   0 0
```

现在，现在可以挂载该磁盘了：

```
# mount /mnt/usb
```

8. 可以通过以下方式卸载磁盘：

```
# umount /mnt/usb/
```

20.9.备份的基础知识

为了有能力从磁盘故障、意外文件删除、随机文件损坏或机器完全毁坏（包括在线备份的毁坏）中恢复，实施备份计划是至关重要的。

备份类型和计划表将有所不同，这取决于数据的重要性、文件恢复所需的粒度以及可接受的停机时间长短。一些可采用的备份技术包括：

- 对整个系统的存档，备份到永久的、离线的介质上。这提供了对上述所有问题的保护，但恢复起来很慢而且不方便，特别是对非特权用户而言。
- 文件系统快照，这对恢复已删除的文件或以前的文件版本很有用。
- 整个文件系统或磁盘的副本，使用周期性的 net/rsync[1233] 与网络上的另一个系统进行同步。
- 硬件或软件 RAID，当一个磁盘发生故障时，可以最大限度地减少或避免停机。

通常情况下，各种备份技术要混合使用。例如，人们可以创建一个计划表，自动进行每周一次的全量备份，并将其离线存放，同时用每小时的 ZFS 快照来补充这一备份。此外，在进行文件编辑或删除之前，可以对个别目录或文件进行手动备份。

这一节介绍了一些可以用来创建和管理 FreeBSD 系统上的备份的工具。

20.9.1.备份文件系统

用于备份文件系统的传统 UNIX® 程序是 dump(8)[1234] 和 restore(8)[1235]，前者用于创建备份，后者用于恢复备份。这些工具在磁盘块级工作，低于文件系统所创建的文件、链接和目录的抽象。与其他备份软件不同，dump 备份的是整个文件系统，不能只备份某个文件系统的一部分或跨越多个文件系统的目录。dump 不写文件和目录，而是写构成文件和目录的原始数据块。

注意

[1233] https://cgit.freebsd.org/ports/tree/net/rsync/pkg-descr

[1234] https://www.freebsd.org/cgi/man.cgi?query=dump&sektion=8&format=html

[1235] https://www.freebsd.org/cgi/man.cgi?query=restore&sektion=8&format=html

如果在根目录上使用 dump，它将不会备份 **/home** 、**/usr** 或许多其他目录，因为这些目录通常是其他文件系统的挂载点或是这些文件系统的符号链接。

当用于恢复数据时，restore 默认在 **/tmp/** 中存储临时文件。当使用较小的 **/tmp** 做恢复盘时，请将 TMPDIR 设置为一个有更多空闲空间的目录，以便成功恢复。

当使用 dump 时，请注意它在 1975 年左右的第 6 版 AT&T UNIX® 中的一些问题仍然存在。默认参数假定备份到九轨磁带，而非其他类型的介质或今天可用的高密度磁带。必须在命令行中覆盖这些默认值。

可以通过网络将一个文件系统备份到另一个系统或连接到另一台计算机的磁带驱动器。虽然 rdump(8)[1236] 和 rrestore(8)[1237] 工具可以用于这个目的，但它们被认为是不安全的。

但是，我们可以通过 SSH 连接，以更安全的方式使用 dump 和 restore。这个例子创建了一个 **/usr** 的完整压缩备份，并通过 SSH 连接将备份文件发送到指定的主机上。

例 **35.** 通过 **SSH** 使用 **``dump``**

```
# /sbin/dump -0uan -f - /usr | gzip -2 | ssh -c blowfish
          \targetuser@targetmachine.example.com dd of=/mybigfiles/dump-usr-
↪l0.gz
```

这个例子设置了 RSH，以便通过 SSH 连接将备份写到远程系统的磁带机上：

例 **36.** 通过 **RSH** 设置 **ssh** 使用 **dump**

```
# env RSH=/usr/bin/ssh /sbin/dump -0uan -f targetuser@targetmachine.example.
↪com:/dev/sa0 /usr
```

20.9.2.备份目录

有几个内置的工具可用于根据需要备份和恢复指定的文件和目录。

tar(1)[1238] 是对一个目录中所有文件进行备份的好选择。这个工具可以追溯到第 6 版的 AT&T UNIX®，默认情况下，它假定递归备份到本地磁带设备。可以用参数来代替指定备份文件的名称。

这个例子创建了一个当前目录的压缩备份，并将其保存到 **/tmp/mybackup.tgz**。当创建一个备份文件时，要确保备份不保存在被备份的同一目录下：

例 **37.** 用 **``tar``** 备份当前目录

```
# tar czvf /tmp/mybackup.tgz .
```

要恢复整个备份，cd 进入要恢复的目录并指定备份的名称。注意，这将覆盖还原目录中任何较新版本的文件。如有疑问，可将其还原到一个临时目录或指定要还原的备份中的文件名称。

[1236] https://www.freebsd.org/cgi/man.cgi?query=rdump&sektion=8&format=html

[1237] https://www.freebsd.org/cgi/man.cgi?query=rrestore&sektion=8&format=html

[1238] https://www.freebsd.org/cgi/man.cgi?query=tar&sektion=1&format=html

例 **38.** 用 **``tar``** 恢复当前目录

```
# tar xzvf /tmp/mybackup.tgz
```

在 tar(1)[1239] 中有几十个可用的参数。这个工具还支持使用排除模式来指定在备份指定目录或从备份中恢复文件时不应该包括哪些文件。

要使用指定的文件和目录列表来创建备份，cpio(1)[1240] 是一个不错的选择。与 `tar` 不同，`cpio` 不知道如何进入目录树，必须提供要备份的文件列表给他。

例如，可以用 `ls` 或 `find` 创建一个文件列表。这个例子创建了一个当前目录的递归列表，然后被输送到 `cpio`，以便创建一个名为 **/tmp/mybackup.cpio** 的输出备份文件。

例 **39.** 使用 **``ls``** 和 **``cpio``** 对当前目录进行递归备份

```
# ls -R | cpio -ovF /tmp/mybackup.cpio
```

pax(1)[1241] 是一个试图结合 `tar` 和 `cpio` 所提供功能的备份工具。多年来，不同版本的 `tar` 和 `cpio` 变得有些不兼容。POSIX® 创造了 `pax`，它试图读写许多不同的 `cpio` 和 `tar` 格式，并加上自己的新格式。

在使用 `pax` 时前面的例子等效为：

例 **40.** 用 **``pax``** 备份当前目录

```
# pax -wf /tmp/mybackup.pax .
```

20.9.3.使用数据磁带进行备份

在磁带技术不断发展的同时，现代的备份系统倾向于将离线备份与本地可移动介质相结合。FreeBSD 支持所有使用 SCSI 的磁带机，如 LTO 或 DAT。对 SATA 和 USB 磁带机则有限支持。

对于 SCSI 磁带设备，FreeBSD 使用驱动程序 sa(4)[1242] 和设备 **/dev/sa0**, **/dev/nsa0**, 和 **/dev/esa0**。物理设备的名字是 **/dev/sa0**。当使用 **/dev/nsa0** 时，备份程序在写完一个文件后不会倒带，这就允许在一盘磁带上写多个文件。使用 **/dev/esa0** 会在设备关闭后弹出磁带。

在 FreeBSD 中，用 `mt` 来控制磁带机的操作，例如在磁带上寻找文件或向磁带上写入磁带控制标记。例如，在写一个新的文件之前，可以通过跳过它们来保留磁带上的前三个文件：

```
# mt -f /dev/nsa0 fsf 3
```

这个工具支持许多操作。详情请参考 mt(1)[1243]。

[1239] https://www.freebsd.org/cgi/man.cgi?query=tar&sektion=1&format=html
[1240] https://www.freebsd.org/cgi/man.cgi?query=cpio&sektion=1&format=html
[1241] https://www.freebsd.org/cgi/man.cgi?query=pax&sektion=1&format=html
[1242] https://www.freebsd.org/cgi/man.cgi?query=sa&sektion=4&format=html
[1243] https://www.freebsd.org/cgi/man.cgi?query=mt&sektion=1&format=html

要用 tar 写一个文件到磁带，指定磁带设备的名称和要备份的文件：

```
# tar cvf /dev/sa0 file
```

要从磁带上的 tar 存档中恢复文件到当前目录：

```
# tar xvf /dev/sa0
```

要备份一个 UFS 文件系统，使用 dump。这个例子备份了 **/usr**，完成后没有倒带：

```
# dump -0aL -b64 -f /dev/nsa0 /usr
```

以交互方式将文件从磁带上的转储文件恢复到当前目录：

```
# restore -i -f /dev/nsa0
```

20.9.4.第三方备份工具

FreeBSD ports 提供了许多第三方的工具，它们可以用来安排备份的创建，简化磁带备份，并使备份更容易和更方便。这些应用程序中有许多是基于客户端/服务器的，可以用来自动备份单个系统或网络中的所有计算机。

流行的工具包括 Amanda 、Bacula 、rsync 和 duplicity。

20.9.5.紧急恢复

除了定期备份外，建议执行以下步骤作为应急准备计划的一部分。

为以下命令的输出创建一个打印副本：

- gpart show
- more /etc/fstab
- dmesg

把这个打印结果和安装介质的副本保存在一个安全的地方。如果需要紧急恢复，请启动安装设备并选择 Live CD 来访问救援 shell。这个救援模式可以用来查看系统的当前状态，如果需要，可以重新格式化磁盘并从备份中恢复数据。

> 注意
>
> FreeBSD/i386 11.2-RELEASE 的安装镜像不包括救援 shell。对于此版本，请从以下地址下载并刻录 Livefs CD 镜像：ftp://ftp.FreeBSD.org/pub/FreeBSD/releases/i386/ISO-IMAGES/11.2/FreeBSD-11.2-RELEASE-i386-livefs.iso[1244].

[1244] ftp://ftp.freebsd.org/pub/FreeBSD/releases/i386/ISO-IMAGES/11.2/FreeBSD-11.2-RELEASE-i386-livefs.iso

接下来，测试救援 shell 和备份，并对该过程做记录。将这些笔记与设备、打印件和备份一起保存。这些记录可以防止在执行紧急恢复的压力下不小心破坏了备份。

为了增加安全性，将最新的备份存放在一个与计算机和磁盘驱动器相隔很远的异地。

20.10.内存盘

除了物理磁盘之外，FreeBSD 还支持创建和使用内存盘。内存盘的一个可能用途是访问 ISO 文件系统的内容，而无需先将其刻录再挂载 CD 或 DVD。

在 FreeBSD 中，驱动程序 md(4)[1245] 被用来提供对内存盘的支持。**GENERIC** 内核包括这个驱动。当使用定制内核配置文件时，请确保它包括这一行:

```
device md
```

20.10.1.附加和分离现有的镜像

要挂载一个现有的文件系统镜像，使用 `mdconfig` 来指定 ISO 文件的名称和一个空闲单元号。然后，参考该单元号将其挂载到一个现有的挂载点上。挂载成功后，ISO 中的文件将出现在挂载点中。这个例子将 *diskimage.iso* 附加到内存设备 **/dev/md0**，然后将该内存设备挂载到 **/mnt**：

```
# mdconfig -f diskimage.iso -u 0
# mount -t cd9660 /dev/md0 /mnt
```

注意，`-t cd9660` 被用来挂载 ISO 格式。如果没有用 `-u` 指定单元号，`mdconfig` 将自动分配一个未使用的内存设备，并输出分配单元的名称，如 `md4`。关于这个命令及其选项的更多细节，请参考 mdconfig(8)[1246]。

当一个内存盘不再被使用时，它的资源应该被释放回系统中。首先，卸载文件系统，然后使用 `mdconfig` 将磁盘从系统中分离出来并释放其资源。继续这个例子:

```
# umount /mnt
# mdconfig -d -u 0
```

要确定是否有任何内存盘仍然连接在系统上，键入 `mdconfig -l`。

[1245] https://www.freebsd.org/cgi/man.cgi?query=md&sektion=4&format=html

[1246] https://www.freebsd.org/cgi/man.cgi?query=mdconfig&sektion=8&format=html

20.10.2.创建一个以文件或内存为基础的内存盘

FreeBSD 也支持内存盘，其中使用的存储空间是从硬盘或内存区域中分配的。第一种方法通常被称为文件支持的文件系统，第二种方法被称为内存支持的文件系统。这两种类型都可以用 `mdconfig` 来创建。

要创建一个新的内存支持的文件系统，指定 `swap` 类型和要创建的内存盘的大小。然后，用文件系统格式化内存盘，像往常一样挂载。这个例子在单元 1 上创建了一个 5M 的内存盘。然后在挂载前用 UFS 文件系统格式化该内存盘:

```
# 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
```

要创建一个新的以文件为基础的内存盘，首先要分配一个磁盘区域来使用。这个例子创建了一个 5MB 的空文件，名为 **newimage** :

```
# dd if=/dev/zero of=newimage bs=1k count=5k
5120+0 records in
5120+0 records out
```

接下来，将该文件附加到一个内存盘上，给内存盘贴上标签，并用 UFS 文件系统对其进行格式化，挂载内存盘，并验证该文件支持的磁盘的大小:

```
# mdconfig -f newimage -u 0
# bsdlabel -w md0 auto
# newfs -U 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
```

使用 `mdconfig` 创建一个文件或内存支持的文件系统需要几个命令。FreeBSD 还配备了 `mdmfs`，可以

用它来自动配置一个内存盘，用 UFS 文件系统格式化，并将其挂载。例如，在用 dd 创建 **newimage** 之后，这条命令就相当于运行上面的 bsdlabel 、newfs 和 mount 命令:

```
# mdmfs -F newimage -s 5m md0 /mnt
```

要使用 mdmfs 创建一个新的基于内存的内存盘，请使用以下命令:

```
# mdmfs -s 5m md1 /mnt
```

如果没有指定单元号，mdmfs 将自动选择一个未使用的内存设备。关于 mdmfs 的更多细节，请参考 mdmfs(8)[1247]。

20.11.文件系统快照

FreeBSD 提供了一个与软更新[1248]相结合的功能：文件系统快照。

UFS 快照允许用户创建指定文件系统的镜像，并将其视为一个文件。必须在执行该操作的文件系统中创建快照文件，并且用户可以在每个文件系统中创建不超过 20 个快照。活动快照被记录在超级区块中，因此它们在卸载和挂载操作以及系统重启时都是不变的。当快照不再需要时，可以用 rm(1)[1249] 将其删除。虽然可以以任何顺序移除快照，但可能不会完全得到使用过的空间，因为另一个快照可能会需要这些被释放的块。

不可更改的快照文件标志是由 mksnap_ffs(8)[1250] 在最初创建快照文件后设置的。unlink(1)[1251] 对快照文件做了例外处理，因为它允许它们被删除。

可通过 mount(8)[1252] 创建快照。要把 **/var** 的快照放在 **/var/snapshot/snap** 文件下，使用下面的命令:

```
# mount -u -o snapshot /var/snapshot/snap /var
```

或者使用 mksnap_ffs(8)[1253] 来创建快照:

```
# mksnap_ffs /var /var/snapshot/snap
```

可以使用 find(1)[1254] 在文件系统上查找快照文件，例如 **/var**:

```
# find /var -flags snapshot
```

[1247] https://www.freebsd.org/cgi/man.cgi?query=mdmfs&sektion=8&format=html
[1248] https://docs.freebsd.org/en/books/handbook/config/index.html#soft-updates
[1249] https://www.freebsd.org/cgi/man.cgi?query=rm&sektion=1&format=html
[1250] https://www.freebsd.org/cgi/man.cgi?query=mksnap_ffs&sektion=8&format=html
[1251] https://www.freebsd.org/cgi/man.cgi?query=unlink&sektion=1&format=html
[1252] https://www.freebsd.org/cgi/man.cgi?query=mount&sektion=8&format=html
[1253] https://www.freebsd.org/cgi/man.cgi?query=mksnap_ffs&sektion=8&format=html
[1254] https://www.freebsd.org/cgi/man.cgi?query=find&sektion=1&format=html

所创建的快照有几种用途：

- 一些管理员会将快照文件用于备份目的，因为快照可以被传输到 CD 或磁带。
- 可以在快照上运行文件系统完整性检查器 fsck(8)[1255]。假设文件系统在被挂载时是干净的，这应该总是提供一个一致的 clean 结果。
- 在快照上运行 dump(8)[1256] 会产生一个与文件系统和快照的时间戳一致的 dump 文件。dump(8)[1257] 也可以使用 `-L` 来获取快照，创建 dump 镜像，然后删除快照。
- 可以作为文件系统的冻结镜像挂载快照。要挂载快照 **/var/snapshot/snap**，请运行:

```
# mdconfig -a -t vnode -o readonly -f /var/snapshot/snap -u 4
# mount -r /dev/md4 /mnt
```

可以通过 **/mnt** 使用被冻结的 **/var**。所有文件最初都将处于快照创建时的状态。唯一的例外是，任何更早的快照将显示为大小为 0 的文件。要解除对快照的挂载，请使用:

```
# umount /mnt
# mdconfig -d -u 4
```

有关软更新和文件系统快照的更多信息，包括技术报告，请访问 Marshall Kirk McKusick 的网站：http://www.mckusick.com/。

20.12.磁盘配额

磁盘配额可以用来限制用户或组的成员在每个文件系统上可以分配的磁盘空间或文件数量。这可以防止某个用户或某组用户消耗掉所有的可用磁盘空间。

本节介绍了如何为 UFS 文件系统配置磁盘配额。要在 ZFS 文件系统上配置配额，请参阅数据集、用户和组配额[1258]

20.12.1.启用磁盘配额

要确定 FreeBSD 内核是否提供对磁盘配额的支持，使用：

```
% sysctl kern.features.ufs_quota
kern.features.ufs_quota: 1
```

[1255] https://www.freebsd.org/cgi/man.cgi?query=fsck&sektion=8&format=html
[1256] https://www.freebsd.org/cgi/man.cgi?query=dump&sektion=8&format=html
[1257] https://www.freebsd.org/cgi/man.cgi?query=dump&sektion=8&format=html
[1258] https://docs.freebsd.org/en/books/handbook/zfs/index.html#zfs-zfs-quota

在这个例子中，`1` 表示支持配额。如果这个值是 `0`，请在定制内核配置文件中加入以下一行，并按照配置 FreeBSD 内核[1259]的说明重建内核:

```
options QUOTA
```

接下来，在 **/etc/rc.conf** 中启用磁盘配额:

```
quota_enable="YES"
```

通常在启动时，每个文件系统的配额完整性由 quotacheck(8)[1260] 检查。这个程序确保配额数据库中的数据正确反映了文件系统上的数据。这是一个耗时的过程，会大大影响系统的启动时间。要跳过这个步骤，请在 **/etc/rc.conf** 中添加这个变量:

```
check_quotas="NO"
```

最后，编辑 **/etc/fstab** 以在每个文件系统上启用磁盘配额。要在某个文件系统上启用每个用户的配额，在文件系统的 **/etc/fstab** 条目中的选项字段中添加 `userquota` 即可，以启用配额。例如:

```
/dev/da1s2g   /home    ufs rw,userquota 1 2
```

要启用组配额，请使用 `groupquota` 代替。要同时启用用户和组配额，请用逗号隔开选项:

```
/dev/da1s2g    /home    ufs rw,userquota,groupquota 1 2
```

默认情况下，配额文件以 **quota.user** 和 **quota.group** 的形式存储在文件系统的根目录下。请参考 fstab(5)[1261] 以了解更多信息。不建议为配额文件指定其他的位置。

配置完成后，重新启动系统，**/etc/rc** 将自动运行适当的命令，为 **/etc/fstab** 中启用的所有配额创建初始配额文件。

在正常的操作过程中，应该无需手动运行 quotacheck(8)[1262]、quoton(8)[1263] 或 quotaoff(8)[1264]。不过，应该阅读这些手册页面以熟悉它们的操作。

[1259] https://docs.freebsd.org/en/books/handbook/kernelconfig/index.html#kernelconfig
[1260] https://www.freebsd.org/cgi/man.cgi?query=quotacheck&sektion=8&format=html
[1261] https://www.freebsd.org/cgi/man.cgi?query=fstab&sektion=5&format=html
[1262] https://www.freebsd.org/cgi/man.cgi?query=quotacheck&sektion=8&format=html
[1263] https://www.freebsd.org/cgi/man.cgi?query=quotaon&sektion=8&format=html
[1264] https://www.freebsd.org/cgi/man.cgi?query=quotaoff&sektion=8&format=html

20.12.2.设置配额限制

要验证配额是否已启用，请运行：

```
# quota -v
```

应该有一行关于磁盘使用情况和每个文件系统的当前配额限制的摘要，这些文件系统已启用配额。

现在，系统已经准备好用 edquota 来分配配额限制。

有几个选项可用于强制限制用户或组可以分配的磁盘空间，以及他们可以创建多少文件。可以根据磁盘空间（块配额）、文件数量（节点配额）或两者的组合来限制分配。每个限制进一步细分为两类：硬限制和软限制。

硬限制是不能超过的。如果有一个用户达到了硬限制，该用户就不能再在该文件系统上进行分配。例如，如果用户在一个文件系统上的硬限制是 500 kb，目前正在使用 490 kb，用户只能再分配 10 kb。若试图分配额外的 11kb 将会失败。

软限制可以在有限的时间内被超过，称为宽限期，默认情况下是一个星期。如果一个用户超过他们的限制超过宽限期，软限制就变成了硬限制，不允许进一步分配。当用户回落到软限制以下时，宽限期被重置。

在下面的例子中，正在编辑测试账户的配额。当调用 edquota 时，将打开由 EDITOR 指定的编辑器，以便编辑配额限制。默认编辑器被设置为 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)
```

每个启用了配额的文件系统通常有两行。一行代表块的限制，另一行代表节点的限制。改变数值来修改配额限制。例如，要把 **/usr** 的块限制提高到 500 的软限制和 600 的硬限制，改变该行的值如下：

```
/usr: kbytes in use: 65, limits (soft = 500, hard = 600)
```

新的配额限制在退出编辑器时生效。

有时，我们希望对一定范围内的用户设置配额限制。这可以通过首先为一个用户分配所需的配额限制来实现。然后，使用 -p 将该配额复制到指定范围的用户 ID（UID）。下面的命令将为 UID 10,000 到 19,999 复制这些配额限制：

```
# edquota -p test 10000-19999
```

更多信息，请参考 edquota(8)[1265]。

[1265] https://www.freebsd.org/cgi/man.cgi?query=edquota&sektion=8&format=html

20.12.3.检查配额限制和磁盘使用情况

要检查单个用户或组的配额和磁盘使用情况，请使用 quota(1)[1266]。一个用户只能检查他自己的配额和他所属的组的配额。只有超级用户可以查看所有用户和组的配额。要获得启用了配额的文件系统的所有配额和磁盘使用情况的摘要，请使用 repquota(8)[1267]。

通常，用户没有使用过的任何磁盘空间中的文件系统都不会显示在 `quota` 的输出中，即使用户为该文件系统分配了配额限制。使用 `-v` 来显示这些文件系统。下面是 `quota -v` 的输出示例，该用户在两个文件系统上有配额限制:

```
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
```

在这个例子中，用户目前在 **/usr** 上的软限制为 50kb，超过了 15kb，并且还有 5 天的宽限期。星号 * 表示该用户目前超过了配额限制

20.12.4.NFS 上的配额

配额是由 NFS 服务器上的配额子系统执行的。rpc.rquotad(8)[1268] 守护进程上的配额信息对 NFS 客户端可用，并允许这些机器上的用户查看他们的配额统计信息。

在 NFS 服务器上，通过删除 **/etc/inetd.conf** 中这一行的 `#` 来启用 `rpc.rquotad`:

```
rquotad/1      dgram rpc/udp wait root /usr/libexec/rpc.rquotad rpc.rquotad
```

然后，重新启动 `inetd`:

```
# service inetd restart
```

20.13.加密磁盘分区

FreeBSD 提供了出色的在线保护，以防止未经授权的数据访问。文件权限和强制访问控制[1269]（MAC）有助于防止未经授权的用户在操作系统运行和计算机开机时访问数据。然而，如果攻击者拥有对计算机的物理访问权限，并且可以将计算机的硬盘移动到另一个系统中去复制和分析数据，那么操作系统所强制执行的权限就没有意义了。

[1266] https://www.freebsd.org/cgi/man.cgi?query=quota&sektion=1&format=html
[1267] https://www.freebsd.org/cgi/man.cgi?query=repquota&sektion=8&format=html
[1268] https://www.freebsd.org/cgi/man.cgi?query=rpc.rquotad&sektion=8&format=html
[1269] https://docs.freebsd.org/en/books/handbook/mac/index.html#mac

无论攻击者是如何占有硬盘或断电的计算机，FreeBSD 内置的基于 GEOM 的加密子系统都能够保护计算机文件系统中的数据，即使是面对拥有大量资源的高度活跃的攻击者。与加密单个文件的方法不同，可以用内置的工具 `gbde` 和 `geli` 来透明地加密整个文件系统。任何明文数据都不会被存储到硬盘上。

本章演示了如何在 FreeBSD 上创建一个加密的文件系统。它首先演示了使用 `gbde` 的过程，然后用 `geli` 对同一个例子进行示范。

20.13.1.用 gbde 进行磁盘加密

工具 gbde(4)[1270] 为攻击者获取冷存储设备的内容提供一个巨大的挑战。然而，如果计算机在运行时被破坏，并且存储设备被主动连接，或者攻击者能够获得有效的口令，那么它就不能为存储设备的内容提供保护。因此，在系统运行时提供物理安全并保护加密机制所使用的口令是很重要的。

该工具提供了几个屏障来保护存储在每个磁盘扇区的数据。它使用 CBC 模式下的 128 位 AES 对磁盘扇区的内容进行加密。磁盘上的每个扇区都用不同的 AES 密钥进行加密。关于加密设计的更多信息，以及如何从用户提供的口令中获得扇区密钥，请参考 gbde(4)[1271]。

FreeBSD 为 gbde 提供了一个内核模块，可以用这个命令加载：

```
# kldload geom_bde
```

如果使用定制内核配置文件，确保它包含这一行：

```
options GEOM_BDE
```

下面的例子演示了在系统中添加一个新的硬盘，该硬盘将存放一个加密的分区，该分区将被挂载为 **/private**。

用 gbde 加密一个分区的步骤

1. 添加新的硬盘驱动器

按照添加硬盘[1272]中的说明，将新硬盘安装到系统中。在这个例子中，一个新的硬盘分区被添加为 **/dev/ad4s1c**。**/dev/ad0s1*** 代表现有的标准 FreeBSD 分区。

```
# 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
```

1. 创建一个存放 `gbde` Lock 文件的目录

```
# mkdir /etc/gbde
```

[1270] https://www.freebsd.org/cgi/man.cgi?query=gbde&sektion=4&format=html

[1271] https://www.freebsd.org/cgi/man.cgi?query=gbde&sektion=4&format=html

[1272] https://docs.freebsd.org/en/books/handbook/book/#disks-adding

gbde Lock 文件包含 gbde 访问加密分区所需的信息。如果不能访问 Lock 文件，在没有大量人工干预的情况下，gbde 将不能解密加密分区中包含的数据，而软件不支持人工干预。每个加密分区都使用一个单独的 Lock 文件。

1. 初始化 gbde 分区

一个 gbde 分区在被使用之前必须被初始化。该初始化只需要执行一次。该命令将打开默认编辑器，以便在一个模板中设置各种配置选项。为了与 UFS 文件系统一起使用，将扇区大小设置为 2048。

```
# gbde init /dev/ad4s1c -i -L /etc/gbde/ad4s1c.lock
# $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
[...]
```

编辑保存之后，用户将被要求两次输入用于保护数据的口令。这两次的口令必须是相同的。gbde 保护数据的能力完全取决于口令的质量。关于如何选择一个容易记住的安全口令的提示，见 http://world.std.com/~reinhold/diceware.htm。

该初始化为 gbde 分区创建一个 Lock 文件。在这个例子中，它被存储为 **/etc/gbde/ad4s1c.lock**。Lock 文件必须以 `.lock` 结尾，以便被 **/etc/rc.d/gbde** 启动脚本正确检测。

> 当心
>
> Lock 文件必须与任何加密分区的内容一起进行备份。如果没有 Lock 文件，合法所有者也将无法访问加密分区上的数据。

1. 将加密的分区附加到内核上

```
# gbde attach /dev/ad4s1c -l /etc/gbde/ad4s1c.lock
```

该命令将提示输入初始化加密分区时选择的口令。新的加密设备将作为 **/dev/device_name.bde** 出现在 **/dev** 中：

```
# 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
```

1. 在加密的设备上创建一个文件系统

加密的设备被连接到内核后，就可以在设备上创建一个文件系统。这个例子创建了一个 UFS 文件系统，启用了软更新。请确保指定具有 ***.bde** 扩展名的分区：

```
# newfs -U /dev/ad4s1c.bde
```

1. 挂载加密的分区

   创建一个挂载点并挂载加密的文件系统：

```
# mkdir /private
# mount /dev/ad4s1c.bde /private
```

1. 验证加密的文件系统是否可用

   加密的文件系统现在应该是可见的并且可供使用：

```
% 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
```

每次启动后，任何加密的文件系统都必须手动重新连接到内核，检查是否有错误，并挂载，然后才能使用这些文件系统。要配置这些步骤，请在 **/etc/rc.conf** 中添加以下几行：

```
gbde_autoattach_all="YES"
gbde_devices="ad4s1c"
gbde_lockdir="/etc/gbde"
```

这需要在启动时在控制台输入密码。输入正确的口令后，加密的分区将被自动挂载。其他 gbde 启动选项也是可用的，在 rc.conf(5)[1273] 中列出。

注意

sysinstall 与 gbde-encrypted 设备不兼容。在启动 sysinstall 之前，所有 ***.bde** 设备必须从内核中分离出来，否则它在最初探测设备时将崩溃。要分离本例中使用的加密设备，使用下面的命令：

```
# gbde detach /dev/ad4s1c
```

[1273] https://www.freebsd.org/cgi/man.cgi?query=rc.conf&sektion=5&format=html

20.13.2.用 geli 进行磁盘加密

使用 geli 可以获得另一种 GEOM 加密级别。这个控制工具增加了一些功能，并使用一个不同的方案来做加密工作。它提供了以下功能：

- 利用 crypto(9)[1274] 框架，在有加密硬件时自动使用。
- 支持多种加密算法，如 AES、Blowfish 和 3DES。
- 允许对根分区进行加密。在系统启动时，将要求提供用于访问加密的根分区的口令。
- 允许使用两个独立的密钥。
- 它是快速的，因为它执行简单的扇区到扇区的加密。
- 允许备份和恢复主密钥。如果用户破坏了他们的密钥，仍有可能通过从备份中恢复密钥来获取数据。
- 允许磁盘附加一个随机的、一次性的密钥，这对交换分区和临时文件系统很有用。

更多的功能和使用实例可以在 geli(8)[1275] 中找到。

下面的例子示范了如何生成一个密钥文件，该文件将被用作安装在 **/private** 下的加密 provider 的主密钥的一部分。密钥文件将提供一些用于加密主密钥的随机数据。主密钥也将受到口令的保护。provider 的扇区大小将是 4kB。这个例子描述了如何附加到 geli provider 上，在它上面创建一个文件系统，挂载它，使用它，最后是如何分离它

用 geli 加密一个分区的步骤

1. 加载 geli

对 geli 的支持是作为一个可加载的内核模块提供的。要配置系统在启动时自动加载该模块，请在 **/boot/loader.conf** 中添加以下一行：

```
geom_eli_load="YES"
```

要立即加载这个内核模块：

```
# kldload geom_eli
```

对于一个定制内核，确保内核配置文件包含这些行：

```
options GEOM_ELI
device crypto
```

1. 生成主密钥

下面的命令生成了一个主密钥，所有的数据都将用这个密钥进行加密。这个密钥永远不能被改变。与其直接使用它，不如用一个或多个用户密钥进行加密。用户密钥是由文件 **/root/da2.key** 中的随机字节和/或口令的可选组合组成的。在这种情况下，密钥文件的数据源是 **/dev/random**。该命令还将 provider（**/dev/da2.eli**）的扇区大小配置为 4kB，以获得更好的性能：

[1274] https://www.freebsd.org/cgi/man.cgi?query=crypto&sektion=9&format=html

[1275] https://www.freebsd.org/cgi/man.cgi?query=geli&sektion=8&format=html

```
# dd if=/dev/random of=/root/da2.key bs=64 count=1
# geli init -K /root/da2.key -s 4096 /dev/da2
Enter new passphrase:
Reenter new passphrase:
```

并非一定要同时使用口令和密钥文件，因为任何一种保护主密钥的方法都可以单独使用。

如果钥匙文件以“-”的形式给出，将使用标准输入。例如，这个命令产生了三个密钥文件：

```
# cat keyfile1 keyfile2 keyfile3 | geli init -K - /dev/da2
```

1. 用生成的密钥连接 provider

要连接 provider，指定密钥文件、磁盘名称和口令：

```
# geli attach -k /root/da2.key /dev/da2
Enter passphrase:
```

这将创建一个带有 **.eli** 扩展名的新设备：

```
# ls /dev/da2*
/dev/da2  /dev/da2.eli
```

1. 创建新的文件系统

接下来，用 UFS 文件系统格式化该设备，并将其挂载到一个现有的挂载点：

```
## dd if=/dev/random of=/dev/>da2.eli bs=1m

## newfs /dev/da2.eli

## mount /dev/da2.eli /private
```

加密的文件系统现在应该可以使用了：

```
# 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
```

加密分区的工作完成后，如果不再需要 **/private** 分区，谨慎的做法是让设备进入冷存储，将 geli 加密分区从内核中卸载并分离出来：

```
# umount /private
# geli detach da2.eli
```

我们提供了一个 **rc.d** 脚本来简化启动时挂载 `geli` 加密设备的过程。对于这个例子，在 **/etc/rc.conf** 中添加这些行：

```
geli_devices="da2"
geli_da2_flags="-k /root/da2.key"
```

这就把 **/dev/da2** 配置为一个 `geli` provider，主密钥为 **/root/da2.key**。在系统关闭之前，系统将自动从内核中分离出 provider。在启动过程中，脚本会在连接 provider 之前提示输入口令。在密码提示前后可能会出现其他内核信息。如果启动过程似乎停滞不前，仔细寻找其他信息中的密码提示。只要输入了正确的口令，provider 就会被连接。然后，文件系统被挂载，通常是通过 **/etc/fstab** 中的一个条目。关于如何配置文件系统在启动时挂载，请参考“挂载和卸载文件系统[1276]”的说明。

20.14.加密交换分区

与加密磁盘分区一样，加密交换空间也是用于保护敏感信息。假想有一个处理密码的应用程序，只要这些密码停留在物理内存中，它们就不会被写入磁盘，并且在重启后会被清除掉。然而，如果 FreeBSD 开始交换内存页以释放空间，密码可能会被未加密地写入磁盘。加密交换空间可以作为这种情况的解决方案。

本节示范了如何使用 gbde(8)[1277] 或 geli(8)[1278] 加密工具配置一个加密的交换分区。它假定 **/dev/ada0s1b** 是交换分区。

20.14.1.配置交换分区加密

在默认情况下是交换分区是不进行加密的，在继续之前应该清除任何敏感数据。要用随机的垃圾覆盖当前的交换分区，请执行以下命令：

```
# dd if=/dev/random of=/dev/ada0s1b bs=1m
```

要使用 gbde(8)[1279] 对交换分区进行加密，为 **/etc/fstab** 中的交换分区添加 `.bde` 后缀：

```
# Device            Mountpoint  FStype  Options     Dump    Pass#
/dev/ada0s1b.bde    none        swap    sw          0       0
```

要使用 geli(8)[1280] 加密交换分区，请使用 `.eli` 后缀：

[1276] https://docs.freebsd.org/en/books/handbook/basics/index.html#mount-unmount
[1277] https://www.freebsd.org/cgi/man.cgi?query=gbde&sektion=8&format=html
[1278] https://www.freebsd.org/cgi/man.cgi?query=geli&sektion=8&format=html
[1279] https://www.freebsd.org/cgi/man.cgi?query=gbde&sektion=8&format=html
[1280] https://www.freebsd.org/cgi/man.cgi?query=geli&sektion=8&format=html

```
# Device        Mountpoint  FStype  Options     Dump    Pass#
/dev/ada0s1b.eli    none        swap    sw      0   0
```

默认情况下，geli(8)[1281] 使用 AES 算法，密钥长度为 128 位。通常情况下，默认设置就足够了。如果需要，这些默认值可以在 **/etc/fstab** 的 `options` 字段中进行修改。可用的参数是：

aalgo

数据完整性验证算法，用于确保加密的数据没有被篡改。支持的算法列表见 geli(8)[1282]。

ealgo

用来保护数据的加密算法。参见 geli(8)[1283] 获取支持的算法列表。

keylen

用于加密算法的密钥的长度。关于每种加密算法所支持的密钥长度，见 geli(8)[1284]。

sectorsize

数据在加密前被分割成的块的大小。较大的扇区大小可以提高性能，但代价是较高的存储开销。推荐的大小是 4096 字节。

这个例子使用 Blowfish 算法配置了一个加密的交换分区，密钥长度为 128 位，扇区大小为 4 kb：

```
# Device        Mountpoint  FStype  Options             Dump    Pass#
/dev/ada0s1b.eli    none        swap    sw,ealgo=blowfish,keylen=128,sectorsize=4096 ␣
↪  0   0
```

20.14.2.校验加密交换空间

系统重新启动之后，可以使用 `swapinfo` 来验证加密的交换空间是否正常运行。

如果使用 gbde(8)[1285]：

```
% swapinfo
Device          1K-blocks     Used    Avail Capacity
/dev/ada0s1b.bde   542720        0   542720     0
```

如果使用 geli(8)[1286]：

[1281] https://www.freebsd.org/cgi/man.cgi?query=geli&sektion=8&format=html
[1282] https://www.freebsd.org/cgi/man.cgi?query=geli&sektion=8&format=html
[1283] https://www.freebsd.org/cgi/man.cgi?query=geli&sektion=8&format=html
[1284] https://www.freebsd.org/cgi/man.cgi?query=geli&sektion=8&format=html
[1285] https://www.freebsd.org/cgi/man.cgi?query=gbde&sektion=8&format=html
[1286] https://www.freebsd.org/cgi/man.cgi?query=geli&sektion=8&format=html

```
% swapinfo
Device          1K-blocks     Used    Avail Capacity
/dev/ada0s1b.eli    542720        0   542720     0
```

20.15.高可用性存储（HAST）

高可用性是商业应用的主要要求之一，而高可用性存储是这种环境下的一个关键组成部分。在 FreeBSD 中，高可用性存储（HAST）框架允许在由 TCP/IP 网络连接的几台物理上分离的机器上透明地存储相同的数据。HAST 可以被理解为基于网络的 RAID1（镜像），并且类似于 GNU/Linux® 平台中使用的 DRBD® 存储系统。与 FreeBSD 的其他高可用性功能（如 CARP）相结合，HAST 使得建立一个高可用性的存储集群成为可能，它可以抵抗硬件故障的发生。

以下是 HAST 的主要特点：

- 可以用来掩盖本地硬盘上的 I/O 错误。
- 文件系统无关，因为它可以与 FreeBSD 支持的任何文件系统一起工作。
- 高效和快速的重新同步机制，只有在一个节点停机期间被修改的块才会被同步。
- 可以在已经部署的环境中使用，以增加额外的冗余。
- 与 CARP 、Heartbeat 或其他工具一起，它可以用来建立一个健壮而可靠的存储系统。

读完本节后，你会了解：

- 什么是 HAST，它是如何工作的，以及它提供哪些功能。
- 如何在 FreeBSD 上设置和使用 HAST。
- 如何整合 CARP 和 devd(8)[1287] 来建立一个强大的存储系统。

在阅读本节之前，你应该：

- 了解 UNIX® 和 FreeBSD 的基础知识 (FreeBSD 基础[1288])。
- 知道如何配置网络接口和其他核心的 FreeBSD 子系统（配置与优化[1289]）。
- 对 FreeBSD 网络有一个很好的理解（网络通信[1290]）

HAST 项目由 FreeBSD 基金会赞助，并得到了 http://www.omc.net/ 和 http://www.transip.nl/ 的支持。

[1287] https://www.freebsd.org/cgi/man.cgi?query=devd&sektion=8&format=html
[1288] https://docs.freebsd.org/en/books/handbook/basics/index.html#basics
[1289] https://docs.freebsd.org/en/books/handbook/config/index.html#config-tuning
[1290] https://docs.freebsd.org/en/books/handbook/partiv/index.html#network-communication

20.15.1. HAST 操作

HAST 在两台物理机之间提供同步的块级复制：*主*（*primary*）节点和 *次*（*secondary*）节点。这两台机器一起被称为一个集群。

由于 HAST 在主—次配置中工作，它只允许集群节点中的一个在任何特定时间处于活动状态。主节点，也叫 *活动*节点，是一个将处理所有 I/O 请求的 HAST 管理的设备。次节点是自动从主要节点同步的。

HAST 系统的物理组件是主节点上的本地磁盘，以及远程的次要节点上的磁盘。

HAST 在块级上同步运行，使它对文件系统和应用程序透明。HAST 在 **/dev/hast/** 中提供了标准的 GEOM 工具，供其他工具或应用程序使用。使用 HAST 提供的设备和普通磁盘或分区之间没有区别。

每个写入、删除或刷新操作都通过 TCP/IP 发送到本地磁盘和远程磁盘。每个读操作都是从本地磁盘提供的，除非本地磁盘不是最新的或发生 I/O 错误。在这种情况下，读取操作被发送到辅助节点。

HAST 试图提供快速的故障恢复。出于这个原因，在一个节点的故障后减少同步时间是很重要的。为了提供快速的同步，HAST 管理一个磁盘上的脏区映射表，并且只在常规的同步过程中同步这些表，但初始同步除外。

有许多方法来处理同步问题。HAST 实现了几种复制模式来处理不同的同步方法：

- *memsync*：这种模式下，当本地写操作完成，远程节点确认数据到达时，但在实际存储数据之前，报告写操作完成。远程节点上的数据将在发送确认后直接存储。这种模式旨在减少延迟，但仍然提供良好的可靠性。这种模式是默认的。
- *fullsync*：当本地写和远程写都完成时，这种模式报告写操作已经完成。这是最安全和最慢的复制模式。
- *async*：这种模式在本地写操作完成后报告为完成。这是最快速和最危险的复制模式。它只应该在复制到远方节点时使用，因为其他模式的延迟太高。

20.15.2.配置 HAST

HAST 框架由几个部分组成:

- hastd(8)[1291] 守护进程，提供数据同步。当这个守护进程被启动时，它将自动加载`geom_gate.ko`。
- 用户区管理工具——hastctl(8)[1292]。
- 配置文件 hast.conf(5)[1293]。这个文件在启动 HAST 之前必须存在。

如果用户希望在内核中静态地编译 `GEOM_GATE` 支持，则应在定制内核配置文件中加入这一行，然后按照配置 FreeBSD 内核[1294] 的说明重建内核:

```
options GEOM_GATE
```

[1291] https://www.freebsd.org/cgi/man.cgi?query=hastd&sektion=8&format=html
[1292] https://www.freebsd.org/cgi/man.cgi?query=hastctl&sektion=8&format=html
[1293] https://www.freebsd.org/cgi/man.cgi?query=hast.conf&sektion=5&format=html
[1294] https://docs.freebsd.org/en/books/handbook/kernelconfig/index.html#kernelconfig

下面的例子示范了如何在主-次操作中配置两个节点，使用 HAST 在两者之间复制数据。一个节点将被称为 hasta，其 IP 地址为 172.16.0.1；另一个是 hastb，其 IP 地址为 172.16.0.2。两个节点将有一个相同大小的专用硬盘 **/dev/ad6** 用于 HAST 操作。HAST 池，有时被称为资源或 **/dev/hast/** 中的 GEOM 设备，将被称为 test。

HAST 的配置是使用 **/etc/hast.conf** 完成的。这个文件在两个节点上应该是相同的。最简单的配置是：

```
resource test {
    on hasta {
        local /dev/ad6
        remote 172.16.0.2
    }
    on hastb {
        local /dev/ad6
        remote 172.16.0.1
    }
}
```

更多高级配置请参考 hast.conf(5)[1295]。

> 技巧
>
> 如果在 **/etc/hosts** 或本地 DNS 中定义的主机是可解析的，也可以在远程语句中使用主机名。

如果配置存在于两个节点上，就可以创建 HAST 池。在两个节点上运行这些命令，将初始元数据放到本地磁盘上并启动 hastd(8)[1296]：

```
# hastctl create test
# service hastd onestart
```

> 注意
>
> 不可能使用已经包含文件系统的 GEOM 设备来创建存储池，也不可能将现有的存储转换为 HAST 管理的池。这个程序需要在设备上存储一些元数据，而在现有的设备上没有足够的所需空间可用。

HAST 节点的主或次角色是由管理员或像 Heartbeat 这样的工具通过 hastctl(8)[1297] 来选择的。

在主节点 hasta 上执行这个命令：

```
# hastctl role primary test
```

在次节点 hastb 上执行这个命令：

[1295] https://www.freebsd.org/cgi/man.cgi?query=hast.conf&sektion=5&format=html
[1296] https://www.freebsd.org/cgi/man.cgi?query=hastd&sektion=8&format=html
[1297] https://www.freebsd.org/cgi/man.cgi?query=hastctl&sektion=8&format=html

```
# hastctl role secondary test
```

通过在每个节点上运行 `hastctl` 来验证结果：

```
# hastctl status test
```

检查输出中的 `status` 行。如果它显示 `degraded`，说明配置文件出了问题。它应该在每个节点上显示 `complete`，即意味着节点之间的同步已经开始。当 `hastctl status` 报告说有 0 byte 的 脏区映射时，同步就完成了。

下一步是在 GEOM 设备上创建一个文件系统并挂载它。这必须在主节点上完成。创建文件系统可能需要几分钟时间，这取决于硬盘的大小。这个例子在 **/dev/hast/test** 上创建了一个 UFS 文件系统:

```
# newfs -U /dev/hast/test
# mkdir /hast/test
# mount /dev/hast/test /hast/test
```

如果 HAST 框架被正确配置，最后一步是确保 HAST 在系统启动时自动启动。添加这一行到 **/etc/rc.conf** 即可：

```
hastd_enable="YES"
```

20.15.2.1.故障转移配置

这个例子的目标是建立一个强大的存储系统，它可以抵御任何特定节点的故障。如果主节点发生故障，副节点就会被无缝地接管，检查和挂载文件系统，并继续工作，不会丢失任何一点数据。

为了完成这项任务，共用地址冗余协议（CARP）被用来在 IP 层提供自动故障转移。CARP 允许同一网段的多个主机共享一个 IP 地址。根据 “共用地址冗余协议（CARP）” [1298] 中提供的文档，在集群的两个节点上设置 CARP。在这个例子中，每个节点将有自己的管理 IP 地址和 `172.16.0.254` 的共享 IP 地址。集群的主要 HAST 节点必须是主要的 CARP 节点。

在上一节中创建的 HAST 池现在已经准备好被导出到网络上的其他主机。这可以通过 NFS 或 Samba 导出，使用共享 IP 地址 `172.16.0.254` 来完成。唯一没有解决的问题是在主节点发生故障时的自动故障转移。

在 CARP 接口开启或关闭时，FreeBSD 操作系统会生成 devd(8)[1299] 事件，从而可以观察 CARP 接口的状态变化。CARP 接口上的状态变化表明其中一个节点失败或重新上线了。这些状态变化事件使运行一个脚本成为可能，该脚本将自动处理 HAST 故障切换。

为了捕捉 CARP 接口上的状态变化，在每个节点上的 **/etc/devd.conf** 中添加这个配置：

[1298] https://docs.freebsd.org/en/books/handbook/advanced-networking/index.html#carp

[1299] https://www.freebsd.org/cgi/man.cgi?query=devd&sektion=8&format=html

```
notify 30 {
    match "system" "IFNET";
    match "subsystem" "carp0";
    match "type" "LINK_UP";
    action "/usr/local/sbin/carp-hast-switch primary";
};

notify 30 {
    match "system" "IFNET";
    match "subsystem" "carp0";
    match "type" "LINK_DOWN";
    action "/usr/local/sbin/carp-hast-switch secondary";
};
```

> 注意
> 如果系统运行的是 FreeBSD 10 及更高版本，把 CARP 配置的接口名称替换 **carp0**。

在两个节点上重新启动 devd(8)[1300] 以使新配置生效：

```
# service devd restart
```

当指定的接口状态因开启或关闭而改变时，系统会产生一个通知，允许 devd(8)[1301] 子系统运行指定的自动故障切换脚本，**/usr/local/sbin/carp-hast-switch**。关于这个配置的进一步说明，请参考 devd.conf(5)[1302]。

下面是一个自动故障转移脚本的例子：

```
#!/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 primary
# make your best guess
delay=3

# logging
log="local0.debug"
```

(continues on next page)

[1300] https://www.freebsd.org/cgi/man.cgi?query=devd&sektion=8&format=html
[1301] https://www.freebsd.org/cgi/man.cgi?query=devd&sektion=8&format=html
[1302] https://www.freebsd.org/cgi/man.cgi?query=devd.conf&sektion=5&format=html

(continued from previous page)

```
name="carp-hast"

# end of user configurable stuff

case "$1" in
    primary)
        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␣
↪resource ${disk}."
                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␣
↪appear."
                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
```

(continues on next page)

(continued from previous page)

```
            mkdir -p /hast/${disk}
            fsck -p -y -t ufs /dev/hast/${disk}
            mount /dev/hast/${disk} /hast/${disk}
        done

    ;;

    secondary)
        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␣
↪resource ${disk}."
                exit 1
            fi
            logger -p $log -t $name "Role switched to secondary for resource ${disk}."
        done
    ;;
esac
```

简而言之，当一个节点成为主节点时，脚本会采取这些行动：

- 将另一个节点上的 HAST 池提升为主池。
- 检查 HAST 池下的文件系统。
- 挂载该池。

当一个节点成为次节点时：

- 解除 HAST 池的挂载。
- 将 HAST 池降级为次级。

当心

这只是一个作为概念证明的脚本例子。它并不能处理所有可能的情况，可以以任何方式对其进行扩展或改变，例如，启动或停止所需的服务。

技巧

在这个例子中，使用了基础的 UFS 文件系统。为了减少恢复所需的时间，可以使用带日志支持的 UFS 或 ZFS 文件系统来代替。

更详细的信息和更多的例子可以在以下网站找到：http://wiki.FreeBSD.org/HAST[1303]。

20.15.3.故障排除

HAST 的工作一般应该没有问题。然而，就像任何其他软件产品一样，可能有的时候它不能像预期的那样工作。问题的来源可能是不同的，但经验法则是确保集群的节点之间的时间是同步的。

当排除 HAST 的故障时，应通过 `hastd -d` 来增加 hastd(8)[1304] 的调试级别。这个参数可以指定多次以进一步提高调试级别。也可以考虑使用 `-F`，它将在前台启动 `hastd`。

20.15.3.1.从“大脑分裂”中恢复过来

当集群的节点无法相互通信，而两个节点都被配置为主节点时，就会出现 *大脑分裂*。这是一个危险的情况，因为它允许两个节点对数据进行不兼容的修改。必须由系统管理员手动纠正这个问题。

管理员必须决定哪个节点含有更重要的变化（或者手动执行合并改动）。然后，让 HAST 对有破损数据的节点执行完全同步。要做到这一点，在需要重新同步的节点上执行这些命令：

```
# hastctl role init test
# hastctl create test
# hastctl role secondary test
```

[1303] http://wiki.freebsd.org/HAST

[1304] https://www.freebsd.org/cgi/man.cgi?query=hastd&sektion=8&format=html

第二一章 GEOM: 模块化磁盘转换框架

21.1.概述

在 FreeBSD 中，GEOM 框架允许通过使用 provider 或 **/dev** 中的磁盘设备来访问和控制类，如主引导记录和 BSD 标签。通过支持各种软件 RAID 配置，GEOM 可提供对操作系统和操作系统工具的透明访问。

本章介绍了在 FreeBSD 的 GEOM 框架下使用磁盘的情况。这包括主要的 RAID 控制工具，这些工具使用该框架进行配置。这一章并非是 RAID 配置的权威指南，只讨论了 GEOM 支持的 RAID 分类。

读完本章后，你会知道：

- GEOM 支持什么类型的 RAID。
- 如何使用基本工具来配置、维护和操作各种级别的 RAID。
- 如何通过 GEOM 镜像、条带、加密和远程连接磁盘设备。
- 如何对连接到 GEOM 框架的磁盘进行故障排除。

在阅读本章之前，你应该：

- 了解 FreeBSD 是如何处理磁盘设备的（存储[1305]）。
- 知道如何配置和安装新的内核（配置 FreeBSD 内核[1306]）。

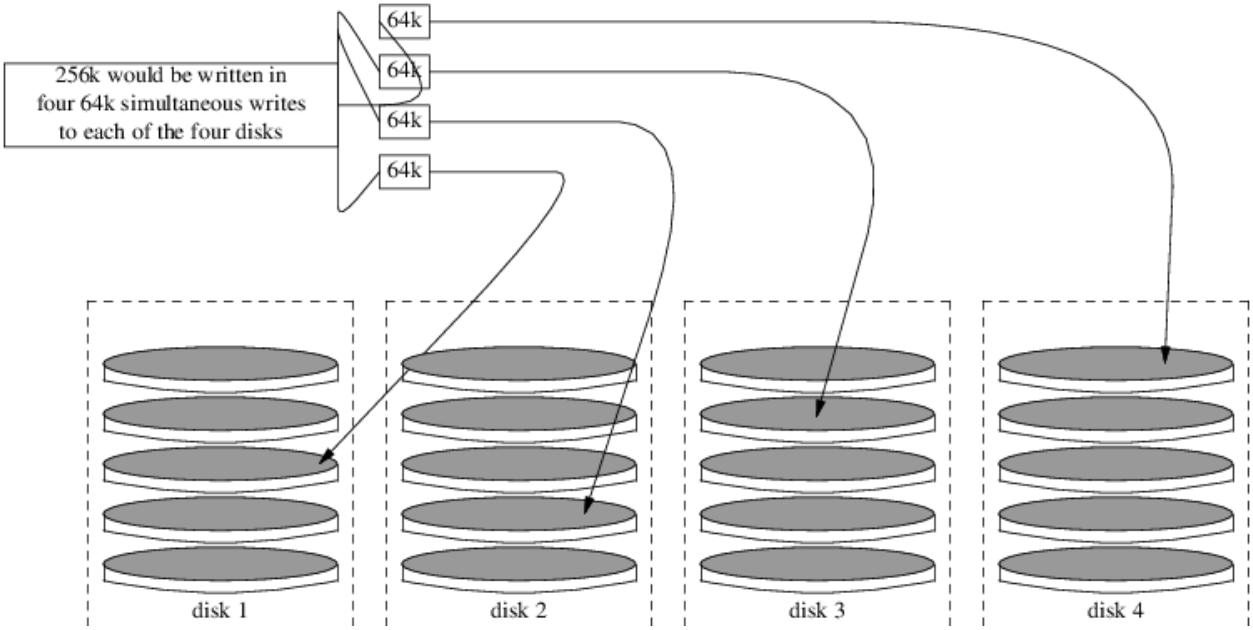
[1305] https://docs.freebsd.org/en/books/handbook/disks/index.html#disks

[1306] https://docs.freebsd.org/en/books/handbook/kernelconfig/index.html#kernelconfig

21.2. RAID0——条带

条带化将几个磁盘驱动器组合成一个卷。可以通过使用硬件 RAID 控制器来进行条带化。GEOM 磁盘子系统为磁盘条带化提供软件支持，也称为 RAID0，不需要 RAID 磁盘控制器。

在 RAID0 中，数据被分割成多个块，被分别写入阵列中的所有磁盘。如下图所示，RAID0 不需要等待系统向一个磁盘写入 256k 的数据，而是可以同时向阵列中的四个磁盘各写入 64k 的数据，从而提供卓越的 I/O 性能。这种性能可以通过使用多个磁盘控制器进一步增强。



RAID0 条带中的每个磁盘必须具有相同的大小，因为 I/O 请求是交错进行的，以并行地读或写到多个磁盘。

注意

RAID0 不提供任何冗余。这意味着，如果阵列中的一个磁盘发生故障，磁盘上的所有数据都会丢失。如果数据很重要，请实施一个备份策略，定期将备份保存到远程系统或设备上。

在 FreeBSD 系统上使用商用磁盘创建一个基于 GEOM 的软件 RAID0 的过程如下。如果创建了条带，请参考 gstripe(8)[1307] 以了解更多关于如何控制现有条带的信息。

创建未格式化的 **ATA** 磁盘条带的步骤

1. 加载 **geom_stripe.ko** 模块：

```
# kldload geom_stripe
```

2. 确保存在一个合适的挂载点。如果这个卷将成为根分区，那么暂时使用另一个挂载点，如 **/mnt**

[1307] https://www.freebsd.org/cgi/man.cgi?query=gstripe&sektion=8&format=html

3. 确定要条带化的磁盘的设备名称，并创建新的条带设备。例如，要对两个未使用和未分区的 ATA 磁盘进行条带化，设备名称为 **/dev/ad2** 和 **/dev/ad3**：

```
# gstripe label -v st0 /dev/ad2 /dev/ad3
Metadata value stored on /dev/ad2.
Metadata value stored on /dev/ad3.
Done.
```

4. 在新卷上写一个标准标签，也称为分区表，并安装默认的引导代码：

```
# bsdlabel -wB /dev/stripe/st0
```

5. 这个过程应该在 **/dev/stripe** 中创建除 **st0** 之外的另外两个设备。这些设备包括 **st0a** 和 **st0c**。此时，可以使用 newfs 在 **st0a** 上创建一个 UFS 文件系统：

```
# newfs -U /dev/stripe/st0a
```

在屏幕上会滑过许多数字，几秒钟后，这个过程就完成了。卷已经被创建，并准备好被挂载。

6. 要手动挂载创建的磁盘条带：

```
# mount /dev/stripe/st0a /mnt
```

7. 为了在启动过程中自动挂载这个带状文件系统，把卷的信息放在 **/etc/fstab** 中。在这个例子中，创建了一个永久的挂载点，名为 **stripe**：

```
# mkdir /stripe
# echo "/dev/stripe/st0a /stripe ufs rw 2 2" \
>> /etc/fstab
```

8. **geom_stripe.ko** 模块也必须在系统初始化时自动加载，方法是在 **/boot/loader.conf** 中添加一行：

```
# echo 'geom_stripe_load="YES"' >> /boot/loader.conf
```

21.3.RAID1——镜像

RAID1，或称 镜像，是将相同的数据写入一个以上的磁盘设备的技术。镜像通常被用来防止因磁盘故障而导致的数据丢失。镜像中的每个磁盘都包含一个相同的数据副本。当一个单独的磁盘发生故障时，镜像继续工作，提供来自仍在运行的磁盘的数据。计算机继续运行，管理员有时间在不影响用户的情况下更换故障磁盘。

这些例子中说明了两种常见的情况。第一个例子是用两个新的磁盘创建一个镜像，用它来替代现有的单个磁盘。第二个例子在一个新的磁盘上创建一个镜像，把旧的磁盘的数据复制到它上面，然后把旧的磁盘插入镜像中。虽然这个过程稍微复杂一些，但它只需要一个新的磁盘。

传统上，镜像中的两个磁盘在型号和容量上是相同的，但是 gmirror(8) 并不要求这样。用不同的磁盘创建的镜像，其容量等于镜像中最小的磁盘的容量。较大磁盘上的额外空间将不被使用。后来插入镜像的硬盘必须至少有与镜像中最小的硬盘一样的容量。

警告

这里显示的镜像步骤是非破坏性的，但与任何主要的磁盘操作一样，先做一个完整的备份。

警告

虽然 dump(8)[1308] 在这些程序中被用来复制文件系统，但它对有软更新日志的文件系统不起作用。参见 tunefs(8)[1309] 以获得关于检测和禁用软更新日志的信息。

21.3.1.元数据问题

许多磁盘系统在每个磁盘的末端存储元数据。在重新使用磁盘做镜像之前，应该擦除旧的元数据。大多数问题是由两种特殊类型的遗留元数据引起的：GPT 分区表和之前镜像的旧元数据。

可以用 gpart(8)[1310] 来擦除 GPT 元数据。这个例子删除了 **ada8** 磁盘上的 GPT 主分区表和 GPT 备份分区表。

```
# gpart destroy -F ada8
```

使用 gmirror(8)[1311] 可以将一个磁盘从活动镜像中移除，并在一个步骤中擦除元数据。这里例子中的磁盘 **ada8** 被从活动镜像 **gm4** 中移除：

```
# gmirror remove gm4 ada8
```

如果镜像没有运行，但是旧的镜像元数据仍然在磁盘上，使用 `gmirror clear` 来移除它：

```
# gmirror clear ada8
```

gmirror(8)[1312] 在磁盘的末端存储一个元数据块。由于 GPT 分区方案也在磁盘的末端存储元数据，所以不推荐用 gmirror(8)[1313] 来镜像整个 GPT 磁盘。这里使用 MBR 分区，因为它只在磁盘的开始部分存储一个分区表，不会与镜像元数据冲突。

[1308] https://www.freebsd.org/cgi/man.cgi?query=dump&sektion=8&format=html
[1309] https://www.freebsd.org/cgi/man.cgi?query=tunefs&sektion=8&format=html
[1310] https://www.freebsd.org/cgi/man.cgi?query=gpart&sektion=8&format=html
[1311] https://www.freebsd.org/cgi/man.cgi?query=gmirror&sektion=8&format=html
[1312] https://www.freebsd.org/cgi/man.cgi?query=gmirror&sektion=8&format=html
[1313] https://www.freebsd.org/cgi/man.cgi?query=gmirror&sektion=8&format=html

21.3.2.用两个新磁盘创建一个镜像

在这个例子中，FreeBSD 已经被安装在一个单一的磁盘上，即 **ada0**。两个新的磁盘，**ada1** 和 **ada2** 已经被连接到系统中。在这两个磁盘上将创建一个新的镜像，用来替换旧的单个磁盘。

geom_mirror.ko 内核模块必须被内置到内核中，或者在启动或运行时加载。现在手动加载内核模块：

```
# gmirror load
```

用两个新的磁盘创建镜像：

```
# gmirror label -v gm0 /dev/ada1 /dev/ada2
```

gm0 是一个用户选择的设备名称，分配给新的镜像。镜像启动后，这个设备名出现在 **/dev/mirror/** 中。

现在可以用 gpart(8)[1314] 在镜像上创建 MBR 和 bsdlabel 分区表。这个例子使用了传统的文件系统布局，有 **/** 、swap 、**/var** 、**/tmp** 和 **/usr** 分区。单一的 **/** 和交换分区也是可以的。

镜像上的分区不一定要和现有磁盘上的分区一样大，但它们必须大到足以容纳 **ada0** 上已有的所有数据。

```
# 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)
```

[1314] https://www.freebsd.org/cgi/man.cgi?query=gpart&sektion=8&format=html

通过在 MBR 和 bsdlabel 中安装 bootcode 并设置活动 slice，使镜像可启动：

```
# gpart bootcode -b /boot/mbr mirror/gm0
# gpart set -a active -i 1 mirror/gm0
# gpart bootcode -b /boot/boot mirror/gm0s1
```

格式化新镜像上的文件系统，启用软更新。

```
# newfs -U /dev/mirror/gm0s1a
# newfs -U /dev/mirror/gm0s1d
# newfs -U /dev/mirror/gm0s1e
# newfs -U /dev/mirror/gm0s1f
```

现在可以用 dump(8)[1315] 和 restore(8)[1316] 将原始 **ada0** 磁盘上的文件系统复制到镜像上。

```
# 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 -)
```

编辑 **/mnt/etc/fstab** 以指向新的镜像文件系统：

```
# 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
```

如果 **geom_mirror.ko** 内核模块没有被内置到内核中，编辑 **/mnt/boot/loader.conf** 以在启动时加载该模块：

```
geom_mirror_load="YES"
```

重新启动系统以测试新的镜像，并验证所有数据是否已经复制。BIOS 会把镜像看作是两个独立的硬盘，而非一个镜像。由于这两个硬盘是相同的，所以选择哪一个来启动并不重要。

如果启动有问题，请参见故障排除[1317]。关掉电源并断开原始 **ada0** 磁盘的连接，可以将其作为离线备份保存。

[1315] https://www.freebsd.org/cgi/man.cgi?query=dump&sektion=8&format=html

[1316] https://www.freebsd.org/cgi/man.cgi?query=restore&sektion=8&format=html

[1317] https://docs.freebsd.org/en/books/handbook/book/#gmirror-troubleshooting

在使用中，镜像的行为就像原来的单盘一样。

21.3.3.用现有的磁盘创建一个镜像

在这个例子中，FreeBSD 已经被安装在一个单一的磁盘上，即 **ada0**。一个新的磁盘，**ada1**，已经被连接到系统中。在新的磁盘上将创建一个单盘镜像，将现有的系统复制到上面，然后将旧的磁盘插入镜像中。这个稍微复杂的过程是有必要的，因为 `gmirror` 需要在每个磁盘的末尾放一个 512 byte 的元数据块，而现有的 **ada0** 通常已经分配了所有的空间。

加载 **geom_mirror.ko** 内核模块：

```
# gmirror load
```

用 `diskinfo` 检查原始磁盘的镜像大小：

```
# diskinfo -v ada0 | head -n3
/dev/ada0
        512             # sectorsize
        1000204821504   # mediasize in bytes (931G)
```

在新磁盘上创建一个镜像。为了确保镜像的容量不比原来的 **ada0** 磁盘大，gnop(8)[1318] 被用来创建一个大小完全相同的假磁盘。这个磁盘不存储任何数据，只是用来限制镜像的大小。当 gmirror(8)[1319] 创建镜像时，它将把容量限制在 **gzero.nop** 的大小，即使新的 **ada1** 磁盘有更多的空间。注意，第二行中的 *1000204821504* 等于上面 diskinfo 显示的 **ada0** 的镜像大小。

```
# geom zero load
# gnop create -s 1000204821504 gzero
# gmirror label -v gm0 gzero.nop ada1
# gmirror forget gm0
```

由于 **gzero.nop** 不存储任何数据，镜像不会将其视为连接。镜像被告知要 “忘记” 未连接的组件，删除对 **gzero.nop** 的引用。结果是一个镜像设备只包含一个磁盘，**ada1**。

创建 **gm0** 后，查看 **ada0** 的分区表。这个输出是来自一个 1TB 的磁盘。如果在磁盘的末端有一些未分配的空间，可以直接从 **ada0** 复制内容到新的镜像。

但是，如果输出显示磁盘上的所有空间都被分配了，就像下面的列表一样，那么磁盘末端的 512 byte 的镜像元数据就没有可用空间了：

```
# gpart show ada0
=>        63  1953525105        ada0  MBR  (931G)
          63  1953525105           1  freebsd  [active]  (931G)
```

[1318] https://www.freebsd.org/cgi/man.cgi?query=gnop&sektion=8&format=html

[1319] https://www.freebsd.org/cgi/man.cgi?query=gmirror&sektion=8&format=html

在这种情况下，必须编辑分区表以减少镜像 **/gm0** 上一个扇区的容量。这个过程将在后面进行解释。

在这两种情况下，应首先使用 `gpart backup` 和 `gpart restore` 来复制主磁盘上的分区表：

```
# gpart backup ada0 > table.ada0
# gpart backup ada0s1 > table.ada0s1
```

这些命令创建了两个文件，**table.ada0** 和 **table.ada0s1**。这个例子来自一个 1TB 的硬盘：

```
# 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
```

如果在磁盘的末端没有显示出自由空间，那么 slice 和最后一个分区的大小都必须减少一个扇区。编辑这两个文件，将 slice 和最后一个分区的大小都减少一个。这些是每个列表中的最后数字：

```
# 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
```

如果在磁盘的末端至少有一个扇区没有被分配，这两个文件可以不加修改地使用。

现在将分区表恢复到 **mirror/gm0**：

```
# gpart restore mirror/gm0 < table.ada0
# gpart restore mirror/gm0s1 < table.ada0s1
```

用 `gpart show` 检查分区表。这个例子中，**gm0s1a** 代表 **/**，**gm0s1d** 代表 **/var**，**gm0s1e** 代表 **/usr**，**gm0s1f** 代表 **/data1**，而 **gm0s1g** 代表 **/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)
```

slice 和最后一个分区都必须在磁盘的末端有至少一个空闲块。

在这些新分区上创建文件系统。分区的数量将有所不同，以配合原始磁盘，即 **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
```

通过在 MBR 和 bsdlabel 中安装 bootcode 并设置活动 slice，使镜像可启动：

```
# gpart bootcode -b /boot/mbr mirror/gm0
# gpart set -a active -i 1 mirror/gm0
# gpart bootcode -b /boot/boot mirror/gm0s1
```

调整 **/etc/fstab** 以使用镜像上的新分区。先把这个文件复制到 **/etc/fstab.orig**，以此来备份：

```
# cp /etc/fstab /etc/fstab.orig
```

编辑 **/etc/fstab**，将 **/dev/ada0** 替换为 **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
```

如果 **geom_mirror.ko** 内核模块没有被内置到内核中，编辑 **/boot/loader.conf** 在启动时加载它：

```
geom_mirror_load="YES"
```

现在可以用 dump(8)[1320] 和 restore(8)[1321] 将原始磁盘的文件系统复制到镜像上。用 `dump -L` 转储的每个文件系统都会先创建一个快照，这可能需要一些时间：

```
# 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 -)
```

重新启动系统，从 **ada1** 启动。如果一切正常，系统将从 **mirror/gm0** 启动，现在 **mirror/gm0** 包含的数据与 **ada0** 之前的一样。如果启动有问题，请看故障排除[1322]。

在这一点上，镜像仍然只包括单一的 **ada1** 磁盘。

从 **mirror/gm0** 成功启动后，最后一步是将 **ada0** 插入镜像中。

> 重要
>
> 当 **ada0** 被插入镜像时，它以前的内容会被镜像的数据覆盖。在将 **ada0** 加入镜像之前，请确定 **mirror/gm0** 的内容与 **ada0** 相同。如果之前通过 dump(8)[1323] 和 restore(8)[1324] 复制的内容与 **ada0** 上的内容不一致，请重新修改 **/etc/fstab**，将文件系统挂载到 **ada0** 上，重新启动，然后重新开始整个过程。

[1320] https://www.freebsd.org/cgi/man.cgi?query=dump&sektion=8&format=html
[1321] https://www.freebsd.org/cgi/man.cgi?query=restore&sektion=8&format=html
[1322] https://docs.freebsd.org/en/books/handbook/book/#gmirror-troubleshooting
[1323] https://www.freebsd.org/cgi/man.cgi?query=dump&sektion=8&format=html
[1324] https://www.freebsd.org/cgi/man.cgi?query=restore&sektion=8&format=html

```
# gmirror insert gm0 ada0
GEOM_MIRROR: Device gm0: rebuilding provider ada0
```

两个磁盘之间的同步将立即开始。使用 `gmirror status` 来查看进度。

```
# gmirror status
      Name    Status  Components
mirror/gm0  DEGRADED  ada1 (ACTIVE)
                      ada0 (SYNCHRONIZING, 64%)
```

一段时间后，同步将结束。

```
GEOM_MIRROR: Device gm0: rebuilding provider ada0 finished.
# gmirror status
      Name    Status  Components
mirror/gm0  COMPLETE  ada1 (ACTIVE)
                      ada0 (ACTIVE)
```

mirror/gm0 现在由两个磁盘 **ada0** 和 **ada1** 组成，内容会自动相互同步。在使用中，**mirror/gm0** 的行为就像原来的单硬盘一样。

21.3.4.故障排除

如果系统不启动，可能需要改变 BIOS 设置，以便从新的镜像磁盘中的一个启动。任何一个镜像磁盘都可以用来启动，因为它们包含相同的数据。

如果启动停止时出现这个信息，说明镜像设备出了问题。

```
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 /)
```

(continues on next page)

(continued from previous page)

```
  ?               List valid disk boot devices
  .               Yield 1 second (for background tasks)
  <empty line>    Abort manual input

mountroot>
```

忘记在 **/boot/loader.conf** 中加载 **geom_mirror.ko** 模块会导致这个问题。要解决这个问题，从 FreeBSD 安装设备启动，在第一个提示符下选择 `Shell`。然后加载镜像模块并安装镜像设备：

```
# gmirror load
# mount /dev/mirror/gm0s1a /mnt
```

编辑 **/mnt/boot/loader.conf**，添加这一行来加载镜像模块:

```
geom_mirror_load="YES"
```

保存该文件并重新启动。

其他导致 `error 19` 的问题需要更多的努力来解决。尽管系统应该从 **ada0** 启动，但如果 **/etc/fstab** 不正确，会出现另一个选择 shell 的提示。在引导加载器提示下输入 `ufs:/dev/ada0s1a`，然后按回车键。撤销在 **/etc/fstab** 中的编辑，然后从原始磁盘（**ada0**）而非镜像中加载文件系统。重新启动系统并再次尝试该过程。

```
Enter full pathname of shell or RETURN for /bin/sh:
# cp /etc/fstab.orig /etc/fstab
# reboot
```

21.3.5.从磁盘故障中恢复

磁盘镜像的好处是，一个单独的磁盘可以发生故障而不会导致镜像丢失任何数据。在上面的例子中，如果 **ada0** 发生故障，镜像将继续工作，从剩余的工作的磁盘，**ada1** 提供数据。

要更换故障的硬盘，请关闭系统，用一个容量相同或更大的新硬盘物理替换故障的硬盘。制造商在以 GB 为单位对硬盘进行评级时，使用了一些任意的数值，真正确定的唯一方法是比较 `diskinfo -v` 所显示的扇区总数。容量大于镜像的磁盘可以工作，尽管新磁盘上的额外空间将不会被使用。

在计算机重新上电后，镜像将以“降级”模式运行，即只有一个磁盘。镜像被告知要忘记当前没有连接的磁盘：

```
# gmirror forget gm0
```

任何旧的元数据都应该按照元数据问题[1325]中的说明从替换磁盘中清除。然后将替换磁盘（本例中为**ada4**）插入镜像中：

```
# gmirror insert gm0 /dev/ada4
```

当新的磁盘被插入镜像时，重新同步开始。这个将镜像数据复制到新磁盘的过程可能需要一段时间。在复制过程中，镜像的性能会大大降低，所以插入新的磁盘最好在计算机上的需求较低时进行。

可以用 `gmirror status` 监控进度，它显示正在同步的磁盘和完成的百分比。在重新同步的过程中，状态将是 `DEGRADED`，当这个过程结束时将变为 `COMPLETE`。

21.4.RAID3——带有专用奇偶校验的字节级条带

RAID3 是一种用于将几个磁盘设备结合成一个具有专用奇偶校验盘的单一卷的方法。在一个 RAID3 系统中，数据被分割成若干字节，除了作为专用奇偶校验盘的一个磁盘外，这些字节被写入阵列中的所有磁盘。这意味着，从一个 RAID3 实现中的磁盘读取会访问阵列中的所有磁盘。通过使用多个磁盘控制器可以提高性能。RAID3 阵列提供 1 个磁盘的容错，同时提供了 1-1/n 倍于阵列中所有磁盘总容量的容量，其中 n 是阵列中的硬盘数量。这样的配置大多适用于存储较大的数据，如多媒体文件。

建立一个 RAID3 阵列，至少需要 3 个物理硬盘。每个磁盘必须是相同的大小，因为 I/O 请求是交错进行的，可以并行地读或写到多个磁盘。另外，由于 RAID3 的性质，硬盘的数量必须等于 3 、5 、9 、17，以此类推，或者 2^n+1。

本节示范了如何在 FreeBSD 系统上创建一个软件 RAID3。

> 注意
> 虽然理论上 FreeBSD 可以从 RAID3 阵列中启动，但这种配置既不常见，也不建议使用。

21.4.1.创建一个专用的 RAID3 阵列

在 FreeBSD 中，对 RAID3 的支持是由 graid3(8)[1326] GEOM 类实现的。在 FreeBSD 上创建一个专用的 RAID3 阵列需要以下步骤。

1. 加载 **geom_raid3.ko** 模块，执行以下任意一条命令：

```
# graid3 load
```

或者

```
# kldload geom_raid3
```

1. 确保存在一个合适的挂载点。这条命令创建了一个新的目录，作为挂载点：

[1325] https://docs.freebsd.org/en/books/handbook/book/#geom-mirror-metadata
[1326] https://www.freebsd.org/cgi/man.cgi?query=graid3&sektion=8&format=html

```
# mkdir /multimedia
```

1. 确定将被添加到阵列中的磁盘的设备名称，并创建新的 RAID3 设备。最后列出的设备将作为专用的奇偶校验盘。这个例子使用了三个未分区的 ATA 硬盘：数据用 **ada1** 和 **ada2**，奇偶校验用 **ada3**。

```
# 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.
```

1. 对新创建的 **gr0** 设备进行分区，并在其上放置 UFS 文件系统。

```
# gpart create -s GPT /dev/raid3/gr0
# gpart add -t freebsd-ufs /dev/raid3/gr0
# newfs -j /dev/raid3/gr0p1
```

在屏幕上会滑过许多数字，经过一段时间后，这个过程就完成了。卷已经被创建，并准备好被挂载：

```
# mount /dev/raid3/gr0p1 /multimedia/
```

现在 RAID3 阵列已经可以使用了。

需要额外的配置，以便在系统重启时保留这一设置。

1.必须在阵列被挂载之前加载模块 **geom_raid3.ko**。为了在系统初始化时自动加载内核模块，在 **/boot/loader.conf** 中添加以下一行：

```
geom_raid3_load="YES"
```

1. 为了在系统启动过程中自动挂载阵列的文件系统，必须在 **/etc/fstab** 中添加下列卷信息：

```
/dev/raid3/gr0p1  /multimedia ufs rw  2   2
```

21.5.软件 RAID 设备

一些主板和扩展卡增加了一些简单的硬件，通常只是一个 ROM，允许计算机从 RAID 阵列中启动。启动后，对 RAID 阵列的访问由运行在计算机主处理器上的软件处理。这种“硬件辅助的软件 RAID”使 RAID 阵列不依赖于任何特定的操作系统，甚至在操作系统被加载之前就能发挥作用。

根据使用的硬件，支持几种级别的 RAID。参见 graid(8)[1327] 以获得完整的列表。

[1327] https://www.freebsd.org/cgi/man.cgi?query=graid&sektion=8&format=html

graid(8)[1328] 需要 **geom_raid.ko** 内核模块，它从 FreeBSD 9.1 开始包含在 **GENERIC** 内核中。如果需要，可以通过 `graid load` 手动加载。

21.5.1.创建阵列

软件 RAID 设备通常有一个菜单，可以在计算机启动时按下特殊键进入。该菜单可以用来创建和删除 RAID 阵列。graid(8)[1329] 也可以直接从命令行创建阵列。

`graid label` 标签用于创建新的阵列。本例中使用的主板有一个 Intel 软件 RAID 芯片组，所以指定了 Intel 元数据格式。新的阵列被赋予 **gm0** 的标签，它是一个镜像（RAID1），并且使用 **ada0** 和 **ada1** 硬盘。

> 当心
> 当硬盘被做成一个新的阵列时，硬盘上的一些空间将被覆盖。请先备份现有的数据！

```
# graid label Intel gm0 RAID1 ada0 ada1
GEOM_RAID: Intel-a29ea104: Array Intel-a29ea104 created.
GEOM_RAID: Intel-a29ea104: Disk ada0 state changed from NONE to ACTIVE.
GEOM_RAID: Intel-a29ea104: Subdisk gm0:0-ada0 state changed from NONE to ACTIVE.
GEOM_RAID: Intel-a29ea104: Disk ada1 state changed from NONE to ACTIVE.
GEOM_RAID: Intel-a29ea104: Subdisk gm0:1-ada1 state changed from NONE to ACTIVE.
GEOM_RAID: Intel-a29ea104: Array started.
GEOM_RAID: Intel-a29ea104: Volume gm0 state changed from STARTING to OPTIMAL.
Intel-a29ea104 created
GEOM_RAID: Intel-a29ea104: Provider raid/r0 for volume gm0 created.
```

状态检查显示新的镜像已经可以使用了：

```
# graid status
   Name   Status  Components
raid/r0  OPTIMAL  ada0 (ACTIVE (ACTIVE))
                  ada1 (ACTIVE (ACTIVE))
```

阵列设备出现在 **/dev/raid/**。第一个阵列被称为 **r0**。其他的阵列，如果存在的话，将是 **r1** 、**r2**，以此类推。

一些设备上的 BIOS 菜单可以创建名称中有特殊字符的阵列。为了避免这些特殊字符的问题，阵列被赋予简单的编号名称，如 **r0**。要显示实际的标签，如上面例子中的 **gm0**，请使用 sysctl(8)[1330]：

```
# sysctl kern.geom.raid.name_format=1
```

[1328] https://www.freebsd.org/cgi/man.cgi?query=graid&sektion=8&format=html
[1329] https://www.freebsd.org/cgi/man.cgi?query=graid&sektion=8&format=html
[1330] https://www.freebsd.org/cgi/man.cgi?query=sysctl&sektion=8&format=html

21.5.2.多个卷

一些软件 RAID 设备支持在一个阵列上有一个以上的卷。卷的工作方式类似于分区，允许物理磁盘上的空间被分割并以不同方式使用。例如，英特尔软件 RAID 设备支持两个卷。这个例子创建了一个 40G 的镜像，用于安全存储操作系统，然后是一个 20G 的 RAID0（条带）卷，用于快速临时存储：

```
# graid label -S 40G Intel gm0 RAID1 ada0 ada1
# graid add -S 20G gm0 RAID0
```

卷在 **/dev/raid/** 中显示为额外的 **rX** 条目。一个有两个卷的阵列将显示为 **r0** 和 **r1**。

关于不同的软件 RAID 设备所支持的卷的数量，请参见 graid(8)[1331]。

21.5.3.将单个磁盘转换为镜像

在某些特定的条件下，有可能将现有的单个硬盘转换成 graid(8)[1332] 阵列，而无需重新格式化。为了避免转换过程中的数据丢失，现有的硬盘必须满足这些最低要求：

- 磁盘必须用 MBR 分区方案进行分区。GPT 或其他分区方案的元数据在硬盘的末端，会被 graid(8)[1333] 的元数据覆盖和破坏。
- 磁盘末端必须有足够的未分区和未使用的空间来容纳 graid(8)[1334] 元数据。这个元数据的大小各不相同，但最大的占据了 64M，所以建议至少有这么多的可用空间。
- 如果磁盘满足这些要求，则首先进行完全备份。然后用这个磁盘创建一个单磁盘镜像：

```
# graid label Intel gm0 RAID1 ada0 NONE
```

graid(8)[1335] 元数据被写到磁盘末端的未使用空间中。现在可以在镜像中插入第二块磁盘：

```
# graid insert raid/r0 ada1
```

原始硬盘的数据将立即开始复制到第二个硬盘。镜像将以降级状态运行，直到复制完成。

[1331] https://www.freebsd.org/cgi/man.cgi?query=graid&sektion=8&format=html
[1332] https://www.freebsd.org/cgi/man.cgi?query=graid&sektion=8&format=html
[1333] https://www.freebsd.org/cgi/man.cgi?query=graid&sektion=8&format=html
[1334] https://www.freebsd.org/cgi/man.cgi?query=graid&sektion=8&format=html
[1335] https://www.freebsd.org/cgi/man.cgi?query=graid&sektion=8&format=html

21.5.4.将新的硬盘插入阵列中

作为故障或丢失的磁盘的替代，可以插入磁盘到阵列中。如果没有故障或丢失的磁盘，新的磁盘就成为备用磁盘。例如，将一块新的磁盘插入到一个工作的双磁盘镜像中，将成为有一个备用磁盘的双磁盘镜像，而不是三磁盘的镜像。

在这个例子的镜像阵列中，数据立即开始被复制到新插入的硬盘。新磁盘上的任何现有信息将被覆盖：

```
# graid insert raid/r0 ada1
GEOM_RAID: Intel-a29ea104: Disk ada1 state changed from NONE to ACTIVE.
GEOM_RAID: Intel-a29ea104: Subdisk gm0:1-ada1 state changed from NONE to NEW.
GEOM_RAID: Intel-a29ea104: Subdisk gm0:1-ada1 state changed from NEW to REBUILD.
GEOM_RAID: Intel-a29ea104: Subdisk gm0:1-ada1 rebuild start at 0.
```

21.5.5.从阵列中移除硬盘

可以从一个阵列中永久移除单个磁盘，其元数据也被删除：

```
# graid remove raid/r0 ada1
GEOM_RAID: Intel-a29ea104: Disk ada1 state changed from ACTIVE to OFFLINE.
GEOM_RAID: Intel-a29ea104: Subdisk gm0:1-[unknown] state changed from ACTIVE to NONE.
GEOM_RAID: Intel-a29ea104: Volume gm0 state changed from OPTIMAL to DEGRADED.
```

21.5.6.停止阵列

阵列可以被停止，而不需要从磁盘上删除元数据。当系统被启动时，该阵列将被重新启动：

```
# graid stop raid/r0
```

21.5.7.检查阵列状态

阵列状态可以在任何时候检查。在上面的例子中，一个磁盘被添加到镜像中后，数据正从原来的磁盘复制到新的磁盘上：

```
# graid status
   Name    Status  Components
raid/r0  DEGRADED  ada0 (ACTIVE (ACTIVE))
                   ada1 (ACTIVE (REBUILD 28%))
```

某些类型的阵列，比如 RAID0 或 CONCAT，如果磁盘发生故障，可能不会显示在状态报告中。要看到这些部分失败的阵列，请添加 -ga：

```
# graid status -ga
          Name  Status  Components
Intel-e2d07d9a  BROKEN  ada6 (ACTIVE (ACTIVE))
```

21.5.8.删除阵列

阵列是通过删除其中所有的卷来销毁的。当最后一个卷被删除时，阵列被停止，元数据从磁盘上被删除：

```
# graid delete raid/r0
```

21.5.9.删除意外的阵列

磁盘可能意外地包含 graid(8)[1336] 元数据，可能是以前使用的或制造商的测试。graid(8)[1337] 会检测到这些磁盘并创建一个阵列，干扰对单个磁盘的访问。要删除不需要的元数据：

1. 启动系统。在启动菜单上，选择 2 为加载器提示。输入：

```
OK set kern.geom.raid.enable=0
OK boot
```

系统将在禁用 graid(8)[1338] 的情况下启动。

1. 备份受影响磁盘上的所有数据。
2. 作为一种变通方法，可以通过以下内容禁用 graid(8)[1339] 阵列检测，添加

```
kern.geom.raid.enable=0
```

到 **/boot/loader.conf**。

要从受影响的磁盘中永久删除 graid(8)[1340] 元数据，请启动 FreeBSD 安装光盘或 U 盘，并选择 Shell。使用 status 找到阵列的名称，通常是 raid/r0：

```
# graid status
  Name   Status  Components
raid/r0  OPTIMAL  ada0 (ACTIVE (ACTIVE))
                  ada1 (ACTIVE (ACTIVE))
```

按名称删除卷：

[1336] https://www.freebsd.org/cgi/man.cgi?query=graid&sektion=8&format=html
[1337] https://www.freebsd.org/cgi/man.cgi?query=graid&sektion=8&format=html
[1338] https://www.freebsd.org/cgi/man.cgi?query=graid&sektion=8&format=html
[1339] https://www.freebsd.org/cgi/man.cgi?query=graid&sektion=8&format=html
[1340] https://www.freebsd.org/cgi/man.cgi?query=graid&sektion=8&format=html

```
# graid delete raid/r0
```

如果显示有多个卷，对每个卷重复该过程。在最后一个阵列被删除后，卷将被销毁。

重新启动并验证数据，必要时从备份中恢复。在删除元数据后，也可以删除 **/boot/loader.conf** 中的 `kern.geom.raid.enable=0` 条目。

21.6.GEOM Gate 网络设备

GEOM 提供了一种简单的机制，通过使用 GEOM Gate 网络守护程序 ggated，其提供了对磁盘、CD 和文件系统等设备的远程访问。拥有设备的系统运行服务器守护程序，处理客户使用 ggatec 提出的请求。这些设备不应包含任何敏感数据，因为客户端和服务器之间的连接没有进行加密。

与网络文件系统（NFS）[1341] 中讨论的 NFS 类似，ggated 也是通过一个 exports 文件来配置的。这个文件指定了哪些系统被允许访问导出的资源，以及提供给他们的访问级别。例如，为了让客户端 `192.168.1.5` 读写第一个 SCSI 磁盘上的第四个分区，创建 **/etc/gg.exports**，添加这一行：

```
192.168.1.5 RW /dev/da0s4d
```

在导出设备之前，确保它目前没有被挂载。然后，启动 ggated：

```
# ggated
```

有几个选项可以用来指定一个备用的监听端口或改变 exports 文件的默认位置。详情请参考 ggated(8)[1342]。

要在客户机上访问导出的设备，首先使用 `ggatec` 来指定服务器的 IP 地址和导出设备的设备名。如果成功，这个命令将显示一个要挂载的 `ggate` 设备名。在一个空闲的挂载点上挂载那个指定的设备名。这个例子连接到 `192.168.1.1` 上的 **/dev/da0s4d** 分区，然后将 **/dev/ggate0** 挂载到 **/mnt**：

```
# ggatec create -o rw 192.168.1.1 /dev/da0s4d
ggate0
# mount /dev/ggate0 /mnt
```

现在可以通过客户端的 **/mnt** 访问服务器上的设备。关于 `ggatec` 的更多细节和一些使用例子，请参考 ggatec(8)[1343]。

注意

如果该设备目前被挂载在服务器或网络上的任何其他客户端上，挂载将失败。如果需要同时访问网络资源，请使用 NFS 代替。

当不再需要该设备时，用 `umount` 卸载它，这样该资源就可以供其他客户使用。

[1341] https://docs.freebsd.org/en/books/handbook/network-servers/index.html#network-nfs

[1342] https://www.freebsd.org/cgi/man.cgi?query=ggated&sektion=8&format=html

[1343] https://www.freebsd.org/cgi/man.cgi?query=ggatec&sektion=8&format=html

21.7.为磁盘设备添加卷标

在系统初始化过程中，FreeBSD 内核会在发现设备时创建设备节点。这种探测设备的方法引起了一些问题。例如，如果一个新的磁盘设备是通过 USB 添加的呢？很可能一个闪存设备会被赋予 **da0** 的设备名，而原来的 **da0** 则被转移到 **da1**。如果文件系统被列在 **/etc/fstab** 中，这将导致文件系统的挂载问题，也可能导致系统无法启动。

一个解决方案是按 SCSI 设备的链接拓扑进行命名，这样添加到 SCSI 卡上的新设备就会被发放未使用的设备号。但是，USB 设备可能会取代主 SCSI 磁盘呢？发生这种情况是因为 USB 设备通常在 SCSI 卡之前被探测到。一种解决办法是只在系统启动后插入这些设备。另一个方法是只使用单一的 ATA 驱动器，并且从不在 **/etc/fstab** 中列出 SCSI 设备。

一个更好的解决方案是使用 `glabel` 来标记磁盘设备，并在 **/etc/fstab** 中使用这些标签。由于 `glabel` 将标签存储在给定 provider 的最后一个扇区中，标签将在重启时保持不变。通过使用这个标签作为设备，文件系统可以始终被挂载，无论它是通过什么设备节点被访问的。

> 注意
>
> `glabel` 可以创建临时性和永久性的标签。只有永久标签在重启时是不变的。请参考 glabel(8)[1344] 了解更多关于标签之间区别的信息。

21.7.1.标签类型和例子

永久标签可以是一个通用标签或一个文件系统标签。永久文件系统标签可以用 tunefs(8)[1345] 或 newfs(8)[1346] 创建。这些类型的标签被创建在 **/dev** 的一个子目录下，并将根据文件系统的类型来命名。例如，UFS2 文件系统的标签将在 **/dev/ufs** 中创建。通用的永久标签可以用 `glabel label` 创建。这些标签不针对文件系统，将被创建在 **/dev/label** 中。

临时标签在下次重启时被销毁。这些标签是在 **/dev/label** 中创建的，适合于实验。可以使用 `glabel create` 来创建一个临时标签。

要为 UFS2 文件系统创建一个永久标签而不破坏任何数据，请执行以下命令：

```
# tunefs -L home /dev/da3
```

现在 **/dev/ufs** 中应该有一个标签，可以添加到 **/etc/fstab** 中：

```
/dev/ufs/home       /home           ufs     rw              2       2
```

> 注意
>
> 在运行 `tunefs` 时，禁止挂载文件系统。

现在可以挂载文件系统了：

[1344] https://www.freebsd.org/cgi/man.cgi?query=glabel&sektion=8&format=html

[1345] https://www.freebsd.org/cgi/man.cgi?query=tunefs&sektion=8&format=html

[1346] https://www.freebsd.org/cgi/man.cgi?query=newfs&sektion=8&format=html

```
# mount /home
```

从这时起，只要在启动时用 **/boot/loader.conf** 加载 **geom_label.ko** 内核模块，或者存在内核选项 GEOM_LABEL，设备节点就可以改变而不会对系统产生任何不良影响。

文件系统也可以通过使用 newfs 的 -L 参数来创建一个默认的标签。更多信息请参考 newfs(8)[1347]。

可以用下面的命令来销毁标签：

```
# glabel destroy home
```

下面的例子显示了如何给启动盘的分区贴标签。

例 **41.**给启动盘上的分区贴标签

通过对启动盘上的分区进行永久标记，系统应该能够继续正常启动，即使磁盘被移到另一个控制器上或者被转移到不同的系统。在这个例子中，假设使用的是一个 ATA 磁盘，目前系统识别为 **ad0**。还假设使用了标准的 FreeBSD 分区方案，包括 **/** 、**/var** 、**/usr** 和 **/tmp**，以及一个交换分区。

重新启动系统，在 loader(8)[1348] 提示下，按 4 键启动到单用户模式。然后输入以下命令：

```
# 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
```

系统将继续以多用户启动。启动完成后，编辑 **/etc/fstab**，用各自的标签替换传统的设备名称。最后的 **/etc/fstab** 将看起来像这样：

```
# 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
```

现在可以重新启动系统了。如果一切顺利，它将正常出现，并显示挂载：

[1347] https://www.freebsd.org/cgi/man.cgi?query=newfs&sektion=8&format=html

[1348] https://www.freebsd.org/cgi/man.cgi?query=loader&sektion=8&format=html

```
# 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)
```

glabel(8)[1349] 类支持 UFS 文件系统的标签类型，基于唯一的文件系统标识 `ufsid`。这些标签可以在 **/dev/ufsid** 中找到，并在系统启动时自动创建。**/etc/fstab** 可以使用 `ufsid` 标签来挂载分区。使用 `glabel status` 来接收文件系统和它们相应的 `ufsid` 标签的列表:

```
% glabel status
                  Name  Status  Components
ufsid/486b6fc38d330916     N/A  ad4s1d
ufsid/486b6fc16926168e     N/A  ad4s1f
```

在上述例子中，**ad4s1d** 代表 **/var**，而 **ad4s1f** 代表 **/usr**。使用所示的 `ufsid` 值，现在可以用 **/etc/fstab** 中的以下条目来挂载这些分区：

```
/dev/ufsid/486b6fc38d330916        /var        ufs        rw        2      2
/dev/ufsid/486b6fc16926168e        /usr        ufs        rw        2      2
```

任何带有 `ufsid` 标签的分区都可以用这种方式挂载，不需要手动创建永久标签，同时还可以享受设备名称独立挂载的好处。

21.8.通过 GEOM 实现 UFS 日志

在 FreeBSD 上，对 UFS 文件系统的日志支持是可用的。该实现是通过 GEOM 子系统提供的，并通过 `gjournal` 进行配置。与其他文件系统的日志实现不同，`gjournal` 方法是基于块的，而不是作为文件系统的一部分来实现。它是一个 GEOM 扩展。

在元数据和文件写入提交到磁盘之前，日志存储了文件系统事务的日志，例如构成一个完整磁盘写入操作的变化。这个事务日志以后可以重放，重做文件系统事务，防止文件系统不一致。

这种方法提供了另一种机制来防止数据丢失和文件系统的不一致。与跟踪和执行元数据更新的软更新和创建文件系统镜像的快照不同，日志是专门为这项任务存储在磁盘空间的。为了提高性能，日志可以存储在另一个磁盘上。在这种配置中，日志 provider 或存储设备应该列在要启用日志的设备之后。

GENERIC 内核提供了对 `gjournal` 的支持。要在启动时自动加载 **geom_journal.ko** 内核模块，请在 **/boot/loader.conf** 中加入以下一行：

[1349] https://www.freebsd.org/cgi/man.cgi?query=glabel&sektion=8&format=html

```
geom_journal_load="YES"
```

如果使用定制内核，确保在内核配置文件中有以下一行：

```
options GEOM_JOURNAL
```

加载该模块之后，就可以通过以下步骤在新的文件系统上创建一个日志。在这个例子中，**da4** 是一个新的 SCSI 磁盘：

```
# gjournal load
# gjournal label /dev/da4
```

这将加载该模块并在 **/dev/da4** 上创建一个设备节点 **/dev/da4.journal**。

现在可以在该日志设备上创建 UFS 文件系统，然后挂载到一个现有的挂载点：

```
# newfs -O 2 -J /dev/da4.journal
# mount /dev/da4.journal /mnt
```

> 注意
>
> 如果有多个 slice，则为每个 slice 都创建一个日志。例如，如果 **ad4s1** 和 **ad4s2** 都是 slice，那么 `gjournal` 就会创建 **ad4s1.journal** 和 **ad4s2.journal**。

也可以通过使用 `tunefs` 在当前的文件系统上启用日志功能。不过，在尝试改变现有文件系统之前，一定要先做备份。在大多数情况下，如果 `gjournal` 无法创建日志，它就会失败，但这并不能防止因误用 `tunefs` 而导致的数据丢失。关于这些命令的更多信息，请参考 gjournal(8)[1350] 和 tunefs(8)[1351]。

可以对 FreeBSD 系统的启动盘进行日记。请参考文章在台式电脑上实现 UFS 日志系统[1352]以了解详细说明。

[1350] https://www.freebsd.org/cgi/man.cgi?query=gjournal&sektion=8&format=html

[1351] https://www.freebsd.org/cgi/man.cgi?query=tunefs&sektion=8&format=html

[1352] https://docs.freebsd.org/en/articles/gjournal-desktop/

第二十二章 Z 文件系统（ZFS）

22.1.是什么使 ZFS 与众不同

ZFS 是一个先进的文件系统，旨在解决以往存储子系统软件中存在的主要问题。

ZFS 最初由 Sun™ 公司开发，正在进行的开源 ZFS 开发已经转移至 OpenZFS 项目[1353]。

ZFS 有三个主要设计目标：

- 数据完整性。所有的数据都包括数据校验码[1354]。ZFS 计算校验和并将其与数据一起写入。当以后读取该数据时，ZFS 重新计算校验和。如果校验和不匹配，意味着检测到一个或多个数据错误，ZFS 将尝试在同位、镜像或同位块可用时自动纠正错误。
- 存储池：将物理存储设备添加到池中，并从该共享池中分配存储空间。空间对所有文件系统和卷都是可用的，并通过向池中添加新的存储设备而增加。
- 性能：缓存机制能提供更高的性能。ARC[1355] 是一个先进的基于内存的读取缓存。ZFS 提供一个带有 L2ARC[1356] 的基于磁盘的二级可读缓存，以及一个名为 ZIL[1357] 的基于磁盘的同步写缓存。

完整的功能和术语列表见 ZFS 功能和术语[1358]。

[1353] http://open-zfs.org/

[1354] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-checksum

[1355] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-arc

[1356] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-l2arc

[1357] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-zil

[1358] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term

22.1.是什么使 ZFS 与众不同

ZFS 不仅仅是一个文件系统，它与传统的文件系统有着本质上的区别。将卷管理器和文件系统这两个传统的独立角色结合起来，为 ZFS 提供独特的优势。现在文件系统知道磁盘的底层结构。传统的文件系统每次只能单独存在于一个磁盘上。如果有两个磁盘，那么就必须创建两个独立的文件系统。可通过传统的硬件 RAID 配置来避免这个问题，它为操作系统提供一个由物理磁盘提供的空间组成的单一逻辑磁盘，操作系统在上面放置一个文件系统。但即使是像 GEOM 提供的软件 RAID 解决方案，在 RAID 上面的 UFS 文件系统也认为它在处理一个单一的设备。ZFS 的卷管理器和文件系统的结合解决这个问题，并允许创建的文件系统都共享一个可用的存储池。ZFS 对物理磁盘分布的了解的一大优势是，当向池中添加额外的磁盘时，现有的文件系统会自动增大。使这些新的空间成为文件系统的可用空间。ZFS 还可以对每个文件系统应用不同的属性。这使得独立的文件系统和数据集的创建成为可能，而非单一的单体文件系统，从而 ZFS 更加实用。

22.2.快速入门指南

FreeBSD 可以在系统初始化时挂载 ZFS 池和数据集。要启用它，请在 `/etc/rc.conf` 中添加这一行：

```
zfs_enable="YES"
```

然后启动该服务：

```
# service zfs start
```

本节中的例子假设有三个 SCSI 磁盘，设备名称分别为 **da0**、**da1** 和 **da2**。SATA 硬件的用户应该使用 **ada** 作为设备名。

22.2.1.单磁盘池

使用单个磁盘设备创建一个简单的、非冗余的池：

```
# zpool create example /dev/da0
```

如需查看新的池，请查看 `df` 的输出：

```
# 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
```

以上输出说明已经创建和挂载 `example` 池，现在可以作为一个文件系统来访问。创建文件供用户浏览：

```
# 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
```

这个池还没有使用任何高级的 ZFS 功能和属性。如需在这个池上创建一个启用压缩功能的数据集：

```
# zfs create example/compressed
# zfs set compression=gzip example/compressed
```

`example/compressed` 数据集现在是一个 ZFS 压缩文件系统。试着复制一些大文件到 **/example/compressed**。

如需禁用压缩：

```
# zfs set compression=off example/compressed
```

如需卸载文件系统，使用 `zfs umount`，然后用 `df` 验证：

```
# 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
```

如需重新挂载文件系统以使其再次被访问，请使用 `zfs mount` 并使用 `df` 验证：

```
# 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
```

运行 `mount` 以显示池和文件系统：

```
# 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/compressed on /example/compressed (zfs, local)
```

创建后像任何文件系统一样使用 ZFS 数据集。需要时，在每个数据集的基础上设置其他可用的特性。下面的例子创建一个名为 `data` 的新文件系统。它假定该文件系统包含重要的文件，并将其配置为存储每个数据块的两个副本：

```
# zfs create example/data
# zfs set copies=2 example/data
```

使用 `df` 来查看数据和空间的使用：

```
# 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
```

注意，池中的所有文件系统都有相同的可用空间。在这些例子中使用 `df` 显示，文件系统使用它们所需要的空间，并且都是从同一个池中提取。ZFS 摆脱了卷和分区等概念，允许多个文件系统共享同一个池。

如果要销毁文件系统并销毁不再需要的池：

```
# zfs destroy example/compressed
# zfs destroy example/data
# zpool destroy example
```

22.2.2.RAID-Z

磁盘故障时，避免磁盘故障造成数据丢失的一个方法是使用 RAID。ZFS 在其池的设计中支持这一功能。RAID-Z 池需要三个或更多的磁盘，但比镜像池提供了更多的可用空间。

这个例子创建一个 RAID-Z 池，指定要添加到池中的磁盘。

```
# zpool create storage raidz da0 da1 da2
```

注意

Sun™ 建议在单个 RAID-Z 配置中使用的设备数量在 3 到 9 之间。对于需要一个由 10 个或更多的磁盘组成的单一磁盘池的环境，可以考虑将其分解成更小的 RAID-Z 组。如果有两个磁盘可用，如果需要的话，ZFS 镜像提供冗余功能。参考 zpool(8)[1359] 以了解更多细节。

前面的例子创建存储池 `zpool`。这个例子在该池中建立一个新的文件系统，名为 `home`：

```
# zfs create storage/home
```

如需启用压缩功能，并存储一个额外的目录和文件副本：

```
# zfs set copies=2 storage/home
# zfs set compression=gzip storage/home
```

为了使这个目录成为用户的新主目录，将用户数据复制到这个目录，并创建适当的符号链接：

```
# cp -rp /home/* /storage/home
# rm -rf /home /usr/home
# ln -s /storage/home /home
# ln -s /storage/home /usr/home
```

用户数据现在存储在新创建的 **/storage/home** 上。通过添加一个新用户并以该用户身份登录来测试。

创建一个文件系统快照，以便日后回滚：

```
# zfs snapshot storage/home@08-30-08
```

ZFS 创建的是数据集的快照，而不是单个目录或文件。

`@` 字符是文件系统名称或卷名称之间的分隔符。在删除一个重要的目录之前，先备份文件系统，然后回滚到该目录仍然存在的早期快照：

```
# zfs rollback storage/home@08-30-08
```

要列出所有可用的快照，在文件系统的 **.zfs/napshot** 目录下运行 `ls`。例如，如需查快照：

```
# ls /storage/home/.zfs/snapshot
```

写一个脚本，定期对用户数据进行快照。随着时间的推移，快照会耗费大量的磁盘空间。使用命令删除之前的快照：

```
# zfs destroy storage/home@08-30-08
```

[1359] https://www.freebsd.org/cgi/man.cgi?query=zpool&sektion=8&format=html

测试后，用这个命令使 **/storage/home** 成为真正的 **/home**：

```
# zfs set mountpoint=/home storage/home
```

运行 `df` 和 `mount` 以确认系统现在将该文件系统视为真正的 **/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
```

这样就完成了 RAID-Z 的配置。通过在 **/etc/periodic.conf** 中添加这一行，在每晚运行的 periodic(8)[1360] 中添加关于已创建文件系统的每日状态更新：

```
daily_status_zfs_enable="YES"
```

22.2.3.恢复 RAID-Z

每个软件 RAID 都有监控其 `state` 的方法。使用以下方法查看 RAID-Z 设备的状态：

```
# zpool status -x
```

如果所有池都是 Online[1361] 的，而且一切正常，则显示消息：

```
all pools are healthy
```

如果有一个问题，也许是一个磁盘处于 Offline[1362] 状态，池的状态将看起来像这样：

```
  pool: storage
 state: DEGRADED
```

(continues on next page)

[1360] https://www.freebsd.org/cgi/man.cgi?query=periodic&sektion=8&format=html

[1361] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-online

[1362] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-offline

(continued from previous page)

```
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
```

OFFLINE 表示管理员使用以下命令将 **da1** 脱机：

```
# zpool offline storage da1
```

现在关闭计算机电源，更换 **da1**。然后打开计算机电源，将新的 **da1** 放回池中：

```
# zpool replace storage da1
```

接下来，再次检查状态，这次不使用 -x 来显示所有池：

```
# 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
```

在这个例子中，一切正常。

22.2.4.数据验证

ZFS 使用校验和来验证存储数据的完整性。创建文件系统时会自动启用该功能。

> 警告
>
> 可以禁用校验和，但不建议这样做。校验和只占用很少的存储空间但可确保数据的完整性。大多数 ZFS 功能在禁用校验和的情况下将不能正常工作。禁用校验和不会明显地提高性能。

验证数据校验（称为 清洗）以确保 `storage` 池的完整性：

```
# zpool scrub storage
```

清洗的时间取决于存储的数据量。较大的数据量将需要相应的时间来验证。由于清洗是 I/O 密集型的，ZFS 一次只允许运行一个清洗。清洗完成后，用 `zpool status` 查看状态：

```
# 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
```

显示上一次清洗的完成日期有助于决定何时开始另一次清洗。例行清洗有助于保护数据免受隐藏的破坏，并确保池的完整性。

参考 zfs(8)[1363] 和 zpool(8)[1364] 以了解其他 ZFS 选项。

[1363] https://www.freebsd.org/cgi/man.cgi?query=zfs&sektion=8&format=html

[1364] https://www.freebsd.org/cgi/man.cgi?query=zfs&sektion=8&format=html

22.3.zpool 管理

ZFS 主要通过两个的工具来进行管理。工具 zpool 控制池的操作，允许添加、删除、替换和管理磁盘。工具 zfs[1365] 允许创建、销毁和管理数据集，包括文件系统[1366]和卷[1367]。

22.3.1.创建和销毁存储池

创建 ZFS 存储池是一个永久性决定，因为池的结构在创建后不能改变。最重要的决定是把物理磁盘分成哪种类型的 vdevs。关于可用选项的细节，请看 vdev 类型[1368]的列表。在创建池之后，大多数类型的 vdev 不允许向 vdev 添加磁盘。镜像和条带是例外，前者允许向 vdev 添加新的磁盘，后者通过将一个新的磁盘附加到 vdev 而升级到镜像。尽管添加新的 vdev 会扩大池，但池的布局在池创建后不能改变。若要更改，需要备份数据，销毁池，然后重新创建。

创建一个简单的镜像池：

```
# zpool create mypool mirror /dev/ada1 /dev/ada2
# zpool status
  pool: mypool
 state: ONLINE
  scan: none requested
config:

        NAME        STATE     READ WRITE CKSUM
        mypool      ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            ada1    ONLINE       0     0     0
            ada2    ONLINE       0     0     0

errors: No known data errors
```

要用一条命令创建多个的 vdev，请指定由 vdev 类型关键字分隔的磁盘组，在这个例子中是 mirror：

```
# zpool create mypool mirror /dev/ada1 /dev/ada2 mirror /dev/ada3 /dev/ada4
# zpool status
  pool: mypool
 state: ONLINE
  scan: none requested
config:
```

(continues on next page)

[1365] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-zfs

[1366] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-filesystem

[1367] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-volume

[1368] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-vdev

(continued from previous page)

```
        NAME        STATE     READ WRITE CKSUM
        mypool      ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            ada1    ONLINE       0     0     0
            ada2    ONLINE       0     0     0
          mirror-1  ONLINE       0     0     0
            ada3    ONLINE       0     0     0
            ada4    ONLINE       0     0     0

errors: No known data errors
```

池也可以使用分区而不是整个磁盘。把 ZFS 放在一个单独的分区里，可允许同一磁盘有其他的分区用于其他目的。特别是，它允许添加启动所需的引导代码和文件系统的分区。这允许从也是一个池的成员的磁盘上启动。当使用一个分区而不是整个磁盘时，ZFS 在 FreeBSD 上没有增加任何性能损失。使用分区还允许管理员对磁盘进行 低配置，使用少于全部容量的磁盘。如果将来有一个与原来的名义大小相同的替换盘，实际的容量略小，那么较小的分区仍然适合使用替换盘。

使用分区创建一个 RAID-Z2[1369] 池：

```
# zpool create mypool raidz2 /dev/ada0p3 /dev/ada1p3 /dev/ada2p3 /dev/ada3p3 /dev/
↪ada4p3 /dev/ada5p3
# zpool status
  pool: mypool
 state: ONLINE
  scan: none requested
config:

        NAME        STATE     READ WRITE CKSUM
        mypool      ONLINE       0     0     0
          raidz2-0  ONLINE       0     0     0
            ada0p3  ONLINE       0     0     0
            ada1p3  ONLINE       0     0     0
            ada2p3  ONLINE       0     0     0
            ada3p3  ONLINE       0     0     0
            ada4p3  ONLINE       0     0     0
            ada5p3  ONLINE       0     0     0

errors: No known data errors
```

可以销毁不再需要的池，以重新使用磁盘。销毁池需要先🄵载该池中的文件系统。如果有任何数据集正在使用，🄵载操作就会失败，且不会销毁这个池。用 `-f` 可强制销毁池。这可能会导致在这些数据集上有已打开文件的应用程序的未定义行为。

[1369] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-vdev-raidz

22.3.2.添加和删除设备

有两种方法可以将磁盘添加到池中：用 `zpool attach` 将磁盘附加到现有的 vdev 上，或者用 `zpool add` 将 vdev 添加到池中。有些 vdev 类型[1370] 允许在创建后向 vdev 添加磁盘。

用单个磁盘创建的池缺乏冗余性。它可以检测到损坏，但不能修复，因为没有其他的数据副本。副本[1371] 属性可能能够从一个小的故障中恢复，如坏扇区，但不能提供与镜像或 RAID-Z 相同的保护水平。从一个由单个磁盘 vdev 组成的池开始，使用 `zpool attach` 向 vdev 添加一个新磁盘，创建一个镜像。也可以使用 `zpool attach` 向镜像组添加新的磁盘，增加冗余度和读取性能。当对用于池的磁盘进行分区时，将第一个磁盘的布局复制到第二个磁盘上。使用 `gpart backup` 和 `gpart restore` 来使这个过程更容易。

通过连接 **ada1p3** 将单磁盘（stripe）vdev **ada0p3** 升级为镜像：

```
# zpool status
  pool: mypool
 state: ONLINE
  scan: none requested
config:

        NAME        STATE     READ WRITE CKSUM
        mypool      ONLINE       0     0     0
          ada0p3    ONLINE       0     0     0

errors: No known data errors
# zpool attach mypool ada0p3 ada1p3
Make sure to wait until resilvering finishes before rebooting.

If you boot from pool 'mypool', you may need to update boot code on newly attached␣
↪disk _ada1p3_.

Assuming you use GPT partitioning and _da0_ is your new boot disk you may use the␣
↪following command:

        gpart bootcode -b /boot/pmbr -p /boot/gptzfsboot -i 1 da0
# gpart bootcode -b /boot/pmbr -p /boot/gptzfsboot -i 1 ada1
bootcode written to ada1
# zpool status
  pool: mypool
 state: ONLINE
status: One or more devices is currently being resilvered.  The pool will
        continue to function, possibly in a degraded state.
action: Wait for the resilver to complete.
```

(continues on next page)

[1370] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-vdev

[1371] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-copies

(continued from previous page)

```
  scan: resilver in progress since Fri May 30 08:19:19 2014
        527M scanned out of 781M at 47.9M/s, 0h0m to go
        527M resilvered, 67.53% done
config:

        NAME        STATE     READ WRITE CKSUM
        mypool      ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            ada0p3  ONLINE       0     0     0
            ada1p3  ONLINE       0     0     0  (resilvering)

errors: No known data errors
# zpool status
  pool: mypool
 state: ONLINE
  scan: resilvered 781M in 0h0m with 0 errors on Fri May 30 08:15:58 2014
config:

        NAME        STATE     READ WRITE CKSUM
        mypool      ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            ada0p3  ONLINE       0     0     0
            ada1p3  ONLINE       0     0     0

errors: No known data errors
```

当向现有的 vdev 添加磁盘不是一种选择，如 RAID-Z，另一种方法是向池中添加另一个 vdev。增加 vdevs 可以通过在 vdevs 上分配写操作而提供更高的性能。每个 vdev 都提供自己的冗余。混合 vdev 类型如镜像和 RAID-Z 是可能的，但不建议这样做。在一个包含镜像或 RAID-Z vdevs 的池中添加一个非冗余的 vdev，会给整个池的数据带来风险。分散写入意味着非冗余磁盘的故障将导致写入池中的每个块的一部分丢失。

ZFS 在每一个 vdevs 上进行数据条带化。例如，对于两个镜像 vdevs，这实际上是一个 RAID 10，它在两组镜像上进行条带化写入。ZFS 分配空间，使每个 vdev 在同一时间达到 100% 满载。拥有不同数量的可用空间的 vdev 将降低性能，因为更多的数据写入到未满的 vdev 上。

当把新设备连接到启动池时，记得要更新引导代码。

在现有的镜像上附加第二个镜像组（**ada2p3** 和 **ada3p3**）：

```
# zpool status
  pool: mypool
 state: ONLINE
  scan: resilvered 781M in 0h0m with 0 errors on Fri May 30 08:19:35 2014
config:
```

(continues on next page)

(continued from previous page)

```
        NAME        STATE     READ WRITE CKSUM
        mypool      ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            ada0p3  ONLINE       0     0     0
            ada1p3  ONLINE       0     0     0

errors: No known data errors
# zpool add mypool mirror ada2p3 ada3p3
# gpart bootcode -b /boot/pmbr -p /boot/gptzfsboot -i 1 ada2
bootcode written to ada2
# gpart bootcode -b /boot/pmbr -p /boot/gptzfsboot -i 1 ada3
bootcode written to ada3
# zpool status
  pool: mypool
 state: ONLINE
  scan: scrub repaired 0 in 0h0m with 0 errors on Fri May 30 08:29:51 2014
config:

        NAME        STATE     READ WRITE CKSUM
        mypool      ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            ada0p3  ONLINE       0     0     0
            ada1p3  ONLINE       0     0     0
          mirror-1  ONLINE       0     0     0
            ada2p3  ONLINE       0     0     0
            ada3p3  ONLINE       0     0     0

errors: No known data errors
```

如果有足够的剩余冗余，从池中移除 vdevs 是不可能的，而从镜像中移除磁盘是代价高昂的。如果在一个镜像组中只保留一个磁盘，该组就不再是镜像，而成为条带，如果剩余的磁盘发生故障，整个池就会有风险。

如需从三路镜像组中删除一个磁盘：

```
# zpool status
  pool: mypool
 state: ONLINE
  scan: scrub repaired 0 in 0h0m with 0 errors on Fri May 30 08:29:51 2014
config:

        NAME        STATE     READ WRITE CKSUM
```

(continues on next page)

(continued from previous page)

```
        mypool      ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            ada0p3  ONLINE       0     0     0
            ada1p3  ONLINE       0     0     0
            ada2p3  ONLINE       0     0     0

errors: No known data errors
# zpool detach mypool ada2p3
# zpool status
  pool: mypool
 state: ONLINE
  scan: scrub repaired 0 in 0h0m with 0 errors on Fri May 30 08:29:51 2014
config:

        NAME        STATE     READ WRITE CKSUM
        mypool      ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            ada0p3  ONLINE       0     0     0
            ada1p3  ONLINE       0     0     0

errors: No known data errors
```

22.3.3.检查池的状态

池的状态是很重要的。如果一个磁盘脱机或 ZFS 检测到一个读、写或校验错误，相应的错误计数会增加。`status` 输出显示池中每个设备的配置和状态，以及整个池的状态。还显示了要采取的操作和关于最后一次清洗[1372] 的详细信息。

```
# zpool status
  pool: mypool
 state: ONLINE
  scan: scrub repaired 0 in 2h25m with 0 errors on Sat Sep 14 04:25:50 2013
config:

        NAME        STATE     READ WRITE CKSUM
        mypool      ONLINE       0     0     0
          raidz2-0  ONLINE       0     0     0
            ada0p3  ONLINE       0     0     0
            ada1p3  ONLINE       0     0     0
            ada2p3  ONLINE       0     0     0
```

(continues on next page)

[1372] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-zpool-scrub

(continued from previous page)

```
            ada3p3  ONLINE       0     0     0
            ada4p3  ONLINE       0     0     0
            ada5p3  ONLINE       0     0     0

errors: No known data errors
```

22.3.4.清除错误

当检测到一个错误时，ZFS 会增加读取、写入或校验的错误计数。用 `zpool clear mypool` 来清除错误信息并重置计数。清除错误状态对自动脚本很重要，这些脚本在池遇到错误的时候会提醒管理员。如果不清除旧的错误，脚本可能无法报告进一步的错误。

22.3.5.替换一个正常工作的设备

用不同的磁盘替换一个磁盘可能是可取的。当替换一个工作磁盘时，该过程在替换期间保持旧磁盘在线。池永远不会进入降级[1373] 状态，减少数据丢失的风险。运行 `zpool replace` 以将数据从旧磁盘复制到新磁盘。在操作完成后，ZFS 将旧磁盘从 vdev 上断开连接。如果新的磁盘比旧的磁盘大，可能有可能使用新的空间来扩大 zpool。请参阅 扩大池[1374]。

替换池中一个正常工作的设备：

```
# zpool status
  pool: mypool
 state: ONLINE
  scan: none requested
config:

        NAME        STATE     READ WRITE CKSUM
        mypool      ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            ada0p3  ONLINE       0     0     0
            ada1p3  ONLINE       0     0     0

errors: No known data errors
# zpool replace mypool ada1p3 ada2p3
Make sure to wait until resilvering finishes before rebooting.

When booting from the pool 'zroot', update the boot code on the newly attached disk
↪'ada2p3'.
```

(continues on next page)

[1373] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-degraded

[1374] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-zpool-online

(continued from previous page)

```
Assuming GPT partitioning is used and [.filename]#da0# is the new boot disk, use the␣
↪following command:

        gpart bootcode -b /boot/pmbr -p /boot/gptzfsboot -i 1 da0
# gpart bootcode -b /boot/pmbr -p /boot/gptzfsboot -i 1 ada2
# zpool status
  pool: mypool
 state: ONLINE
status: One or more devices is currently being resilvered.  The pool will
        continue to function, possibly in a degraded state.
action: Wait for the resilver to complete.
  scan: resilver in progress since Mon Jun  2 14:21:35 2014
        604M scanned out of 781M at 46.5M/s, 0h0m to go
        604M resilvered, 77.39% done
config:

        NAME             STATE     READ WRITE CKSUM
        mypool           ONLINE       0     0     0
          mirror-0       ONLINE       0     0     0
            ada0p3       ONLINE       0     0     0
            replacing-1  ONLINE       0     0     0
              ada1p3     ONLINE       0     0     0
              ada2p3     ONLINE       0     0     0  (resilvering)

errors: No known data errors
# zpool status
  pool: mypool
 state: ONLINE
  scan: resilvered 781M in 0h0m with 0 errors on Mon Jun  2 14:21:52 2014
config:

        NAME        STATE     READ WRITE CKSUM
        mypool      ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            ada0p3  ONLINE       0     0     0
            ada2p3  ONLINE       0     0     0

errors: No known data errors
```

22.3.6.处理故障设备

当池中的一个磁盘发生故障时，该磁盘所属的 vdev 会进入降级[1375] 状态。数据仍然可用，但性能下降，因为 ZFS 从可用的冗余中计算缺失的数据。为了将 vdev 恢复到完全功能的状态，替换故障的物理设备。然后 ZFS 被指示开始 resilver[1376] 操作。ZFS 从可用的冗余中重新计算故障设备上的数据，并将其写入替换设备中。完成后，vdev 返回到在线状态。

如果 vdev 没有任何冗余，或者如果设备已经故障，并且没有足够的冗余来补偿，池就会进入故障[1377]状态。除非有足够的设备可以重新连接，否则这个池就无法运行，需要从备份中进行数据恢复。

当替换一个故障磁盘时，故障磁盘的名称将变为新磁盘的 GUID。如果替换的设备具有相同的设备名称，则无需为 `zpool replace` 提供新的设备名称参数。

使用 `zpool replace` 替换一个故障磁盘：

```
# zpool status
  pool: mypool
 state: DEGRADED
status: One or more devices could not be opened.  Sufficient replicas exist for
        the pool to continue functioning in a degraded state.
action: Attach the missing device and online it using 'zpool online'.
   see: http://illumos.org/msg/ZFS-8000-2Q
  scan: none requested
config:

        NAME                     STATE     READ WRITE CKSUM
        mypool                   DEGRADED     0     0     0
          mirror-0               DEGRADED     0     0     0
            ada0p3               ONLINE       0     0     0
            316502962686821739   UNAVAIL      0     0     0  was /dev/ada1p3

errors: No known data errors
# zpool replace mypool 316502962686821739 ada2p3
# zpool status
  pool: mypool
 state: DEGRADED
status: One or more devices is currently being resilvered.  The pool will
        continue to function, possibly in a degraded state.
action: Wait for the resilver to complete.
  scan: resilver in progress since Mon Jun  2 14:52:21 2014
        641M scanned out of 781M at 49.3M/s, 0h0m to go
        640M resilvered, 82.04% done
```

(continues on next page)

[1375] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-degraded
[1376] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-resilver
[1377] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-faulted

(continued from previous page)

```
config:

        NAME                          STATE     READ WRITE CKSUM
        mypool                        DEGRADED     0     0     0
          mirror-0                    DEGRADED     0     0     0
            ada0p3                    ONLINE       0     0     0
            replacing-1               UNAVAIL      0     0     0
              15732067398082357289    UNAVAIL      0     0     0  was /dev/ada1p3/old
              ada2p3                  ONLINE       0     0     0  (resilvering)

errors: No known data errors
# zpool status
  pool: mypool
 state: ONLINE
  scan: resilvered 781M in 0h0m with 0 errors on Mon Jun  2 14:52:38 2014
config:

        NAME        STATE     READ WRITE CKSUM
        mypool      ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            ada0p3  ONLINE       0     0     0
            ada2p3  ONLINE       0     0     0

errors: No known data errors
```

22.3.7.刷新池

定期清洗[1378] 池，最好是每月一次以上。`scrub` 操作是磁盘密集型的，在运行中会降低性能。在调度 `scrub` 时要避开高需求期，或者使用 `vfs.zfs.scrub_delay <https://docs.freebsd.org/en/books/handbook/zfs/#zfs-advanced-tuning-scrub_delay>`__ 来调整 `scrub` 的相对优先级，以免拖累其他工作负载。

```
# zpool scrub mypool
# zpool status
  pool: mypool
 state: ONLINE
  scan: scrub in progress since Wed Feb 19 20:52:54 2014
        116G scanned out of 8.60T at 649M/s, 3h48m to go
        0 repaired, 1.32% done
config:
```

(continues on next page)

[1378] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-scrub

(continued from previous page)

```
        NAME        STATE     READ WRITE CKSUM
        mypool      ONLINE       0     0     0
          raidz2-0  ONLINE       0     0     0
            ada0p3  ONLINE       0     0     0
            ada1p3  ONLINE       0     0     0
            ada2p3  ONLINE       0     0     0
            ada3p3  ONLINE       0     0     0
            ada4p3  ONLINE       0     0     0
            ada5p3  ONLINE       0     0     0

errors: No known data errors
```

如果需要，可以运行 `zpool scrub -s mypool` 以取消刷新操作。

22.3.8.自我修复

与数据块一起存储的校验和使文件系统能够 *自我修复*。这个功能将自动修复那些校验和与另一个属于存储池的设备上记录的校验和不一致的数据。例如，有两个磁盘的镜像配置，其中一个磁盘开始出现故障，无法再正常存储数据。当数据很长时间没有被访问时，这种情况就更严重了，就像长期的档案存储一样。传统的文件系统需要运行检查和修复数据的命令，如 fsck(8)[1379]。这些命令需要时间，而且在严重的情况下，管理员必须决定执行哪种修复操作。当 ZFS 检测到一个具有不匹配校验和的数据块时，它试图从镜像磁盘读取数据。如果该磁盘能够提供正确的数据，ZFS 将把它交给应用程序，并纠正磁盘上有错误校验和的数据。在正常的池操作中，这种情况无需系统管理员的任何互动。

下一个例子通过创建一个 **/dev/ada0** 和 **/dev/ada1** 磁盘的镜像池来展示这种自修复行为：

```
# zpool create healer mirror /dev/ada0 /dev/ada1
# zpool status healer
  pool: healer
 state: ONLINE
  scan: none requested
config:

    NAME        STATE     READ WRITE CKSUM
    healer      ONLINE       0     0     0
      mirror-0  ONLINE       0     0     0
       ada0     ONLINE       0     0     0
       ada1     ONLINE       0     0     0

errors: No known data errors
# zpool list
```

(continues on next page)

[1379] https://www.freebsd.org/cgi/man.cgi?query=fsck&sektion=8&format=html

(continued from previous page)

```
NAME     SIZE  ALLOC   FREE   CKPOINT  EXPANDSZ   FRAG   CAP  DEDUP  HEALTH  ALTROOT
healer   960M  92.5K   960M        -         -     0%    0%  1.00x  ONLINE  -
```

将一些重要的数据复制到池中，利用自修复功能防止数据错误，并创建池的校验和，以便以后进行比较：

```
# cp /some/important/data /healer
# zfs list
NAME     SIZE  ALLOC   FREE    CAP  DEDUP  HEALTH  ALTROOT
healer   960M  67.7M   892M     7%  1.00x  ONLINE  -
# sha1 /healer > checksum.txt
# cat checksum.txt
SHA1 (/healer) = 2753eff56d77d9a536ece6694bf0a82740344d1f
```

通过向镜像中的一个磁盘的开头写入随机数据来模拟数据损坏。为了防止 ZFS 在检测到数据时进行修复，在损坏之前导出池，之后再导入。

> 警告
>
> 这是一个危险的操作，可能会破坏重要的数据，这里仅作示范。在存储池的正常操作中，不要尝试它。这个故意破坏的例子也不应该在任何不使用 ZFS 文件系统的磁盘或分区上运行。不要使用不在池中的任何其他磁盘设备名称。在运行该命令之前，请确保有适当的存储池备份，并对其进行测试！

```
# zpool export healer
# dd if=/dev/random of=/dev/ada1 bs=1m count=200
200+0 records in
200+0 records out
209715200 bytes transferred in 62.992162 secs (3329227 bytes/sec)
# zpool import healer
```

pool status 显示，有一个设备出现错误。请注意，从池中读取数据的应用程序并没有收到任何不正确的数据。ZFS 从 **ada0** 设备提供具有正确校验和的数据。要找到有错误校验和的设备，寻找其 CKSUM 列包含一个非零值的设备。

```
# zpool status healer
    pool: healer
   state: ONLINE
  status: One or more devices has experienced an unrecoverable error.  An
          attempt was made to correct the error.  Applications are unaffected.
  action: Determine if the device needs to be replaced, and clear the errors
          using 'zpool clear' or replace the device with 'zpool replace'.
     see: http://illumos.org/msg/ZFS-8000-4J
    scan: none requested
```

(continues on next page)

(continued from previous page)

```
  config:

      NAME        STATE     READ WRITE CKSUM
      healer      ONLINE       0     0     0
        mirror-0  ONLINE       0     0     0
          ada0    ONLINE       0     0     0
          ada1    ONLINE       0     0     1

errors: No known data errors
```

ZFS 检测到这个错误，并通过使用未受影响的 **ada0** 镜像盘中存在的冗余来处理它。与原始磁盘的校验和比较将揭示池是否再次一致：

```
# sha1 /healer >> checksum.txt
# cat checksum.txt
SHA1 (/healer) = 2753eff56d77d9a536ece6694bf0a82740344d1f
SHA1 (/healer) = 2753eff56d77d9a536ece6694bf0a82740344d1f
```

在故意篡改之前和之后生成校验和，而池的数据仍然匹配。这表明当校验和不同时，ZFS 能够自动检测并纠正任何错误。注意这是在池中存在足够的冗余的情况下才可能实现的。一个由单一设备组成的池没有自我修复的能力。这也是为什么校验和在 ZFS 中如此重要的原因；不要以任何理由禁用它们。ZFS 不需要 fsck(8)[1380] 或类似的文件系统一致性检查程序来检测和纠正这一点，并可在有问题时保持池的可用性。现在需要一次清洗操作来覆盖 **ada1** 上的损坏的数据。

```
# zpool scrub healer
# zpool status healer
  pool: healer
 state: ONLINE
status: One or more devices has experienced an unrecoverable error.  An
            attempt was made to correct the error.  Applications are unaffected.
action: Determine if the device needs to be replaced, and clear the errors
            using 'zpool clear' or replace the device with 'zpool replace'.
   see: http://illumos.org/msg/ZFS-8000-4J
  scan: scrub in progress since Mon Dec 10 12:23:30 2012
        10.4M scanned out of 67.0M at 267K/s, 0h3m to go
        9.63M repaired, 15.56% done
config:

    NAME        STATE     READ WRITE CKSUM
    healer      ONLINE       0     0     0
      mirror-0  ONLINE       0     0     0
```

(continues on next page)

[1380] https://www.freebsd.org/cgi/man.cgi?query=fsck&sektion=8&format=html

(continued from previous page)

```
       ada0     ONLINE       0     0     0
       ada1     ONLINE       0     0   627  (repairing)
```

清洗操作从 **ada0** 读取数据，并在 **ada1** 上重写任何有错误校验的数据，由 `zpool status` 的 `(repairing)` 输出显示。操作完成后，池的状态变成：

```
# zpool status healer
  pool: healer
 state: ONLINE
status: One or more devices has experienced an unrecoverable error.  An
        attempt was made to correct the error.  Applications are unaffected.
action: Determine if the device needs to be replaced, and clear the errors
             using 'zpool clear' or replace the device with 'zpool replace'.
   see: http://illumos.org/msg/ZFS-8000-4J
  scan: scrub repaired 66.5M in 0h2m with 0 errors on Mon Dec 10 12:26:25 2012
config:

    NAME        STATE     READ WRITE CKSUM
    healer      ONLINE       0     0     0
      mirror-0  ONLINE       0     0     0
       ada0     ONLINE       0     0     0
       ada1     ONLINE       0     0 2.72K

errors: No known data errors
```

在清洗操作完成后，所有数据会从 **ada0** 同步到 **ada1**，通过运行 `zpool clear` 清除[1381] 池状态中的错误信息。

```
# zpool clear healer
# zpool status healer
  pool: healer
 state: ONLINE
  scan: scrub repaired 66.5M in 0h2m with 0 errors on Mon Dec 10 12:26:25 2012
config:

    NAME        STATE     READ WRITE CKSUM
    healer      ONLINE       0     0     0
      mirror-0  ONLINE       0     0     0
       ada0     ONLINE       0     0     0
       ada1     ONLINE       0     0     0
```

(continues on next page)

[1381] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-zpool-clear

(continued from previous page)

```
errors: No known data errors
```

现在池已经恢复到完全工作的状态，所有的错误计数都是零。

22.3.9.扩大池

每个 vdev 中最小的设备限制冗余池的可用大小。用一个更大的设备替换最小的设备。在完成替换[1382] 或 resilver[1383] 操作后，池就可以增长到使用新设备的容量。以一个 1TB 磁盘和一个 2TB 磁盘组成的镜像为例。可用的空间是 1TB。当用另一个 2TB 的磁盘替换 1TB 的磁盘时，resilver 程序会将现有数据复制到新的磁盘上。由于这两台设备现在都有 2TB 的容量，镜像的可用空间增长到 2TB。

通过在每个设备上使用 `zpool online -e` 开始扩展。在扩展所有设备后，额外的空间就成为池的可用空间。

22.3.10.导入和导出池

在把池移到另一个系统之前，先 导出池。ZFS 解除对所有数据集的挂载，将每个设备标记为已导出但仍被锁定，以防止被其他磁盘使用。这允许在其他机器、其他支持 ZFS 的操作系统、甚至不同的硬件架构上 导入池（有一些注意事项，见 zpool(8)[1384]）。当一个数据集有已打开的文件时，使用 `zpool export -f` 来强制导出池。使用这个方法时要注意。数据集被强制卸载，可能会导致在这些数据集上有已打开文件的应用程序出现意外行为。

导出一个不使用的池：

```
# zpool export mypool
```

导入一个池会自动挂载数据集。`zpool import -o` 为这个特定的导入设置临时属性。`zpool import altroot=` 允许导入一个具有基本挂载点而不是文件系统根目录的池。如果这个池最后是在不同的系统上使用，并且没有被正确导出，请使用 `zpool import -f` 强制导入。`zpool import -a` 会导入所有看起来没有被其他系统使用的池。

列出所有可供导入的池：

```
# zpool import
   pool: mypool
     id: 9930174748043525076
  state: ONLINE
 action: The pool can be imported using its name or numeric identifier.
 config:
```

(continues on next page)

[1382] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-zpool-replace

[1383] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-resilver

[1384] https://www.freebsd.org/cgi/man.cgi?query=zpool&sektion=8&format=html

(continued from previous page)

```
        mypool      ONLINE
          ada2p3    ONLINE
```

用另一个根目录导入池。

```
# zpool import -o altroot=/mnt mypool
# zfs list
zfs list
NAME                 USED  AVAIL  REFER  MOUNTPOINT
mypool               110K  47.0G    31K  /mnt/mypool
```

22.3.11.升级存储池

在升级 FreeBSD 之后，或者从使用旧版本的系统中导入一个池，请手动将池升级到最新的 ZFS 版本，以支持更多的新功能。在升级之前，请考虑是否需要在旧系统上导入这个池。升级是一个单向的过程。可以升级较旧的池，但是不可以降级具有较新功能的池。

升级池到 v28 以支持 `Feature Flags`。

```
# zpool status
  pool: mypool
 state: ONLINE
status: The pool is formatted using a legacy on-disk format.  The pool can
        still be used, but some features are unavailable.
action: Upgrade the pool using 'zpool upgrade'.  Once this is done, the
        pool will no longer be accessible on software that does not support feat
        flags.
  scan: none requested
config:

        NAME        STATE     READ WRITE CKSUM
        mypool      ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
        ada0    ONLINE       0     0     0
        ada1    ONLINE       0     0     0

errors: No known data errors
# zpool upgrade
This system supports ZFS pool feature flags.

The following pools are formatted with legacy version numbers and are upgraded to use␣
```

(continues on next page)

(continued from previous page)

```
↪feature flags.
After being upgraded, these pools will no longer be accessible by software that does␣
↪not support feature flags.

VER  POOL
---  ------------
28   mypool

Use 'zpool upgrade -v' for a list of available legacy versions.
Every feature flags pool has all supported features enabled.
# zpool upgrade mypool
This system supports ZFS pool feature flags.

Successfully upgraded 'mypool' from version 28 to feature flags.
Enabled the following features on 'mypool':
  async_destroy
  empty_bpobj
  lz4_compress
  multi_vdev_crash_dump
```

在 `zpool upgrade` 完成之前，ZFS 的较新功能将无法使用。使用 `zpool upgrade -v` 查看升级提供哪些新功能，以及哪些功能已经被支持。

升级池以支持新的 feature flags：

```
# zpool status
  pool: mypool
 state: ONLINE
status: Some supported features are not enabled on the pool. The pool can
        still be used, but some features are unavailable.
action: Enable all features using 'zpool upgrade'. Once this is done,
        the pool may no longer be accessible by software that does not support
        the features. See zpool-features(7) for details.
  scan: none requested
config:

        NAME        STATE     READ WRITE CKSUM
        mypool      ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
        ada0    ONLINE       0     0     0
        ada1    ONLINE       0     0     0

errors: No known data errors
```

(continues on next page)

(continued from previous page)

```
# zpool upgrade
This system supports ZFS pool feature flags.

All pools are formatted using feature flags.

Some supported features are not enabled on the following pools. Once a
feature is enabled the pool may become incompatible with software
that does not support the feature. See zpool-features(7) for details.

POOL  FEATURE
---------------
zstore
      multi_vdev_crash_dump
      spacemap_histogram
      enabled_txg
      hole_birth
      extensible_dataset
      bookmarks
      filesystem_limits
# zpool upgrade mypool
This system supports ZFS pool feature flags.

Enabled the following features on 'mypool':
  spacemap_histogram
  enabled_txg
  hole_birth
  extensible_dataset
  bookmarks
  filesystem_limits
```

警告

在从池中启动的系统上更新引导代码，以支持新的池版本。在包含引导代码的分区上使用 `gpart bootcode`。根据系统引导的方式，有两种类型的引导码可用。GPT（最常见的选项）和 EFI（用于更现代的系统）。

对于使用 GPT 的传统启动，使用以下命令：

```
# gpart bootcode -b /boot/pmbr -p /boot/gptzfsboot -i 1 ada1
```

对于使用 EFI 启动的系统，执行以下命令：

```
# gpart bootcode -p /boot/boot1.efifat -i 1 ada1
```

将启动代码应用于池中的所有可启动磁盘。参见 gpart(8)[1385] 以了解更多信息。

[1385] https://www.freebsd.org/cgi/man.cgi?query=gpart&sektion=8&format=html

22.3.12.显示已记录的池历史

ZFS 记录改变池的命令，包括创建数据集、改变属性或替换磁盘。回顾池的创建历史是很有用的，就像检查哪个用户在什么时候执行一个特定的动作一样。历史记录不保存在日志文件中，而是池本身的一部分。查看这个历史的命令被恰当地命名为 `zpool history`：

```
# zpool history
History for 'tank':
2013-02-26.23:02:35 zpool create tank mirror /dev/ada0 /dev/ada1
2013-02-27.18:50:58 zfs set atime=off tank
2013-02-27.18:51:09 zfs set checksum=fletcher4 tank
2013-02-27.18:51:18 zfs create tank/backup
```

输出显示以某种方式改变池的 `zpool` 和 `zfs` 命令，以及一个时间戳。像 `zfs list` 这样的命令不包括在内。当没有指定池的名称时，ZFS 显示所有池的历史。

当提供 `-i` 或 `-l` 选项时，`zpool history` 可以显示更多信息。`-i` 显示用户发起的事件以及内部记录的 ZFS 事件。

```
# zpool history -i
History for 'tank':
2013-02-26.23:02:35 [internal pool create txg:5] pool spa 28; zfs spa 28; zpl 5;uts ␣
↪9.1-RELEASE 901000 amd64
2013-02-27.18:50:53 [internal property set txg:50] atime=0 dataset = 21
2013-02-27.18:50:58 zfs set atime=off tank
2013-02-27.18:51:04 [internal property set txg:53] checksum=7 dataset = 21
2013-02-27.18:51:09 zfs set checksum=fletcher4 tank
2013-02-27.18:51:13 [internal create txg:55] dataset = 39
2013-02-27.18:51:18 zfs create tank/backup
```

通过添加 `-l` 显示更多细节。以显示长格式的历史记录，包括执行命令的用户名称和发生变化的主机名等信息。

```
# zpool history -l
History for 'tank':
2013-02-26.23:02:35 zpool create tank mirror /dev/ada0 /dev/ada1 [user 0 (root) on␣
↪:global]
2013-02-27.18:50:58 zfs set atime=off tank [user 0 (root) on myzfsbox:global]
2013-02-27.18:51:09 zfs set checksum=fletcher4 tank [user 0 (root) on myzfsbox:global]
2013-02-27.18:51:18 zfs create tank/backup [user 0 (root) on myzfsbox:global]
```

输出显示，`root` 用户用磁盘 **/dev/ada0** 和 **/dev/ada1** 创建了镜像池。主机名 `myzfsbox` 也显示在池创建后的命令中。当从一个系统导出池并导入另一个系统时，主机名的显示变得很重要。可以通过每条命令记录的主机名来区分其他系统上执行的命令。

将这两个选项合并到 `zpool history` 中，可以为任何给定的池提供最详细的信息。当追踪所执行的操作或需要更详细的输出进行调试时，池的历史记录提供宝贵的信息。

22.3.13.性能监控

一个内置的监控系统可以实时显示池的 I/O 统计数据。它显示池上的可用空间和使用空间的数量，每秒执行的读写操作，以及使用的 I/O 带宽。默认情况下，ZFS 监控并显示系统中的所有池。提供一个池的名字来限制对该池的监控。一个基本的例子：

```
# zpool iostat
               capacity     operations    bandwidth
pool        alloc   free   read  write   read  write
----------  -----  -----  -----  -----  -----  -----
data         288G  1.53T      2     11  11.3K  57.1K
```

要看到连续的 I/O 活动，可以指定一个数字作为最后一个参数，表示更新之间的间隔时间，单位是秒。每隔一段时间就会打印出下一个统计行。按 `Ctrl+C` 停止这种连续监控。在间隔时间之后，在命令行中给出第二个数字，以指定要显示的统计数字的总数。

用 `-v` 显示更详细的 I/O 统计数据。池中的每个设备都会出现一个统计行。这对于查看每个设备上执行的读写操作很有用，可以帮助确定是否有任何单个设备拖累池子。这个例子显示了一个有两个设备的镜像池：

```
# zpool iostat -v
                            capacity     operations    bandwidth
pool                      alloc   free   read  write   read  write
------------------------  -----  -----  -----  -----  -----  -----
data                       288G  1.53T      2     12  9.23K  61.5K
  mirror                   288G  1.53T      2     12  9.23K  61.5K
    ada1                      -      -      0      4  5.61K  61.7K
    ada2                      -      -      1      4  5.04K  61.7K
------------------------  -----  -----  -----  -----  -----  -----
```

22.3.14.分割一个存储池

ZFS 可以将一个由一个或多个镜像 vdev 组成的池分割成两个池。除非另有规定，否则 ZFS 将分离每个镜像的最后一个成员，并创建一个包含相同数据的新池。请确保先用 `-n` 对该操作进行模拟运行。这将显示所请求的操作的细节，而不实际执行。这有助于确认该操作将完成用户的意图。

22.4.zfs 管理

使用 `zfs` 工具可以创建、销毁和管理池中所有现有的 ZFS 数据集。要管理池本身，请使用 zpool[1386]。

22.4.1.创建和销毁数据集

与传统的磁盘和卷管理器不同，ZFS 中的空间不是预先分配的。在传统的文件系统中，在分区和分配空间之后，如果不增加一个新的磁盘，就没有办法增加一个新的文件系统。使用 ZFS，创建新的文件系统在任何时候都是可能的。每个数据集[1387] 的属性包括压缩、重复数据删除、缓存和配额等功能，以及其他有用的属性，如只读、大小写敏感性、网络文件共享和挂载点。数据集之间的嵌套是可能的，子数据集将继承其祖先的属性。委托[1388]、复制[1389]、快照[1390]、jail[1391] 允许将每个数据集作为一个单元进行管理和销毁。为每个不同类型或一组文件创建一个单独的数据集有其优势。拥有大量的数据集的缺点是一些命令，如 `zfs list` 会比较慢，而且挂载数百甚至数千的数据集会减慢 FreeBSD 的引导过程。

创建一个新的数据集并启用 LZ4 压缩[1392]：

```
# zfs list
NAME                  USED  AVAIL  REFER  MOUNTPOINT
mypool                781M  93.2G   144K  none
mypool/ROOT           777M  93.2G   144K  none
mypool/ROOT/default   777M  93.2G   777M  /
mypool/tmp            176K  93.2G   176K  /tmp
mypool/usr            616K  93.2G   144K  /usr
mypool/usr/home       184K  93.2G   184K  /usr/home
mypool/usr/ports      144K  93.2G   144K  /usr/ports
mypool/usr/src        144K  93.2G   144K  /usr/src
mypool/var           1.20M  93.2G   608K  /var
mypool/var/crash      148K  93.2G   148K  /var/crash
mypool/var/log        178K  93.2G   178K  /var/log
mypool/var/mail       144K  93.2G   144K  /var/mail
mypool/var/tmp        152K  93.2G   152K  /var/tmp
# zfs create -o compress=lz4 mypool/usr/mydataset
# zfs list
NAME                   USED  AVAIL  REFER  MOUNTPOINT
mypool                 781M  93.2G   144K  none
mypool/ROOT            777M  93.2G   144K  none
```

(continues on next page)

[1386] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-zpool
[1387] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-dataset
[1388] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-zfs-allow
[1389] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-zfs-send
[1390] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-zfs-snapshot
[1391] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-zfs-jail
[1392] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-compression-lz4

(continued from previous page)

```
mypool/ROOT/default    777M  93.2G   777M  /
mypool/tmp             176K  93.2G   176K  /tmp
mypool/usr             704K  93.2G   144K  /usr
mypool/usr/home        184K  93.2G   184K  /usr/home
mypool/usr/mydataset  87.5K  93.2G  87.5K  /usr/mydataset
mypool/usr/ports       144K  93.2G   144K  /usr/ports
mypool/usr/src         144K  93.2G   144K  /usr/src
mypool/var            1.20M  93.2G   610K  /var
mypool/var/crash       148K  93.2G   148K  /var/crash
mypool/var/log         178K  93.2G   178K  /var/log
mypool/var/mail        144K  93.2G   144K  /var/mail
mypool/var/tmp         152K  93.2G   152K  /var/tmp
```

销毁一个数据集比删除数据集上的文件要快得多，因为它不涉及扫描文件以及更新相应的元数据。

销毁已创建的数据集：

```
# zfs list
NAME                  USED  AVAIL  REFER  MOUNTPOINT
mypool                880M  93.1G   144K  none
mypool/ROOT           777M  93.1G   144K  none
mypool/ROOT/default   777M  93.1G   777M  /
mypool/tmp            176K  93.1G   176K  /tmp
mypool/usr            101M  93.1G   144K  /usr
mypool/usr/home       184K  93.1G   184K  /usr/home
mypool/usr/mydataset  100M  93.1G   100M  /usr/mydataset
mypool/usr/ports      144K  93.1G   144K  /usr/ports
mypool/usr/src        144K  93.1G   144K  /usr/src
mypool/var           1.20M  93.1G   610K  /var
mypool/var/crash      148K  93.1G   148K  /var/crash
mypool/var/log        178K  93.1G   178K  /var/log
mypool/var/mail       144K  93.1G   144K  /var/mail
mypool/var/tmp        152K  93.1G   152K  /var/tmp
# zfs destroy mypool/usr/mydataset
# zfs list
NAME                 USED  AVAIL  REFER  MOUNTPOINT
mypool               781M  93.2G   144K  none
mypool/ROOT          777M  93.2G   144K  none
mypool/ROOT/default  777M  93.2G   777M  /
mypool/tmp           176K  93.2G   176K  /tmp
mypool/usr           616K  93.2G   144K  /usr
mypool/usr/home      184K  93.2G   184K  /usr/home
mypool/usr/ports     144K  93.2G   144K  /usr/ports
```

(continues on next page)

(continued from previous page)

```
mypool/usr/src        144K  93.2G   144K  /usr/src
mypool/var           1.21M  93.2G   612K  /var
mypool/var/crash      148K  93.2G   148K  /var/crash
mypool/var/log        178K  93.2G   178K  /var/log
mypool/var/mail       144K  93.2G   144K  /var/mail
mypool/var/tmp        152K  93.2G   152K  /var/tmp
```

在现代版本的 ZFS 中，`zfs destroy` 是异步的，释放的空间可能需要几分钟才能出现在池中。使用 `zpool get freeing poolname` 查看 `freeing` 属性，显示哪些数据集的块在后台被释放了。如果有子数据集，比如快照[1393]或其他数据集，销毁父数据集是不可能的。要销毁一个数据集和它的子集，使用 `-r` 来递归销毁数据集和它的子集。使用 `-n -v` 来列出由该操作销毁的数据集和快照，而不是实际销毁任何东西。通过销毁快照回收的空间也会被显示出来。

22.4.2.创建和销毁卷

卷是一种特殊的数据集类型。与其作为一个文件系统挂载，不如把它作为一个块设备暴露在 **/dev/zvol/poolname/dataset** 下。这将允许把卷用于其他文件系统，备份虚拟机的磁盘，或者使用 iSCSI 或 HAST 等协议使其对其他网络主机可用。

用任何文件系统格式化卷，或者不使用文件系统来存储原始数据。对用户来说，卷看起来就是一个普通的磁盘。把普通文件系统放在这些 zvols 上，可以提供普通磁盘或文件系统所不具备的功能。例如，在一个 250MB 的卷上使用压缩属性可以创建一个压缩的 FAT 文件系统。

```
# zfs create -V 250m -o compression=on tank/fat32
# zfs list tank
NAME USED AVAIL REFER MOUNTPOINT
tank 258M  670M   31K /tank
# newfs_msdos -F32 /dev/zvol/tank/fat32
# mount -t msdosfs /dev/zvol/tank/fat32 /mnt
# df -h /mnt | grep fat32
Filesystem           Size Used Avail Capacity Mounted on
/dev/zvol/tank/fat32 249M  24k  249M     0%   /mnt
# mount | grep fat32
/dev/zvol/tank/fat32 on /mnt (msdosfs, local)
```

销毁卷与销毁普通的文件系统数据集是一样的。该操作几乎是瞬间完成的，但可能需要几分钟的时间来回收后台的空闲空间。

[1393] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-snapshot

22.4.3.重命名数据集

要改变数据集的名称，使用 `zfs rename`。要改变数据集的父数据集，也可以使用这个命令。重命名数据集以拥有一个不同的父数据集将改变那些从父数据集继承的属性值。重新命名一个数据集，然后在新的位置重新挂载（从新的父数据集继承的）。要防止这种行为，请使用 `-u`。

重命名数据集并将其移动到不同的父数据集下：

```
# zfs list
NAME                   USED  AVAIL  REFER  MOUNTPOINT
mypool                 780M  93.2G   144K  none
mypool/ROOT            777M  93.2G   144K  none
mypool/ROOT/default    777M  93.2G   777M  /
mypool/tmp             176K  93.2G   176K  /tmp
mypool/usr             704K  93.2G   144K  /usr
mypool/usr/home        184K  93.2G   184K  /usr/home
mypool/usr/mydataset  87.5K  93.2G  87.5K  /usr/mydataset
mypool/usr/ports       144K  93.2G   144K  /usr/ports
mypool/usr/src         144K  93.2G   144K  /usr/src
mypool/var            1.21M  93.2G   614K  /var
mypool/var/crash       148K  93.2G   148K  /var/crash
mypool/var/log         178K  93.2G   178K  /var/log
mypool/var/mail        144K  93.2G   144K  /var/mail
mypool/var/tmp         152K  93.2G   152K  /var/tmp
# zfs rename mypool/usr/mydataset mypool/var/newname
# zfs list
NAME                   USED  AVAIL  REFER  MOUNTPOINT
mypool                 780M  93.2G   144K  none
mypool/ROOT            777M  93.2G   144K  none
mypool/ROOT/default    777M  93.2G   777M  /
mypool/tmp             176K  93.2G   176K  /tmp
mypool/usr             616K  93.2G   144K  /usr
mypool/usr/home        184K  93.2G   184K  /usr/home
mypool/usr/ports       144K  93.2G   144K  /usr/ports
mypool/usr/src         144K  93.2G   144K  /usr/src
mypool/var            1.29M  93.2G   614K  /var
mypool/var/crash       148K  93.2G   148K  /var/crash
mypool/var/log         178K  93.2G   178K  /var/log
mypool/var/mail        144K  93.2G   144K  /var/mail
mypool/var/newname    87.5K  93.2G  87.5K  /var/newname
mypool/var/tmp         152K  93.2G   152K  /var/tmp
```

重命名快照使用同样的命令。由于快照的性质，重命名不能改变其父数据集。要重命名一个递归快照，请指定 `-r`；这也将重命名子数据集中所有具有相同名称的快照。

```
# zfs list -t snapshot
NAME                                  USED  AVAIL  REFER  MOUNTPOINT
mypool/var/newname@first_snapshot        0      -  87.5K  -
# zfs rename mypool/var/newname@first_snapshot new_snapshot_name
# zfs list -t snapshot
NAME                                    USED  AVAIL  REFER  MOUNTPOINT
mypool/var/newname@new_snapshot_name       0      -  87.5K  -
```

22.4.4.设置数据集属性

每个 ZFS 数据集都有各自属性，这些属性用来控制数据集的如何操作。大多数属性会自动从父数据集继承，但是可以被本地覆盖。用 `zfs set property=value dataset` 来进行数据集属性设置。大多数属性有效赋值范围会受到限制，`zfs get` 可以用来显示每种可能的属性和有效值。用 `zfs inherit` 来恢复属性的原继承来的值。另外还可以自定义新的属性，它们成为数据集设置的一部分并进一步完善数据集和其内容等信息。可以使用冒号（:）来为属性创建自定义命名空间用以区分自定义属性和 ZFS 自带属性。

```
# zfs set custom:costcenter=1234 tank
# zfs get custom:costcenter tank
NAME PROPERTY           VALUE SOURCE
tank custom:costcenter  1234  local
```

要删除自定义属性，可以使用 `zfs inherit` 并带上参数 `-r`。如果自定义属性没有在任何父数据集中被定义过，用这个参数就可以进行删除（存储池的历史日志仍然会记录这种更改操作）。

```
# zfs inherit -r custom:costcenter tank
# zfs get custom:costcenter tank
NAME    PROPERTY           VALUE              SOURCE
tank    custom:costcenter  -                  -
# zfs get all tank | grep custom:costcenter
#
```

22.4.4.1.获取和设置共享属性

两个最常用的数据集属性是共享 NFS 和 SMB 参数。设置这些会定义 ZFS 是否以及如何通过网络共享数据集。目前来说，FreeBSD 支持单独设置 NFS 共享。如要获取当前共享的状态，可以输入：

```
# zfs get sharenfs mypool/usr/home
NAME             PROPERTY  VALUE    SOURCE
mypool/usr/home  sharenfs  on       local
# zfs get sharesmb mypool/usr/home
```

(continues on next page)

(continued from previous page)

```
NAME             PROPERTY  VALUE     SOURCE
mypool/usr/home  sharesmb  off       local
```

如要启用数据集共享，可以输入：

```
#  zfs set sharenfs=on mypool/usr/home
```

设置 NFS 共享的其它参数，如 `-alldirs`, `-maproot` 和 `network`。要设置数据集 NFS 共享参数，输入：

```
#  zfs set sharenfs="-alldirs,-maproot=root,-network=192.168.1.0/24" mypool/usr/home
```

22.4.5.管理快照

快照[1394]功能是 ZFS 最强大的功能之一。快照提供了数据集只读和时间指针引用的复制功能。使用写入时复制（Copy-On-Write，COW）功能，ZFS 可以在保留磁盘上老版本数据的同时快速创建快照。如果没有快照存在，当数据被重写覆盖或删除时 ZFS 就会回收磁盘空间供将来使用。快照功能通过仅记录数据集当前和过去版本的变化来节约磁盘空间。要打开基于整个数据集上的快照功能，而不是在文件或目录层面。数据集的快照会复制数据集中的所有内容，包括文件系统属性，文件，目录，权限等等。首次创建时快照不占用额外空间，但当它引用的数据块发生改变时就会开始消耗磁盘空间。用 `-r` 来创建的数据集及其子集的同名快照提供了文件系统在同一时间片段上的即时快照。这种功能在应用程序的文件位于相关数据集上或这些文件互相依赖时非常有用。如果没有了快照功能，备份时需要保存文件在不同时间点的许多份副本。

ZFS 里的快照功能还提供了其它文件系统快照所不具备的许多不同特性。举例来说快照的一个典型用处是当执行软件安装或系统升级等有风险的操作时用来快速备份文件系统的当前状态。如果操作失败，就可以退回创建快照时的系统状态。如果升级顺利，则可以删除快照来释放磁盘空间。没有快照功能的话，升级失败往往需要进行备份恢复，而那是及其繁琐和耗费时间的，而且可能还会带来系统无法使用的宕机状态。快照恢复非常迅速，甚至可以在系统正常运行时进行，带来极少或不带来任何宕机时间。考虑到需要从备份恢复的数据复制时间的话快照为 TB 级存储系统带来的时间节约是及其巨大的。快照不是存储池完整备份的替代方案，但提供了一种在特定时间保存数据集的快速而又简便的方法。

22.4.5.1.创建快照

用 `zfs snapshot` *dataset@snapshotname* 来创建快照，添加 `-r` 参数来创建包含子数据集的递归快照。

创建整个存储池的递归快照：

```
# zfs list -t all
NAME                                  USED  AVAIL  REFER  MOUNTPOINT
```

(continues on next page)

[1394] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-snapshot

(continued from previous page)

```
mypool                                  780M  93.2G   144K  none
mypool/ROOT                             777M  93.2G   144K  none
mypool/ROOT/default                     777M  93.2G   777M  /
mypool/tmp                              176K  93.2G   176K  /tmp
mypool/usr                              616K  93.2G   144K  /usr
mypool/usr/home                         184K  93.2G   184K  /usr/home
mypool/usr/ports                        144K  93.2G   144K  /usr/ports
mypool/usr/src                          144K  93.2G   144K  /usr/src
mypool/var                             1.29M  93.2G   616K  /var
mypool/var/crash                        148K  93.2G   148K  /var/crash
mypool/var/log                          178K  93.2G   178K  /var/log
mypool/var/mail                         144K  93.2G   144K  /var/mail
mypool/var/newname                     87.5K  93.2G  87.5K  /var/newname
mypool/var/newname@new_snapshot_name       0      -  87.5K  -
mypool/var/tmp                          152K  93.2G   152K  /var/tmp
# zfs snapshot -r mypool@my_recursive_snapshot
# zfs list -t snapshot
NAME                                        USED  AVAIL  REFER  MOUNTPOINT
mypool@my_recursive_snapshot                   0      -   144K  -
mypool/ROOT@my_recursive_snapshot              0      -   144K  -
mypool/ROOT/default@my_recursive_snapshot      0      -   777M  -
mypool/tmp@my_recursive_snapshot               0      -   176K  -
mypool/usr@my_recursive_snapshot               0      -   144K  -
mypool/usr/home@my_recursive_snapshot          0      -   184K  -
mypool/usr/ports@my_recursive_snapshot         0      -   144K  -
mypool/usr/src@my_recursive_snapshot           0      -   144K  -
mypool/var@my_recursive_snapshot               0      -   616K  -
mypool/var/crash@my_recursive_snapshot         0      -   148K  -
mypool/var/log@my_recursive_snapshot           0      -   178K  -
mypool/var/mail@my_recursive_snapshot          0      -   144K  -
mypool/var/newname@new_snapshot_name           0      -  87.5K  -
mypool/var/newname@my_recursive_snapshot       0      -  87.5K  -
mypool/var/tmp@my_recursive_snapshot           0      -   152K  -
```

用常规的 `zfs list` 命令不会显示快照。要显示快照列表。添加 `-t snapshot` 参数到 `zfs list`。`-t all` 则会同时显示文件系统和快照。

快照不会直接被挂载，在 `MOUNTPOINT` 列中不会显示路径。由于快照在创建后是以只读方式存在，ZFS 在 `AVAIL` 列中不会显示可用空间。对比快照和原始数据集：

```
# zfs list -rt all mypool/usr/home
NAME                                 USED  AVAIL  REFER  MOUNTPOINT
mypool/usr/home                      184K  93.2G   184K  /usr/home
```

(continues on next page)

(continued from previous page)

```
mypool/usr/home@my_recursive_snapshot      0      -   184K  -
```

同时显示数据集和快照揭示快照如何以 COW[1395] 方式工作。他们保存更新的变化（*delta*）而不是完整的文件系统内容。这意味着快照在带来改变的同时只占用很少磁盘空间。通过将文件复制到数据集来更详细地观察磁盘使用情况并创建第二个快照：

```
# cp /etc/passwd /var/tmp
# zfs snapshot mypool/var/tmp@after_cp
# zfs list -rt all mypool/var/tmp
NAME                                   USED  AVAIL  REFER  MOUNTPOINT
mypool/var/tmp                         206K  93.2G   118K  /var/tmp
mypool/var/tmp@my_recursive_snapshot    88K      -   152K  -
mypool/var/tmp@after_cp                  0      -   118K  -
```

第二个快照包含了在复制操作之后的数据集变化，这极大地节约了存储空间。可以看到快照 *mypool/var/tmp@my_recursive_snapshot* 的大小在 `USED` 列发生了改变，显示出之前快照和当前快照所占的不同磁盘空间。

22.4.5.2.快照比对

ZFS 提供了自带的命令来比对两个不同快照的内容。这在用户创建了许多快照并希望看看随着时间推移文件系统发生哪些变化时非常有用。例如。`zfs diff` 让用户可以找到还保留某个被意外删除的文件的最近一次快照。在之前小节创建的两个快照上进行这种比较得到如下结果：

```
# zfs list -rt all mypool/var/tmp
NAME                                   USED  AVAIL  REFER  MOUNTPOINT
mypool/var/tmp                         206K  93.2G   118K  /var/tmp
mypool/var/tmp@my_recursive_snapshot    88K      -   152K  -
mypool/var/tmp@after_cp                  0      -   118K  -
# zfs diff mypool/var/tmp@my_recursive_snapshot
M       /var/tmp/
+       /var/tmp/passwd
```

这条命令列出了指定快照（如此例 *mymypool/var/tmp@my_recursive_snapshot*）和当前文件系统的变化。第一列显示变化类别：

[1395] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-cow

•	增加路径或文件
•	删除路径或文件
M	更改路径或文件
R	重命名路径或文件

对照表格和输出结果，显然 ZFS 在创建了快照 *mypool/var/tmp@my_recursive_snapshot* 后添加了 **passwd**。这还导致了挂载在 `/var/tmp` 的上级目录的变化。

当使用 ZFS 复制功能把一个数据集转移到另一台主机实现备份时来进行两个快照之间的比对是很有帮助的。

通过完整数据集名称和快照名称来比对两个数据集的快照：

```
# cp /var/tmp/passwd /var/tmp/passwd.copy
# zfs snapshot mypool/var/tmp@diff_snapshot
# zfs diff mypool/var/tmp@my_recursive_snapshot mypool/var/tmp@diff_snapshot
M       /var/tmp/
+       /var/tmp/passwd
+       /var/tmp/passwd.copy
# zfs diff mypool/var/tmp@my_recursive_snapshot mypool/var/tmp@after_cp
M       /var/tmp/
+       /var/tmp/passwd
```

22.4.5.3.快照恢复

可以在任何时候恢复已有快照。在很多情况下当前数据集状态已经不需要或更愿意使用老版本时就经常需要这样做。或者在本地开发测试出问题时，有缺陷的系统更新妨害了系统的正常功能，或者需要恢复删除的文件或目录时也往往需要这样做。要恢复到某个快照，使用 `zfs roolback` *snapshotname* 命令。如果发生过许多改变，这项操作就需要较长时间。在那期间数据集总保持在一致的状态，很像符合 ACID 标准的数据库执行一次回滚任务那样。当数据集不需要宕机就可以正常访问时就是如此。当快照恢复以后，数据集就恢复到和创建快照之前的相同状态。恢复快照会导致不存在于快照数据集中的数据的丢失。在恢复操作前创建当前数据集状态的下快照是今后需要这些数据时的一个好方法。这样，用户就可以在这些快照之间来回恢复而不丢失有价值的数据。

在第一个例子中，恢复快照是由于粗心的 `rm` 操作不小心删除了太多数据。

```
# zfs list -rt all mypool/var/tmp
NAME                                USED  AVAIL  REFER  MOUNTPOINT
mypool/var/tmp                      262K  93.2G   120K  /var/tmp
```

(continues on next page)

(continued from previous page)

```
mypool/var/tmp@my_recursive_snapshot    88K      -   152K  -
mypool/var/tmp@after_cp               53.5K      -   118K  -
mypool/var/tmp@diff_snapshot              0      -   120K  -
# ls /var/tmp
passwd          passwd.copy     vi.recover
# rm /var/tmp/passwd*
# ls /var/tmp
vi.recover
```

这时，用户注意到误删除了额外的文件并想要进行恢复。只要平时有规律地对重要数据执行快照，ZFS 就提供了便捷的方式来进行恢复。为了从最近的快照恢复文件，执行以下命令：

```
# zfs rollback mypool/var/tmp@diff_snapshot
# ls /var/tmp
passwd          passwd.copy     vi.recover
```

恢复操作把数据集恢复到上一次快照的状态。也可以从这之后的快照恢复到早得多的快照也是可行的，当尝试这样操作时，ZFS 会给出警告提示：

```
# zfs list -rt snapshot mypool/var/tmp
AME                                   USED  AVAIL  REFER  MOUNTPOINT
mypool/var/tmp@my_recursive_snapshot    88K      -   152K  -
mypool/var/tmp@after_cp               53.5K      -   118K  -
mypool/var/tmp@diff_snapshot              0      -   120K  -
# zfs rollback mypool/var/tmp@my_recursive_snapshot
cannot rollback to 'mypool/var/tmp@my_recursive_snapshot': more recent snapshots exist
use '-r' to force deletion of the following snapshots:
mypool/var/tmp@after_cp
mypool/var/tmp@diff_snapshot
```

这个警告表示在数据集当前状态和快照恢复状态之间有其它快照。为了完成恢复操作需要删除这些快照。ZFS 不能保留数据集不同状态之间的所有变化，因为快照是只读的。ZFS 不会删除受影响的快照除非用户输入 `-r` 来确认这是他想要的操作。如果那真是用户的意图并且用户知道这样会丢失所有中间快照，就执行如下命令：

```
# zfs rollback -r mypool/var/tmp@my_recursive_snapshot
# zfs list -rt snapshot mypool/var/tmp
NAME                                    USED  AVAIL  REFER  MOUNTPOINT
mypool/var/tmp@my_recursive_snapshot     8K      -   152K  -
# ls /var/tmp
vi.recover
```

`zfs list -t snapshot` 命令的返回结果确认了中间快照已被删除，那是 `zfs rollback -r` 命令

的执行结果。

22.4.5.4.从快照中恢复部分文件

快照保存在父数据集的一个隐藏目录下：**.zfs/snapshots/snapshotname**。默认情况下，这些目录在执行标准的 `ls -a` 命令下也不会显示。虽然目录不显示，但可以像普通目录一样进行访问。名为 `snapdir` 的属性控制这些隐藏目录是否在显示目录命令时别列出。把这个属性设置为 `visible` 允许用 `ls` 命令和其它与目录内容有关命令来显示这些隐藏目录。

```
# zfs get snapdir mypool/var/tmp
NAME            PROPERTY  VALUE    SOURCE
mypool/var/tmp  snapdir   hidden   default
# ls -a /var/tmp
.               ..              passwd          vi.recover
# zfs set snapdir=visible mypool/var/tmp
# ls -a /var/tmp
.               ..              .zfs            passwd          vi.recover
```

要恢复个别文件到之前的状态时可以把这些文件从快照复制回父数据集就可以。**.zfs/snapshot** 里面的目录结构有个目录名字和之前快照类似的目录，可以更方便地找到他们。下面的例子显示如何把一个文件从隐藏的 **.zfs** 目录里恢复，只要把它从包含这个文件最近版本的快照里复制出来就可以：

```
# rm /var/tmp/passwd
# ls -a /var/tmp
.               ..              .zfs            vi.recover
# ls /var/tmp/.zfs/snapshot
after_cp                my_recursive_snapshot
# ls /var/tmp/.zfs/snapshot/after_cp
passwd          vi.recover
# cp /var/tmp/.zfs/snapshot/after_cp/passwd /var/tmp
```

即使 `snapdir` 属性被设置为隐藏，运行 `ls .zfs/snapshot` 还是可以列出那个目录的内容。管理员可以决定是否显示这些目录。每个数据集都有这样的设置。把文件或目录从隐藏的 **.zfs/snapshot** 复制出来非常简单。而用其它方法就会报错：

```
# cp /etc/rc.conf /var/tmp/.zfs/snapshot/after_cp/
cp: /var/tmp/.zfs/snapshot/after_cp/rc.conf: Read-only file system
```

这个错误提醒用户快照是只读的，在创建后就不能更改。复制或者移动文件都是不允许的操作，因为那会改变它们引用的数据集的状态。

快照消耗的磁盘空间决定于父文件系统在创建快照后发生了多大的改变。快照的 `written` 属性跟踪快照使用的磁盘空间。

要销毁快照并回收磁盘空间。可以使用 `zfs destroy` *dataset@snapshot* 命令。添加 `-r` 参数可以递归删除父数据集下有同样名称的所有快照。添加 `-n -v` 到命令行可以在不执行实际销毁操作下显示将被删除的快照和预计回收的磁盘空间大小。

22.4.6.管理 camino

camino 是快照的一份副本，处理起来更近似于一个普通的数据集。与快照不同，camino 可写也可挂载，有它自己的属性。在使用 `zfs clone` 之后，可以销毁原始快照。要对调 camino 和快照的 child/parent 关系可以使用 `zfs promote` 命令。`child/parent` 关系对调之后快照就会变成 camino 的 child，而不是最开始的父数据集。这回改变 ZFS 所占磁盘空间的方式，但不实际改变消耗的磁盘空间。把 camino 挂载到 ZFS 文件系统的任何层级中的位置都是可以的，并不只是在原快照位置之下的层级。

下面的示例数据集可以显示 camino 的一些特性：

```
# zfs list -rt all camino/home/joe
NAME                    USED  AVAIL  REFER  MOUNTPOINT
camino/home/joe         108K   1.3G    87K  /usr/home/joe
camino/home/joe@plans    21K      -  85.5K  -
camino/home/joe@backup    0K      -    87K  -
```

camino 的典型作用是在一些特定的数据集上做实验，当出现问题时还可以使用快照进行恢复。因为快照内容是不能改变的，所以需要创建一个可读也可写的快照的 camino。在 camino 中取得想要的结果后，就可以提升 camino 让它成为一个数据集并删除老的文件系统。删除父数据集严格来说并不必要，因为 camino 和数据集可以无障碍地同时存在。

```
# zfs clone camino/home/joe@backup camino/home/joenew
# ls /usr/home/joe*
/usr/home/joe:
backup.txz     plans.txt

/usr/home/joenew:
backup.txz     plans.txt
# df -h /usr/home
Filesystem          Size    Used   Avail Capacity  Mounted on
usr/home/joe        1.3G     31k    1.3G     0%    /usr/home/joe
usr/home/joenew     1.3G     31k    1.3G     0%    /usr/home/joenew
```

创建 camino 等于是产生了一个快照创建时的数据集的实际副本。这样就可以在不影响原数据集的情况下改变 camino 里的内容。他们之间通过快照进行关联。ZFS 在名为 `origin` 的属性中保存了这种关连。用 `zfs promote` 提升这个 camino 可以让 camino 成为一个独立的数据集。这会删除 `origin` 属性的值并切断新的独立的数据集与快照的关连。示例如下：

```
# zfs get origin camino/home/joenew
NAME                 PROPERTY  VALUE                     SOURCE
camino/home/joenew   origin    camino/home/joe@backup    -
# zfs promote camino/home/joenew
# zfs get origin camino/home/joenew
NAME                 PROPERTY  VALUE   SOURCE
camino/home/joenew   origin    -       -
```

在通过复制 **loader.conf** 到 promote 之后的 camino 对它进行改变后，例如，老的目录就失效了。然而，提升之后的 camino 可以取代它。要这样做，可以用 `zfs destroy` 来先销毁原来的数据集然后再用 `zfs rename` 命令来把 camino 命名为原来的数据集名称（也可以命名为完全不同的名称）。

```
# cp /boot/defaults/loader.conf /usr/home/joenew
# zfs destroy -f camino/home/joe
# zfs rename camino/home/joenew camino/home/joe
# ls /usr/home/joe
backup.txz     loader.conf     plans.txt
# df -h /usr/home
Filesystem          Size    Used   Avail Capacity  Mounted on
usr/home/joe        1.3G    128k    1.3G     0%    /usr/home/joe
```

快照的 camino 现在已经是一个普通的数据集。它含有源快照的所有数据和新增加的文件，如 **loader.conf**。camino 在不同场景下为 ZFS 用户提供了有用的功能。例如。把包含不同软件集合的快照作为 jail 来使用。如果需要的话用户可以 camino 这些快照并加入他们自己的应用程序。如果对变更感到满意，就提升 camino 成为完成的数据集，对最终用户来说用起来就和真正的数据集别无两样。提供这些 jail 可以节约时间和管理开销。

22.4.7.数据复制

把单独的存储池的数据存放在一个地方会有被盗，自然或人为灾害的风险。定期为整个存储池创建备份至关重要。ZFS 提供了自带的序列化功能，它可以发送把数据以流的形式发送到标准输出。使用这一功能，就把数据存储在连接到本地系统的另一个存储池，例如通过网络连接的另一个系统。快照是复制的基础（参见 ZFS 快照[1396]章节）。用来复制数据的命令为 `zfs send` 和 `zfs receive`。

下面的例子演示了在两个存储池之间进行 ZFS 数据复制：

```
# zpool list
NAME    SIZE  ALLOC   FREE   CKPOINT  EXPANDSZ   FRAG    CAP  DEDUP  HEALTH  ALTROOT
backup  960M    77K   896M         -         -     0%     0%  1.00x  ONLINE  -
mypool  984M  43.7M   940M         -         -     0%     4%  1.00x  ONLINE  -
```

[1396] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-zfs-snapshot

名为 *mypool* 的存储池是主存储池，定期会有数据读写操作发生。当主存储池不可用时，使用第二个名为 *backup* 的待机存储池来接替。注意发生故障时的切换并不是由 ZFS 自动完成的，而是当需要时必须经由系统管理员的人工操作。使用快照来提供复制时的一致性文件系统版本。在创建 *mypool* 的快照后。通过复制快照的方式把它复制到 *backup* 存储池。这并不会包含最近快照之后发生的改变。

```
# zfs snapshot mypool@backup1
# zfs list -t snapshot
NAME                    USED  AVAIL  REFER  MOUNTPOINT
mypool@backup1             0      -  43.6M  -
```

现在已经有了快照，可以使用 `zfs send` 来创建包含快照内容的数据流。把数据流在另一个存储池中以文件的形式来保存。把数据流写入标准输出，但重定向到文件或管道否则会出现错误：

```
# zfs send mypool@backup1
Error: Stream can not be written to a terminal.
You must redirect standard output.
```

要用 `zfs send` 来备份数据集，重定向到一个已挂载的备份存储池上的文件。确保存储池有足够的空间来容纳发送的快照，这里的快照指快照所包含的完整数据，不是自前一个快照之后发生的改变。

```
# zfs send mypool@backup1 > /backup/backup1
# zpool list
NAME    SIZE  ALLOC   FREE   CKPOINT  EXPANDSZ   FRAG    CAP  DEDUP  HEALTH  ALTROOT
backup  960M  63.7M   896M         -         -     0%     6%  1.00x  ONLINE  -
mypool  984M  43.7M   940M         -         -     0%     4%  1.00x  ONLINE  -
```

`zfs send` 命令可以把名为 *backup1* 快照里的数据传送到名为 *backup* 的存储池。要创建并自动发送这些快照，可以使用 cron(8)[1397] 调度任务命令。

ZFS 可以把接收到的数据作为实时的文件系统来对待而不是存储为备份文件，允许直接访问这些数据。要获取这些确切的流数据，可以使用 `zfs receive` 来把流转换回文件和目录。下面的例子包含使用管道来在存储池间复制数据的 `zfs send` 和 `zfs receive` 命令。在传输完成后在接收的存储池一端可以直接使用这些数据。向一个空数据集复制已有数据集也是可行的。

```
# zfs snapshot mypool@replica1
# zfs send -v mypool@replica1 | zfs receive backup/mypool
send from @ to mypool@replica1 estimated size is 50.1M
total estimated size is 50.1M
TIME        SENT   SNAPSHOT

# zpool list
NAME    SIZE  ALLOC   FREE   CKPOINT  EXPANDSZ   FRAG    CAP  DEDUP  HEALTH  ALTROOT
```

(continues on next page)

[1397] https://www.freebsd.org/cgi/man.cgi?query=cron&sektion=8&format=html

(continued from previous page)

```
backup  960M  63.7M   896M         -         -     0%     6%  1.00x  ONLINE  -
mypool  984M  43.7M   940M         -         -     0%     4%  1.00x  ONLINE  -
```

22.4.7.1.增量备份

`zfs send` 也可以判断两个快照的不同之处并把差异部分进行发送。这能够节约磁盘空间和传输时间，例如：

```
# zfs snapshot mypool@replica2
# zfs list -t snapshot
NAME                    USED  AVAIL  REFER  MOUNTPOINT
mypool@replica1         5.72M      -  43.6M  -
mypool@replica2             0      -  44.1M  -
# zpool list
NAME    SIZE  ALLOC   FREE   CKPOINT  EXPANDSZ   FRAG   CAP  DEDUP  HEALTH  ALTROOT
backup  960M  61.7M   898M         -         -     0%    6%  1.00x  ONLINE  -
mypool  960M  50.2M   910M         -         -     0%    5%  1.00x  ONLINE  -
```

创建叫做 *replica2* 的第二个快照。第二个快照含有之前快照 *replica1* 到现在发生的文件系统的变化。使用 `zfs send -i` 来指明产生含有更新数据的增量复制流的快照配对。如果初始快照已经存在于接收端，就可以继续执行这条指令：

```
# zfs snapshot mypool@replica2
# zfs list -t snapshot
NAME                    USED  AVAIL  REFER  MOUNTPOINT
mypool@replica1         5.72M      -  43.6M  -
mypool@replica2             0      -  44.1M  -
# zpool list
NAME    SIZE  ALLOC   FREE   CKPOINT  EXPANDSZ   FRAG   CAP  DEDUP  HEALTH  ALTROOT
backup  960M  61.7M   898M         -         -     0%    6%  1.00x  ONLINE  -
mypool  960M  50.2M   910M         -         -     0%    5%  1.00x  ONLINE  -
```

增量流仅复制改变的数据而不是整个 *replica1*。发送差异化部分只要花少的多的时间并能够节约磁盘空间，不需要每次复制整个存储池。这个功能非常有用，尤其是在通过较慢的网络来复制或按传送的字节来收费时。

一个新的文件系统 *backup/mupool*，在来自存储池 *mypool* 的文件和数据到位之后就可以随时待命了。通过 `-p` 来指定复制数据集的属性，包含压缩设置，容量配额，挂载点等等。通过 `-R` 来指定复制数据集的所有子数据集及其属性。自动化式发送和接收可以在第二个存储池上建立定期备份。

22.4.7.2.通过 SSH 发送加密备份

通过网络发送流数据是进行远程备份的好方法，但也带来了一些缺陷。通过网络链接发送的数据是不加密的，任何人可以在数据发送方不知道的情况下截取流数据并把它们传输到其它地方。通过互联网向远程主机进行流数据传输时就非常危险，这样就需要使用 SSH 在网络传输时进行安全加密。ZFS 本来就需要从标准输出重定向流数据，使用 SSH 就很方便。如果在传输时还要对文件系统的内容在远端系统进行加密，就可以考虑使用 PEFS[1398]。

先更改一些设置并做安全预防。这里介绍使用 `zfs send` 命令进行操作前的必要步骤，更多关于 SSH 的详细信息，请参考 OpenSSH[1399]。

按下列步骤更改设置：

- 使用 SSH 密钥来启用发送端和接收端主机的 SSH 功能。
- ZFS 需要 `root` 用户权限来发送和接收流数据。用 `root` 账号登录接收端系统。
- 默认情况下禁止使用 `root` 来登录系统。
- 使用 ZFS 授权[1400]系统来允许系统中普通用户执行各自的发送和接收操作。在发送端：

```
# zfs allow -u someuser send,snapshot mypool
```

- 普通用户需是目录的所有者才能挂载存储池，为普通用户开放挂载文件系统的相关权限。

在接收端：

-

```
# sysctl vfs.usermount=1
vfs.usermount: 0 -> 1
# echo vfs.usermount=1 >> /etc/sysctl.conf
# zfs create recvpool/backup
# zfs allow -u someuser create,mount,receive recvpool/backup
# chown someuser /recvpool/backup
```

现在普通用户就可以接收和挂载数据集了，也可以把 *home* 数据集复制到远端系统：

```
% zfs snapshot -r mypool/home@monday
% zfs send -R mypool/home@monday | ssh someuser@backuphost zfs recv -dvu recvpool/
↪backup
```

为存储池 *mypool* 上的文件系统数据集 *home* 创建名为 *monday* 的递归快照，然后用 `zfs send -R` 命令来把数据集和其所有子集，快照，camino，设置等等包含进需要发送的数据流。用 SSH 方式传递到远端使用

[1398] https://wiki.freebsd.org/PEFS

[1399] https://docs.freebsd.org/en/books/handbook/security/index.html#openssh

[1400] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-zfs-allow

`zfs receive` 命令的主机 *backuphost*。这里也可以使用主机的 IP 地址或合格的域名。接收端主机把数据写入存储池 *recvpool* 中的 *backup* 数据集。添加 `-d` 到 `zfs recv` 来用快照名替换接收端的存储池名称。`-u` 参数则不会让接收端文件系进行 mount。使用 `-v` 来显示传输情况的更多详细信息，包括已用时间和已传输的数据总量。

22.4.8.数据集，用户和组配额

使用数据集配额[1401]来限制特定数据集的磁盘占用。参考配额[1402]也有类似功能，但它只计算数据集本身所占磁盘空间，并不包含快照和所有子数据集。同样地，可以使用用户[1403]或组[1404]配额来防止用户或组占用存储池或数据集中的所有磁盘空间。

下例例子假设系统中已经存在这些用户。在向系统添加用户时，先确保创建用户的 home 数据集并将挂载点设为 `/home/bob`。然后创建用户并让 home 目录指向数据集的挂载点位置。这样就可以正确地设置用户和组相关权限而不覆盖现有可能存在的 home 目录路径。

强制设置数据集 **storage/home/bob** 配额为 10 GB:

```
# zfs set quota=10G storage/home/bob
```

强制设置数据集 **storage/home/bob** 引用配额为 10 GB:

```
# zfs set refquota=10G storage/home/bob
```

删除数据集 **storage/home/bob** 配额：

```
# zfs set quota=none storage/home/bob
```

常规格式是 `userquota@user=size`，用户名必须符合以下标准之一：

- POSIX 兼容命名如 *joe*
- POSIX 数字 ID 如 *789*
- SID 命名如 *joe.bloggs@example.com*
- SID 数字 ID 如 *S-1-123-456-789*

例如，要设置用户 *joe* 的用户配额为 50 GB：

```
# zfs set userquota@joe=50G
```

要删除相关用户配额：

[1401] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-quota
[1402] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-refquota
[1403] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-userquota
[1404] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-groupquota

```
# zfs set userquota@joe=none
```

> 注意
>
> 用户配额属性并不会通过 zfs get all 命令显示出来。非 root 权限用户并不会看到其他人的配额除非被通过 userquota 进行授权。获得授权的用户就可以看到并设置所有人的磁盘空间配额，

设置组配额的格式通常为：groupquota@group=size

如要设置 *firstgroup* 的组配额为 50 GB，可以使用：

```
# zfs set groupquota@firstgroup=50G
```

使用用户配额属性，非 root 权限用户可以看到他们所属的组的配额。有 groupquota 授权的用户或 root 用户可以看到和设置所有组别的配额。

如要显示每个用户在文件系统或快照中的配额，使用 zfs userspace。用 zfs groupspace 可以看到组相关的配额信息。若要了解更多其它参数或如何单独查看特定参数，可以参考zfs(1)[1405]。

授权用户和 root 用户可以使用以下命令查看 **storage/home/bob** 的配额：

```
# zfs get quota storage/home/bob
```

22.4.9.保留（磁盘空间）

保留[1406] 是保障了数据集上有永久可用磁盘空间的一种方式。保留的空间不会被其它数据集占用。这个有用的功能确保一个重要数据集或日志文件有可用的剩余磁盘空间。

reservation 属性的通常格式为 reservation=size，如要为 **storage/home/bob** 设置 10 GB 的保留空间，使用命令：

```
# zfs set reservation=10G storage/home/bob
```

取消所有保留空间：

```
# zfs set reservation=none storage/home/bob
```

设置引用保留空间[1407]时同样的规则也适用于 reservation 属性，通常格式为 refreservation=size

下面命令会显示和 **storage/home/bob** 相关的任何保留空间或引用保留空间：

[1405] https://www.freebsd.org/cgi/man.cgi?query=zfs&sektion=1&format=html

[1406] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-reservation

[1407] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-refreservation

```
# zfs get reservation storage/home/bob
# zfs get refreservation storage/home/bob
```

22.4.10.压缩

ZFS 提供了透明化的压缩功能。在数据块层面的写入上进行数据压缩可以节约磁盘空间，也可以提高存盘的吞吐效率。如果数据压缩 25%，压缩后的数据会和未压缩时相同的速率被写入磁盘，有效写入速度就达到了 125%。压缩也可以成为去重[1408]功能之外的另一个好选择，因为它并不需要额外的内存。

ZFS 提供了不同的压缩算法，每种都有各自一些劣势。ZFS v5000 的 LZ4 压缩方式可以将整个存储池进行压缩，不会像其它算法那样带来大的性能损失。LZ4 的最大优势是 *early abort* 功能，如果 LZ4 不能在数据 header 部分获得至少 12.5% 以上的压缩率，它就不会对数据块进行压缩以避免浪费 CPU 资源来压缩那些之前要么已被压缩或未被压缩的数据。要了解 ZFS 中可用的不同压缩算法，可以参考术语部分的压缩[1409]条目。

系统管理员可以使用数据集的一些属性来查看有效压缩率：

```
# zfs get used,compressratio,compression,logicalused mypool/compressed_dataset
NAME        PROPERTY          VALUE     SOURCE
mypool/compressed_dataset  used              449G      -
mypool/compressed_dataset  compressratio     1.11x     -
mypool/compressed_dataset  compression       lz4       local
mypool/compressed_dataset  logicalused       496G      -
```

这个数据集使用了 449 GB 的磁盘空间（属性 `used`）。如果不用压缩，它会占用 496 GB 磁盘空间（属性 `logicalused`）。最终的压缩比为 1.11:1。

在和用户配额[1410] 一起使用时压缩会带来一些意想不到的副作用。用户配额限制用户在数据集压缩之后的实际磁盘占用空间。如果一个用户有 10 GB 配额，写入 10 GB 压缩后的数据，他们还可以存储更多数据。如果之后他们要更新一个文件，如数据库，压缩后的数据变得更多或更少，可用空间的总量对他们就会改变。这会导致一些奇怪的情形出现，比如虽然用户的数据没有增加实际的磁盘空间占用（属性 `logicalused`），但压缩率的意外变化让他们触发了空间配额的限制。

压缩在进行备份操作时也会有类似意想不到的影响。空间配额经常被用来限制数据存储以确保有足够的备份空间，但计算配额时又不会考虑 ZFS 处理未压缩备份时可能会实际写入更多压缩数据量情形。

[1408] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-zfs-deduplication
[1409] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-compression
[1410] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-userquota

22.4.11.Z 标准压缩

OpenZFS 2.0 增加了一个新的压缩算法。Z 标准（Zstd) 比默认的 LZ4 提供了更高的压缩比，同时有着比其它算法如 gzip 快得多得速度。OpenZFS 2.0 从 FreeBSD 12.1-RELEASE 版本起可以通过 sysutils/openzfs[1411] 来使用，并在 FreeBSD 13.0-RELEASE 中成为了默认的 zfs 版本。

Zstd 提供了很多压缩级别供选择，在传统压缩比之外还额外提供了性能上的精细控制。Zstd 一项主要优势就是解压速度和压缩级别无关，对于一些写入一次但经常读取的数据，Zstd 允许使用最高压缩级别而不会有读取性能上的损失。

甚至在进行定期的数据更新时，启用压缩功能经常有着更高的性能，这是 ARC 压缩功能带来的最大的优势之一。ZFS 的自适应替换缓存（ARC）技术可以缓存内存中的压缩数据并每次进行解压。这可以让同样大小内存存储更多数据和元数据，增加了缓存的命中率。

ZFS 提供了 19 个 Zstd 压缩级别，每个级别都带来更多磁盘空间节约当然以更慢压缩时间为代价。默认的压缩级别为 `zstd-3`，它提供了优于 LZ4 的压缩率，而且压缩速度不会慢很多。级别 10 以上就需要更多内存来压缩系统中每个区块，低于 16 GB 内存就不要去使用。ZFS 也使用 Zst_fast_level 参数，获得更快的压缩速度，但压缩比会更低。ZFS 支持 `zstd-fast-` 到 `zstd-fast-10`，`zstd-fast-20` 到 `zstd-fast-100` (以单位 10 为步进)，`zstd-fast-500` 和 `zstd-fast-1000` 提供了最小的压缩率，但有着最高的性能。

如过使用 Zstd 后 ZFS 不能获得足够的内存来压缩数据块，它就会按未压缩的方式来存储数据块。这种情形不太可能发生，除非最高级别的 Zstd 在某些系统上被限制了内存使用。ZFS 在 `kstat.zfs.misc.zstd.compress_alloc_fail` 加载 ZFS 模块后就会开始计算这种情况的出现频率。

22.4.12.Z 去重

当启用后，去重[1412]功能利用每个区块数据的校验值来删除重复的区块。当有一个新的区块和已有区块重复时，ZFS 会将已有数据的引用写入而不是整个重复区块。如果数据包含大量重复文件或重复信息的话这样可以节约巨大的磁盘空间。警告：去重功能需要消耗大量的内存，最好启用压缩功能而不是不计代价的来提高节磁盘空间节约量。

要使用去重，在需要的存储池上设置 `dedup` 属性：

```
# zfs set dedup=on pool
```

去重只会影响新写入存储池的数据，已有数据不会被重新去重。新启用去重功能的存储池示例如下：

```
# zpool list
NAME  SIZE ALLOC  FREE   CKPOINT  EXPANDSZ   FRAG   CAP   DEDUP   HEALTH   ALTROOT
pool 2.84G 2.19M 2.83G         -         -     0%    0%   1.00x   ONLINE   -
```

`DEDUP` 列显示了存储池实际的去重率，数值 1.00x 代表数据还没有被去重，下个例子会复制三次系统二进制文件到上例去重存储池中的不同的目录：

[1411] https://cgit.freebsd.org/ports/tree/sysutils/openzfs/pkg-descr

[1412] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-deduplication

```
# for d in dir1 dir2 dir3; do
> mkdir $d && cp -R /usr/bin $d &
> done
```

查看去重了多少冗余数据：

```
# zpool list
NAME SIZE  ALLOC  FREE   CKPOINT  EXPANDSZ   FRAG  CAP   DEDUP   HEALTH   ALTROOT
pool 2.84G 20.9M 2.82G         -         -     0%   0%   3.00x   ONLINE   -
```

DEDUP 列显示了数值 3.00x。检测和去重的数据使用了三分之一的磁盘空间。潜在的可节约空间非常巨大，但需要以足够的内存为代价来追踪去重的数据块。

当存储池中的数据并不冗余时去重并不带来好处，ZFS 可以通过在现有存储池上进行模拟去重来显示潜在的空间节约：

```
# zdb -S pool
Simulated DDT histogram:

bucket              allocated                       referenced
______   ______________________________   ______________________________
refcnt   blocks   LSIZE   PSIZE   DSIZE   blocks   LSIZE   PSIZE   DSIZE
------   ------   -----   -----   -----   ------   -----   -----   -----
     1    2.58M    289G    264G    264G    2.58M    289G    264G    264G
     2     206K   12.6G   10.4G   10.4G     430K   26.4G   21.6G   21.6G
     4    37.6K    692M    276M    276M     170K   3.04G   1.26G   1.26G
     8    2.18K   45.2M   19.4M   19.4M    20.0K    425M    176M    176M
    16      174   2.83M   1.20M   1.20M    3.33K   48.4M   20.4M   20.4M
    32       40   2.17M    222K    222K    1.70K   97.2M   9.91M   9.91M
    64        9     56K   10.5K   10.5K      865   4.96M    948K    948K
   128        2   9.50K      2K      2K      419   2.11M    438K    438K
   256        5   61.5K     12K     12K    1.90K   23.0M   4.47M   4.47M
    1K        2      1K      1K      1K    2.98K   1.49M   1.49M   1.49M
 Total    2.82M    303G    275G    275G    3.20M    319G    287G    287G

dedup = 1.05, compress = 1.11, copies = 1.00, dedup * compress / copies = 1.16
```

在 zdb -S 分析完存储池后，会显示启用去重后的空间缩小比率。在这个例子中。1.16 是一个糟糕的空间节约系数，在这个存储池上使用去重不会节约任何更多磁盘空间，不值得相关的内存开销代价。使用公式 *ratio = dedup * compress / copies*，系统管理员可以规划存储分配，判断工作量是否会包含足够的重复区块来证明额外内存开销的合理性。如果数据可合理压缩，空间节约就会可观。好的习惯是先启用压缩功能因为压缩也可以极大提供性能增长。在可获得更多空间节约并有足够内存完成 DDT[1413] 的情况下启用去重。

[1413] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-deduplication

22.4.13.ZFS 和 Jail

使用 `zfs jail` 和相应的 `jailed` 属性来把 ZFS 数据集分配给 Jail[1414]。`zfs jail` *jailid* 将一个数据关联到指定的 jail。`zfs unjail` 则取消这种关联。要在 jail 内控制数据集，可以设置 `jailed` 属性。ZFS 禁止在主机上挂载 jail 数据集，因为一些挂载节点会对主机安全带来威胁。

22.5.委托管理

完善的权限委托体系可允许普通用户执行 ZFS 管理功能。例如，如果每个用户的主目录是一个数据集，用户需要有权限来创建和销毁他们主目录的快照。执行备份的用户可以获得使用的备份功能的权限。ZFS 允许只有所有用户磁盘使用情况的权限来运行统计脚本。授权进行“代理委托”管理也是可行的。每个子命令和大多数属性都可以进行权限委托。

22.5.1.数据集创建的委托

`zfs allow someuser create` *mydataset* 给予指定的用户在选定的父数据集下创建子数据集的权限。注意：创建一个新的数据集涉及到挂载操作。这需要将 FreeBSD 的sysctl(8)[1415] 参数 `vfs.usermount` 设置为 `1`，以允许非 root 用户挂载文件系统。另一个旨在防止滥用的限制是：非 `root` 用户必须具有文件系统的挂载点的所有者权限。

22.5.2.授权“权限委托”

`zfs allow someuser allow` *mydataset* 可让指定的用户将他们在目标数据集或其子集上的各种权限分配给其他用户。如果一个用户拥有 `snapshot` 权限和 `allow` 权限，该用户就可以将 `snapshot` 权限分配给其他用户。

22.6.高级主题

22.6.1.优化

调整可优化的因素，可使 ZFS 在不同的工作负荷下都呈现最佳表现。

- *13.x 使用 ``vfs.zfs.arc.max``*(12.x 使用 `vfs.zfs.arc_max`)——ARC[1416] 的上限大小。默认是 RAM 的总量减 1GB，或所有 RAM 的 5/8，以多者为准。如果系统上运行的任何其他守护程序或进程可能需要内存，请使用较低的值。在运行时用 sysctl(8)[1417] 调整此值，并在 **/boot/loader.conf** 或 **/etc/sysctl.conf** 中设置它。

[1414] https://docs.freebsd.org/en/books/handbook/jails/index.html#jails

[1415] https://www.freebsd.org/cgi/man.cgi?query=sysctl&sektion=8&format=html

[1416] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-arc

[1417] https://www.freebsd.org/cgi/man.cgi?query=sysctl&sektion=8&format=html

- *13.x 使用 ``vfs.zfs.arc.meta_limit``*(12.x 使用 `vfs.zfs.arc_meta_limit`)——用于限制存储元数据的 ARC[1418] 的数量。默认是 `vfs.zfs.arc_max` 的四分之一。如果工作负载涉及对大量文件和目录的操作，或频繁的元数据操作，增加这个值将提高性能，代价是 ARC[1419] 中拟合的文件数据减少。在运行时用 **/boot/loader.conf** 或 **/etc/sysctl.conf** 中的 sysctl(8)[1420] 调整此值。

- *13.x 使用 ``vfs.zfs.arc.min``*(12.x 使用 `vfs.zfs.arc_min`)——ARC[1421] 的最小大小。默认是 `vfs.zfs.arc_meta_limit` 的一半。调整此值以防止其他应用程序压榨出整个 ARC[1422]。在运行时用 sysctl(8)[1423] 和 **/boot/loader.conf** 或 **/etc/sysctl.conf** 来调整此值。

- *vfs.zfs.vdev.cache.size*——预先分配的内存量，作为池中每个设备的缓存。使用的总内存量将是这个值乘以设备的数量。在启动时或 **/boot/loader.conf** 中设置此值。

- *vfs.zfs.min_auto_ashift*——在创建池时自动使用的较低 `ashift`（扇区大小）。该值是二的幂值。默认值 `9` 代表 `2^9=512`，即扇区大小为 512 字节。为了避免写入放大并获得最佳性能，将此值设置为池中设备使用的最大扇区大小。

  普通磁盘的扇区是 4KB。在这些磁盘上使用默认的 `9` 作为 `ashift` 会导致这些设备的写入放大。单个 4KB 的写入所包含的数据被写成 8 个 512 字节的写入。在创建池时，ZFS 试图从所有设备中读取本地扇区大小，但具有 4 KB 扇区的驱动器报告说它们的扇区是 512 字节，以实现兼容。在创建池之前，将 `vfs.zfs.min_auto_ashift` 设置为 `12`（`2^12=4096`），迫使 ZFS 在这些磁盘上使用 4KB 块以获得最佳性能。

  强制 4KB 区块对于有计划的磁盘升级的池也很有用。未来的磁盘使用 4KB 扇区，而且在创建池后，`ashift` 值不能改变。

  在一些特定的情况下，较小的 512 字节的块大小可能是更好的。当使用 512 字节的磁盘用于数据库或作为虚拟机的存储时，在小的随机读取过程中传输的数据较少。这可以在使用较小的 ZFS 记录大小时提供更好的性能。

- *vfs.zfs.prefetch_disable*——禁用预取。值为 `0` 时启用，`1` 时停用。默认值是 `0`，除非系统的内存小于 4GB。预取的作用是将比请求的更大的块读入 ARC[1424]，希望很快就能需要这些数据。如果工作负载有大量的随机读取，禁用预取实际上可能会通过减少不必要的读取来提高性能。在任何时候都可以用 sysctl(8)[1425] 调整此值。

- *vfs.zfs.vdev.trim_on_init*——控制添加到池中的新设备是否对其运行 `TRIM` 命令。这可以确保 SSD 的最佳性能和寿命，但需要消耗额外的时间。如果设备已经被安全擦除，禁用此设置将使新设备的添加更快。在任何时候都可用 sysctl(8)[1426] 调整此值。

- *vfs.zfs.vdev.max_pending*——限制每个设备的挂起 I/O 请求的数量。一个较高的值会使设备的命令队列保

[1418] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-arc
[1419] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-arc
[1420] https://www.freebsd.org/cgi/man.cgi?query=sysctl&sektion=8&format=html
[1421] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-arc
[1422] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-arc
[1423] https://www.freebsd.org/cgi/man.cgi?query=sysctl&sektion=8&format=html
[1424] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-arc
[1425] https://www.freebsd.org/cgi/man.cgi?query=sysctl&sektion=8&format=html
[1426] https://www.freebsd.org/cgi/man.cgi?query=sysctl&sektion=8&format=html

持满员，并可能带来更高的吞吐量。一个较低的值将减少延迟。在任何时候都可以用 sysctl(8)[1427] 调整此值。

- *vfs.zfs.top_maxinflight*——每个顶层 vdev[1428] 的未完成 I/O 的最高数量。限制命令队列的深度以防止高延迟。这个限制是每个顶级 vdev 的，意味着这个限制独立地适用于每个 镜像[1429]、RAID-Z[1430] 或其他 vdev。在任何时候都可以用 sysctl(8)[1431] 来调整此值。

- *vfs.zfs.l2arc_write_max*——限制每秒写入 L2ARC[1432] 的数据量。这个调节器通过限制写入设备的数据量来延长 SSD 的寿命。在任何时候都可以用 sysctl(8)[1433] 调整此值。

- *vfs.zfs.l2arc_write_boost*—— 在 `vfs.zfs.l2arc_write_max <https://docs.freebsd.org/en/books/handbook/zfs/#zfs-advanced-tuning-l2arc_write_max>`__ 的基础上增加这个可调控的值，并增加对 SSD 的写入速度，直到驱逐 L2ARC[1434] 的第一个块。这个“涡轮增压热身阶段”可以减少重启后空的 L2ARC[1435] 带来的性能损失。在任何时候都可以用 sysctl(8)[1436] 调整此值。

- *vfs.zfs.scrub_delay*——在 `scrub <https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-scrub>`__ 过程中，每个 I/O 之间延迟的刻度数。为了确保 `scrub` 不干扰池的正常运行，如果有任何其他 I/O 正在发生，`scrub` 将在每个命令之间延迟。这个值控制 `scrub` 所产生的总 IOPS（每秒 I/O 数）的限制。设置的颗粒度由 `kern.hz` 的值决定，默认为每秒 1000 次。改变这个设置会导致不同的有效 IOPS 限制。默认值是 4，导致的限制是: 1000 ticks/sec / 4 = 250 IOPS。使用一个 20 的值会得到一个限制: 1000 ticks/sec / 20 = 50 IOPS。池子上的最近活动限制 `scrub` 的速度，这由 `vfs.zfs.scan_idle <https://docs.freebsd.org/en/books/handbook/zfs/#zfs-advanced-tuning-scan_idle>`__ 决定。在任何时候都可以用 sysctl(8)[1437] 调整此值。

- *vfs.zfs.resilver_delay*——在 resilver[1438] 过程中每次 I/O 之间插入延迟的毫秒数量。为了确保 resilver 不会干扰池的正常运行，如果有任何其他 I/O 正在发生，resilver 将在每个命令之间延迟。这个值控制 resilver 产生的总 IOPS（每秒 I/O 数）的限制。ZFS 通过 `kern.hz` 的值来决定设置的粒度，其默认值为每秒 1000 次。改变这个设置会导致不同的有效 IOPS 限制。默认值是 2，导致的限制是: 1000 ticks/sec / 2 = 500 IOPS。如果另一个设备失败可能导致池 Fault[1439]，导致数据丢失，那么将池返回到 Online[1440] 状态可能更为重要。一个 0 值将给予 resilver 操作与其他操作相同的优先级，加速 resilver 过程。池上的其他最近活动限制 resilver 的速度，由 `vfs.zfs.scan_idle <https://docs.freebsd.org/en/books/handbook/zfs/#zfs-advanced-tuning-scan_idle>`__ 决定。在任何时候都可以用 sysctl(8)[1441] 调整此值。

[1427] https://www.freebsd.org/cgi/man.cgi?query=sysctl&sektion=8&format=html
[1428] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-vdev
[1429] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-vdev-mirror
[1430] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-vdev-raidz
[1431] https://www.freebsd.org/cgi/man.cgi?query=sysctl&sektion=8&format=html
[1432] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-l2arc
[1433] https://www.freebsd.org/cgi/man.cgi?query=sysctl&sektion=8&format=html
[1434] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-l2arc
[1435] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-l2arc
[1436] https://www.freebsd.org/cgi/man.cgi?query=sysctl&sektion=8&format=html
[1437] https://www.freebsd.org/cgi/man.cgi?query=sysctl&sektion=8&format=html
[1438] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-resilver
[1439] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-faulted
[1440] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-online
[1441] https://www.freebsd.org/cgi/man.cgi?query=sysctl&sektion=8&format=html

- *vfs.zfs.scan_idle*——在考虑池是空闲的之前，从最后一次操作开始的毫秒数量。当池子处于空闲状态时，ZFS 会禁用 `scrub <https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-scrub>`__ 和 resilver[1442] 的速率限制。在任何时候都可以用 sysctl(8)[1443] 调整此值。
- *vfs.zfs.txg.timeout*——事务组[1444]之间的最高秒数。当前的事务组将写入池中，如果从上一个事务组开始已经过这个时间，就会启动一个新的事务组。如果写入足够的数据，事务组可能会提前触发。默认值是 5 秒。一个更大的值可能会通过延迟异步写入来提高读取性能，但这可能会导致写入事务组时的性能不均衡。在任何时候都可以用 sysctl(8)[1445] 调整此值。

22.6.2.i386 上的 ZFS

ZFS 提供的一些功能是内存密集型的，可能需要在内存有限的系统上进行优化以提高效率。

22.6.2.1.内存

作为一个较低的值，总的系统内存应该至少是 1GB。推荐的内存数量取决于池的大小和 ZFS 使用的功能。一般的经验法则是每 1TB 的存储量需要使用 1GB 的内存。如果使用重复数据删除功能，一般的经验法则是每 TB 存储使用 5 GB 内存进行重复数据删除。虽然有些用户使用的 ZFS 内存更少，但负载过重的系统可能会因为内存耗尽而 panic。ZFS 可能需要对低于推荐内存要求的系统进行进一步的优化。

22.6.2.2.内核配置

由于 i386™ 平台的地址空间限制，i386™ 架构上的 ZFS 用户必须在定制内核配置文件中加入这个选项，重新编译内核，然后重新启动：

```
options         KVA_PAGES=512
```

这就扩展了内核地址空间，允许 `vm.kvm_size` 可调值超过 1GB 的强制限制，或者 PAE 的 2GB 限制。为了找到这个选项的最合适的值，将所需的地址空间（以 MB 为单位）除以 4。在这个例子中，`512` 代表 2GB。

22.6.2.3.引导器的可优化选项

在所有的 FreeBSD 架构上增加 **kmem** 地址空间。在 **/boot/loader.conf** 中加入以下选项，然后重新启动，一个拥有 1GB 物理内存的测试系统就会从中受益：

[1442] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-resilver
[1443] https://www.freebsd.org/cgi/man.cgi?query=sysctl&sektion=8&format=html
[1444] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-txg
[1445] https://www.freebsd.org/cgi/man.cgi?query=sysctl&sektion=8&format=html

```
vm.kmem_size="330M"
vm.kmem_size_max="330M"
vfs.zfs.arc.max="40M"
vfs.zfs.vdev.cache.size="5M"
```

关于 ZFS 相关优化的更详细的建议列表，请参见 https://wiki.freebsd.org/ZFSTuningGuide。

22.7.更多资源

- OpenZFS[1446]
- FreeBSD 维基——ZFS 优化[1447]
- Calomel 博客——ZFS Raidz 的性能、容量和完整性[1448]

22.8.ZFS 特性和术语

与文件系统相比，ZFS 具有根本性的不同。ZFS 结合文件系统和卷管理器的角色，使新的存储设备能够添加到一个实时系统中，并使该池中现有的文件系统上的新空间立即可用。通过结合传统上独立的角色，ZFS 能够克服以前阻碍 RAID 组成长的限制。*vdev* 是一个池中的顶级设备，可以是一个简单的磁盘或一个 RAID 转换，如镜像或 RAID-Z 阵列。ZFS 文件系统（称为数据集）每个都可以访问整个池的综合自由空间。从池中使用的块会减少每个文件系统的可用空间。这种方法避免了广泛分区的常见缺陷，即自由空间在各分区之间变得支离破碎。

池

存储 池是 ZFS 的最基本的构成要素。池由一个或多个 vdev 组成，是存储数据的基础设备。然后，池被用来创建一个或多个文件系统（数据集）或块设备（卷）。这些数据集和卷共享池中的剩余可用空间。每个池由一个名称和一个 GUID 来唯一识别。池的 ZFS 版本号决定了可用的功能。

[1446] https://openzfs.org/
[1447] https://wiki.freebsd.org/ZFSTuningGuide
[1448] https://calomel.org/zfs_raid_speed_capacity.html

vdev 类型

一个池由一个或多个 vdev 组成，它们本身是一个单一的磁盘或一组磁盘，被转化为 RAID。当使用大量的 vdev 时，ZFS 将数据分散到各个 vdev 中，以提高性能并最大限度地提高可用空间。所有的 vdev 必须至少有 128MB 大小。

- 磁盘—最基本的 vdev 类型是一个标准块设备。这可以是整个磁盘（例如 `/dev/ada0` 或 `/dev/da0`）或一个分区（`/dev/ada0p3`）。在 FreeBSD 上，使用一个分区而不是整个磁盘不会有任何性能损失。这与 Solaris 文档中的建议不同。

  当心

  强烈不建议使用整个磁盘作为可启动池的一部分，因为这可能导致池无法启动。同样地，你也不应该使用整个磁盘作为镜像或 RAID-Z vdev 的一部分。在启动时可靠地确定一个未分区的磁盘的大小是不可能的，因为没有地方可以放入启动代码。

- 文件 - 普通文件可以组成 ZFS 池，这对测试和实验很有用。在`zpool create`中使用文件的完整路径作为设备路径。

- 镜像 - 当创建一个镜像时，指定镜像关键字，然后是镜像的成员设备的列表。镜像由两个或多个设备组成，将所有数据写入所有成员设备。一个镜像 vdev 将容纳与其最小的成员一样多的数据。一个镜像 vdev 可以承受除一个成员外的所有故障而不丢失任何数据。

  注意

  要在任何时候将一个普通的单磁盘 vdev 升级为镜像 vdev，请使用zpool attach[1449]。

- RAID-Z - ZFS 使用 RAID-Z，这是标准 RAID-5 的一个变种，它提供更好的奇偶校验分布，并消除“RAID-5 write hole”，即数据和奇偶校验信息在意外重新启动后变得不一致。ZFS 支持三种级别的 RAID-Z，提供不同级别的冗余，以换取不断减少的可用存储空间。ZFS 使用 RAID-Z1 到 RAID-Z3，基于阵列中的奇偶校验设备的数量和在磁盘池停止运行前可能发生故障的磁盘数量。

  - 在一个有四个磁盘的 RAID-Z1 配置中，每个磁盘都是 1TB，可用的存储空间是 3TB，在有一个故障磁盘的情况下，磁盘池仍然能够在降级模式下运行。如果在更换和恢复故障磁盘之前，另一个磁盘脱机将导致所有的池数据丢失。
  - 在有 8 个 1TB 磁盘的 RAID-Z3 配置中，该卷将提供 5TB 的可用空间，并且在有 3 个故障磁盘的情况下仍能运行。Sun™ 建议在一个 vdev 中不超过九个磁盘。如果更多的磁盘组成配置，建议将它们分成独立的 vdev，并在它们之间进行磁盘池数据的剥离。
  - 一个由两个 RAID-Z2 vdev 组成的配置，每个 vdev 有 8 个磁盘，将创建类似 RAID-60 的阵列。一个 RAID-Z 组的存储容量大约是最小的磁盘的大小乘以非奇偶性磁盘的数量。在 RAID-Z1 中 4 个 1TB 的磁盘有大约 3TB 的有效大小，而在 RAID-Z3 中 8 个 1TB 的磁盘阵列将产生 5TB 的可用空间。

- 备用 - ZFS 有一个特殊的伪 vdev 类型，用于跟踪可用的热备用。注意已安装的热备设备不会自动部署；使用`zfs replace`手动配置它们来替换故障设备。

[1449] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-zpool-attach

- 日志 - ZFS 日志设备，也被称为 ZFS 意图日志ZIL[1450]，将意图日志从常规池设备转移到一个专用设备，通常是一个 SSD。拥有一个专用的日志设备可以提高像数据库这样的有大量同步写入的应用程序的性能。日志设备的镜像是可能的，但不支持 RAID-Z。如果使用大量的日志设备，写操作将在它们之间进行负载平衡。
- 缓存 - 在池里添加一个缓存 vdev 将把缓存的存储添加到L2ARC[1451]中。镜像高速缓存设备是不可能的。因为缓存设备只存储现有数据的新副本，所以没有数据丢失的风险。

事务组 (TXG)

事务组是 ZFS 将块变化分组并写入池中的方式。事务组是 ZFS 用来确保一致性的原子单元。ZFS 给每个事务组分配一个唯一的 64 位连续标识符。一次最多可以有三个活动的事务组，在这三种状态中各有一个。

- Open - 一个新的事务组在开放状态下开始，接受新的写入。总是有一个交易组处于开放状态，但如果交易组达到一个限制，它可能会拒绝新的写入。一旦开放的事务组达到限制，或达到 vfs.zfs.txg.timeout[1452]，事务组将进入下一个状态。
- Quiescing - 一个短暂的状态，允许任何未决的操作完成，而不阻止创建一个新的开放事务组。一旦该组中的所有事务都完成，该事务组将进入最后一个状态。
- 同步 - 将交易组中的所有数据写入稳定的存储。这个过程将反过来改变其他数据，如元数据和空间图，ZFS 也将把这些数据写到稳定存储中。同步的过程包括几个环节。在第一次和最大的一次，所有改变的数据块；接下来是元数据，这可能需要几次才能完成。由于为数据块分配空间会产生新的元数据，同步状态不能完成，直到一个不使用任何新空间的过程完成。同步状态也是同步任务完成的地方。同步任务是一些管理操作，如创建或销毁快照和数据集，完成 uberblock 的变化。一旦同步状态完成，处于静止状态的事务组就会推进到同步状态。所有的管理功能，如 快照[1453] 写入作为事务组的一部分。ZFS 将创建的同步任务添加到开放的事务组中，该事务组尽可能快地推进到同步状态，以减少管理命令的延迟。

自适应替换缓存（ARC）

ZFS 使用自适应替换缓存（ARC），而不是更传统的最近使用最少的缓存（LRU）。LRU缓存是一个简单的缓存项目列表，按照对象最近被使用的时间排序，将新项目添加到列表的头部。当缓存已满时，从列表的尾部驱逐项目，为更多的活动对象腾出空间。一个 ARC 由四个列表组成；最近使用（MRU）和最常使用（MFU）的对象，加上每个对象的幽灵列表。这些幽灵列表跟踪被驱逐的对象，以防止将它们重新添加到缓存中。这样就避免那些有偶尔使用历史的对象，从而提高了缓存的命中率。同时使用 MRU 和 MFU 的另一个好处是，扫描整个文件系统会驱逐 MRU 或 LRU 缓存中的所有数据，以支持这些新访问的内容。在 ZFS 中，也有一个 MFU 来跟踪最经常使用的对象，而最经常访问的块的缓存仍然存在。

[1450] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-zil

[1451] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-l2arc

[1452] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-advanced-tuning-txg-timeout

[1453] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-snapshot

L2ARC

L2ARC 是 ZFS 缓存系统的第二层。RAM 存储主 ARC。由于可用的 RAM 数量通常是有限的，ZFS 也可以使用 cache vdevs[1454]。固态硬盘（SSD）经常被用作这些缓存设备，因为与传统的旋转磁盘相比，它们的速度更高，延迟更低。L2ARC 完全是可有可无的，但是拥有它可以提高 SSD 上缓存文件的读取速度，而不必从普通磁盘上读取。L2ARC 也可以加快 重复数据删除[1455] 的速度，因为不适合在 RAM 中，但适合在 L2ARC 中的重复数据删除表（DDT）将比必须从磁盘读取的 DDT 快得多。对添加到缓存设备的数据速率的限制可以防止因额外的写入而过早地磨损 SSD。在缓存满之前（第一个块被驱逐以腾出空间），对 L2ARC 的写入限制在写入限制和提升限制之和，之后限制在写入限制。一对 sysctl(8)[1456] 的值控制着这些速率限制。vfs.zfs.l2arc_write_max[1457] 控制每秒写入缓存的字节数，而 vfs.zfs.l2arc_write_boost[1458] 在 “Turbo Warmup Phase”（Write Boost）增加这个限制。

ZIL

ZIL 通过使用比主存储池中使用的存储设备（如 SSD）更快的存储设备，来加速同步。当一个应用程序要求同步写入时（保证数据被存储到磁盘，而不仅仅是为以后的写入进行缓存），将数据写入更快的 ZIL 存储，然后再将其冲出到常规磁盘上，这就大大降低延迟，提高性能。像数据库这样的同步工作负载将单独从 ZIL 中获益。常规的异步写操作，如复制文件，则根本不会用到 ZIL。

写入时复制

与传统的文件系统不同，ZFS 写一个不同的块，而不是覆盖原地的旧数据。当完成这个写入时，元数据会更新以指向新的位置。当发生 shorn write（在写文件的过程中系统崩溃或断电）时，文件的整个原始内容仍然可用，ZFS 会丢弃不完整的写操作。这也意味着 ZFS 在意外关机后不需要进行 fsck(8)[1459]。

数据集

数据集是 ZFS 文件系统、卷、快照或 Clone 的通用术语。每个数据集都有一个唯一的名字，格式为 *poolname/path@snapshot*。池的根也是一个数据集。子数据集有像目录一样的分层名称。例如，*mypool/home*，home 数据集，是 mypool 的一个子集，从它那里继承属性。通过创建 *mypool/home/user* 进一步扩展。这个孙子数据集将继承父母和祖父母的属性。在子集上设置属性以覆盖从父集和祖集上继承的默认值。数据集及其子集的管理可以被 委托[1460]。

[1454] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-vdev-cache
[1455] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-deduplication
[1456] https://www.freebsd.org/cgi/man.cgi?query=sysctl&sektion=8&format=html
[1457] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-advanced-tuning-l2arc_write_max
[1458] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-advanced-tuning-l2arc_write_boost
[1459] https://www.freebsd.org/cgi/man.cgi?query=fsck&sektion=8&format=html
[1460] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-zfs-allow

文件系统

一个 ZFS 数据集最常被用作一个文件系统。像大多数其他文件系统一样，ZFS 文件系统挂载在系统目录层次中的某个地方，并包含自己的文件和目录，有权限、标志和其他元数据。

卷

ZFS 也可以创建卷，它看起来像磁盘设备。卷有很多与数据集相同的功能，包括写时拷贝、快照、Clone 和校验。卷对于在 ZFS 之上运行其他文件系统格式非常有用，比如 UFS 虚拟化，或者导出 iSCSI 扩展。

快照

ZFS 的 写入时复制[1461]（COW）设计允许几乎即时的、一致的、具有任意名称的快照。在对数据集进行快照后，或对包括所有子数据集的父数据集进行递归快照后，新数据会进入新的区块，但不会将旧的区块回收为自由空间。快照包含原始文件系统的版本，而实时文件系统包含自拍摄快照以来的任何变化，不使用其他空间。写入实时文件系统的新数据使用新的块来存储这些数据。当这些区块不再用于实时文件系统，而是单独用于快照时，快照将增长。将这些快照挂载为只读允许恢复以前的文件版本。将实时文件系统 回滚[1462] 到一个特定的快照是可能的，撤消在拍摄快照后发生的任何变化。池中的每个区块都有一个参考计数器，用于跟踪使用该区块的快照、Clone、数据集或卷。当文件和快照被删除时，参考计数就会减少，当不再参考一个区块时，就会回收自由空间。用 hold[1463] 标记快照的结果是，任何试图销毁它的行为都会返回一个 `EBUSY` 错误。每个快照都可以有一个独特名称的 hold。release[1464] 命令会移除 hold，这样快照就可以被删除。快照、Clone 和回滚可以在卷上进行，但独立挂载则不行。

Clone

Clone 一个快照也是可能的。Clone 是快照的一个可写版本，允许文件系统作为一个新的数据集分叉。与快照一样，Clone 最初不消耗新的空间。当写入 Clone 的新数据使用新的块时，Clone 的大小会增长。当 Clone 的文件系统或卷中的块被覆盖时，先前块上的参考计数会减少。移除作为 Clone 基础的快照是不可能的，因为 Clone 依赖于它。快照是父母，而 Clone 是孩子。Clone 可以被提升，扭转这种依赖关系，使 Clone 成为父，而先前的父成为子。这个操作不需要新的空间。由于父代和子代所使用的空间量是相反的，它可能会影响现有的配额和保留。

[1461] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-cow
[1462] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-zfs-snapshot
[1463] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-zfs-snapshot
[1464] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-zfs-snapshot

校验和

每个区块也会被校验。使用的校验算法是每个数据集的属性，见 set[1465]。每个块的校验和在读取时被透明地验证，允许 ZFS 检测无声的损坏。如果读取的数据与预期的校验和不一致，ZFS 将尝试从任何可用的冗余中恢复数据，如镜像或 RAID-Z。使用 scrub[1466] 对所有校验和进行验证。校验和算法包括 `fletcher2*`、`fletcher4*`、`sha256`。`fletcher` 算法更快，但是 `sha256` 是一个强大的加密散列，以牺牲一些性能为代价，碰撞的机会更少。停用校验和是可能的，但强烈不建议这样做。

压缩

每个数据集都有一个压缩属性，其默认值为关闭。把这个属性设置为一个可用的压缩算法。这将导致对写入数据集的所有新数据进行压缩。除了减少使用的空间，读和写的吞吐量通常会增加，因为需要读或写的块更少。

- LZ4 - 在 ZFS 池版本 5000 中添加（feature flags），LZ4 现在是推荐的压缩算法。当对可压缩数据进行操作时，LZ4 的工作速度比 LZJB 快 50% 左右，而在对不可压缩数据进行操作时，LZ4 的工作速度超过三倍。LZ4 的解压速度也比 LZJB 快大约 80%。在现代 CPU 上，LZ4 的压缩速度通常超过 500MB/s，解压速度超过 1.5GB/s（每个 CPU 核心）。
- LZJB - 默认的压缩算法。由 Jeff Bonwick（ZFS 的原始创建者之一）创建。与 GZIP 相比，LZJB 提供良好的压缩效果，而且 CPU 开销较少。在未来，默认的压缩算法将改变为 LZ4。
- GZIP - ZFS 中可用的一种流行的流压缩算法。使用 GZIP 的主要优势之一是其可配置的压缩级别。当设置`compress` 属性时，管理员可以选择压缩级别，范围从最低压缩级别的 `gzip1` 到最高压缩级别的`gzip9`。这使管理员能够控制用多少 CPU 时间来换取节省的磁盘空间。
- ZLE - 零长度编码是一种特殊的压缩算法，可以单独压缩连续运行的零。当数据集包含大量的零块时，这种压缩算法很有用。

Copies

当设置为一个大于 1 的值时，`copies` 属性指示 ZFS 维护文件系统[1467]或卷[1468] 中每个块的副本。在重要的数据集上设置这个属性提供额外的冗余，可以从中恢复不符合其校验和的块。在没有冗余的池中，副本功能是唯一的冗余形式。复制功能可以从单个坏扇区或其他形式的轻微损坏中恢复，但它不能保护池免受整个磁盘的损失。

[1465] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-zfs-set

[1466] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-scrub

[1467] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-filesystem

[1468] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-volume

去重

校验和使其有可能在写入数据时发现重复的块。通过重复数据删除，一个现有的、相同的块的参考计数增加，节省存储空间。ZFS 在内存中保留一个重复数据删除表（DDT）来检测重复的块。该表包含一个唯一的校验和的列表，这些块的位置，以及参考计数。当写入新数据时，ZFS 计算校验和并将其与列表进行比较。当找到一个匹配时，它使用现有的块。使用 SHA256 校验算法和重复数据删除提供一个安全的加密哈希。重复数据删除是可调整的。如果 `dedup` 设置为 `on`，那么一个匹配的校验和意味着数据是相同的。将 `dedup` 设置为 `verify`，ZFS 会对数据进行逐个字节的检查，确保它们实际上是相同的。如果数据不完全相同，ZFS 将注意到哈希碰撞，并分别存储这两个块。由于 DDT 必须存储每个独特区块的哈希值，它消耗大量的内存。一般的经验法则是每 1TB 的重复数据需要 5-6GB 的内存）。在不实际的情况下，要有足够的内存来保持整个 DDT 在内存中，性能将受到很大影响，因为 DDT 必须在写入每个新块之前从磁盘读取。重复数据删除可以使用 L2ARC 来存储 DDT，在快速系统内存和较慢的磁盘之间提供一个中间地带。考虑使用压缩来代替，这通常能提供几乎同样多的空间节省，而不需要增加内存。

清洗

ZFS 有 `scrub`，而无需 fsck(8)[1469] 那样的一致性检查。`scrub` 读取存储在池中的所有数据块，并根据存储在元数据中的已知良好的校验和验证它们。对存储在池中的所有数据的定期检查确保在需要之前恢复任何损坏的块。在一次不正常的关机后无需清洗，但好的做法是至少每三个月清洗一次。ZFS 在正常使用期间会验证每个块的校验和，但是清洗可以确保检查即使是不经常使用的块是否有不易发现的损坏。ZFS 提高了存档存储情况下的数据安全性。。用 `vfs.zfs.scrub_delay` 调整清洗的相对优先级，以防止清洗降低池上其他工作负载的性能。

数据集配额

ZFS 提供快速和准确的数据集、用户和组空间核算，以及配额和空间保留。这使管理员能够精细地控制空间分配，并允许为关键文件系统保留空间。ZFS支持不同类型的配额：数据集配额、参考 配额（refquota）[1470]、用户配额[1471] 和 组配额[1472]。配额限制一个数据集及其后代的总大小，包括数据集的快照、子数据集和这些数据集的快照。

注意

卷不支持配额，因为 `volsize` 属性作为一个隐含的配额。

[1469] https://www.freebsd.org/cgi/man.cgi?query=fsck&sektion=8&format=html

[1470] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-refquota%22

[1471] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-userquota

[1472] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-groupquota

引用配额

引用配额通过强制执行一个硬限制来限制数据集可以消耗的空间量。这个硬性限制包括仅由数据集引用的空间，不包括由后代使用的空间，如文件系统或快照。

用户配额

用户配额对于限制指定用户使用的空间数量很有用。

组配额

组的配额限制一个指定的组所能消耗的空间量。

数据集保留

`reservation` 属性使得为一个特定的数据集及其后代保证一定的空间成为可能。这意味着在 `storage/home/bob` 上设置一个 10GB 的保留，可以防止其他数据集使用所有的自由空间，为这个数据集保留至少 10GB 的空间。与普通的 refreservation[1473] 不同，快照和后裔使用的空间不计入保留中。例如，如果拍摄 `storage/home/bob` 的快照，必须有足够的磁盘空间，而不是刷新 `refreservation` 的数量，这样操作才能成功。主数据集的后代不会被计算在 `refreservation` 数量中，因此不会占用空间集。任何形式的保留在一些情况下都是有用的，比如在一个新系统中计划和测试磁盘空间分配的适宜性，或者确保文件系统中有足够的空间用于音频日志或系统恢复程序和文件。

引用保留

`refreservation` 属性使得保证一个特定数据集的使用空间量成为可能，但不包括其后代。这意味着在`storage/home/bob` 上设置一个 10GB 的保留，另一个数据集试图使用空闲空间时会为这个数据集保留至少 10GB 的空间。与普通的 refreservation[1474] 相比，快照和子代数据集使用的空间不计入保留。例如，如果拍摄storage/home/bob 的快照，必须有足够的磁盘空间，而不是 `refreservation` 总量，这样操作才能成功。主数据集的后代不会被计算在 `refreservation` 数量中，因此不会占用空间集。

[1473] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-refreservation

[1474] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-refreservation

Resilver

当替换一个损坏的磁盘时，ZFS 必须用丢失的数据填充新的磁盘。*Resilvering* 是使用分布在剩余驱动器上的奇偶校验信息来计算并将丢失的数据写入新的驱动器的过程。

Online

处于 `Online` 状态的池或 vdev 有其成员设备连接并完全运行。处于 `Online` 状态的单个设备正在运行。

Offline

如果存在足够的冗余以避免将池或vdev置于 Faulted[1475] 状态，则管理员将单个设备置于 `Offline` 状态。管理员可能会选择将一个磁盘脱机，以准备替换它，或者使它更容易被识别。

Degraded

一个处于`Degraded`状态的池或 vdev 有一个或多个磁盘消失或失效。该池仍然可以使用，但是如果其他设备发生故障，该池可能变得无法恢复。重新连接丢失的设备或替换失败的磁盘，在重新连接的设备或新设备完成 Resilver[1476] 过程后，池将恢复到 Online 状态。

Faulted

处于`Faulted`状态的池或 vdev 不再运行。不能再访问数据。当丢失或故障的设备数量超过 vdev 的冗余水平时，池或 vdev 将进入故障状态。如果重新连接丢失的设备，池将返回到 Online 状态。如果没有足够的冗余来补偿故障磁盘的数量，就会丢失池里的数据，需要从备份中恢复。

[1475] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-faulted

[1476] https://docs.freebsd.org/en/books/handbook/zfs/#zfs-term-resilver

第二十三章其他文件系统

23.1.概述

任何操作系统都由文件系统作为其一部分组成。它们允许用户上传和存储文件，提供对数据的访问，并使硬盘发挥作用。不同的操作系统在它们的本地文件系统上有所不同。传统上，FreeBSD 的本地文件系统是 Unix 文件系统 UFS（已被现代化为 UFS2）。从 FreeBSD 7.0 开始，Z 文件系统（ZFS）也可以作为本地文件系统使用。请参阅 Z 文件系统 (ZFS)[1477] 了解更多信息。

除了原生文件系统之外，FreeBSD 还支持许多其他文件系统，这样就可以在本地访问来自其他操作系统的数据，例如存储在本地连接的 USB 存储设备、闪存驱动器和硬盘上的数据。这包括对 Linux® 扩展文件系统（EXT）的支持。

FreeBSD 对各种文件系统的支持程度不同。有些需要加载内核模块，有些可能需要安装工具集。一些非原生文件系统的是支持完全读写的，而另一些是只读的。

读完本章后，你会了解到：

- 原生文件系统和被支持的文件系统之间的区别。
- 哪些文件系统被 FreeBSD 所支持。
- 如何启用、配置、访问和使用非本地文件系统。

在阅读本章之前，你应该：

- 理解 UNIX® 和 FreeBSD 基础[1478]。
- 熟悉配置和编译 FreeBSD 内核[1479]。
- 对在 FreeBSD 中安装软件[1480]感觉从容自在。

[1477] https://docs.freebsd.org/en/books/handbook/zfs/index.html#zfs
[1478] https://docs.freebsd.org/en/books/handbook/basics/index.html#basics
[1479] https://docs.freebsd.org/en/books/handbook/kernelconfig/index.html#kernelconfig
[1480] https://docs.freebsd.org/en/books/handbook/ports/index.html#ports

- 对 FreeBSD 中的磁盘[1481]、存储和设备名称有一定的熟悉程度。

23.2. Linux® 文件系统

FreeBSD 提供了对几个 Linux® 文件系统的内置支持。这一节演示了如何开启对可用的 Linux® 文件系统的支持以及如何挂载这些文件系统。

23.2.1. ext2 / ext3 / ext4

对 ext2 文件系统的内核级支持从 FreeBSD 2.2 起就开始了。ext2fs(5)[1482] 驱动程序允许 FreeBSD 内核读取和写入 ext2, ext3, 和 ext4 文件系统。

> 注意
> 目前还不支持日志和加密功能。

通过指定其 FreeBSD 分区名称和现有的挂载点来挂载 ext 卷，可访问 ext 文件系统。该例子将 **/dev/ada1s1** 挂载到 **/mnt** 上：

```
# mount -t ext2fs /dev/ada1s1 /mnt
```

[1481] https://docs.freebsd.org/en/books/handbook/disks/index.html#disks

[1482] https://www.freebsd.org/cgi/man.cgi?query=ext2fs&sektion=5&format=html

第二十四章虚拟化

24.1.概述

虚拟化软件可让一台计算机同时运行多个操作系统。这种用于个人电脑的系统软件通常涉及一个运行虚拟化软件的宿主机（host）操作系统，并支持任何数量的客户机（guest）操作系统。

读完本章后，你将知道：

- 宿主机操作系统与客户机操作系统的区别。
- 如何在如下虚拟化平台中安装 FreeBSD：
  - Parallels Desktop（Apple® macOS®）
  - VMware Fusion（Apple® macOS®）
  - VirtualBox™（Microsoft® Windows®、基于 Intel® 芯片的 Apple® macOS®、Linux）
  - bhyve（FreeBSD）
- 如何优化 FreeBSD 系统以在虚拟化环境下发挥最佳性能。

在你阅读本章之前，你应该：

- 理解 FreeBSD 基础[1483]。
- 了解如何安装 FreeBSD[1484]。
- 了解如何设置网络连接[1485]。
- 了解如何安装额外的第三方软件[1486]。

[1483] https://docs.freebsd.org/en/books/handbook/basics/index.html#basics

[1484] https://docs.freebsd.org/en/books/handbook/bsdinstall/index.html#bsdinstall

[1485] https://docs.freebsd.org/en/books/handbook/advanced-networking/index.html#advanced-networking

[1486] https://docs.freebsd.org/en/books/handbook/ports/index.html#ports

24.2.使用 macOS® 上的 Parallels Desktop 安装 FreeBSD

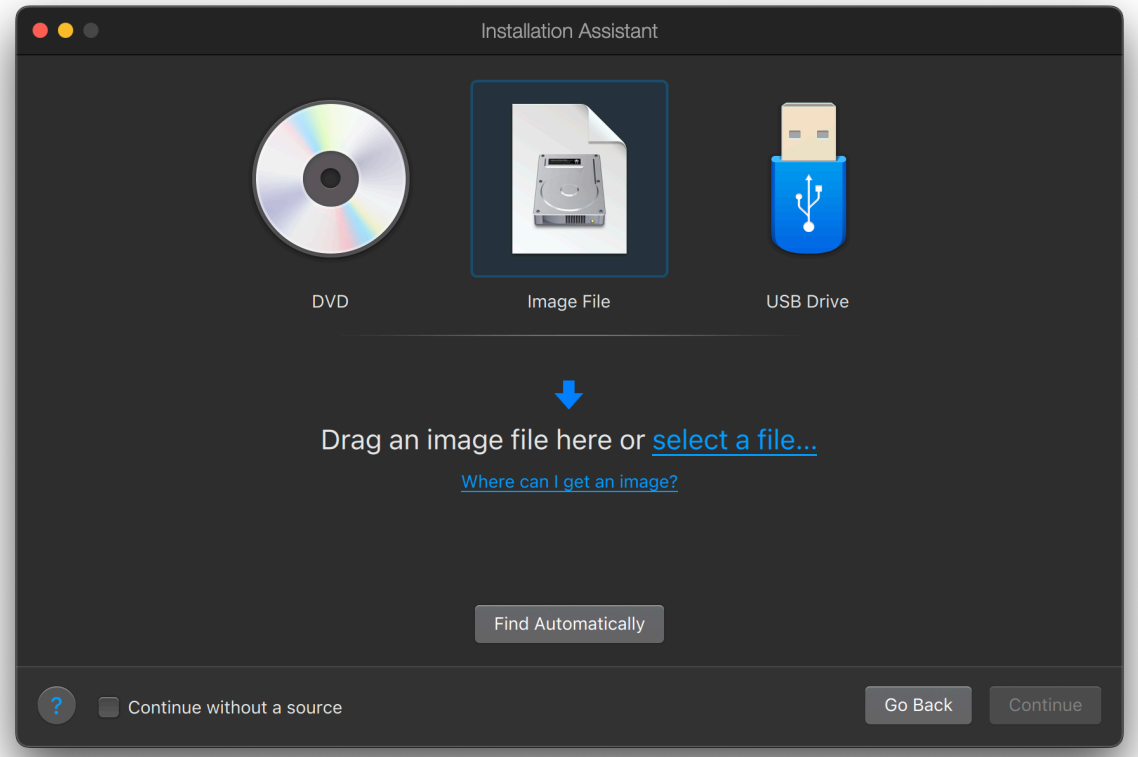
Parallels Desktop for Mac® 是为 Apple® Mac® 计算机所开发的商用软件，支持 macOS® 10.14.6 及更高版本。FreeBSD 是得到其完整支持的客户机系统。在 macOS® 上安装好 Parallels 之后，用户需要配置虚拟机并安装所需的客户机操作系统。

24.2.1.在 Parallels Desktop on Mac® 上安装 FreeBSD

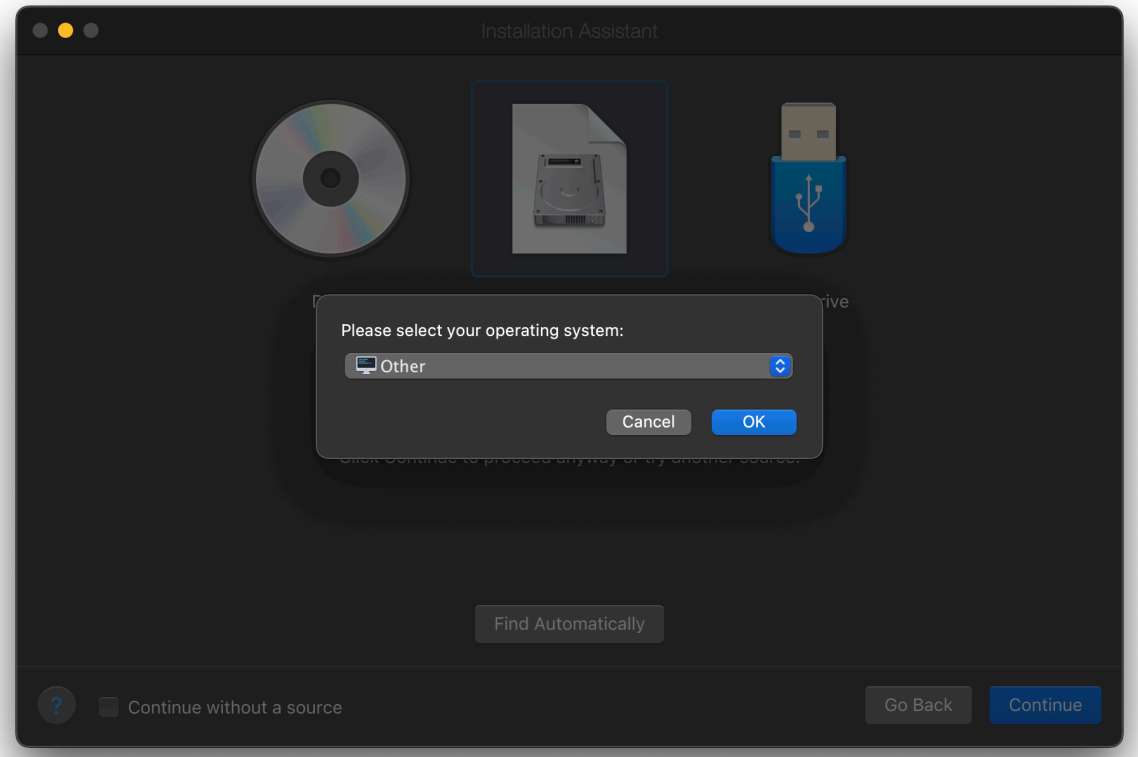
在 Parallels 上安装 FreeBSD 的第一步是创建一个新的用于安装的虚拟机。

选择**从 DVD 或镜像文件安装 Windows 或其他操作系统**，然后继续。

选择 FreeBSD 镜像文件。


Installation Assistant
DVD
Image File
USB Drive
Drag an image file here or select a file...
Where can I get an image?
Find Automatically
Continue without a source
Go Back
Continue

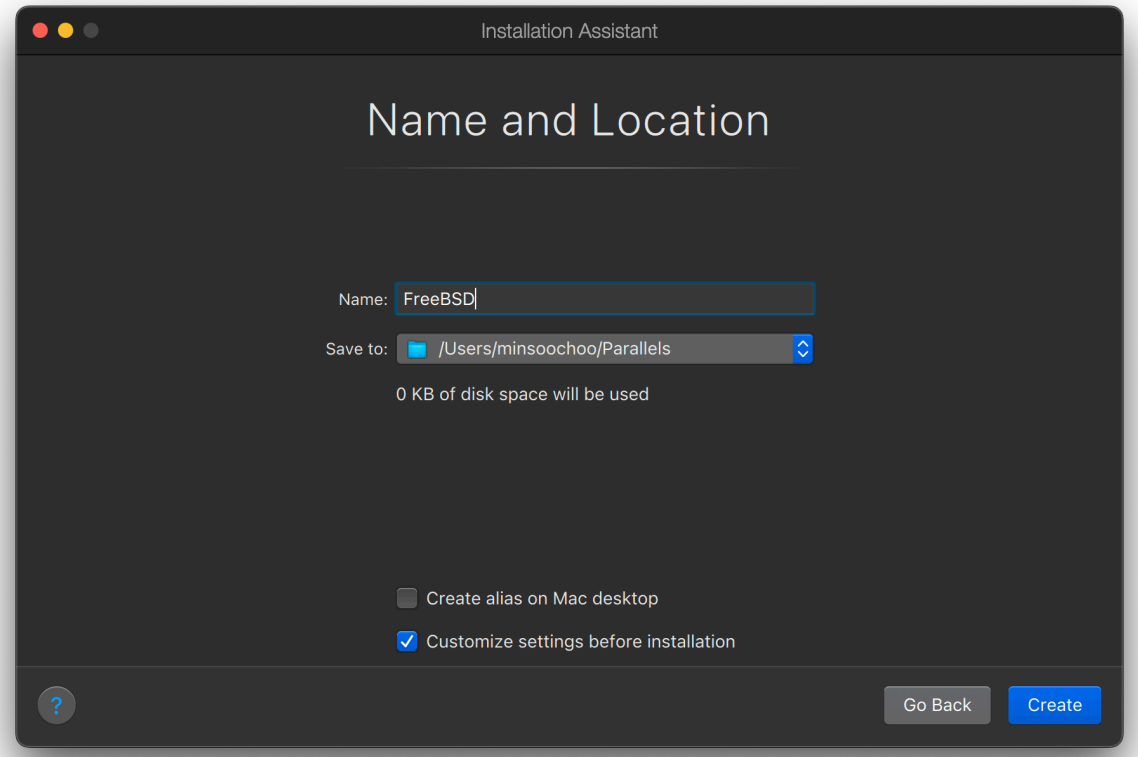
选择**其他操作系统**。



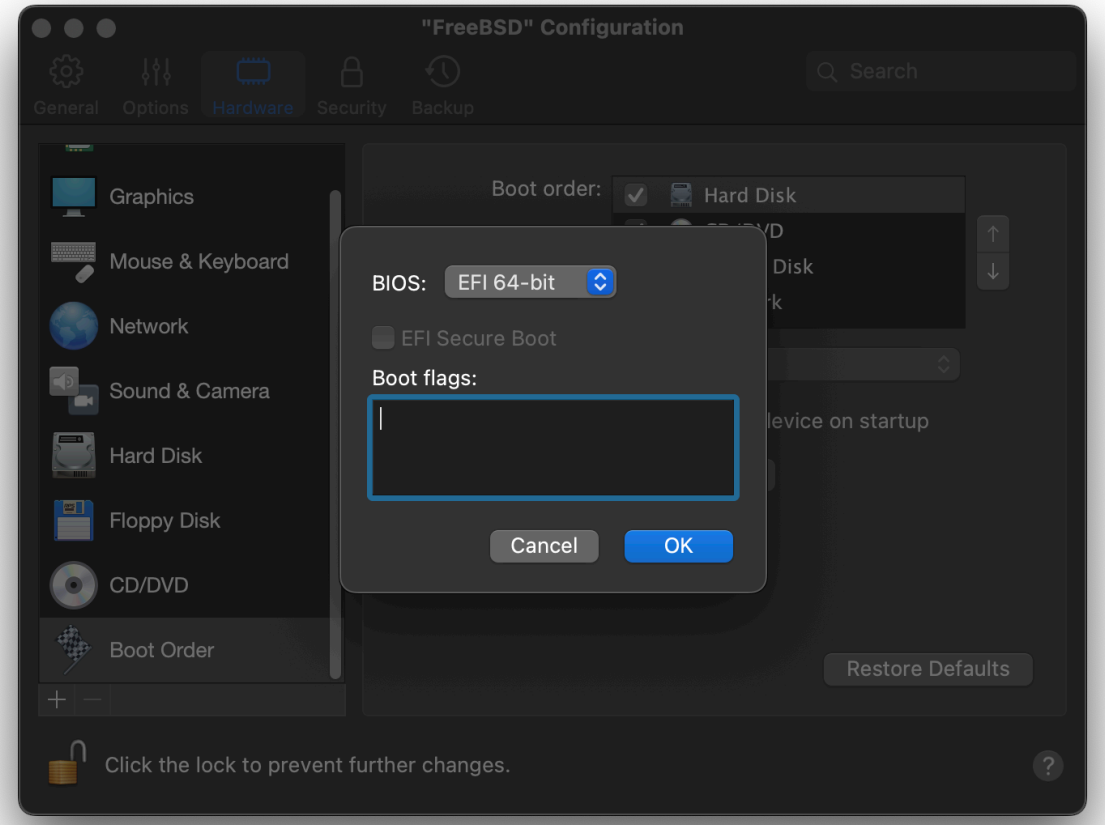
警告

选择 FreeBSD 会在启动时导致引导错误。

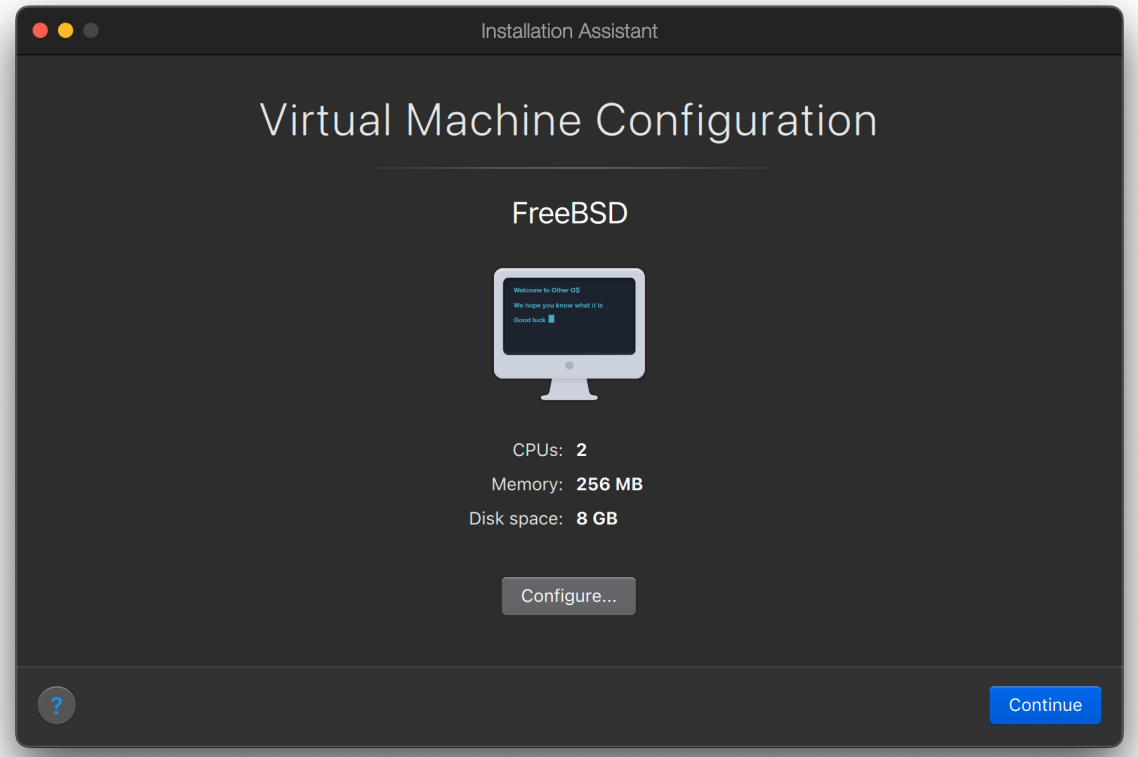
为虚拟机命名并检查**自定义设置**



弹出配置窗口时，进入**硬件**标签页，选择**启动顺序**，单击**高级**。然后选择 **EFI 64-bit** 作为 **BIOS**。



单击**确定**，关闭配置窗口，单击**继续**。



虚拟机将自动引导。按照常规步骤安装 FreeBSD。

24.2.2.在 Parallels 上配置 FreeBSD

在 macOS® 的 Parallels 上成功安装 FreeBSD 之后，可以进行如下几个配置来对虚拟化操作系统进行优化。

1. 设置引导加载器变量

   最重要的一步是调整可优化的 `kern.hz` 来减少 FreeBSD 在 Parallels 环境下的 CPU 使用率。这可通过在 **/boot/loader.conf** 中加入如下一行来实现：

   ```
   kern.hz=100
   ```

   在没有此项设置的情况下，Parallels 中运行的 FreeBSD 在闲置状态下会占用大约 15% 的 CPU（在一台搭载单颗处理器的 iMac® 上）。进行设置后，占用率可以降低到 5% 左右。

2. 创建新的内核配置文件

   所有 SCSI、火线和 USB 设备驱动都可以从定制内核配置文件中移除。Parallels 提供了一个虚拟网络适

配器并使用了驱动 ed(4)[1487]，所以除了 ed(4)[1488] 和 miibus(4)[1489] 之外的网络驱动都可以从内核中移除。

3. 配置网络

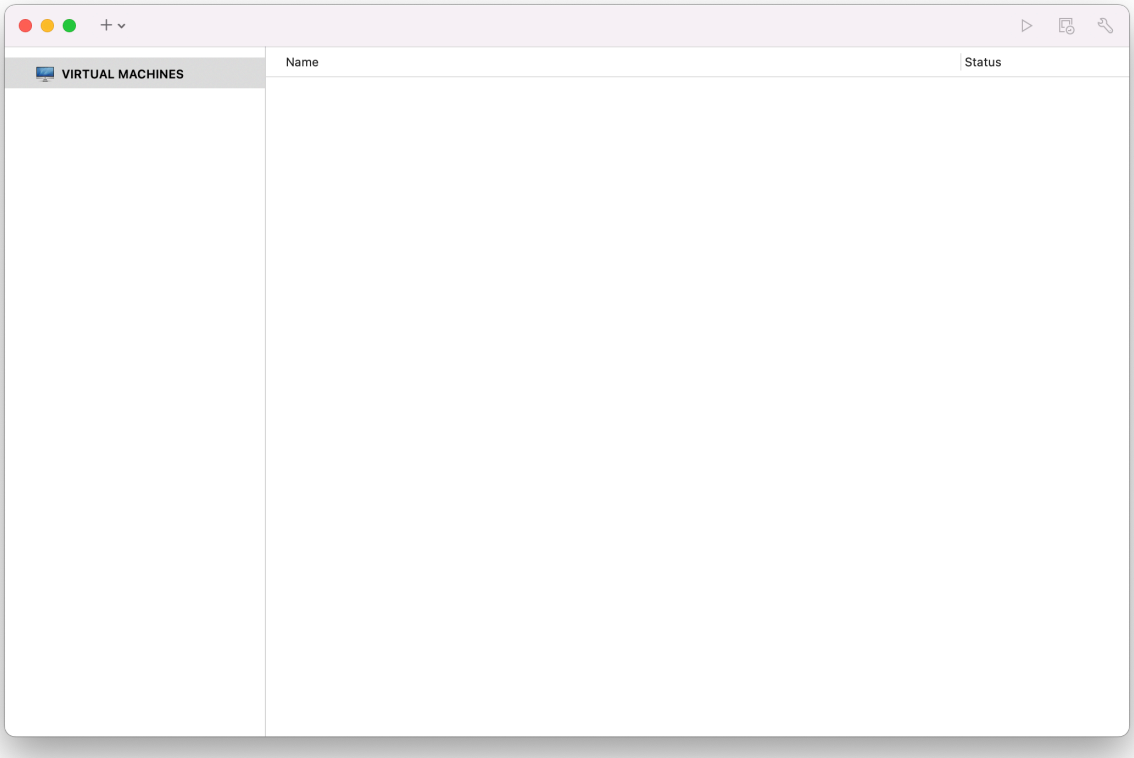
   最基本的网络配置使用 DHCP 来将虚拟机连接到宿主机 Mac® 所在的同一个局域网中。这可以通过向 **/etc/rc.conf** 中添加 `ifconfig_ed0="DHCP"` 来实现。更高级的网络设置请参见高级网络[1490]。

24.3.使用 macOS® 上的 VMware Fusion 安装 FreeBSD

VMware Fusion for Mac® 是为基于 Intel® 的 Apple® Mac® 计算机所开发的商用软件，支持 macOS® 10.11 及更高版本。FreeBSD 是得到其完整支持的客户机系统。在 macOS® 上安装好 VMware Fusion 之后，用户需要配置一个虚拟机并安装所需的客户机操作系统。

24.3.1.在 VMware Fusion 上安装 FreeBSD

第一步是启动 VMware Fusion 并加载虚拟机资源库。点击 + → New⋯来创建虚拟机：



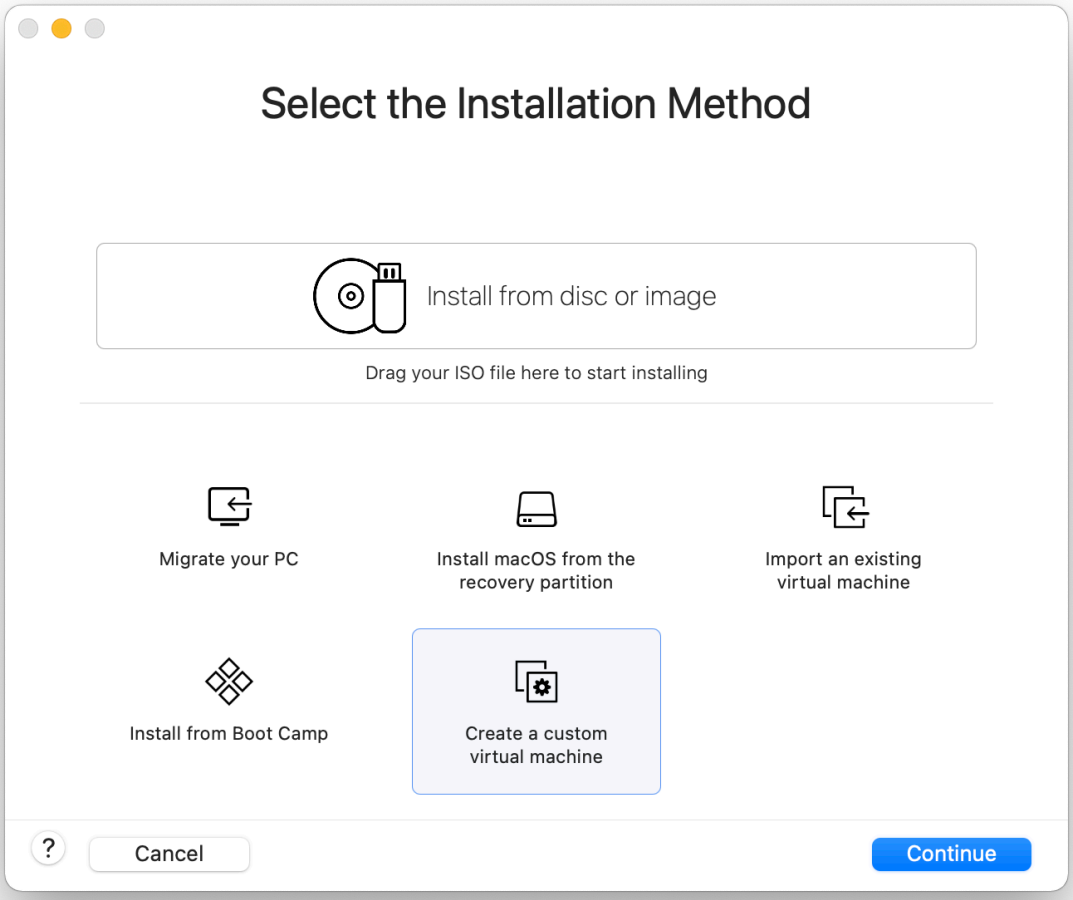
[1487] https://www.freebsd.org/cgi/man.cgi?query=ed&sektion=4&format=html

[1488] https://www.freebsd.org/cgi/man.cgi?query=ed&sektion=4&format=html

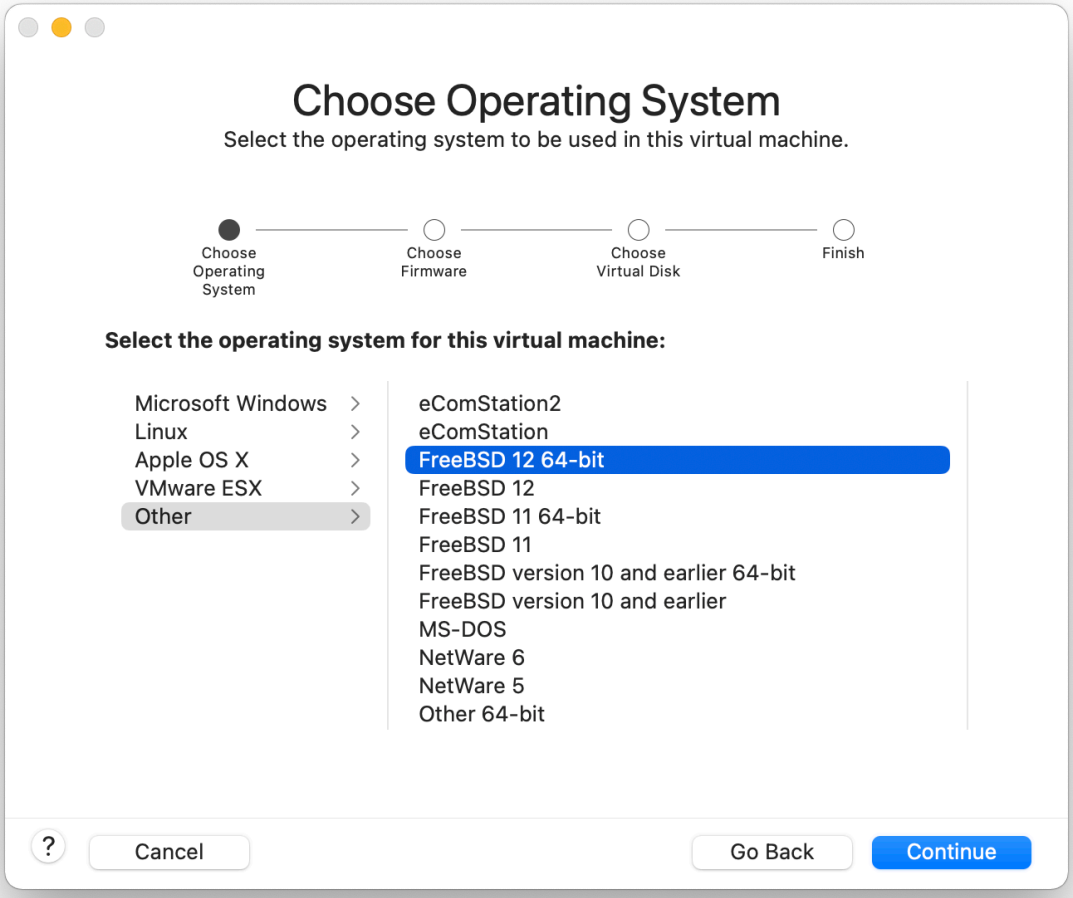
[1489] https://www.freebsd.org/cgi/man.cgi?query=miibus&sektion=4&format=html

[1490] https://docs.freebsd.org/en/books/handbook/advanced-networking/index.html#advanced-networking

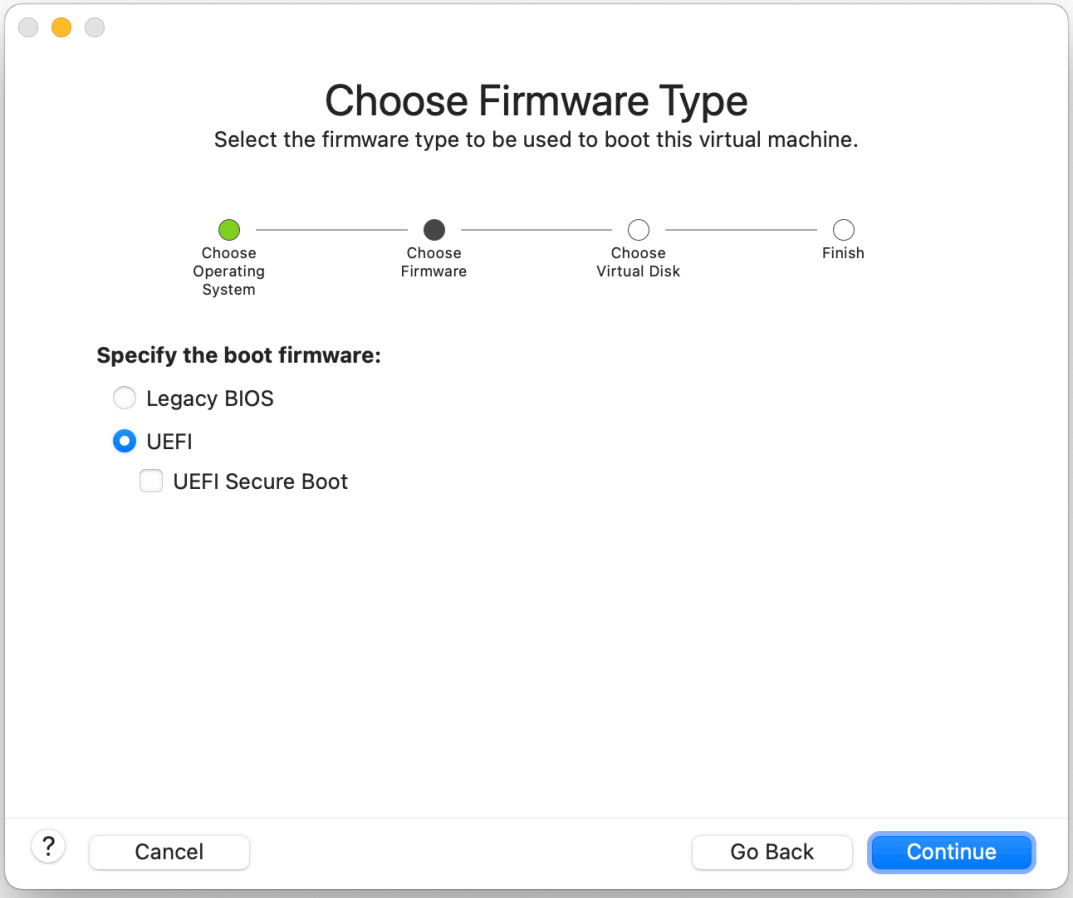
接下来会弹出新建虚拟机助理，选择 Create a custom virtual machine 并点击 Continue：



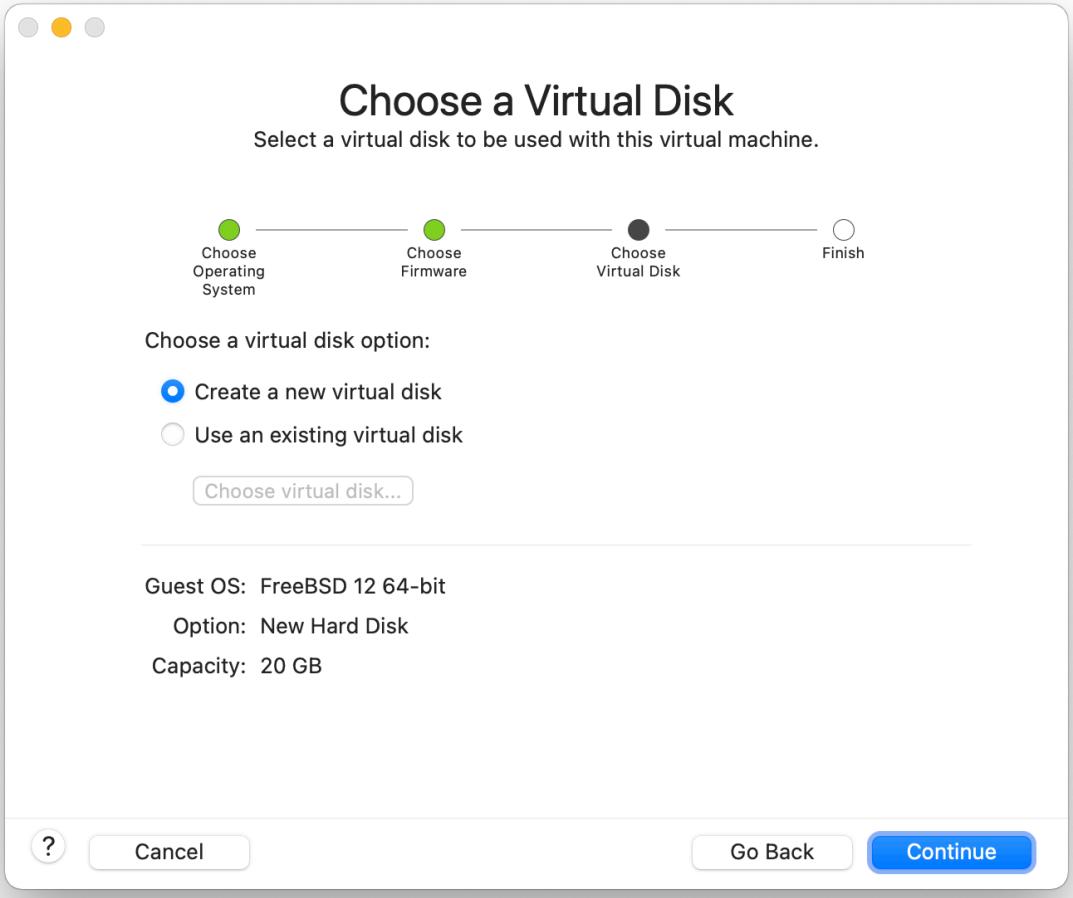
当出现提示时，选择 Other 作为 Operating System，并选择 FreeBSD X 或 FreeBSD X 64-bit，作为 **Version**：



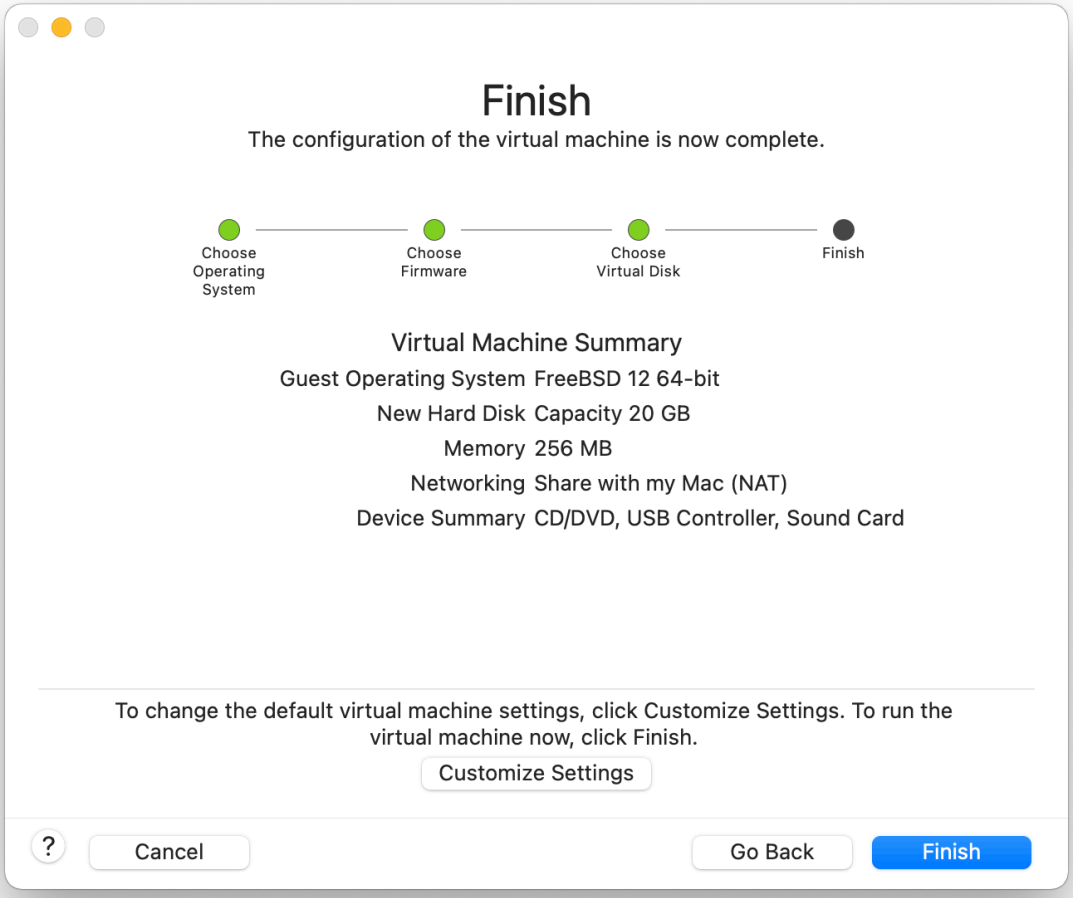
选择指定引导固件（建议选择 UEFI）：



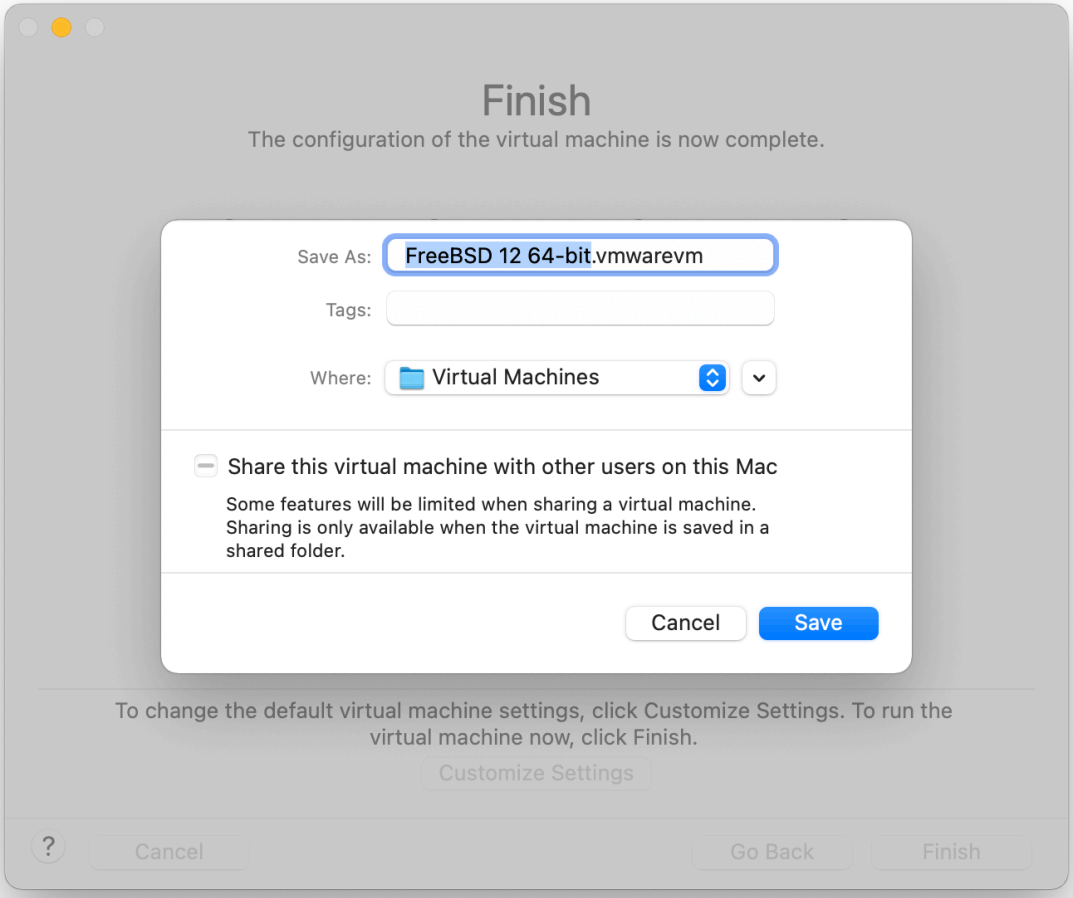
选择 Create a new virtual disk 然后点击 Continue：



检查配置是否正确，然后点击 Finish：

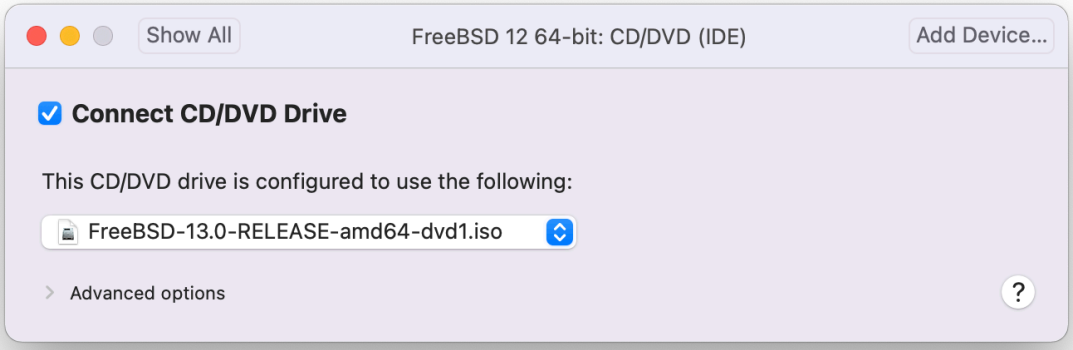


为虚拟机命名并选择保存的位置：



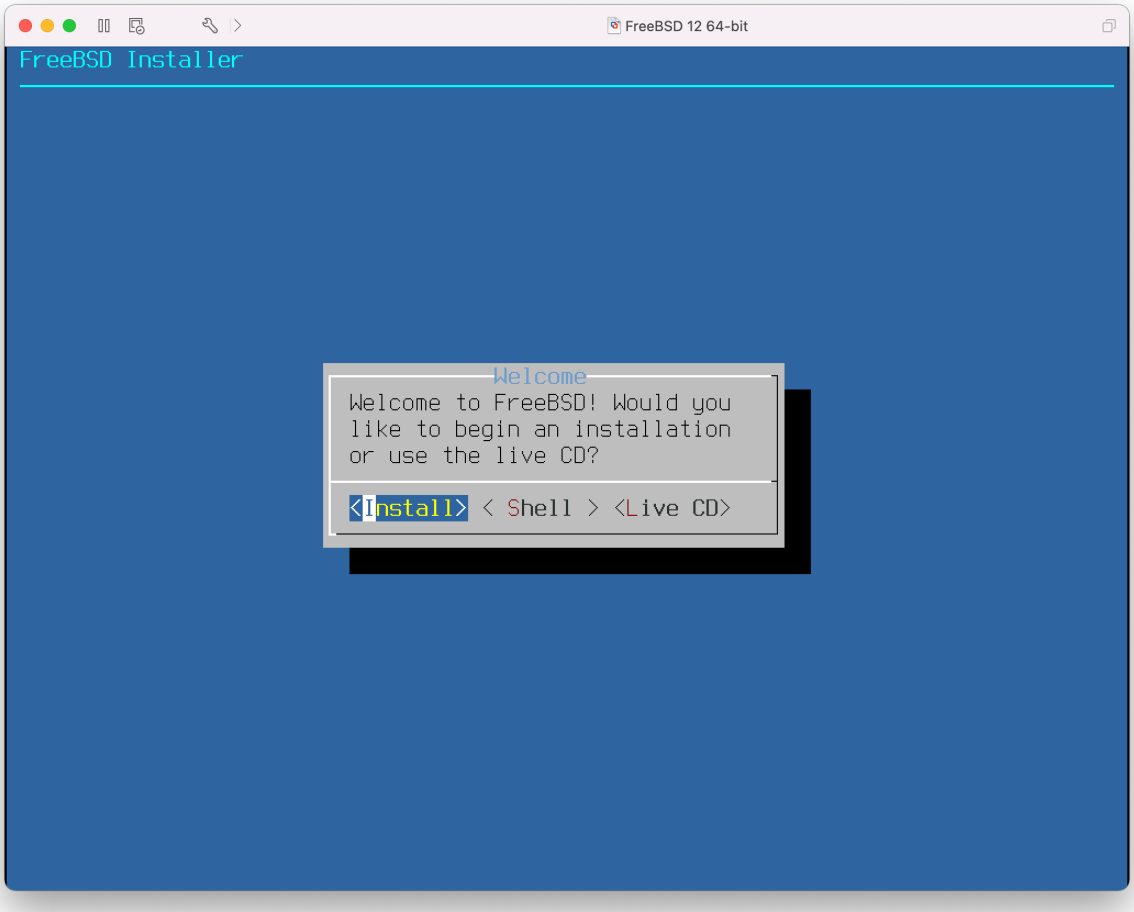
按 command + E 打开虚拟机配置，然后点击 CD/DVD：

选择 FreeBSD 的 ISO 镜像文件，或者关联到实体的 CD/DVD：



启动虚拟机：

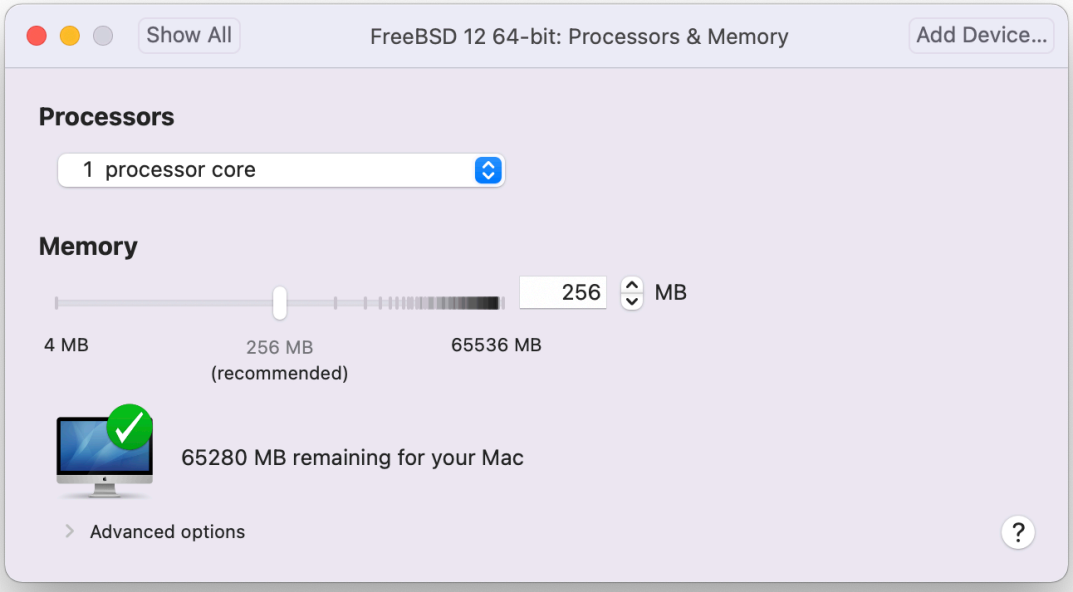
进行正常的 FreeBSD 安装流程：



在安装完成之后，你可以更改虚拟机配置，例如内存大小和分配给虚拟机的 CPU 核心数量：

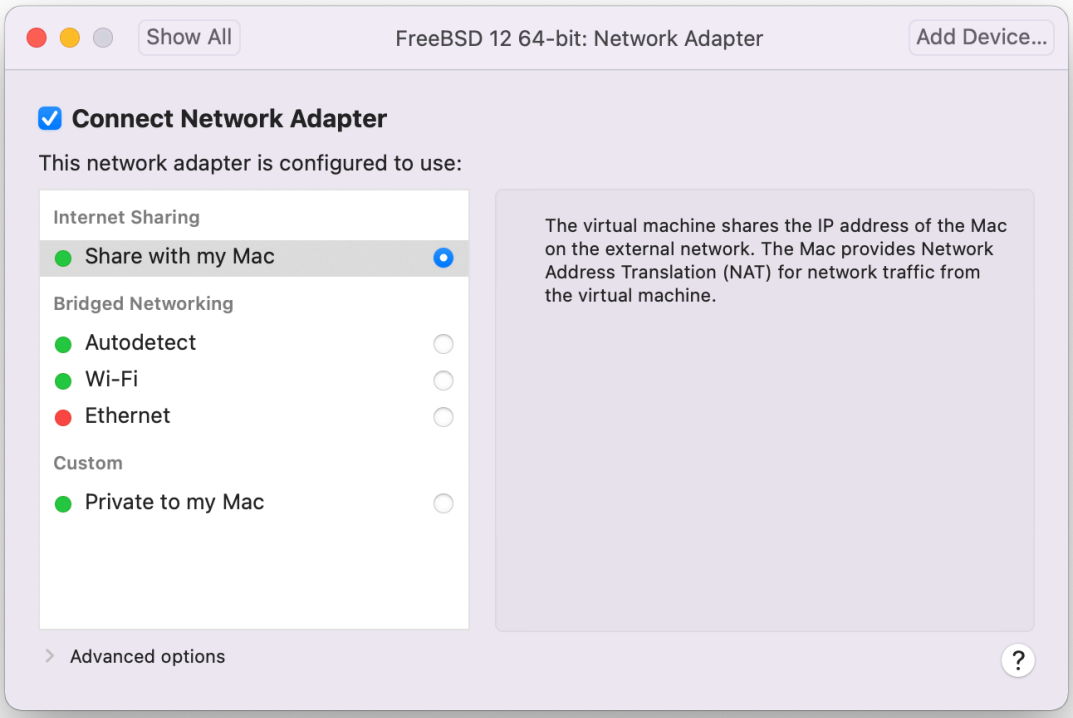
提示

虚拟机的 System Hardware settings 在虚拟机运行期间无法被修改。



在一般情况下，如果不再需要，将 CD/DVD/ISO 从虚拟机断开连接。

最后设置虚拟机连接到网络的方式。若要允许除宿主之外的机器连接到虚拟机，请选择连接到物理网络接口（桥接模式网络连接）。其他情况下，推荐使用与我的 Mac 共享（NAT），该模式允许虚拟机连接到互联网，但外部网络无法直接访问虚拟机。



编辑设置之后，启动进入全新安装的 FreeBSD 虚拟机。

24.3.2.在 VMware Fusion 上配置 FreeBSD

在 macOS® 的 VMware Fusion 上成功安装 FreeBSD 之后，可以进行如下几个配置来对虚拟化操作系统进行优化。

1. 设置引导加载器变量

   最重要的一步是调整可调节的 `kern.hz` 来减少 FreeBSD 在 VMware Fusion 环境下的 CPU 使用率。这可通过在 **/boot/loader.conf** 中加入如下一行来实现：

   ```
   kern.hz=100
   ```

   在没有此项设置的情况下，VMware Fusion 中运行的 FreeBSD 在闲置状态下会占用大约 15% 的 CPU（在一台搭载单颗处理器的 iMac® 上）。进行设置后，占用率可以降低到 5% 左右。

2. 创建新的内核配置文件

   所有 SCSI、火线和 USB 设备驱动都可以从定制内核配置文件中移除。VMware Fusion 提供一个虚拟网

络适配器并由 em(4)[1491] 驱动使用，所以除了 em(4)[1492] 之外的网络驱动都可以从内核中移除。

3. 配置网络

   最基本的网络配置使用 DHCP 来将虚拟机连接到宿主机 Mac® 所在的同一个局域网中。这可以通过向 **/etc/rc.conf** 中添加 `ifconfig_em0="DHCP"` 来实现。更高级的网络设置请参见高级网络[1493]。

4. 安装驱动和 open-vm-tools

   若要在 VMware 上顺畅地运行 FreeBSD，你应该安装驱动：

```
# pkg install xf86-video-vmware xf86-input-vmmouse open-vm-tools
```

24.4.使用 VirtualBox™ 安装 FreeBSD

FreeBSD 作为客户机操作系统在 VirtualBox™ 中表现良好。该虚拟化软件可用于大多数常见的操作系统，包括 FreeBSD。

VirtualBox™ 客户机额外提供如下支持：

- 同步剪切板。
- 整合鼠标指针。
- 同步宿主机时间。
- 窗口缩放。
- 无缝模式。

> 注意
> 请在 FreeBSD 客户机中执行下文命令。

首先，在 FreeBSD 客户机中安装 virtualbox-ose-additions[1494]。可以通过软件包或者 port 安装。以 port 为例：

```
# cd /usr/ports/emulators/virtualbox-ose-additions && make install clean
```

然后在 **/etc/rc.conf** 中添加如下内容：

```
vboxguest_enable="YES"
vboxservice_enable="YES"
```

[1491] https://www.freebsd.org/cgi/man.cgi?query=em&sektion=4&format=html
[1492] https://www.freebsd.org/cgi/man.cgi?query=em&sektion=4&format=html
[1493] https://docs.freebsd.org/en/books/handbook/advanced-networking/index.html#advanced-networking
[1494] https://cgit.freebsd.org/ports/tree/emulators/virtualbox-ose-additions/pkg-descr

如果使用了 ntpd(8)[1495] 或者 ntpdate(8)[1496]，可禁用宿主机时间同步：

```
vboxservice_flags="--disable-timesync"
```

Xorg 会自动识别 vboxvideo 驱动。也可以在 **/etc/X11/xorg.conf** 中手动指定：

```
Section "Device"
    Identifier "Card0"
    Driver "vboxvideo"
    VendorName "InnoTek Systemberatung GmbH"
    BoardName "VirtualBox Graphics Adapter"
EndSection
```

若要使用 vboxmouse 驱动，请在 **/etc/X11/xorg.conf** 中修改鼠标部分配置：

```
Section "InputDevice"
    Identifier "Mouse0"
    Driver "vboxmouse"
EndSection
```

使用 mount_vboxvfs 来挂在用于在宿主和虚拟机之间传输文件的共享文件夹。共享文件夹可以在宿主机上通过 VirtualBox 的图形界面或者 vboxmanage 工具创建。例如，要为虚拟机 *BSDBox* 创建一个名为 *myshare* 的共享文件夹，其位于 **/mnt/bsdboxshare**，请执行：

```
# vboxmanage sharedfolder add 'BSDBox' --name myshare --hostpath /mnt/bsdboxshare
```

请注意共享文件夹名称中不能包含空格。要在客户机中挂载这个共享文件夹，请执行：

```
# mount_vboxvfs -w myshare /mnt
```

24.5.在 FreeBSD 上安装 VirtualBox™

VirtualBox™ 是一个正在活跃开发的，完备的虚拟化软件。可运行于大多操作系统，包括 Windows®，macOS®，Linux® 和 FreeBSD。它也同样能够运行 Windows® 或者类 UNIX® 客户机系统。该软件作为开源软件发行，但是在单独的扩展包中提供了闭源组件。这些组件包括对 USB 2.0 设备的支持。更多的信息可以在 VirtualBox™ 维基下载页[1497]中找到。截至目前为止，这些扩展尚不支持 FreeBSD。

[1495] https://www.freebsd.org/cgi/man.cgi?query=ntpd&sektion=8&format=html

[1496] https://www.freebsd.org/cgi/man.cgi?query=ntpdate&sektion=8&format=html

[1497] http://www.virtualbox.org/wiki/Downloads

24.5.1.安装 VirtualBox™

VirtualBox™ 在 FreeBSD 软件包和 port 名为 emulators/virtualbox-ose[1498]。其 port 可以通过如下命令安装：

```
# cd /usr/ports/emulators/virtualbox-ose
# make install clean
```

port 配置菜单中的一个实用选项是 `GuestAdditions` 程序套件。这些程序提供了在数个客户机操作系统中有用的功能，如鼠标指针整合（允许鼠标在宿主和客户机之间共享，而不需要按下特定的按键组合来切换）以及更快的视频渲染，特别是在 Windows® 客户机当中。在客户机安装完成后，这些附加套件可以在 **设备**菜单中找到。

VirtualBox™ 初次启动时需要修改几个配置。该 port 会在 **/boot/modules** 安装一个内核模块，它需要被加载到运行中的内核：

```
# kldload vboxdrv
```

为了保证该模块始终能在重启后被加载，请在 **/boot/loader.conf** 中添加这样一行：

```
vboxdrv_load="YES"
```

若要使用允许桥接或仅宿主网络的内核模块，请在 **/etc/rc.conf** 中添加这样一行，并重启计算机：

```
vboxnet_enable="YES"
```

在 VirtualBox™ 安装中会新建 `vboxusers` 组。所有需要访问 VirtualBox™ 的用户都需要作为成员被添加到这个组中。你可以使用 `pw` 命令向其中添加成员：

```
# pw groupmod vboxusers -m yourusername
```

/dev/vboxnetctl 的默认权限是受限的，若要使用桥接网络需对其进行修改：

```
# chown root:vboxusers /dev/vboxnetctl
# chmod 0660 /dev/vboxnetctl
```

为了使该权限永久有效，请在 **/etc/devfs.conf** 中添加：

```
own     vboxnetctl root:vboxusers
perm    vboxnetctl 0660
```

要启动 VirtualBox™，请在 Xorg 会话中输入：

[1498] https://cgit.freebsd.org/ports/tree/emulators/virtualbox-ose/pkg-descr

```
% VirtualBox
```

若要获取关于配置和使用 VirtualBox™ 的更多信息，请参阅官方网站[1499]。关于 FreeBSD 特定的信息和疑难解答，请参阅 FreeBSD 维基上的相关页面[1500]。

24.5.2.VirtualBox™ USB 支持

VirtualBox™ 可被配置为 USB 设备直接连接到客户机操作系统。OSE 版本的主控制器仅限模拟 USB 1.1 设备，除非用以支持 USB 2.0 和 3.0 设备的扩展组件开始支持 FreeBSD。

为了使 VirtualBox™ 获知 USB 设备连接到计算机，当前用户需要作为 `operator` 组的成员。

```
# pw groupmod operator -m yourusername
```

然后，将以下内容添加到 **/etc/devfs.rules**，如果文件并不存在请新建该文件：

```
[system=10]
add path 'usb/*' mode 0660 group operator
```

添加以下内容到 **/etc/rc.conf** 来加载新建的规则：

```
devfs_system_ruleset="system"
```

然后重启 `devfs` 使之生效：

```
# service devfs restart
```

重启登录会话和 VirtualBox™ 来使这些更改生效，还可按需要创建 USB 过滤规则。

24.5.3.访问 VirtualBox™ 宿主的 DVD/CD

从客户机访问宿主的 DVD/CD 驱动器是通过共享物理光驱来实现的。在 VirtualBox™ 中，从虚拟机设置中的存储视窗进行设置。如果需要，请先创建一个空白的 IDECD/DVD 设备，然后在虚拟 CD/DVD 驱动器的弹出菜单中选择宿主光驱。这里会出现穿透模式复选框，这个模式可以让客户机直接访问宿主光驱硬件。例如，音频 CD 和刻录软件只有在启用该模式时能够正常工作。

为了使用户能够使用 VirtualBox™ DVD/CD 功能，用户需要有访问 **/dev/xpt0**，**/dev/cdN** 和 **/dev/passN** 的权限。通常来说，将用户加入 `operator` 组即可。请向 **/etc/devfs.conf** 添加以下内容来纠正这些设备的权限：

[1499] http://www.virtualbox.org/

[1500] http://wiki.freebsd.org/VirtualBox

```
perm cd* 0660
perm xpt0 0660
perm pass* 0660
```

然后重启 devfs 使之生效：

```
# service devfs restart
```

24.6.使用 FreeBSD 上的 bhyve 虚拟机

BSD 许可的 bhyve 虚拟机管理器随着 FreeBSD 10.0-RELEASE 开始成为底层系统的一部分。这个虚拟机管理器支持数种客户机，包括 FreeBSD，OpenBSD 和数个 Linux® 发行版。默认情况下，bhyve 提供对串行控制台的访问而不模拟图形化控制台。其利用较新的 CPU 上的虚拟化 offload 功能来避免使用过时方法翻译指令和手动管理内存映射。

bhyve 在设计上需要支持 Intel® 扩展页表（EPT）或 AMD® 快速虚拟化索引（RVI）或 Nested 页表（NPT）的 CPU。以超过一枚 vCPU 运行 Linux 或者 FreeBSD 客户机需要 VMX unrestricted 模式（UG）支持。大多数新的处理器，特别是 Intel® Core™ i3/i5/i7 和 Intel® Xeon™ E3/E5/E7 都支持这些功能。对 UG 的支持是从 Intel® 的 Westmere 微架构引入的。若要查阅一份完整的支持 EPT 的 Intel® 处理器列表，请参见 https://ark.intel.com/content/www/us/en/ark/search/featurefilter.html?productType=873&0_ExtendedPageTables=True。RVI 在第三代以及之后的 AMD Opteron™（Barcelona）处理器中应用。判断处理器是否支持 bhyve 的方法是运行 demsg，或者在 **/var/run/dmesg.boot** 中的 Feature2 一行查找 POPCNT 处理器功能标识（针对 AMD® 处理器）；或者在 VT-x 一行查找 EPT 和 UG 处理器功能标识（针对 Intel® 处理器）。

24.6.1.准备宿主机

在 bhyve 中创建虚拟机的第一步是配置宿主操作系统。首先，加载 bhyve 内核模块：

```
# kldload vmm
```

然后，创建 tap 接口以供虚拟机中的网络设备进行连接。为了让该网络设备加入网络，另外创建一个桥接接口，其中包含 tap 接口和物理网络接口作为成员。在下面的例子中，物理网络接口为 igb0：

```
# ifconfig tap0 create
# sysctl net.link.tap.up_on_open=1
net.link.tap.up_on_open: 0 -> 1
# ifconfig bridge0 create
# ifconfig bridge0 addm igb0 addm tap0
# ifconfig bridge0 up
```

24.6.2.创建一个 FreeBSD 客户机

创建一个文件用作客户机的虚拟磁盘。指定这个虚拟磁盘的大小和名称：

```
# truncate -s 16G guest.img
```

下载 FreeBSD 的安装镜像：

```
# fetch https://download.freebsd.org/releases/ISO-IMAGES/13.1/FreeBSD-13.1-RELEASE-
↪amd64-bootonly.iso
FreeBSD-13.1-RELEASE-amd64-bootonly.iso                366 MB   16 MBps    22s
```

FreeBSD 附带了一个在 bhyve 中运行虚拟机的示例脚本。这个脚本将会启动虚拟机并循环执行，所以可在其崩溃时自动重新启动。这个脚本可接受数个选项来配置虚拟机：`-c` 控制虚拟 CPU 的数量，`-m` 限制客户机的可用内存，`-t` 定义使用的 **tap** 设备，`-d` 指定其使用的磁盘镜像，`-i` 控制 bhyve 从 CD 镜像引导，而非从磁盘引导，`-I` 指定要使用的 CD 镜像。最后一个参数是虚拟机的名称，该名称将被用来追踪运行中的虚拟机。这个例子将在安装模式下启动虚拟机：

```
# sh /usr/share/examples/bhyve/vmrun.sh -c 1 -m 1024M -t tap0 -d guest.img -i -I␣
↪FreeBSD-13.1-RELEASE-amd64-bootonly.iso guestname
```

这个虚拟机将会引导并启动安装器。在虚拟机中完成系统安装并进入最后阶段，当系统提示是否进入 shell 时，选择 **Yes**。

重新启动虚拟机。重启会导致 bhyve 退出，这时循环执行 `bhyve` 的 **vmrun.sh** 脚本会自动重启虚拟机。这时请从引导器菜单中选择重启来跳出循环。之后，客户机就可以从虚拟磁盘中启动了：

```
# sh /usr/share/examples/bhyve/vmrun.sh -c 4 -m 1024M -t tap0 -d guest.img guestname
```

24.6.3.创建一个 Linux® 客户机

要引导进入 FreeBSD 以外的系统，必须先安装 port sysutils/grub2-bhyve[1501] 。

接下来，创建一个文件用作客户机的虚拟磁盘：

```
# truncate -s 16G linux.img
```

使用 bhyve 启动虚拟机分为两个步骤。首先要加载一个内核，然后启动客户机。通过 sysutils/grub2-bhyve[1502] 加载 Linux® 内核。创建 **device.map** 令 grub 将虚拟设备映射到宿主机上的镜像文件：

[1501] https://cgit.freebsd.org/ports/tree/sysutils/grub2-bhyve/pkg-descr

[1502] https://cgit.freebsd.org/ports/tree/sysutils/grub2-bhyve/pkg-descr

```
(hd0) ./linux.img
(cd0) ./somelinux.iso
```

使用 sysutils/grub2-bhyve[1503] 从 ISO 文件中加载 Linux® 内核：

```
# grub-bhyve -m device.map -r cd0 -M 1024M linuxguest
```

这将启动 grub。如果安装 CD 中包含 **grub.cfg**，将会显示一个菜单；否则你需要手动指明并加载 `vmlinuz` 和 `initrd` 文件：

```
grub> ls
(hd0) (cd0) (cd0,msdos1) (host)
grub> ls (cd0)/isolinux
boot.cat boot.msg grub.conf initrd.img isolinux.bin isolinux.cfg memtest
splash.jpg TRANS.TBL vesamenu.c32 vmlinuz
grub> linux (cd0)/isolinux/vmlinuz
grub> initrd (cd0)/isolinux/initrd.img
grub> boot
```

现在 Linux® 内核就加载完毕了，可以启动客户机：

```
# bhyve -A -H -P -s 0:0,hostbridge -s 1:0,lpc -s 2:0,virtio-net,tap0 -s 3:0,virtio-
↪blk,./linux.img \
    -s 4:0,ahci-cd,./somelinux.iso -l com1,stdio -c 4 -m 1024M linuxguest
```

系统将会引导进入并启动安装器。在虚拟机中完成系统安装之后，重新启动虚拟机。bhyve 会在这时退出。请先销毁这个虚拟机实例，然后重新启动：

```
# bhyvectl --destroy --vm=linuxguest
```

现在客户机可以直接从磁盘镜像中启动了。首先加载内核：

```
# grub-bhyve -m device.map -r hd0,msdos1 -M 1024M linuxguest
grub> ls
(hd0) (hd0,msdos2) (hd0,msdos1) (cd0) (cd0,msdos1) (host)
(lvm/VolGroup-lv_swap) (lvm/VolGroup-lv_root)
grub> ls (hd0,msdos1)/
lost+found/ grub/ efi/ System.map-2.6.32-431.el6.x86_64 config-2.6.32-431.el6.x
86_64 symvers-2.6.32-431.el6.x86_64.gz vmlinuz-2.6.32-431.el6.x86_64
initramfs-2.6.32-431.el6.x86_64.img
grub> linux (hd0,msdos1)/vmlinuz-2.6.32-431.el6.x86_64 root=/dev/mapper/VolGroup-lv_
```

(continues on next page)

[1503] https://cgit.freebsd.org/ports/tree/sysutils/grub2-bhyve/pkg-descr

(continued from previous page)

```
↪root
grub> initrd (hd0,msdos1)/initramfs-2.6.32-431.el6.x86_64.img
grub> boot
```

引导进入虚拟机：

```
# bhyve -A -H -P -s 0:0,hostbridge -s 1:0,lpc -s 2:0,virtio-net,tap0 \
    -s 3:0,virtio-blk,./linux.img -l com1,stdio -c 4 -m 1024M linuxguest
```

虚拟机中的 Linux® 将会启动并提示你登录。当你不再使用虚拟机时，重新启动虚拟机来退出 bhyve，并销毁虚拟机实例：

```
# bhyvectl --destroy --vm=linuxguest
```

24.6.4.使用 UEFI 固件引导 bhyve 虚拟机

除了 bhyveload 和 grub-bhyve，bhyve 虚拟机管理器还可使用 UEFI 用户空间固件来引导虚拟机。这可以兼容其他引导器不支持的客户机操作系统。

要利用 bhyve 的 UEFI 支持，首先要生成 UEFI 固件镜像。你可以通过 port 或软件包安装 sysutils/bhyve-firmware[1504] 来实现。

准备好固件之后，向 bhyve 命令添加标识 `-l bootrom,/path/to/firmware`，例如：

```
# bhyve -AHP -s 0:0,hostbridge -s 1:0,lpc \
-s 2:0,virtio-net,tap1 -s 3:0,virtio-blk,./disk.img \
-s 4:0,ahci-cd,./install.iso -c 4 -m 1024M \
-l bootrom,/usr/local/share/uefi-firmware/BHYVE_UEFI.fd \
guest
```

sysutils/bhyve-firmware[1505] 还包含了启用 CSM 的固件，若要在 legacy BIOS 模式中启动不支持 UEFI 的客户机，请这样运行：

```
# bhyve -AHP -s 0:0,hostbridge -s 1:0,lpc \
-s 2:0,virtio-net,tap1 -s 3:0,virtio-blk,./disk.img \
-s 4:0,ahci-cd,./install.iso -c 4 -m 1024M \
-l bootrom,/usr/local/share/uefi-firmware/BHYVE_UEFI_CSM.fd \
guest
```

[1504] https://cgit.freebsd.org/ports/tree/sysutils/bhyve-firmware/pkg-descr
[1505] https://cgit.freebsd.org/ports/tree/sysutils/bhyve-firmware/pkg-descr

24.6.5.为客户机设置图形化 UEFI 帧缓冲

UEFI 固件支持对主要使用图形化界面的客户机操作系统格外有用，例如 Microsoft Windows®。

对 UEFI-GOP framebuffer 的支持也可以用参数 `-s 29,fbuf,tcp=0.0.0.0:5900` 来启用。帧缓冲的分辨率可以通过如 `w=800 h=600` 来设置。bhyve 也可以通过添加 `wait` 来等待 VNC 连接，之后再引导进入客户机。framebuffer 可以从宿主机或者 VNC 协议的网络连接访问。此外，可以添加 `-s 30,xhci,tablet` 来实现与宿主机之间精确的鼠标同步。

最后的 bhyve 命令可能类似这样：

```
# bhyve -AHP -s 0:0,hostbridge -s 31:0,lpc \
-s 2:0,virtio-net,tap1 -s 3:0,virtio-blk,./disk.img \
-s 4:0,ahci-cd,./install.iso -c 4 -m 1024M \
-s 29,fbuf,tcp=0.0.0.0:5900,w=800,h=600,wait \
-s 30,xhci,tablet \
-l bootrom,/usr/local/share/uefi-firmware/BHYVE_UEFI.fd \
guest
```

注意，在模拟 BIOS 模式下，当控制从固件移交到宿主机操作系统后，帧缓冲会停止接收更新。

24.6.6.在 bhyve 客户机中使用 ZFS

如果在宿主机上可以使用 ZFS，在客户机上使用 ZFS 卷取代磁盘镜像可以显著提高性能。你可以这样新建一个 ZFS 卷：

```
# zfs create -V16G -o volmode=dev zroot/linuxdisk0
```

启动虚拟机时，指定 ZFS 卷作为磁盘驱动器：

```
# bhyve -A -H -P -s 0:0,hostbridge -s 1:0,lpc -s 2:0,virtio-net,tap0 -s3:0,virtio-blk,
↪/dev/zvol/zroot/linuxdisk0 \
    -l com1,stdio -c 4 -m 1024M linuxguest
```

24.6.7.虚拟机控制台

你可以使用类似于 sysutils/tmux[1506] 或者 sysutils/screen[1507] 这样的管理工具将 bhyve 控制台封装在一个会话中用来连接和断开控制台，这将带来一些好处。你还可以将一个空的调制解调器作为 bhyve 的控制台，然后使用 `cu` 来访问。若要这样做，请加载 **nmdm** 内核模块，然后将 `-l com1,stdio` 替换为 `-l com1,/`

[1506] https://cgit.freebsd.org/ports/tree/sysutils/tmux/pkg-descr
[1507] https://cgit.freebsd.org/ports/tree/sysutils/screen/pkg-descr

dev/nmdm0A。设备 **/dev/nmdm** 会自动按需创建并组成一对，对应着空的调制解调器的两端（**/dev/nmdm0A** 和 **/dev/nmdm0B**）。查阅 nmdm(4)[1508] 获取更多信息。

```
# kldload nmdm
# bhyve -A -H -P -s 0:0,hostbridge -s 1:0,lpc -s 2:0,virtio-net,tap0 -s 3:0,virtio-
↪blk,./linux.img \
    -l com1,/dev/nmdm0A -c 4 -m 1024M linuxguest
# cu -l /dev/nmdm0B
Connected

Ubuntu 13.10 handbook ttyS0

handbook login:
```

24.6.8.管理虚拟机

对于每个虚拟机，在 **/dev/vmm** 下都会创建一个设备节点。这使得管理员很容易就能看到运行中的虚拟机列表：

```
# ls -al /dev/vmm
total 1
dr-xr-xr-x   2 root  wheel    512 Mar 17 12:19 ./
dr-xr-xr-x  14 root  wheel    512 Mar 17 06:38 ../
crw-------   1 root  wheel  0x1a2 Mar 17 12:20 guestname
crw-------   1 root  wheel  0x19f Mar 17 12:19 linuxguest
crw-------   1 root  wheel  0x1a1 Mar 17 12:19 otherguest
```

通过 bhyvectl 可以销毁指定的虚拟机：

```
# bhyvectl --destroy --vm=guestname
```

24.6.9.持久化配置

要配置系统在启动时运行 bhyve 虚拟机，请进行如下配置：

1. **/etc/sysctl.conf**

   ```
   net.link.tap.up_on_open=1
   ```

2. **/etc/rc.conf**

[1508] https://www.freebsd.org/cgi/man.cgi?query=nmdm&sektion=4&format=html

```
cloned_interfaces="bridge0 tap0"
ifconfig_bridge0="addm igb0 addm tap0"
kld_list="nmdm vmm"
```

24.7.使用 FreeBSD 上的 Xen™ 虚拟机

Xen 是一个 GPLv2 许可的第 1 类虚拟机管理器[1509]，适用于 Intel® 和 ARM® 架构。FreeBSD 从 FreeBSD 8.0 开始就包含了对 i386™，AMD® 64-Bit DomU[1510] 和 Amazon EC2[1511] 非特权域（虚拟机）的支持，并在 FreeBSD 11.0 中包含了对 Dom0 控制域（宿主机）的支持。在 FreeBSD 11 中，取消了对准虚拟化（PV）域的支持，转而支持硬件虚拟化（HVM）域，这提供了更好的性能。

Xen™ 是一个裸虚拟机管理器，这意味着它是在 BIOS 之后加载的第一个程序。然后会启动一个称为 Domain-0（简称 `Dom0`）的特殊权限客户机。Dom0 利用其特殊权限直接访问底层物理硬件，使其成为一个高性能的解决方案。它能够直接访问磁盘控制器和网络适配器。用于管理和控制 Xen™ 虚拟机管理器的 Xen™ 管理工具也被 Dom0 用来创建、列出和销毁虚拟机。Dom0 为非特权域提供虚拟磁盘和网络，通常称为 `DomU`。Xen™ Dom0 可以比作其他虚拟机管理器解决方案的服务控制台，而 DomU 是运行单个客户虚拟机的地方。

Xen™ 可以在不同的 Xen™ 服务器之间迁移虚拟机。当两台 xen 主机共享相同的底层存储时，迁移过程可以不必先关闭虚拟机。取而代之的是，迁移是在 DomU 运行过程中实时进行的，不需要重新启动或计划停机。这在维护场景或升级窗口中非常有用，可以确保 DomU 提供的服务持续在线。Xen™ 的许多其他功能都罗列在 Xen Wiki 概述页面[1512]上。请注意，FreeBSD 还不支持其全部功能。

24.7.1.Xen™ Dom0 的硬件要求

要在宿主机上运行 Xen™ hypervisor，需要特定的硬件功能。硬件虚拟化域需要宿主机的处理器支持扩展页表（EPT[1513]）和输入输出内存管理单元（IOMMU[1514]）。

> 注意
>
> 要运行 FreeBSD Xen™ Dom0，这个容器必须使用 legacy boot（BIOS）引导。

[1509] https://en.wikipedia.org/wiki/Hypervisor#Classification

[1510] https://wiki.xenproject.org/wiki/DomU

[1511] https://en.wikipedia.org/wiki/Amazon_Elastic_Compute_Cloud

[1512] https://wiki.xenproject.org/wiki/Category:Overview

[1513] http://en.wikipedia.org/wiki/Extended_Page_Table

[1514] http://en.wikipedia.org/wiki/List_of_IOMMU-supporting_hardware

24.7.2.设置 Xen™ Dom0 控制域

FreeBSD 11 用户应该安装基于 Xen 4.7 的软件包 emulators/xen-kernel47[1515] 和 sysutils/xen-tools47[1516]。FreeBSD-12.0 及更新版本的用户可以使用基于 Xen 4.11 的 emulators/xen-kernel411[1517] 和 sysutils/xen-tools411[1518]。

在安装 Xen 软件包后，必须编辑配置文件来为宿主机整合 Dom0 做准备。在 **/etc/sysctl.conf** 中加入一个条目，禁用对 wired 内存页数的限制。否则，对内存要求较高的 DomU 虚拟机将无法运行：

```
# echo 'vm.max_wired=-1' >> /etc/sysctl.conf
```

另一个内存相关的设置需要修改 **/etc/login.conf**，将 `memorylocked` 选项设置为 `unlimited`。否则，创建 DomU 时可能遇到错误 `Cannot allocate memory` 而失败。在修改 **/etc/login.conf** 后，运行 `cap_mkdb` 来更新功能数据库。参阅资源限制[1519]获得更多信息。

```
# sed -i '' -e 's/memorylocked=64K/memorylocked=unlimited/' /etc/login.conf
# cap_mkdb /etc/login.conf
```

为 Xen™ 控制台在 **/etc/ttys** 添加一个条目：

```
# echo 'xc0     "/usr/libexec/getty Pc"         xterm   onifconsole  secure' >> /etc/
↪ttys
```

在 **/boot/loader.conf** 中选择 Xen™ 内核就会启用 Dom0。Xen™ 还需要宿主机上的 CPU 和内存等资源以供自身和其他 DomU 域使用。分配多少 CPU 和内存取决于个人需求和硬件性能。在这个例子中，我们为 Dom0 分配了 8 GB 的内存和 4 颗虚拟的 CPU，同时启用串行控制台，并定义了日志设置。

用于 Xen 4.7 的命令为：

```
# echo 'hw.pci.mcfg=0' >> /boot/loader.conf
# echo 'if_tap_load="YES"' >> /boot/loader.conf
# echo 'xen_kernel="/boot/xen"' >> /boot/loader.conf
# echo 'xen_cmdline="dom0_mem=8192M dom0_max_vcpus=4 dom0pvh=1 console=com1,vga␣
↪com1=115200,8n1 guest_loglvl=all loglvl=all"' >> /boot/loader.conf
```

而在 Xen 4.11 或更高版本中，应使用：

```
# echo 'if_tap_load="YES"' >> /boot/loader.conf
# echo 'xen_kernel="/boot/xen"' >> /boot/loader.conf
```

(continues on next page)

[1515] https://cgit.freebsd.org/ports/tree/emulators/xen-kernel47/pkg-descr
[1516] https://cgit.freebsd.org/ports/tree/sysutils/xen-tools47/pkg-descr
[1517] https://cgit.freebsd.org/ports/tree/emulators/xen-kernel411/pkg-descr
[1518] https://cgit.freebsd.org/ports/tree/sysutils/xen-tools411/pkg-descr
[1519] https://docs.freebsd.org/en/books/handbook/security/index.html#security-resourcelimits

(continued from previous page)

```
# echo 'xen_cmdline="dom0_mem=8192M dom0_max_vcpus=4 dom0=pvh console=com1,vga␣
↪com1=115200,8n1 guest_loglvl=all loglvl=all"' >> /boot/loader.conf
```

技巧

Xen™ 为 DomU 虚拟机创建的日志存储在 **/var/log/xen**。如果遇到问题，请记得查看该目录中的内容。

在系统启动时启用 `xencommons` 服务：

```
# sysrc xencommons_enable=yes
```

进行这些设置便可以启动 Dom-0 系统了，然而我们还没有为 DomU 虚拟机提供网络功能。为了修复这一点，请在系统的主网络接口控制器定义一个桥接网络接口，以供 DomU 虚拟机连接到网络。请用你的主网络接口名称替换 *em0*：

```
# sysrc cloned_interfaces="bridge0"
# sysrc ifconfig_bridge0="addm em0 SYNCDHCP"
# sysrc ifconfig_em0="up"
```

重启宿主机以加载 Xen™ 内核并启动 Dom0：

```
# reboot
```

成功引导 Xen™ 内核并再次登录进入系统后，使用 Xen™ 管理工具 `xl` 显示域的信息：

```
# xl list
Name                                        ID   Mem VCPUs      State   Time(s)
Domain-0                                     0  8192     4     r-----     962.0
```

该输出结果确认了 Dom0（显示为 `Domain-0`）具有 ID `0` 并且正在运行。它也拥有先前在 **/boot/loader.conf** 中定义的内存和虚拟 CPU。可以在 Xen™ 文档[1520] 中找到更多信息。现在就可以创建 DomU 客户虚拟机了。

24.7.3.Xen™ DomU 客户虚拟机配置

非特权域由一个配置文件和虚拟或物理磁盘组成。如创建和销毁卷[1521]所述，DomU 的虚拟磁盘可以是由 truncate(1)[1522] 创建的文件或者 ZFS 卷。在这个例子中，我们使用一个 20 GB 的卷。该虚拟机由这个 ZFS 卷、FreeBSD ISO 镜像、1GB 内存和两颗虚拟的 CPU 组成。使用 fetch(1)[1523] 获取 ISO 安装文件，文件在本地储存为 *freebsd.iso*：

[1520] https://www.xenproject.org/help/documentation.html
[1521] https://docs.freebsd.org/en/books/handbook/zfs/index.html#zfs-zfs-volume
[1522] https://www.freebsd.org/cgi/man.cgi?query=truncate&sektion=1&format=html
[1523] https://www.freebsd.org/cgi/man.cgi?query=fetch&sektion=1&format=html

```
# fetch https://download.freebsd.org/releases/ISO-IMAGES/13.1/FreeBSD-13.1-RELEASE-
↪amd64-bootonly.iso -o freebsd.iso
```

创建一个 20 GB，名为 **xendisk0** 的 ZFS 卷用作虚拟机的磁盘空间。

```
# zfs create -V20G -o volmode=dev zroot/xendisk0
```

这个 DomU 客户虚拟机定义在一个文件中。诸如主机名、键盘布局和 VNC 连接的细节定义也在其中。如下的例子中，**freebsd.cfg** 包含了一个最简单化的 DomU 配置：

```
# cat freebsd.cfg
builder = "hvm"   ①
name = "freebsd"    ②
memory = 1024   ③
vcpus = 2   ④
vif = [ 'mac=00:16:3E:74:34:32,bridge=bridge0' ]    ⑤
disk = [
'/dev/zvol/tank/xendisk0,raw,hda,rw',   ⑥
'/root/freebsd.iso,raw,hdc:cdrom,r'   ⑦
  ]
vnc = 1   ⑧
vnclisten = "0.0.0.0"
serial = "pty"
usbdevice = "tablet"
```

这些命令的详细解释：

① 定义使用的虚拟化类型。`hvm` 表示硬件辅助的虚拟化或者硬件虚拟机。客户机操作系统可以通过虚拟化插件运行在未经修改的 CPU 上，提供近乎在物理硬件上运行的性能。而 `generic` 是默认值，将会创建一个 PV 域。

② 指定虚拟机名称，以便和其他同样运行在 Dom0 的虚拟机区分开来。这是必需项。

③ 分配给虚拟机的内存容量（用 MB 表示）。这个容量是从虚拟机管理器的总可用内存中分配的，而不是 Dom0 的内存。

④ 分配给虚拟机的虚拟 CPU 数量。为了实现最佳性能，请不要为创建的虚拟机分配多于宿主机物理 CPU 数量的虚拟 CPU。

⑤ 设定虚拟网络适配器。这个网桥将网络适配器连接到宿主机上的网络接口。`mac` 参数是为虚拟网络接口设定的 MAC 地址，为可选项；如果没有提供，Xen™ 将会随机生成一个。

⑥ 用作虚拟机磁盘空间的磁盘、文件或 ZFS 卷的完整路径。选项和多个磁盘定义之间用逗号隔开。

⑦ 定义初始化系统安装的启动介质。在这个例子中为先前下载的 ISO 镜像。关于其他类型的设备和选项设置请参阅 Xen™ 文档。

⑧ 设定 DomU 控制台连接 VNC。按照顺序为：激活 VNC 支持，定义监听的 IP 地址，串行控制台的设备节点，精确鼠标指针定位和其他输入设备的输入方式。`keymap` 定义使用何种键位，默认为 `english`。

设定所有这些必要参数并创建文件之后，将它作为参数输入 `xl create` 来创建 DomU：

```
# xl create freebsd.cfg
```

> 提示
>
> 每次 Dom0 重启后，必须将这个配置文件再次传递到 `xl create` 来重新创建 DomU。默认状态下，重启后只会创建 Dom0 而不会创建虚拟机。这些虚拟机可以从在虚拟磁盘中存储的操作系统中恢复之前的状态。你可以修改虚拟机配置文件（比如增加内存）。虚拟机配置文件应该得到妥当备份，并在需要重新创建虚拟机时保证可用。

`xl list` 的输出结果确认 DomU 已被创建：

```
# xl list
Name                                        ID   Mem VCPUs      State   Time(s)
Domain-0                                     0  8192     4     r-----  1653.4
freebsd                                      1  1024     1     -b----   663.9
```

开始安装基本操作系统，启动 VNC 客户端，将其指向主机的主网络地址或 **freebsd.cfg** 中 `vnclisten` 一行定义的 IP 地址。操作系统安装完毕后，关闭 DomU 并断开与 VNC 查看器的连接。编辑 **freebsd.cfg**，删除带有 cdrom 定义的一行，或者在该行的开头插入一个 `#` 字符，将其注释出来。要加载这个新的配置，必须将其名称或 id 作为参数，用 `xl destroy` 删除旧的 DomU。然后用修改过的 **freebsd.cfg** 重新创建它：

```
# xl destroy freebsd
# xl create freebsd.cfg
```

之后，这个虚拟机可以通过 VNC 查看器再次访问。这次，它会从安装操作系统的虚拟磁盘中引导，并可以作为虚拟机使用了。

24.7.4.疑难解答

这一节包含一些基础信息，帮助你解决使用 FreeBSD 作为 Xen™ 的宿主机或客户机时遇到的问题。

24.7.4.1.宿主机引导疑难解答

请注意，接下来的疑难解答适用于 Xen™ 4.11 或更新版本。如果你还在使用 Xen™ 4.7 并遇到问题，请考虑迁移到更新版本的 Xen™。

为了排查宿主机启动问题，你可能需要一根串口线或 USB 调试线。通过在 **loader.conf** 中添加 `xen_cmdline` 选项，可以获得详细的 Xen™ 启动输出。几个相关的调试选项是：

- `iommu=debug`：可用于输出关于 IOMMU 的额外诊断信息。

- `dom0=verbose`：可用于输出关于 Dom0 构建过程的额外诊断信息。
- `sync_console`：强制同步控制台输出。有助于在调试过程中避免由于更新频率限制导致丢失消息。请不要在生产环境中使用这个选项，因为它可能使恶意客户机使用控制台对 Xen™ 进行 DoS 攻击。

为了识别任何可能存在的问题，FreeBSD 也应该在详细输出模式中启动。要激活详细的引导输出，请运行：

```
# echo 'boot_verbose="YES"' >> /boot/loader.conf
```

如果上述的选项都没有帮助你解决问题，请将串口启动日志发送到 freebsd-xen@FreeBSD.org 和 xen-devel@lists.xenproject.org 以供进一步分析。

24.7.4.2.客户机创建疑难解答

在创建客户机时也可能出现问题，下面将尝试为那些试图诊断客户机创建问题的人提供一些帮助。

导致客户机创建失败的最常见原因是 `xl` 命令提示的一些错误，并以非 0 的返回结果退出。如果提示的错误不足以帮助你识别问题，也可以通过叠加使用 `v` 选项来从 `xl` 获得更详细的输出。

```
# xl -vvv create freebsd.cfg
Parsing config from freebsd.cfg
libxl: debug: libxl_create.c:1693:do_domain_create: Domain 0:ao 0x800d750a0: create:␣
↪how=0x0 callback=0x0 poller=0x800d6f0f0
libxl: debug: libxl_device.c:397:libxl__device_disk_set_backend: Disk vdev=xvda spec.
↪backend=unknown
libxl: debug: libxl_device.c:432:libxl__device_disk_set_backend: Disk vdev=xvda,␣
↪using backend phy
libxl: debug: libxl_create.c:1018:initiate_domain_create: Domain 1:running bootloader
libxl: debug: libxl_bootloader.c:328:libxl__bootloader_run: Domain 1:not a PV/PVH␣
↪domain, skipping bootloader
libxl: debug: libxl_event.c:689:libxl__ev_xswatch_deregister: watch w=0x800d96b98:␣
↪deregister unregistered
domainbuilder: detail: xc_dom_allocate: cmdline="", features=""
domainbuilder: detail: xc_dom_kernel_file: filename="/usr/local/lib/xen/boot/hvmloader
↪"
domainbuilder: detail: xc_dom_malloc_filemap    : 326 kB
libxl: debug: libxl_dom.c:988:libxl__load_hvm_firmware_module: Loading BIOS: /usr/
↪local/share/seabios/bios.bin
...
```

如果这些详细的输出不能帮助你诊断问题，在 **/var/log/xen** 中还有 QEMU 和 Xen™ toolstack 日志。请注意，域的名称会被附加到日志名称中，所以如果域名为 freebsd，你应该会找到 **/var/log/xen/xl-freebsd.log**，并可能找到 **/var/log/xen/qemu-dm-freebsd.log**。这两个日志文件都可能包含用于调试的有用信息。如果这些

都不能帮助你解决问题，请将你所面临的问题的描述和尽可能多的信息发送到 freebsd-xen@FreeBSD.org 和 xen-devel@lists.xenproject.org，以便获得帮助。

第二十五章本地化——i18n/L10n 的使用和设置

25.1.概述

FreeBSD 是一个用户和贡献者遍布全球的软件项目。因此，FreeBSD 支持多种语言的本地化，允许用户用非英语的语言来查看、输入或处理数据。人们可以从大多数的主流语言中进行选择，其中包括但不限于以下语言：中文、德语、日语、韩语、法语、俄语和越南语。

国际化一词被简称为 i18n，它代表了国际化（`internationalization`）的第一个字母和最后一个字母之间的数量。L10n 这个简称也使用相同的命名方案，但来自本地化（`localization`）一词。i18n/L10n 输入法、协议和应用程序可以让用户选择使用他们的语言。

这一章讨论了 FreeBSD 的国际化和本地化特性。读完这一章后，你将知道：

- 本地化名称是如何构建的。
- 如何为登录的 shell 进行区域设置。
- 如何为非英语语言配置控制台。
- 如何为不同的语言配置 Xorg。
- 如何找到符合 i18n 的应用程序。
- 在哪里可以找到更多关于配置特定语言的信息。

在阅读本章之前，你应该：

- 知道如何安装额外的第三方应用程序[1524]。

[1524] https://docs.freebsd.org/en/books/handbook/ports/index.html#ports

25.2.使用本地化

本地化设置主要基于三个部分：语言代码、国家代码（地区）和编码。本地化名称是由这些部分构成的，如下所示：

```
语言代码_国家代码（地区）.编码
```

语言代码和国家代码用于确定国家（地区）和具体的语言变化。通用语言和国家（地区）代码提供了一些语言代码_国家（地区）代码的例子。

表 15. 常见的语言和国家（地区）代码

语言代码_国家（地区）代码	代指
en_US	英语，美国
ru_RU	俄语，俄罗斯
zh_TW	繁体中文，中国台湾

完整的可用语言列表可以通过输入以下内容找到：

```
% locale -a | more
```

要确定当前的地区设置。

```
% locale
```

特定语言的字符集，如 ISO8859-1、ISO8859-15、KOI8-R 和 CP437，在 multibyte(3)[1525] 中有说明。可以在 IANA 注册中心[1526]找到有效的字符集列表。

有些语言，如中文或日文，不能用 ASCII 字符表示，需要用宽字符或多字节字符进行扩展语言编码。宽字符或多字节编码的例子包括 EUC 和 Big5。老的应用程序可能会误认为这些编码是控制字符，而新的应用程序通常会识别这些字符。根据具体实现，用户可能需要编译软件以支持宽字符或多字节字符，或者对其进行正确的配置。

> 注意
> FreeBSD 使用与 Xorg 兼容的地区编码。

本节的其余部分将说明在 FreeBSD 系统上配置 locale 的各种方法。下一节将讨论寻找和编译支持 i18n 的应用程序的注意事项。

[1525] https://www.freebsd.org/cgi/man.cgi?query=multibyte&sektion=3&format=html
[1526] http://www.iana.org/assignments/character-sets

25.2.1.为登录 shell 设置地区设置

地区设置是在用户的 **~/.login_conf** 或用户 shell 的启动文件 **~/.profile**，**~/.bashrc**，或 **~/.cshrc** 中配置的。

有需要对两个环境变量进行设置：

- `LANG`，用于设置语言环境
- `MM_CHARSET`，设置应用程序使用的 MIME 字符集

除了配置用户的 shell 之外，还应该为特定的应用程序配置和 Xorg 配置设置这些变量。

有两种方法可用于进行所需的变量分配：登录分级[1527]（推荐）和启动文件[1528]。接下来的两节将演示如何使用这两种方法。

25.2.1.1.登录分级

这第一种是推荐的方法，因为它为每个可能的 shell 都分配了所需的环境变量，如本地化名称和 MIME 字符集。这个设置可以由每个用户执行，也可由超级用户为所有用户配置。

这个最简单的例子在单个用户的主目录的 **.login_conf** 中为 `Latin-1` 编码设置了两个变量：

```
me:\
    :charset=ISO-8859-1:\
    :lang=de_DE.ISO8859-1:
```

下面是一个用户 **~/.login_conf** 的例子，它为 BIG-5 编码的繁体中文设置了变量。因为有些应用程序不能正确地遵守中文、日文和韩文的地区变量，所以需要设置更多的变量：

```
#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,LC_COLLATE=zh_TW.Big5,LC_CTYPE=zh_TW.Big5,LC_
↪MESSAGES=zh_TW.Big5,LC_MONETARY=zh_TW.Big5,LC_NUMERIC=zh_TW.Big5,LC_TIME=zh_TW.
↪Big5:\
```

另外，超级用户可以对系统的所有用户进行本地化配置。在 **/etc/login.conf** 中的以下变量可用于设置 locale 和 MIME 字符集：

```
language_name|Account Type Description:\
    :charset=MIME_charset:\
    :lang=locale_name:\
    :tc=default:
```

[1527] https://docs.freebsd.org/en/books/handbook/book/#login-class

[1528] https://docs.freebsd.org/en/books/handbook/book/#startup-file

因此，之前的 Latin-1 例子就会是这样的：

```
german|German Users Accounts:\
:charset=ISO-8859-1:\
:lang=de_DE.ISO8859-1:\
:tc=default:
```

关于这些变量的更多细节，见 login.conf(5)[1529]。注意，它已经包含了预先定义的 *russian * 分级。

每当编辑 **/etc/login.conf** 后，记得执行以下命令来更新数据库能力：

```
# cap_mkdb /etc/login.conf
```

> 注意
> 对于终端用户来说，需要在他们的 **~/.login_conf** 上运行 `cap_mkdb` 命令，以使任何改变生效。

25.2.1.1.1.改变登录分级的实用程序

除了手动编辑 **/etc/login.conf** 之外，还有几个工具可以用来为新创建的用户设置 locale。

当使用 `vipw` 添加新用户时，指定 *language* 来设置 locale：

```
user:password:1111:language:0:0:User Name:/home/user:/bin/sh
```

当使用 `adduser` 添加新用户时，可以为所有新用户预先配置默认语言，也可以为单个用户指定。

如果所有新用户都使用相同的语言，在 **/etc/adduser.conf** 中设置 `defaultclass=` *language*。

要在创建用户时覆盖这个设置，可以在这个提示下输入所需的地区语言：

```
Enter login class: default []:
```

或者在调用 `adduser` 时指定要设置的 locale：

```
# adduser -class language
```

如果使用 `pw` 来添加新的用户，请按以下方式指定 locale：

```
# pw useradd user_name -L language
```

要改变一个现有用户的登录分级，可以使用 `chpass`。以超级用户身份调用它，并提供要编辑的用户名作为参数：

[1529] https://www.freebsd.org/cgi/man.cgi?query=login.conf&sektion=5&format=html

```
# chpass user_name
```

25.2.1.2.shell 启动文件方法

不推荐使用这种方法，因为每个使用的 shell 都需要进行手动配置，每个 shell 都有不同的配置文件和不同的语法。举个例子，为了给 `sh` 设置德语，可以在 **~/.profile** 中加入这些行，以设置仅适用于该用户的 shell。这些行也可以添加到 **/etc/profile** 或 **/usr/share/skel/dot.profile** 中，以便为所有用户都设置该 shell：

```
LANG=de_DE.ISO8859-1; export LANG
MM_CHARSET=ISO-8859-1; export MM_CHARSET
```

然而，对于 csh 来说，配置文件的名称和使用的语法是不同的。这些是以上设置的同等文件 **~/.login**、**/etc/csh.login** 和 **/usr/share/skel/dot.login**：

```
setenv LANG de_DE.ISO8859-1
setenv MM_CHARSET ISO-8859-1
```

使问题复杂化的是，在 **~/.xinitrc** 中配置 Xorg 所需的语法也取决于 shell。第一个例子是针对 sh 的，第二个是针对 csh 的：

```
LANG=de_DE.ISO8859-1; export LANG
```

```
setenv LANG de_DE.ISO8859-1
```

25.2.2.控制台设置

有几种本地化的字体可用于控制台。要查看可用字体的列表，请输入 `ls /usr/share/syscons/fonts`。要配置控制台字体，在 **/etc/rc.conf** 中指定 *font_name*，不含 *.fnt* 后缀：

```
font8x16=font_name
font8x14=font_name
font8x8=font_name
```

可以通过在 **/etc/rc.conf** 中添加以下内容来设置键盘映射和屏幕映射：

```
scrnmap=screenmap_name
keymap=keymap_name
keychange="fkey_number sequence"
```

要查看可用的屏幕映射列表，输入 `ls /usr/share/syscons/scrnmaps`。在指定 *screenmap_name* 时，无需 **.scm** 后缀。通常需要一个带有相应映射字体的屏幕映射，作为 VGA 适配器字体字符矩阵上的第 8 位扩展到第 9 位的变通方法，这样，如果屏幕字体使用第 8 位列，字母就会被移出伪图区。

要查看可用的键盘映射，请输入 `ls /usr/share/syscons/keymaps`。当指定 *keymap_name* 时，无需 **.kbd** 后缀。要在不重启的情况下测试键盘映射，请使用 kbdmap(1)[1530]。

由于不能在键盘映射中定义功能键序列，因此通常需要使用 `keychange` 条目来对功能键进行编程以匹配所选的终端类型。

接下来，在 **/etc/ttys** 中为所有虚拟终端条目设置正确的控制台终端类型。为字符集定义的终端类型[1531]总结了可用的终端类型：

表 16. 为字符集定义的终端类型

字符集	终端类型
ISO8859-1 或 ISO8859-15	cons25l1
ISO8859-2	cons25l2
ISO8859-7	cons25l7
KOI8-R	cons25r
KOI8-U	cons25u
CP437 (VGA 默认)	cons25
US-ASCII	cons25w

对于使用宽字符或多字节字符的语言，应从 FreeBSD ports 中安装适合该语言的控制台。可用的 port 的可用终端的 port[1532] 中进行了总结。安装完毕后，请参考 port 的 **pkg-message** 或 man 页面以了解配置和使用说明。

语言	port 位置
繁体中文 (BIG-5)	chinese/big5con[1533]
中文/日语/韩语	chinese/cce[1534]
中文/日语/韩语	chinese/zhcon[1535]
日语	chinese/kon2[1536]
日语	japanese/kon2-14dot[1537]
日语	japanese/kon2-16dot[1538]

[1530] https://www.freebsd.org/cgi/man.cgi?query=kbdmap&sektion=1&format=html

[1531] https://docs.freebsd.org/en/books/handbook/book/#locale-charset

[1532] https://docs.freebsd.org/en/books/handbook/book/#locale-console

如果在 **/etc/rc.conf** 中启用了 moused，可能需要额外的配置。默认情况下，驱动 syscons(4)[1539] 的鼠标光标在字符集中占据范围 `0xd0-0xd3`。如果语言使用了这个范围，请在 **/etc/rc.conf** 中添加以下一行来移动光标的范围：

```
mousechar_start=3
```

25.2.3.设置 Xorg

X Window 系统[1540] 介绍了如何安装和配置 Xorg。在为本地化配置 Xorg 时，可以从 FreeBSD ports 中获得额外的字体和输入法。可以在 **~/.Xresources** 中调整应用程序特定的 i18n 设置，如字体和菜单，并可让用户在图形化的应用程序菜单中查看他们选择的语言。

X 输入法（XIM）协议是符合 Xorg 标准，用于输入非英语字符。可用的输入法[1541] 总结了 FreeBSD ports 中可用的输入法应用程序。另外还有一些 Fcitx 和 Uim 应用。

语言	输入法
中文	chinese/gcin[1542]
中文	chinese/ibus-chewing
中文	chinese/ibus-pinyin[1543]
中文	chinese/oxim[1544]
中文	chinese/scim-fcitx[1545]
中文	chinese/scim-pinyin[1546]
中文	chinese/scim-tables[1547]
日语	japanese/ibus-anthy[1548]
日语	japanese/ibus-mozc[1549]
日语	japanese/ibus-skk[1550]
日语	japanese/im-ja[1551]
日语	japanese/kinput2[1552]
日语	japanese/scim-anthy[1553]
日语	japanese/scim-canna[1554]
日语	japanese/scim-honnoka[1555]
日语	japanese/scim-honoka-plugin-romkan[1556]

continues on next page

[1533] https://cgit.freebsd.org/ports/tree/chinese/big5con/pkg-descr
[1534] https://cgit.freebsd.org/ports/tree/chinese/cce/pkg-descr
[1535] https://cgit.freebsd.org/ports/tree/chinese/zhcon/pkg-descr
[1536] https://cgit.freebsd.org/ports/tree/chinese/kon2/pkg-descr
[1537] https://cgit.freebsd.org/ports/tree/japanese/kon2-14dot/pkg-descr
[1538] https://cgit.freebsd.org/ports/tree/japanese/kon2-16dot/pkg-descr
[1539] https://www.freebsd.org/cgi/man.cgi?query=syscons&sektion=4&format=html
[1540] https://docs.freebsd.org/en/books/handbook/x11/index.html#x11
[1541] https://docs.freebsd.org/en/books/handbook/book/#locale-xim

表 4 – continued from previous page

语言	输入法
日语	japanese/scim-honoka-plugin-wnn[1557]
日语	japanese/scim-prime[1558]
日语	japanese/scim-skk[1559]
日语	japanese/scim-tables[1560]
日语	japanese/scim-tomoe[1561]
日语	japanese/scim-uim[1562]
日语	japanese/skkinput[1563]
日语	japanese/skkinput3[1564]
日语	japanese/uim-anthy[1565]
韩语	korean/imhangul[1566]
韩语	korean/nabi[1567]
韩语	korean/scim-hangul[1568]
韩语	korean/scim-tables[1569]
越南语	vietnamese/xvnkb[1570]
越南语	vietnamese/x-unikey[1571]

25.3.寻找 i18n 应用程序

i18n 应用程序是使用库中的 i18n 工具包进行编程的。这些程序可让开发人员编写一个简单的文件，并将显示的菜单和文本翻译成各种语言。

FreeBSD ports[1572] 包含了许多内置支持多种语言的宽字符或多字节字符的应用程序。这样的应用程序在它们的名字中包含了 `i18n`，以便于识别。然而，它们并不总是支持所需要的语言。

一些应用程序可以用特定的字符集进行编译。这通常是在 port 的 **Makefile** 中完成的，或通过向 configure 传递一个值。关于如何确定所需的 configure 值，或在 port 的 **Makefile** 中确定在编译 port 时应使用哪些编译选项，请参考相应 FreeBSD port 源代码中的 i18n 文档。

[1542] https://cgit.freebsd.org/ports/tree/chinese/gcin/pkg-descr
[1543] https://cgit.freebsd.org/ports/tree/chinese/ibus-chewing/pkg-descr
[1544] https://cgit.freebsd.org/ports/tree/chinese/oxim/pkg-descr
[1545] https://cgit.freebsd.org/ports/tree/chinese/scim-fcitx/pkg-descr
[1546] https://cgit.freebsd.org/ports/tree/chinese/scim-pinyin/pkg-descr
[1547] https://cgit.freebsd.org/ports/tree/chinese/scim-tables/pkg-descr
[1548] https://cgit.freebsd.org/ports/tree/japanese/ibus-anthy/pkg-descr
[1549] https://cgit.freebsd.org/ports/tree/japanese/ibus-mozc/pkg-descr
[1550] https://cgit.freebsd.org/ports/tree/japanese/ibus-skk/pkg-descr
[1551] https://cgit.freebsd.org/ports/tree/japanese/im-ja/pkg-descr
[1552] https://cgit.freebsd.org/ports/tree/japanese/kinput2/pkg-descr
[1553] https://cgit.freebsd.org/ports/tree/japanese/scim-anthy/pkg-descr
[1554] https://cgit.freebsd.org/ports/tree/japanese/scim-canna/pkg-descr
[1555] https://cgit.freebsd.org/ports/tree/japanese/scim-honoka/pkg-descr
[1556] https://cgit.freebsd.org/ports/tree/japanese/scim-honoka-plugin-romkan/pkg-descr
[1557] https://cgit.freebsd.org/ports/tree/japanese/scim-honoka-plugin-wnn/pkg-descr
[1558] https://cgit.freebsd.org/ports/tree/japanese/scim-prime/pkg-descr
[1559] https://cgit.freebsd.org/ports/tree/japanese/scim-skk/pkg-descr
[1560] https://cgit.freebsd.org/ports/tree/japanese/scim-tables/pkg-descr
[1561] https://cgit.freebsd.org/ports/tree/japanese/scim-tomoe/pkg-descr
[1562] https://cgit.freebsd.org/ports/tree/japanese/scim-uim/pkg-descr
[1563] https://cgit.freebsd.org/ports/tree/japanese/skkinput/pkg-descr
[1564] https://cgit.freebsd.org/ports/tree/japanese/skkinput3/pkg-descr
[1565] https://cgit.freebsd.org/ports/tree/japanese/uim-anthy/pkg-descr
[1566] https://cgit.freebsd.org/ports/tree/korean/imhangul/pkg-descr
[1567] https://cgit.freebsd.org/ports/tree/korean/nabi/pkg-descr
[1568] https://cgit.freebsd.org/ports/tree/korean/scim-hangul/pkg-descr
[1569] https://cgit.freebsd.org/ports/tree/korean/scim-tables/pkg-descr
[1570] https://cgit.freebsd.org/ports/tree/vietnamese/xvnkb/pkg-descr
[1571] https://cgit.freebsd.org/ports/tree/vietnamese/x-unikey/pkg-descr
[1572] https://www.freebsd.org/ports/

25.4.特定语言的区域配置

这一节提供了将 FreeBSD 系统本地化为俄语的配置示例。然后，本文还提供了一些用于本地化其他语言的额外资源。

25.4.1.俄语 (KOI8-R 编码)

这一节介绍了对 FreeBSD 系统进行俄语本地化所需的具体设置。关于每一种设置的更完整介绍，请参考使用本地化[1573]。

要为登录的 shell 设置这种语言，请在每个用户的 **~/.login_conf** 中添加以下几行：

```
me:My Account:\
    :charset=KOI8-R:\
    :lang=ru_RU.KOI8-R:
```

要配置控制台，在 **/etc/rc.conf** 中添加以下几行：

```
keymap="ru.utf-8"
scrnmap="utf-82cp866"
font8x16="cp866b-8x16"
font8x14="cp866-8x14"
font8x8="cp866-8x8"
mousechar_start=3
```

对于 **/etc/ttys** 中的每个 `ttyv` 条目，使用 `cons25r` 作为终端类型。

要配置打印，需要一个特殊的输出过滤器来将 KOI8-R 转换为 CP866，因为大多数带有俄文字符的打印机都带有硬件代码页 CP866。FreeBSD 包括一个默认的过滤器，**/usr/libexec/lpr/ru/koi2alt**。要使用这个过滤器，请在 **/etc/printcap** 中添加这个条目：

```
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:
```

请参考 printcap(5)[1574] 以获得更详细的解释。

要配置对安装的 MS-DOS® 文件系统中的俄文文件名的支持，在向 **/etc/fstab** 添加条目时包括 `-L` 和区域名称：

```
/dev/ad0s2 /dos/c msdos rw,-Lru_RU.KOI8-R 0 0
```

[1573] https://docs.freebsd.org/en/books/handbook/book/#using-localization

[1574] https://www.freebsd.org/cgi/man.cgi?query=printcap&sektion=5&format=html

更多细节请参考 mount_msdosfs(8)[1575]。

要为 Xorg 配置俄语字体，请安装软件包 x11-fonts/xorg-fonts-cyrillic[1576]。然后，检查 **/etc/X11/xorg.conf** 中的“Files”部分。必须在任何其他 `FontPath` 条目之前添加以下一行：

```
FontPath "/usr/local/lib/X11/fonts/cyrillic"
```

可以在 ports 中找到其他的西里尔文字体。

要激活俄语键盘，请在 **/etc/xorg.conf** 的“Keyboard”部分添加以下内容：

```
Option "XkbLayout"   "us,ru"
Option "XkbOptions"  "grp:toggle"
```

确保在该文件中被注释掉了 `XkbDisable`。

对于 `grp:toggle` 使用 `right Alt`，对于 `grp:ctrl_shift_toggle` 使用 `Ctrl+Shift`。对于 grp:caps_toggle 使用 `CapsLock`。旧的 `CapsLock` 功能在 LAT 模式下仍然可用，只能使用 `Shift+CapsLock`。`grp:caps_toggle` 在 Xorg 中不能使用，原因不明。

如果键盘有“Windows®”键，并且一些非字母键被错误地映射，请在 **/etc/xorg.conf** 中添加以下一行：

```
Option "XkbVariant" ",winkeys"
```

> 注意
>
> 俄罗斯的 XKB 键盘可能无法在非本地化的应用程序中使用。最低限度的本地化应用程序应该在程序的早期调用 `XtSetLanguageProc(NULL, NULL, NULL);` 函数。

更多有关 Xorg 应用程序本地化的说明，请参见 http://koi8.pp.ru/xwin.html。更多关于 KOI8-R 编码的一般信息，请参考 http://koi8.pp.ru/。

25.4.2.其他特定语言的资源

本节列出了一些配置其他语言的额外资源。

中国台湾的繁体中文

FreeBSD 台湾项目在 http://netlab.cse.yzu.edu.tw/~statue/freebsd/zh-tut/ 网站上提供了 FreeBSD 的中文使用教程。

希腊语本地化

这里[1577]有一篇关于 FreeBSD 希腊语支持的完整文章，只有希腊语版本，是 FreeBSD 希腊语官方文档的一部分。

[1575] https://www.freebsd.org/cgi/man.cgi?query=mount_msdosfs&sektion=8&format=html

[1576] https://cgit.freebsd.org/ports/tree/x11-fonts/xorg-fonts-cyrillic/pkg-descr

[1577] https://www.freebsd.org/doc/gr/articles/greek-language-support/

日语和韩语的本地化

日语请参考 http://www.jp.FreeBSD.org/[1578]，韩语请参考 http://www.kr.FreeBSD.org/[1579]。

非英语的 FreeBSD 文档

一些 FreeBSD 的贡献者已经将 FreeBSD 的部分文档翻译成了其他语言。它们可以通过 FreeBSD 网站[1580]上的链接或 **/usr/share/doc** 获得。

[1578] http://www.jp.freebsd.org/

[1579] http://www.kr.freebsd.org/

[1580] https://www.freebsd.org/

第二十六章 FreeBSD 更新与升级

26.1.概述

FreeBSD 在不同的分支之间不断迭代。有些人喜欢使用官方的 release 版本，而另一些人则喜欢与最新的开发版本保持同步。然而，即使是官方的 release 版本也会经常进行安全和其他关键性修复的更新。无论你使用哪个版本，FreeBSD 都提供了所必须的工具来维护系统的更新，并可在不同版本之间轻松切换。本章介绍了如何跟踪开发系统以及确保 FreeBSD 系统更新的基本工具。

读完本章后，你将知道：

- 如何使用 freebsd-update 或 Git 来使 FreeBSD 系统保持最新状态。
- 如何将已安装的系统与已知的原始副本进行状态比对。
- 如何使用 Git 或文档 port 来确保所安装的文档是最新的。
- 两个开发分支之间的区别：FreeBSD-STABLE 和 FreeBSD-CURRENT。
- 如何编译和重新安装整个基本系统。

在阅读本章之前，你应该：

- 正确设置网络连接高级网络[1581]。
- 知道如何安装额外的第三方软件安装应用程序：软件包和 port[1582]。

> 注意
>
> 在本章中，使用 `git` 来获取和更新 FreeBSD 的源代码。你也可以选择使用软件包或 port devel/git[1583]。

[1581] https://docs.freebsd.org/en/books/handbook/advanced-networking/index.html#advanced-networking

[1582] https://docs.freebsd.org/en/books/handbook/ports/index.html#ports

[1583] https://cgit.freebsd.org/ports/tree/devel/git/pkg-descr

26.2.更新 FreeBSD

及时应用安全补丁和升级到新版的操作系统是系统持续性管理的一个重要方面。FreeBSD 有一个叫做 `freebsd-update` 的工具，可以用来执行这两项任务。

这个工具支持 FreeBSD 安全和勘误的二进制更新，而不需要手动编译和安装补丁或新内核。二进制更新适用于目前安全团队所支持的所有架构和版本。所支持的版本列表和它们的预计结束日期列在 https://www.freebsd.org/security/。

该工具也支持将操作系统进行次要版本更新，或升级到另一个开发分支。在升级到一个新的版本之前，请查看其发行公告，因为它包含了与此版本有关的重要信息。发行公告可从 https://www.freebsd.org/releases/ 获得。

> 注意
> 如果在 crontab(5)[1584] 使用了 freebsd-update(8)[1585] 功能，在升级操作系统之前必须将其禁用。

本节介绍了 `freebsd-update` 使用的配置文件，示范了如何应用安全补丁以及如何升级到一个次要或主要的操作系统版本，并讨论了升级操作系统时的一些注意事项。

26.2.1.配置文件

`freebsd-update` 依据默认的配置文件运行。一些用户可能想对 **/etc/freebsd-update.conf** 中的默认配置进行调整，以便更好的控制这个过程。该文件中的注释解释了可用的选项，但是以下内容可能需要更详细的说明：

```
# Components of the base system which should be kept updated.
Components world kernel
```

这个参数控制了 FreeBSD 的哪些部分会被保持在最新状态。默认情况是更新整个基本系统和内核。也可以指定个别组件，例如 **src/base** 或 **src/sys**。然而，最好的选择是保持默认，因为改变它以包括特定的项目需要列出每个需要的项目。随着时间的推移，这可能会产生灾难性的后果，因为源代码和二进制文件可能变得不同步。

```
# Paths which start with anything matching an entry in an IgnorePaths
# statement will be ignored.
IgnorePaths /boot/kernel/linker.hints
```

要想在更新过程中不触动指定的目录，例如 **/bin** 或 **/sbin**，可以在该语句中添加它们的路径。这个选项可以用来防止 `freebsd-update` 覆盖本地的修改。

[1584] https://www.freebsd.org/cgi/man.cgi?query=crontab&sektion=5&format=html
[1585] https://www.freebsd.org/cgi/man.cgi?query=freebsd-update&sektion=8&format=html

```
# 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
```

这个选项将只更新指定目录中未修改的配置文件。用户所做的任何修改都会阻止这些文件的自动更新。还有一个选项 KeepModifiedMetadata，它将指示 freebsd-update 在合并过程中保存更改。

```
# 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/ /boot/device.hints
```

带有配置文件的目录列表，freebsd-update 应该尝试合并这些文件。文件合并过程是一系列类似于 mergemaster(8)[1586] 的 diff(1)[1587] 补丁，但选项较少。合并后要么接受，要么打开一个编辑器，要么导致 freebsd-update 中止。如果有疑问，可以备份 **/etc** 并接受合并。关于 mergemaster 的更多信息，见 mergemaster(8)[1588]。

```
# Directory in which to store downloaded updates and temporary
# files used by FreeBSD Update.
# WorkDir /var/db/freebsd-update
```

这个目录是放置所有修补程序和临时文件的地方。在用户进行版本升级的情况下，这个位置应该至少有 1GB 的可用磁盘空间。

```
# 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
```

当这个选项设置为 yes 时，freebsd-update 将认为组件列表是完整的，并且不会尝试在列表之外进行修改。实际上，freebsd-update 将尝试更新属于组件列表的每个文件。

更多细节，请参阅 freebsd-update.conf(5)[1589]。

[1586] https://www.freebsd.org/cgi/man.cgi?query=mergemaster&sektion=8&format=html
[1587] https://www.freebsd.org/cgi/man.cgi?query=diff&sektion=1&format=html
[1588] https://www.freebsd.org/cgi/man.cgi?query=mergemaster&sektion=8&format=html
[1589] https://man.freebsd.org/cgi/man.cgi?query=freebsd-update.conf&sektion=5&format=html

26.2.2.应用安全补丁

应用 FreeBSD 安全补丁的过程被简化了，使得管理员能够使用 `freebsd-update` 来保持系统的补丁完整性。更多关于 FreeBSD 安全公告的信息可以在 FreeBSD 安全公告[1590]中找到。

可以通过下列命令来下载和安装 FreeBSD 的安全补丁。第一条命令将确定是否有任何未完成的补丁，如果有的话，将列出在应用补丁时将被修改的文件。第二条命令将应用这些补丁。

```
# freebsd-update fetch
# freebsd-update install
```

如果更新应用了任何内核补丁，系统将需要重启，以便启动到打了补丁的内核。如果该补丁应用于任何正在运行的二进制文件，受影响的应用程序应该被重新启动，以便使用二进制文件的补丁版本。

> 注意
>
> 通常情况下，用户需要准备好重启系统。要知道系统是否因为内核更新而需要重启，执行命令 `freebsd-version -k` 和 `uname -r`。如果输出结果不同，就重新启动系统。

可以通过在 **/etc/crontab** 中添加这个条目，将系统配置为每天自动检查一次更新。

```
@daily                                  root    freebsd-update cron
```

如果存在补丁，它们将被自动下载，但不会被应用。根用户将被发送一封电子邮件，这样就可以通过 `freebsd-update install` 来查看和手动安装补丁。

如果出了什么问题，`freebsd-update` 有能力通过下面的命令回滚到上一组修改：

```
# freebsd-update rollback
Uninstalling updates... done.
```

同样，如果内核或任何内核模块被修改，应该重新启动系统，任何受影响的二进制文件也应该被重新启动。

只有 **GENERIC** 内核可以被 `freebsd-update` 自动更新。如果安装的是定制内核，那么在 `freebsd-update` 完成安装更新后，必须重新编译并重新安装内核。默认的内核名称是 **GENERIC**。可以用 uname(1)[1591] 命令来验证它的安装。

> 注意
>
> 在 **/boot/GENERIC** 中始终保留了一份 **GENERIC** 内核的副本。它在诊断各种问题和进行版本升级时都会有帮助。请参阅 FreeBSD 9.X 及更高版本的定制内核[1592] 以了解如何获得 **GENERIC** 内核的副本。

除非 **/etc/freebsd-update.conf** 中的默认配置被改变，否则 `freebsd-update` 将和其他的更新一起安装更新的内核源码。可以按照常规方法重新编译和安装新的定制内核。

[1590] https://docs.freebsd.org/en/books/handbook/security/index.html#security-advisories

[1591] https://www.freebsd.org/cgi/man.cgi?query=uname&sektion=1&format=html

[1592] https://docs.freebsd.org/en/books/handbook/cutting-edge/#freebsd-update-custom-kernel-9x

由 freebsd-update 发布的更新并不总是涉及内核。如果内核源码没有被 freebsd-update 安装修改，就没有必要重新编译定制内核。然而，freebsd-update 会一直更新 **/usr/src/sys/conf/newvers.sh**,当前的补丁级别，由 uname -r 报告的 p 数字表示，可以从该文件获得。编译定制内核，即使没有其他变化，也可以让 uname 准确地报告系统的当前补丁级别。这在维护多个系统时特别有帮助，因为它可以快速评估每个系统中安装的更新。

26.2.3.执行次要和主要版本的升级

从 FreeBSD 的一个小版本升级到另一个小版本被称为 次要版本升级。例如：

- 从 FreeBSD 13.1 到 13.2

主要版本升级会增加主要版本号，例如：

- 从 FreeBSD 12.4 到 13.2

这两种类型的升级都可以通过包含 freebsd-update 支持的 release 版本来完成。

警告

在每个新的 RELEASE 发布之后，FreeBSD 软件包编译服务器在一定时期内将并 不使用新版本的操作系统。这为许多在（系统）发布后不立即进行升级的用户提供了连续性。比如说：

- 使用 13.1 和 13.2 的用户的软件包将在运行 13.1 的服务器上进行编译，直到 13.1 的支持结束。

---—但是，严格来讲：

- 一个在 13.1 上编译的内核模块可能并 不适合于 13.2。

因此，在任何次要或主要的操作系统版本升级中，如果你的软件包要求包括任一内核模块：

- 准备好从源代码编译该模块。

注意

如果系统正在运行定制内核，在开始升级之前，请确保在 **/boot/GENERIC** 中有一份 **GENERIC** 内核的副本。请参考 FreeBSD 9.X 及以后版本的定制内核[1593]，以了解如何获得 **GENERIC** 内核的副本。

在 FreeBSD 13.1 系统上运行下面的命令时，会将其升级到 FreeBSD 13.2：

```
# freebsd-update -r 13.2-RELEASE upgrade
```

在收到命令后，freebsd-update 将评估配置文件和当前系统，试图收集必要的信息来执行升级。屏幕上会显示哪些组件已经被检测到，哪些没有被检测到。比方说:

```
Looking up update.FreeBSD.org mirrors... 1 mirrors found.
Fetching metadata signature for 13.2-RELEASE from update1.FreeBSD.org... done.
Fetching metadata index... done.
Inspecting system... done.
```

(continues on next page)

[1593] https://docs.freebsd.org/en/books/handbook/book/#freebsd-update-custom-kernel-9x

(continued from previous page)

```
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
```

在这时候，`freebsd-update` 将尝试下载所有升级所需的文件。在某些情况下，用户可能会被提示关于安装什么或如何进行的问题。

当使用定制内核时，上述步骤将产生类似于以下的警告：

```
WARNING: This system is running a "MYKERNEL" kernel, which is not a
kernel configuration distributed as part of FreeBSD 13.1-RELEASE.
This kernel will not be updated: you MUST update the kernel manually
before running "/usr/sbin/freebsd-update install"
```

这个警告在此时可以被安全地忽略。更新后的 **GENERIC** 内核将被用作升级过程中的一个中间步骤。

所有的补丁被下载到本地系统后，它们将被应用。这个过程可能需要一些时间，取决于机器的速度和工作量。然后，配置文件将被合并。合并过程需要一些用户干预，因为文件可能会被合并，或者屏幕上会出现一个编辑器用于手动合并。在这个过程中，每次成功的合并结果都会显示给用户。合并失败或被忽略将导致进程中止。用户可能希望对 **/etc** 进行备份，并在以后手动合并重要文件（如 **master.passwd** 和 **group**）。

> 注意
>
> 系统还没有被改变，因为所有的补丁和合并都在另一个目录中进行。一旦所有的补丁都被成功应用，所有的配置文件都被合并，而且看起来这个过程会很顺利，用户就可以使用下面的命令将修改提交到磁盘：
>
> ```
> # freebsd-update install
> ```

内核和内核模块将首先被打上补丁。如果系统使用的是定制内核，请使用 nextboot(8)[1594] 将下次启动时的内核设置为更新的 **/boot/GENERIC**：

```
# nextboot -k GENERIC
```

[1594] https://www.freebsd.org/cgi/man.cgi?query=nextboot&sektion=8&format=html

警告

在使用 **GENERIC** 内核重启之前，如果被更新的机器是通过远程访问的话，请确保它包含了系统正常启动和连接网络所需的所有驱动程序。特别是，如果正在运行的定制内核包含了通常由内核模块提供的内置功能，请确保使用 **/boot/loader.conf** 工具将这些模块临时加载到 **GENERIC** 内核中。建议禁用非必要的服务，以及任何磁盘和网络挂载，直到升级过程完成。

现在应该用升级后的内核重新启动机器：

```
# shutdown -r now
```

系统重新上线后，使用下面的命令重新启动 `freebsd-update`。由于进程的状态已经被保存，`freebsd-update` 不会从头开始，而是进入下一阶段并删除所有旧的共享库和对象文件。

```
# freebsd-update install
```

注意

取决于是否有任何库的版本号被提升，安装阶段可能只有两个，而非三个。

现在升级已经完成。如果这是一次主要版本的升级，请按照在主要版本升级软件包[1595]中的说明来重新安装所有的 port 和软件包。

26.2.3.1.在 FreeBSD 9.X 及以后版本的定制内核

在使用 `freebsd-update` 之前，请确保在 **/boot/GENERIC** 中存在一个 **GENERIC**内核的副本。如果一个定制内核只被编译过一次，那么 **/boot/kernel.old** 中的内核就是 **GENERIC** 内核，只需将这个目录重命名为 **/boot/GENERIC**。

如果一个定制的内核被编译了不止一次，或者不知道这个定制的内核被编译了多少次，那么请获取一份与当前操作系统版本相匹配的 `GENERIC` 内核。如果可以对系统进行物理访问，可以从安装介质中安装 `GENERIC` 内核的副本：

```
# mount /cdrom
# cd /cdrom/usr/freebsd-dist
# tar -C/ -xvf kernel.txz boot/kernel/kernel
```

另外，也可以从源代码中重新编译和安装 `GENERIC` 内核：

```
# cd /usr/src
# make kernel __MAKE_CONF=/dev/null SRCCONF=/dev/null
```

要使这个内核被 `freebsd-update` 识别为 **GENERIC** 内核，必须没有以任何方式对 **GENERIC** 配置文件进行修改。此外，我们还建议在编译内核时不要使用任何其他特殊的选项。

[1595] https://docs.freebsd.org/en/books/handbook/book/#freebsdupdate-portsrebuild

由于 `freebsd-update` 只需要 **/boot/GENERIC** 的存在，因此不需要重新启动到 **GENERIC** 内核。

26.2.3.2.在主要版本升级后升级软件包

一般来说，已安装的应用程序在小版本升级后会继续工作，不会有问题。主要版本使用不同的应用程序二进制接口 (ABI)，这将破坏大多数第三方应用程序。在主要版本升级后，所有安装的软件包和 port 都需要升级。包可以用 `pkg upgrade` 来升级。要升级已安装的 port，请使用诸如 ports-mgmt/portmaster[1596] 等工具。

要对所有已安装的软件包进行强制升级并用软件库中的新版本替换这些软件包，即使版本号没有增加。这是由于在 FreeBSD 的主要版本之间进行升级时 ABI 版本的变化所要求的。可以通过执行以下操作来完成强制升级：

```
# pkg-static upgrade -f
```

重新编译所有已安装的应用程序可以用这个命令来完成：

```
# portmaster -af
```

这个命令将显示每个有可配置选项的应用程序的配置屏幕，并等待用户与这些屏幕互动。要防止这种行为，只使用默认选项，在上述命令中加上 `-G`。

软件升级完成后，最后一次调用 `freebsd-update` 来结束升级过程，以便把升级过程中的所有问题都解决。

```
# freebsd-update install
```

如果临时使用了 **GENERIC** 内核，这时可以按照配置 FreeBSD 内核[1597]的说明建立并安装一个新的定制内核。

重新启动机器进入新的 FreeBSD 版本。升级过程现在已经完成。

26.2.4.系统状态比对

可以使用 `freebsd-update IDS` 来测试所安装的 FreeBSD 版本与已知的良好副本的状态。这个命令可以评估系统工具、库和配置文件的当前版本，并可以作为一个内置的入侵检测系统（IDS）使用。

> 警告
>
> 这个命令不能替代真正的 IDS，如 security/snort[1598]。由于 `freebsd-update` 将数据存储在磁盘上，遭篡改的可能性是很大的。虽然可以通过使用 `kern.securelevel` 和在不使用时将 `freebsd-update` 数据存储在只读文件系统中来减少这种可能性，但更好的解决方案是将系统

[1596] https://cgit.freebsd.org/ports/tree/ports-mgmt/portmaster/pkg-descr

[1597] https://docs.freebsd.org/en/books/handbook/kernelconfig/index.html#kernelconfig

[1598] https://cgit.freebsd.org/ports/tree/security/snort/pkg-descr

与安全磁盘进行比较，如 DVD 或安全存储的外部 USB 磁盘设备。关于使用内置工具提供 IDS 功能的替代方法在二进制验证[1599]中有所说明。

要开始比对，指定输出文件，将结果保存到：

```
# freebsd-update IDS >> outfile.ids
```

现在将检查系统，输出一个冗长的文件列表，以及版本中的已知值和当前安装的 SHA256 哈希值，将被传输到指定的输出文件。

清单中的条目非常长，但输出格式可以很容易地被解析。例如，要获得与发行版中不同的所有文件的列表，请执行以下命令：

```
# cat outfile.ids | awk '{ print $1 }' | more
/etc/master.passwd
/etc/motd
/etc/passwd
/etc/pf.conf
```

这个样本输出已经被截断了，因为还有很多文件存在。一些文件有自然的修改:例如，如果用户被添加到系统中，**/etc/passwd** 将被修改。内核模块可能有所不同，因为 `freebsd-update` 可能已经更新了它们。要排除特定的文件或目录，可以在 **/etc/freebsd-update.conf** 中的 `IDSIgnorePaths` 选项中添加它们。

26.3.更新 Bootcode

以下手册对 Bootcode 和 Boot Loader 的升级过程进行了说明：gpart(8)[1600]、gptboot(8)[1601]、gptzfsboot(8)[1602] 和 loader.efi(8)[1603]。

26.4.更新文档

文档是 FreeBSD 操作系统的一个组成部分。虽然总是可以在 FreeBSD 网站（文档门户[1604]）上找到最新版本的 FreeBSD 文档，但如果有一份最新的 FreeBSD 网站、手册、FAQ 和文章的本地副本，也会很方便。

这一节介绍了如何使用源代码或 FreeBSD Ports 来保持 FreeBSD 文档的本地副本的更新。

关于编辑和提交文档更正的信息，请参考 FreeBSD 文档项目新贡献者入门（FreeBSD 文档项目新贡献者入门[1605]）。

[1599] https://docs.freebsd.org/en/books/handbook/security/index.html#security-ids
[1600] https://www.freebsd.org/cgi/man.cgi?query=gpart&sektion=8&format=html
[1601] https://www.freebsd.org/cgi/man.cgi?query=gptboot&sektion=8&format=html
[1602] https://www.freebsd.org/cgi/man.cgi?query=gptzfsboot&sektion=8&format=html
[1603] https://www.freebsd.org/cgi/man.cgi?query=loader.efi&sektion=8&format=html
[1604] https://docs.freebsd.org/
[1605] https://docs.freebsd.org/en/books/fdp-primer/

26.4.1.从源代码更新文档

从源代码重新编译 FreeBSD 文档需要一系列 FreeBSD 基本系统之外的工具。可以按照 FreeBSD 文档项目入门的这些步骤来安装所需的工具。

安装完成后，使用 `git` 来获取一份干净的文档源代码：

```
# git clone https://git.FreeBSD.org/doc.git /usr/doc
```

文档的源代代码的初始下载可能需要一些时间。让它运行，直到完成。

要获取后续更新的文件源代码，可以执行：

```
# git pull
```

文档源代码的最新快照被提取到 **/usr/doc** 后，一切都为更新已安装的文档做好准备。

要进行全面更新，键入：

```
# cd /usr/doc
# make
```

26.5.追踪开发分支

FreeBSD 有两个开发分支：FreeBSD-CURRENT 和 FreeBSD-STABLE。

本节对每个分支及其受众进行了解释，并介绍了如何使系统与各分支保持同步。

26.5.1.使用 FreeBSD-CURRENT

FreeBSD-CURRENT 是 FreeBSD 开发的“血锋”，FreeBSD-CURRENT 的用户应该有很高的技术水平。那些希望追踪开发分支但技术水平较低的用户，应该追踪 FreeBSD-STABLE。

FreeBSD-CURRENT 是 FreeBSD 的最新源代码，它包括正在进行的工作及实验性的修改，以及在下一个正式版本中可能出现也可能不出现的过渡机制。尽管许多 FreeBSD 开发人员每天都在编译 FreeBSD-CURRENT 源代码，但仍有一些短时期内的源代码可能无法编译。这些问题会被尽快解决，但 FreeBSD-CURRENT 是否带来灾难或新的功能，可能与源代码同步的时间有关。

FreeBSD-CURRENT 适用于三个主要的利益群体：

1. 正在积极研究某些部分的源代码的 FreeBSD 社区成员。
2. FreeBSD 社区的成员是积极的测试者。他们愿意花时间解决问题，对 FreeBSD 的变化和总体方向提出专题建议，并提交补丁。
3. 希望关注事物的用户，使用当前的源码作为参考，或偶尔发表评论或贡献代码。

FreeBSD-CURRENT 不应该被认为是在下一个版本之前获得新功能的快速通道，因为预发布的功能还没有经过充分的测试，而且很可能包含有错误。它不是一个快速获得错误修正的方法，因为任何给定的提交都有可能修正现有错误并引入新的错误。FreeBSD-CURRENT 没有任何“官方支持”。

要追踪 FreeBSD-CURRENT：

1. 加入 FreeBSD-CURRENT 邮件列表[1606]以及 src 仓库主分支的提交信息[1607]列表。这对于看到人们对系统现状的评论以及收到有关 FreeBSD-CURRENT 现状的重要公告是 *必不可少*的。

   src 仓库列表的主分支的提交信息[1608]记录了每个改动的提交日志条目，以及任何可能的副作用的相关信息。

   要加入这些列表，请到 FreeBSD 列表服务器[1609]，点击要订阅的列表，并按照说明操作。为了跟踪整个源码的变化，而不仅仅是 FreeBSD-CURRENT 的变化，请订阅 src 仓库中所有分支的 Commit 信息[1610]。

2. 与 FreeBSD-CURRENT 源码同步。通常情况下，用 `git` 来从 FreeBSD Git 仓库的 `main` 分支检出 -CURRENT 代码 (详见使用 Git[1611])。

3. 出于仓库的大小，一些用户选择只同步他们感兴趣的部分或他们正在贡献补丁的部分的源代码。然而，计划从源代码编译操作系统的用户必须下载全部的 FreeBSD-CURRENT，而不能只选择部分。

   在编译 FreeBSD-CURRENT 之前，请仔细阅读 **/usr/src/Makefile**，并遵循从源代码更新 FreeBSD[1612] 中的说明。阅读 FreeBSD-CURRENT 邮件列表[1613] 和 **/usr/src/UPDATING** 来了解其他引导程序的最新情况，这些程序在通往下一个版本的过程中有时是必须的。

4. 动起手来！我们鼓励 FreeBSD-CURRENT 用户提交他们的改进或错误修正建议。我们随时欢迎附有代码的建议。

26.5.2.使用 FreeBSD-STABLE

FreeBSD-STABLE 是开发分支，主要的版本都是从它开始的。进入这个分支的修改速度较慢，而且一般认为这些修改已经在 FreeBSD-CURRENT 中进行了测试。这 *仍*然是一个开发分支，而且在任何时候，FreeBSD-STABLE 的源代码可能都不适合用于一般的用途。它只是另一个工程开发轨道，而不是终端用户可依赖的。没有设备进行测试的用户应该运行最新的 FreeBSD release。

那些对追踪或贡献于 FreeBSD 开发过程感兴趣的人，特别是与 FreeBSD 下一版本有关的人，应该考虑关注 FreeBSD-STABLE。

尽管 FreeBSD-STABLE 分支应该在任何时候都能编译和运行，但这并不能保证。由于运行 FreeBSD-STABLE 的人比运行 FreeBSD-CURRENT 的人多，有时不可避免地会在 FreeBSD-STABLE 中发现一些在 FreeBSD-CURRENT 中不明显的 bug 和角落案例。由于这个原因，人们不应盲目地追踪 FreeBSD-STABLE。

[1606] https://lists.freebsd.org/subscription/freebsd-current
[1607] https://lists.freebsd.org/subscription/dev-commits-src-main
[1608] https://lists.freebsd.org/subscription/dev-commits-src-main
[1609] https://lists.freebsd.org/
[1610] https://lists.freebsd.org/subscription/dev-commits-src-all
[1611] https://docs.freebsd.org/en/books/handbook/mirrors/index.html#git
[1612] https://docs.freebsd.org/en/books/handbook/book/#makeworld
[1613] https://lists.freebsd.org/subscription/freebsd-current

尤其重要的是，在没有在开发或测试环境中彻底测试代码之前，不要把任何生产服务器更新到 FreeBSD-STABLE。

要追踪 FreeBSD-STABLE：

1. 加入 FreeBSD-STABLE 邮件列表[1614]，以便随时了解 FreeBSD-STABLE 中可能出现的联编依赖关系或其他需要特别注意的问题。开发人员在考虑进行一些有争议的修复或更新时，也会在这个邮件列表中发布公告，如果用户对提议的修改有任何问题要提出，可以给他们一个回应的机会。

   加入被追踪的分支的相关 git 列表。例如，追踪 13-STABLE 分支的用户应该加入 src 仓库的稳定分支的提交信息[1615]。这个列表记录了每个改动的提交日志条目，以及任何可能产生的副作用的相关信息。

   要加入这些列表，请到 FreeBSD 列表服务器[1616]，点击要订阅的列表，并按照说明操作。为了跟踪整个源代码的变化，请订阅 src 仓库所有分支的提交信息[1617]。

2. 要安装一个新的 FreeBSD-STABLE 系统，需要从 FreeBSD 镜像站点安装最新的 FreeBSD-STABLE 版本，或者使用由 FreeBSD-STABLE 编译的每月快照。关于快照的更多信息，请参考 www.freebsd.org/snapshots[1618]。

   要将现有的 FreeBSD 系统编译或升级到 FreeBSD-STABLE，可以使用 `git` 来检查所需分支的源代码。分支名称，例如 `stable/9`，会在 www.freebsd.org/releng[1619] 列出。

3. 在编译或升级到 FreeBSD-STABLE 之前，请仔细阅读 **/usr/src/Makefile** 并遵循从源代码更新 FreeBSD[1620] 中的说明。阅读 FreeBSD-STABLE 邮件列表[1621] 和 **/usr/src/UPDATING** 来了解在通往下一版本的道路上有时需要的其他引导程序的最新情况。

26.5.3.数值 N

当追踪 bug 时，知道被用来创建出现问题的系统来自哪些版本的源代码是很重要的。FreeBSD 提供了编译在内核中的版本信息，例如 uname(1)[1622] 可以检索到这些信息：

```
% uname -v
FreeBSD 14.0-CURRENT #112 main-n247514-031260d64c18: Tue Jun 22 20:43:19 MDT 2021     ␣
↪fred@machine:/usr/home/fred/obj/usr/home/fred/git/head/amd64.amd64/sys/FRED
```

最后一个字段给出了关于内核名称、构建它的人以及它被编译的位置的信息。看一下第四列，它是由几个部分组成的：

[1614] https://lists.freebsd.org/subscription/freebsd-stable
[1615] https://lists.freebsd.org/subscription/dev-commits-src-branches
[1616] https://lists.freebsd.org/
[1617] https://lists.freebsd.org/subscription/dev-commits-src-all
[1618] https://www.freebsd.org/snapshots/
[1619] https://www.freebsd.org/releng/
[1620] https://docs.freebsd.org/en/books/handbook/book/#makeworld
[1621] https://lists.freebsd.org/subscription/freebsd-stable
[1622] https://www.freebsd.org/cgi/man.cgi?query=uname&sektion=1&format=html

```
main-n247514-031260d64c18

main        ①
n247514     ②
031260d64c18    ③
        ④
```

① Git 分支的名称。注意：数值 N 只对发布的项目分支（`main`, `stable/XX` 和 `releng/XX`）有效。本地分支也会有数值 n，这些编号会与它们的父分支的提交相重合。

② 数值 n 是指从该行包含的 Git 散列开始，回溯到 Git 仓库开始的提交的线性计数。

③ 检出的 Git 哈希值

④ 有时，当内核是在未提交的仓库上建立的时候，会出现后缀带有 -dirty 的情况。在这个例子中，它是不存在的，因为 FRED 的内核是由原始检出编译的。

`git rev-list` 命令用于查找与 Git 哈希值对应的数值 n。比如说：

```
% git rev-list --first-parent --count 031260d64c18    ①
247514     ②
```

① 用来转换的 git 的哈希值（上面例子中的哈希值是重复使用的）。

② 数值 n。

通常情况下，这个数字并不那么重要。然而，当错误修复被提交时，这个数字可以让我们很容易地确定该修复是否存在于当前运行的系统中。开发者通常会提到提交的哈希值（或提供一个有该哈希值的链接），但不会提到数值 n，因为哈希值是一个变化的易见标识，而数值 n 则不是。安全公告和勘误通知也会注明数值 n，可以直接与你的系统进行比较。当你需要使用浅层 Git 克隆时，你不能可靠地比较数值 n，因为 `git rev-list` 命令会计算仓库中的所有修订，而浅层克隆会省略这些修订。

26.6.从源代码更新 FreeBSD

与二进制更新相比，通过从源代码编译来更新 FreeBSD 具有一些优势。可以用不同的参数来编译代码，以利用特定的硬件。基本系统的某些部分可以用非默认的配置来编译，或者完全丢弃不需要或不想要的地方。与安装二进制更新相比，编译过程需要更长的时间来更新系统，但可完全定制，以产生自定义的 FreeBSD 版本。

26.6.1.快速开始

这是对通过从源代码编译来更新 FreeBSD 的典型步骤的快速参考。后面的章节会更详细地介绍这个过程。

警告

当从 mergemaster(8)[1623] 切换到 etcupdate(8)[1624] 时，第一次运行可能会错误地合并变化，产生虚假的冲突。为了防止这种情况，在更新源代码和编译新的世界之 前，请执行以下步骤：

```
# etcupdate extract ①
# etcupdate diff    ②
```

① 自举旧的 **/etc** 文件的数据库；更多信息见 etcupdate(8)[1625]。

② 在自举后检查差异。修剪任何不再需要的本地修改，以减少在未来更新中发生冲突的机会。

- 更新和编译

```
# git pull /usr/src  ①
check /usr/src/UPDATING  ②
# cd /usr/src          ③
# make -j4 buildworld  ④
# make -j4 kernel      ⑤
# shutdown -r now      ⑥
# etcupdate -p         ⑦
# cd /usr/src          ⑧
# make installworld    ⑨
# etcupdate -B         ⑩
# shutdown -r now      ⓫
```

① 获取最新版本的源代码。关于获取和更新源代码的更多信息，请参见更新源代码[1626]。

② 检查 **/usr/src/UPDATING**，看看在从源代码编译之前或之后是否需要任何手动步骤。

③ 切换到源代码目录。

④ 编译世界（除了内核以外的一切）。

⑤ 编译和安装内核。这等同于 `make buildkernel installkernel`。

⑥ 重新启动系统到新的内核。

⑦ 在安装世界之前，更新和合并 **/etc/** 中所需要的配置文件。

⑧ 切换到源代码目录。

⑨ 安装世界。

⑩ 更新和合并 **/etc/** 中的配置文件。

⓫ 重新启动系统以使用新编译的世界和内核。

[1623] https://www.freebsd.org/cgi/man.cgi?query=mergemaster&sektion=8&format=html0

[1624] https://www.freebsd.org/cgi/man.cgi?query=etcupdate&sektion=8&format=html

[1625] https://www.freebsd.org/cgi/man.cgi?query=etcupdate&sektion=8&format=html

[1626] https://docs.freebsd.org/en/books/handbook/book/#updating-src-obtaining-src

26.6.2. 为从源代码更新做准备

阅读 **/usr/src/UPDATING**。在更新之前或之后必须执行的任何手动步骤在这个文件中都有说明。

26.6.3. 更新源代码

FreeBSD 的源代码位于 **/usr/src/**。更新此源代码的首选方法是通过 Git 版本控制系统。验证源代码是否在版本控制之下：

```
# cd /usr/src
# git remote --v
origin  https://git.freebsd.org/src.git (fetch)
origin  https://git.freebsd.org/src.git (push)
```

这表明 **/usr/src/** 处于版本控制之下，可以用 git(1)[1627] 更新：

```
# git pull /usr/src
```

如果目录最近没有被更新，更新过程可能需要一些时间。在它完成后，源代码是最新的，可以开始下一节所述的编译过程。

注意

获得源代码：

如果输出显示 `fatal: not a git repository`，说明那里的文件缺失或是用不同的方法安装的。需要重新检出该源代码。

表 **19. FreeBSD** 版本和仓库的分支

[1627] https://www.freebsd.org/cgi/man.cgi?query=git&sektion=1&format=html

uname -r 输出	仓库路径	说明
X.Y-RELEASE	releng/*X.Y*	RELEASE 版只附带有关键的安全和错误修复补丁。建议大多数用户使用该分支。
X.Y-STABLE	stable/*X*	RELEASE 版加上该分支上的所有额外开发。*STABLE* 指的是应用程序二进制接口（ABI）没有改变，因此为早期版本编译的软件仍然可以运行。例如，在 FreeBSD 10.1 上编译的软件仍然可以在后来编译的 FreeBSD 10-STABLE 上运行。STABLE 分支偶尔会有可能影响用户的错误或不兼容，尽管这些通常会很快被修复。
X-CURRENT	main	最新的未发布的 FreeBSD 开发版本。CURRENT 分支可能会有重大的错误或不兼容，因此只推荐给高级用户。

用 uname(1)[1628] 确定正在使用哪个版本的 FreeBSD：

```
# uname -r
13.2-RELEASE
```

根据 FreeBSD 版本和仓库分支[1629]，用于更新 13.2-RELEASE 的源代码的仓库路径是 releng/13.2。这个路径在检出源代码时被使用：

```
# mv /usr/src /usr/src.bak   ①
# git clone --branch releng/13.2 https://git.FreeBSD.org/src.git /usr/src   ②
```

① 把旧的目录移开。如果这个目录中没有本地的修改，就可以删除它。

② FreeBSD 版本和仓库分支[1630]的路径会被添加到仓库的链接中。第三个参数是本地系统上源代码的目标目录。

26.6.4.从源代码开始编译

世界，即除了内核以外的所有操作系统，都被编译了。这样做首先是为了提供最新的工具来编译内核。然后是内核本身的编译。

```
# cd /usr/src
# make buildworld
# make buildkernel
```

[1628] https://www.freebsd.org/cgi/man.cgi?query=uname&sektion=1&format=html

[1629] https://docs.freebsd.org/en/books/handbook/book/#updating-src-obtaining-src-repopath

[1630] https://docs.freebsd.org/en/books/handbook/book/#updating-src-obtaining-src-repopath

编译后的代码被写到 **/usr/obj**。

这些是基本步骤。下面将介绍控制编译的其他选项。

26.6.4.1.执行清洁编译

某些版本的 FreeBSD 联编系统会将先前编译的代码留在临时对象目录 **/usr/obj** 中。这可以通过避免重新编译没有变化的代码来加快以后的联编速度。要强制编译所有内容，可以在开始联编前使用 `cleanworld`：

```
# make cleanworld
```

26.6.4.2.设置作业的数量

在多核处理器上增加编译作业的数量可以提高编译速度。用 `sysctl hw.ncpu` 来确定核心数。处理器各不相同，不同版本的 FreeBSD 所使用的联编系统也各不相同，因此测试是唯一可以确定不同数量的作业对联编速度有何影响的方法。作为一个开始，可以考虑核心数的一半到两倍之间的数值。作业数量是用 `-j` 指定的。

例 **42.** 增加编译作业的数量

用四项作业来编译世界和内核：

```
# make -j4 buildworld buildkernel
```

26.6.4.3.只编译内核

如果源代码有变化，必须完成 `buildworld`。除此之外，可以在任何时候运行 `buildkernel` 来编译内核。要想只编译内核：

```
# cd /usr/src
# make buildkernel
```

26.6.4.4.编译定制内核

标准的 FreeBSD 内核基于一个叫做 **GENERIC** 的 内核配置文件。**GENERIC** 内核包括最常用的设备驱动和选项。有时，建立一个定制内核是有用的或必要的，可以添加或删除设备驱动程序或选项以适应特定的需要。

例如，有人在开发一台内存严重受限的小型嵌入式计算机时，可以删除不需要的设备驱动程序或选项，使内核略微缩小。

内核配置文件位于 **/usr/src/sys/arch/conf/**，其中 *arch* 是 `uname -m` 的输出。在大多数计算机上，它是 `amd64`，给出的配置文件目录是 **/usr/src/sys/amd64/conf/**。

技巧

/usr/src 可以被删除或重新创建，所以最好把定制的内核配置文件放在一个单独的目录下，比如 **/root**。将内核配置文件链接到 conf 目录中。如果该目录被删除或覆盖，内核配置文件可以被重新链接到新目录中。

通过复制配置文件 **GENERIC** 可以创建一个自定义的配置文件。在这个例子中，新的定制内核是用于存储服务器的，所以被命名为 **STORAGESERVER**：

```
# cp /usr/src/sys/amd64/conf/GENERIC /root/STORAGESERVER
# cd /usr/src/sys/amd64/conf
# ln -s /root/STORAGESERVER .
```

然后编辑 **/root/STORAGESERVER**，添加或删除设备或选项，如 config(5)[1631] 所示。

定制的内核是通过在命令行上设置 KERNCONF 到内核配置文件来编译的：

```
# make buildkernel KERNCONF=STORAGESERVER
```

26.6.5.安装已编译的代码

在 buildworld 和 buildkernel 的步骤完成后，安装新的内核和世界：

```
# cd /usr/src
# make installkernel
# shutdown -r now
# cd /usr/src
# make installworld
# shutdown -r now
```

如果编译了定制内核，KERNCONF 也必须被设置为使用新的定制内核：

```
# cd /usr/src
# make installkernel KERNCONF=STORAGESERVER
# shutdown -r now
# cd /usr/src
# make installworld
# shutdown -r now
```

[1631] https://www.freebsd.org/cgi/man.cgi?query=config&sektion=5&format=html

26.6.6.完成更新

最后几项任务完成了更新。任何修改过的配置文件都将与新版本合并，过时的库被定位并删除，然后系统被重新启动。

26.6.6.1.用 etcupdate(8)[Page 644, 1632] 合并配置文件

etcupdate(8)[1633] 是一个管理更新文件的工具，这些文件没有作为安装世界的一部分被更新，比如位于 **/etc/** 的文件。它通过对这些文件的修改与本地版本进行三方合并来管理更新。与 mergemaster(8)[1634] 的交互式提示相比，它还被设计为尽量减少用户的干预。

注意

一般来说，etcupdate(8)[1635] 在执行中不需要任何特殊的参数。然而，有一个很方便的中间命令，用于检查第一次使用 etcupdate(8)[1636] 时将会做什么：

```
# etcupdate diff
```

该命令允许用户审计配置的变化。

如果 etetcupdate(8)[1637] 不能自动合并文件，可以通过执行手动交互来解决合并冲突：

```
# etcupdate resolve
```

警告

当从 mergemaster(8)[1638] 切换到 etcupdate(8)[1639] 时，第一次运行可能会不正确地合并变化，产生虚假的冲突。为了防止这种情况，在更新源代码和编译新的世界之前，请执行以下步骤：

```
# etcupdate extract ①
# etcupdate diff ②
```

① 启动库存 **/etc** 文件的数据库，更多信息见 etcupdate(8)[1640]。

② 在启动后检查差异。修剪任何不再需要的本地修改，以减少在未来更新中发生冲突的机会。

[1632] https://www.freebsd.org/cgi/man.cgi?query=etcupdate&sektion=8&format=html
[1633] https://www.freebsd.org/cgi/man.cgi?query=etcupdate&sektion=8&format=html
[1634] https://www.freebsd.org/cgi/man.cgi?query=mergemaster&sektion=8&format=html
[1635] https://www.freebsd.org/cgi/man.cgi?query=etcupdate&sektion=8&format=html
[1636] https://www.freebsd.org/cgi/man.cgi?query=etcupdate&sektion=8&format=html
[1637] https://www.freebsd.org/cgi/man.cgi?query=etcupdate&sektion=8&format=html
[1638] https://www.freebsd.org/cgi/man.cgi?query=mergemaster&sektion=8&format=html
[1639] https://www.freebsd.org/cgi/man.cgi?query=etcupdate&sektion=8&format=html
[1640] https://www.freebsd.org/cgi/man.cgi?query=etcupdate&sektion=8&format=html

26.6.6.2.用 mergemaster(8)[Page 645, 1641] 合并配置文件

mergemaster(8)[1642] 提供了一种将对系统配置文件的修改与这些文件的新版本进行合并的方法。mergemaster(8)[1643] 是替代的 etcupdate(8)[1644] 的首选方法。使用 `-Ui`，mergemaster(8)[1645] 可自动更新未被用户修改的文件，并安装尚未存在的新文件：

```
# mergemaster -Ui
```

如果一个文件必须被手动合并，一个交互式的显示允许用户选择文件的哪些部分被保留。参见 mergemaster(8)[1646] 以了解更多信息。

26.6.6.3.检查是否有过期的文件和库

一些过时的文件或目录在更新后可能仍然存在。可以找到这些文件：

```
# make check-old
```

并删除：

```
# make delete-old
```

一些过时的库也可能保留下来。这些可以用以下方法检测：

```
# make check-old-libs
```

并删除：

```
# make delete-old-libs
```

当库被删除后，仍然使用这些旧库的程序将停止工作。这些程序必须在删除旧库后重新编译或替换。

> 技巧
> 当知道所有的旧文件或目录都可以安全删除时，可以通过在命令中设置 BATCH_DELETE_OLD_FILES来避免按 y 和回车键来删除每个文件。比如说：
>
> ```
> # make BATCH_DELETE_OLD_FILES=yes delete-old-libs
> ```

[1641] https://www.freebsd.org/cgi/man.cgi?query=mergemaster&sektion=8&format=html
[1642] https://www.freebsd.org/cgi/man.cgi?query=mergemaster&sektion=8&format=html
[1643] https://www.freebsd.org/cgi/man.cgi?query=mergemaster&sektion=8&format=html
[1644] https://www.freebsd.org/cgi/man.cgi?query=etcupdate&sektion=8&format=html
[1645] https://www.freebsd.org/cgi/man.cgi?query=mergemaster&sektion=8&format=html
[1646] https://www.freebsd.org/cgi/man.cgi?query=mergemaster&sektion=8&format=html

26.6.6.4.更新后重新启动

更新后的最后一步是重新启动计算机，以便所有的变化都生效。

```
# shutdown -r now
```

26.7.多台机器的追踪

当多台机器需要追踪同一个源代码时，让每个系统下载源码并重新编译一切，是对磁盘空间、网络带宽和 CPU 周期的一种浪费。解决方案是让一台机器做大部分的工作，而其余的机器通过 NFS 挂载这些工作。本节概述了一种这样做的方法。关于使用 NFS 的更多信息，请参阅网络文件系统（NFS）[1647]。

首先，确定一组机器，它们将运行同一组二进制文件，称为 *构建集*。每台机器可以有一个定制内核，但将运行相同的用户空间二进制文件。从这组机器中，选择一台机器作为 *构建机器*，在上面构建 world 和内核。理想情况下，这是一台快速的机器，它有足够的空闲 CPU 来运行 `make buildworld` 和 `make buildkernel`。

选择一台机器作为 *测试机*，在软件更新投入生产之前对其进行测试。这台机器 *必须*能够承受长时间的故障。它可以是构建机，但不一定是。

这个构建集中的所有机器都需要通过 NFS 从构建机上挂载 **/usr/obj** 和 **/usr/src**。对于多个构建集，**/usr/src** 应该在一台构建机上，其余的则通过 NFS 安装。

确保构建集中所有机器上的 **/etc/make.conf** 和 **/etc/src.conf** 与构建机一致。这意味着，构建机器必须构建构建组中任何机器要安装的基础系统的所有部分。另外，每个构建机器都应该在 **/etc/make.conf** 中用 `KERNCONF` 设置其内核名称，构建机器应该在其 `KERNCONF` 中列出所有的内核，将自己的内核列在前面。构建机器必须在其 **/usr/src/sys/arch/conf** 中为每个机器准备好内核配置文件。

在联编机上，按照从源代码更新 FreeBSD[1648] 中的说明，联编内核和世界，但不要在联编机上安装任何东西。相反，将构建好的内核安装到测试机上。在测试机上，通过 NFS 挂载 **/usr/src** 和 **/usr/obj**。然后，运行 `shutdown now` 进入单用户模式，以便安装新的内核和世界，像往常一样运行 `mergemaster`。完成后，重新启动，回到正常的多用户操作。

在验证了测试机器上的所有东西都正常工作后，使用同样的程序将新的软件安装到构建集中的其他每台机器上。

同样的方法也可以用在 ports 上。第一步是通过 NFS 将 **/usr/ports** 共享给联编集中的所有机器。要配置 **/etc/make.conf** 来共享 distfiles，将 `DISTDIR` 设置为一个共同的共享目录，这个目录可以被 NFS 挂载的任何一个 `root` 用户所写入。如果要在本地构建 ports，每台机器都应将 `WRKDIRPREFIX` 设置为本地构建目录。另外，如果联编系统要联编并向联编组中的机器分发软件包，则应将联编系统上的 `PACKAGES` 设置为与 `DISTDIR` 类似的目录。

[1647] https://docs.freebsd.org/en/books/handbook/network-servers/index.html#network-nfs

[1648] https://docs.freebsd.org/en/books/handbook/book/#makeworld

第二十七章 DTrace

27.1.概述

DTrace，即动态跟踪，是一个由 Sun™ 开发的工具，用来在生产系统和预生产系统中定位性能瓶颈。除了诊断性能问题以外，DTrace 也可以被用来调查和调试 FreeBSD 内核与用户态程序中的异常行为。

DTrace 是一个卓越的性能分析工具，有着众多用于诊断系统问题的功能。它还可以用来执行预先编写好的脚本，以充分利用其功能。用户可以使用 DTrace 的 D 语言打造他们自己的工具，令用户能够根据特殊需求自定义他们的性能分析。

其在 FreeBSD 中提供了对内核态 DTrace 的完整支持与用户态 DTrace 的实验性支持。用户态 DTrace 允许用户使用 `pid` 执行对用户态程序的函数边界跟踪，并在用户态程序中插入静态探测器以便之后进行跟踪。像 databases/postgresql12-server[1649] 和 lang/php74[1650] 这样的 port 提供了一个 DTrace 选项来启用静态探测器。

DTrace 官方指南由 Illumos 项目维护，位于 DTrace 指南[1651]。

读完本章后，你将知道：

- DTrace 是什么，有什么用。
- Solaris™ 和在 FreeBSD 中提供的 DTrace 实现有何区别。
- 如何在 FreeBSD 中启用和使用 DTrace。

在你阅读本章之前，你应该：

- 理解 UNIX® 和 FreeBSD 基础（FreeBSD 基础[1652]）。
- 知悉有关 FreeBSD 的安全措施（安全[1653]）。

[1649] https://cgit.freebsd.org/ports/tree/databases/postgresql12-server/pkg-descr
[1650] https://cgit.freebsd.org/ports/tree/lang/php74/pkg-descr
[1651] http://dtrace.org/guide
[1652] https://docs.freebsd.org/en/books/handbook/basics/index.html#basics
[1653] https://docs.freebsd.org/en/books/handbook/security/index.html#security

27.2.实现上的差异

虽然在 FreeBSD 的 DTrace 与 Solaris™ 的类似，但仍有一些区别。主要区别是 FreeBSD 中的 DTrace 是由一组内核模块实现的，在模块加载之前无法使用 DTrace。要加载所有必须的模块，请执行：

```
# kldload dtraceall
```

自 FreeBSD 10.0-RELEASE 起，运行 `dtrace` 时会自动加载这些模块。

FreeBSD 使用内核选项 `DDB_CTF` 启用内核模块和内核自身对 `CTF` 加载的支持。`CTF` 是 Solaris™ Compact C Type Format，它封装了一种类似于 DWARF 和古老的 stabs 的调试信息的简化形式。。`CTF` 数据通过 `ctfconvert` 和 `ctfmerge` 构建工具添加到二进制库中。`ctfconvert` 工具解析由编译器创建的 `DWARFELF` 调试部分，`ctfmerge` 将 `CTFELF` 部分从对象合并到可执行文件或共享库。

一些 provider 程序是为 FreeBSD 而非 Solaris™ 提供的。最显著的是 `dtmalloc` provider 程序，它允许在 FreeBSD 内核中通过类型跟踪 `malloc()`。Solaris™ 中的一些 provider 程序，比如 `cpc` 和 `mib` 并不存在于 FreeBSD 中，虽然在日后可能支持。此外，一些同时存在于两种操作系统中的提供程序互不兼容，比如它们探测器有不同的参数。所以，在 Solaris™ 中编写的 D 语言脚本在 FreeBSD 中不做修改就可能无法运行，反之亦然。

由于安全措施的差异，在 FreeBSD 上只有 `root` 能够使用 DTrace。Solaris™ 拥有一些在 FreeBSD 中还不存在的底层安全校验。因此 **/dev/dtrace/dtrace** 仅限 `root` 使用。

DTrace 受到通用开发和发行许可证（Common Development and Distribution License，`CDDL`）保护。要在 FreeBSD 中查看此许可证，见 **/usr/src/cddl/contrib/opensolaris/OPENSOLARIS.LICENSE** 或者在 http://opensource.org/licenses/CDDL-1.0 在线查看。虽然 FreeBSD 内核对于 DTrace 的实现采用 `BSD` 许可证，但是当模块以二进制形式分发或加载时会使用 `CDDL` 许可证。

27.3.开启 DTrace 支持

在 FreeBSD 9.2 和 10.0 中，对 DTrace 的支持被内置于 **GENERIC** 内核中。使用早先版本或是希望静态编译 DTrace 支持的用户应当在定制内核配置文件中添加如下内容，并按照配置 FreeBSD 内核[1654]中的指导重新编译内核：

```
options         KDTRACE_HOOKS
options         DDB_CTF
makeoptions DEBUG=-g
makeoptions WITH_CTF=1
```

使用 AMD64 架构的用户还应添加：

[1654] https://docs.freebsd.org/en/books/handbook/kernelconfig/index.html#kernelconfig

```
options         KDTRACE_FRAME
```

这个选项提供了 FBT 支持。虽然 DTrace 在没有这个选项的时候也能够工作，但是对函数边界测试时的支持将会受限。

FreeBSD 系统重启进入新的内核后，或是通过 kldload dtraceall 加载了 DTrace 内核模块后，系统将会需要 ksh，因为 DTrace Toolkit 中有几个工具是通过 ksh 编写的。请确保你通过软件包或 port 安装了 shells/ksh93[1655]。你也可以在 shells/pdksh[1656] 或者 shells/mksh[1657] 上运行这些工具。

最后，安装当前版本的 DTrace Toolkit，这是一些收集系统信息的脚本。其中包括检查打开文件、内存、CPU 使用率等脚本。FreeBSD 10 在 **/usr/share/dtrace** 下已经安装了其中几个脚本。在其他版本的 FreeBSD 中，或者要安装完整的 DTrace Toolkit，请安装通过软件包或 port sysutils/dtrace-toolkit[1658]。

> 注意
>
> **/usr/share/dtrace** 下的脚本是专门用于 FreeBSD 的。并非所有包括在 DTrace Toolkit 中的脚本都能在 FreeBSD 上如期工作，一些脚本可能需要额外的修改。

DTrace Toolkit 包含了许多用 DTrace 专有的 D 语言编写的脚本。这种语言非常接近 C++。对其进行深入的探讨不属于本文的范围，这在 Illumos 动态跟踪指南[1659]中有所说明。

27.4.使用 DTrace

DTrace 脚本由一个或多个 探测器或测量点组成，每个探测器关联一个动作。当探测器的条件被满足时，与之相关联的动作就会被执行。例如，当打开一个文件，启动一个进程或者运行一行代码时可能执行一个动作。这个动作可能是记录某些信息，或者变更上下文变量。对上下文变量的读取和写入使得探测器之间能够共享信息，并且对不同事件协同分析其相关性。

要查看所有探测器，管理员可以执行以下命令：

```
# dtrace -l | more
```

每个探测器都有一个 ID，一个 PROVIDER（dtrace 或 fbt），一个 MODULE 和一个 FUNCTION NAME。关于这个命令的更多信息请见 dtrace(1)[1660]。

这节中的例子大致演示了如何使用 DTrace Toolkit 中两个得到完整支持的脚本：**hotkernel** 和 **procsystime**。

脚本 **hotkernel** 被设计用来识别占用内核时间最多的函数。它会生成类似下文的输出：

[1655] https://cgit.freebsd.org/ports/tree/shells/ksh93/pkg-descr
[1656] https://cgit.freebsd.org/ports/tree/shells/pdksh/pkg-descr
[1657] https://cgit.freebsd.org/ports/tree/shells/mksh/pkg-descr
[1658] https://cgit.freebsd.org/ports/tree/sysutils/dtrace-toolkit/pkg-descr
[1659] http://www.dtrace.org/guide
[1660] https://www.freebsd.org/cgi/man.cgi?query=dtrace&sektion=1&format=html

```
# cd /usr/local/share/dtrace-toolkit
# ./hotkernel
Sampling... Hit Ctrl-C to end.
```

如提示，用快捷键 `Ctrl + C` 结束进程。结束时，脚本会显示内核函数和用时信息列表，按用时升序排列：

```
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%
```

这个脚本也同样适用于内核模块。要使用此功能，请用 `-m` 执行这个脚本：

```
# ./hotkernel -m
Sampling... Hit Ctrl-C to end.
^C
MODULE                                                COUNT   PCNT
0xc107882e                                                1   0.0%
0xc10e6aa4                                                1   0.0%
0xc1076983                                                1   0.0%
```

(continues on next page)

(continued from previous page)

```
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%
```

脚本 **procsystime** 会记录并输出给定 ID（PID）或进程名称的系统调用时间。在如下示例中，将会生成一个新的 **/bin/csh** 实例，然后 **procsystime** 将会执行并等待。这时，在另一个 csh 中输入几个命令。这是该测试的结果：

```
# ./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
```

如结果显示的那样，read() 系统调用占用的时间（以纳秒显示）最长，而 getpid() 占用的时间最少。

第二十八章 USB 设备模式/USB OTG

28.1.概述

这一章介绍了 FreeBSD 中 USB 设备模式和 USB On The Go (USB OTG) 的使用。这包括虚拟串行控制台、虚拟网络接口和虚拟 USB 驱动器。

当在支持 USB 设备模式或 USB OTG 的硬件上运行时，例如许多嵌入式板卡中内置的硬件，FreeBSD USB 栈可以在 **设备模式**下运行。设备模式使得计算机有可能将自己表现为不同种类的 USB 设备类别，包括串行端口、网络适配器和大容量存储，或者它们的组合。像笔记本电脑或台式电脑这样的 USB 主机能够像物理 USB 设备一样访问它们。设备模式有时被称为“USB 小工具模式”。

硬件有两种基本方式可以提供设备模式功能：一种是单独的“客户端端口”，只支持设备模式；另一种是 USB OTG 端口，可以同时提供设备和主机模式。对于 USB OTG 端口，USB 堆栈会自动在主机端和设备端之间切换，这取决于连接到该端口的东西。将一个 USB 设备（如记忆棒）连接到这个端口，会使 FreeBSD 切换到主机模式。连接一个像电脑一样的 USB 主机会使 FreeBSD 切换到设备模式。单一用途的“客户端口”总是在设备模式下工作。

FreeBSD 向 USB 主机展示什么取决于 sysctl 的 `hw.usb.template` 参数。一些模板提供了一个单一的设备，例如一个串行终端；其他模板提供了多个设备，这些设备可以同时使用。一个例子是模板 10，它提供了一个大容量存储设备、一个串行控制台和一个网络接口。参见 usb_template(4)[1661] 获取可用值的列表。

请注意，在某些情况下，根据硬件和主机操作系统的不同，为了让主机注意到配置的变化，它必须在物理上断开并重新连接，或者强制以系统特定的方式重新扫描 USB 总线。当 FreeBSD 在主机上运行时，可以使用 usbconfig(8)[1662] 重置。如果 USB 主机已经连接到 USBOTG 插座，这也必须在加载 **usb_template.ko** 之后进行。

读完本章后，你将知道:

- 如何在 FreeBSD 上设置 USB 设备模式功能。
- 如何在 FreeBSD 上配置虚拟串口。

[1661] https://www.freebsd.org/cgi/man.cgi?query=usb_template&sektion=4&format=html

[1662] https://www.freebsd.org/cgi/man.cgi?query=usbconfig&sektion=8&format=html

- 如何从各种操作系统连接到虚拟串口。
- 如何配置 FreeBSD 以提供一个虚拟的 USB 网络接口。
- 如何配置 FreeBSD 来提供一个虚拟的 USB 存储设备。

28.2.USB 虚拟串行端口

28.2.1.配置 USB 设备模式的串行端口

3 号、8 号和 10 号模板提供了虚拟串口支持。请注意，模板 3 可以与微软 Windows 10 一起使用，不需要特殊的驱动程序和 INF 文件。其他主机操作系统可以与所有三个模板一起工作。usb_template(4)[1663] 和 umodem(4)[1664] 内核模块都必须被加载。

要启用 USB 设备模式的串行端口，请将这些行添加到 **/etc/ttys**：

```
ttyU0   "/usr/libexec/getty 3wire"  vt100   onifconsole secure
ttyU1   "/usr/libexec/getty 3wire"  vt100   onifconsole secure
```

然后在 **/etc/devd.conf** 中添加这些行：

```
notify 100 {
    match "system"      "DEVFS";
    match "subsystem"   "CDEV";
    match "type"        "CREATE";
    match "cdev"        "ttyU[0-9]+";
    action "/sbin/init q";
};
```

如果 devd(8)[1665] 已经在运行，则重新加载配置：

```
# service devd restart
```

通过在 **/boot/loader.conf** 中添加这些行，确保必要的模块被加载，并在启动时设置正确的模板，如果它不存在的话，则创建它：

```
umodem_load="YES"
hw.usb.template=3
```

要加载模块和设置模板而不重启，请使用：

[1663] https://www.freebsd.org/cgi/man.cgi?query=usb_template&sektion=4&format=html
[1664] https://www.freebsd.org/cgi/man.cgi?query=umodem&sektion=4&format=html
[1665] https://www.freebsd.org/cgi/man.cgi?query=devd&sektion=8&format=html

```
# kldload umodem
# sysctl hw.usb.template=3
```

28.2.2.从 FreeBSD 连接到 USB 设备模式的串行端口

要连接到配置为提供 USB 设备模式的串行端口的板子，将 USB 主机，如笔记本电脑，连接到板子的 USB OTG 或 USB 客户端端口。在主机上使用‘pstat -t’来列出终端连接。在列表的最后，你应该看到一个 USB 串口，例如“ttyU0”。要打开这个连接，使用:

```
# cu -l /dev/ttyU0
```

按几次回车键后，你会看到一个登录提示。

28.2.3.从 macOS 连接到 USB 设备模式的串行端口

要连接到配置为提供 USB 设备模式的串行端口的板子，将 USB 主机，如笔记本电脑，连接到板子的 USB OTG 或 USB 客户端端口。要打开连接，请使用：

```
# cu -l /dev/cu.usbmodemFreeBSD1
```

28.2.4.从 Linux 连接到 USB 设备模式的串行端口

要连接到配置为提供 USB 设备模式串行端口的电路板，请将 USB 主机（如笔记本电脑）连接到电路板的 USB OTG 或 USB 客户端端口。要打开连接，请使用：

```
# minicom -D /dev/ttyACM0
```

28.2.5.从微软 Windows 10 连接到 USB 设备模式的串行端口

要连接到配置为提供 USB 设备模式的串行端口的板子，将 USB 主机，如笔记本电脑，连接到板子的 USB OTG 或 USB 客户端端口。要打开连接，你将需要一个串行终端程序，如 PuTTY。要检查 Windows 使用的 COM 端口名称，运行设备管理器，展开“端口（COM 和 LPT）”。你会看到一个类似于“USB 串行设备（COM4）”的名称。运行你选择的串行终端程序，例如 PuTTY。在 PuTTY 对话框中，将“连接类型”设置为“串行”，在“串行线”对话框中输入从设备管理器获得的 COMx，然后点击“打开”。

28.3.USB Device 模式网络接口

1 号、8 号和 10 号模板提供虚拟网络接口支持。请注意，它们都不能与 Microsoft Windows 一起使用。其他主机操作系统可以与所有三个模板一起工作。usb_template(4)[1666] 和 if_cdce(4)[1667] 内核模块都必须被加载。

通过在 **/boot/loader.conf** 中添加这些行来确保必要的模块被加载，并且在启动时设置正确的模板，如果它不存在，则创建它:

```
if_cdce_load="YES"
hw.usb.template=1
```

要加载模块并设置模板而不重启，请使用:

```
# kldload if_cdce
# sysctl hw.usb.template=1
```

28.4.USB 虚拟存储设备

> **注意**
>
> cfumass(4)[1668] 驱动程序是一个 USB 设备模式的驱动程序。在 FreeBSD 12.0 中首次出现。

大规模存储目标是由模板 0 和 10 提供的。usb_template(4)[1669] 和 cfumass(4)[1670] 两个内核模块都必须被加载。cfumass(4)[1671] 与 CTL 子系统的接口，也就是用于 iSCSI 或光纤通道目标的那个。在主机方面,USB Mass Storage 启动器只能访问一个 LUN，LUN 0。

28.4.1.使用 cfumass 启动脚本配置 USB 大容量存储目标

设置只读 USB 存储目标的最简单方法是使用 **cfumass** rc 脚本。要这样配置，把要提交给 USB 主机的文件复制到‘/var/cfumass’目录，并在 **/etc/rc.conf** 中添加这一行。

```
cfumass_enable="YES"
```

要在不重启的情况下配置目标，请运行此命令。

[1666] https://www.freebsd.org/cgi/man.cgi?query=usb_template&sektion=4&format=html
[1667] https://www.freebsd.org/cgi/man.cgi?query=if_cdce&sektion=4&format=html
[1668] https://www.freebsd.org/cgi/man.cgi?query=cfumass&sektion=4&format=html
[1669] https://www.freebsd.org/cgi/man.cgi?query=usb_template&sektion=4&format=html
[1670] https://www.freebsd.org/cgi/man.cgi?query=cfumass&sektion=4&format=html
[1671] https://www.freebsd.org/cgi/man.cgi?query=cfumass&sektion=4&format=html

```
# service cfumass start
```

与串行和网络功能不同，模板不应该在 **/boot/loader.conf** 中设置为 0 或 10。这是因为在设置模板之前必须先设置 LUN。cfumass 启动脚本在启动时自动设置正确的模板号。

28.4.2.使用其他方式配置 USB 大容量存储器

本章的其余部分将详细说明如何在不使用 cfumass rc 文件的情况下设置目标。如果想提供一个可写的 LUN，这是必要的。

USB 大容量存储不需要运行 ctld(8)[1672] 守护程序，尽管在需要时可以使用它。这与 iSCSI 不同。因此，有两种方法来配置目标：ctladm(8)[1673]，或 ctld(8)[1674]。两者都需要加载 **cfumass.ko** 内核模块。该模块可以手动加载:

```
# kldload cfumass
```

如果 **cfumass.ko** 没有被内置到内核中, **/boot/loader.conf** 可以设置在启动时加载该模块:

```
cfumass_load="YES"
```

可以在没有 ctld(8)[1675] 守护程序的情况下创建一个 LUN：

```
# ctladm create -b block -o file=/data/target0
```

这将把映像文件 **/data/target0** 的内容作为一个 LUN 呈现给 USB 主机。在执行该命令之前，该文件必须存在。要在系统启动时配置 LUN，请将该命令添加到 **/etc/rc.local**。

ctld(8)[1676] 也可以用来管理 LUN。创建 **/etc/ctl.conf**，在 **/etc/rc.conf** 中添加一行，确保 ctld(8)[1677] 在启动时自动启动，然后启动守护程序。

这是一个简单的 **/etc/ctl.conf** 配置文件的例子。请参考 ctl.conf(5)[1678] 以获得更完整的选项说明。

```
target naa.50015178f369f092 {
    lun 0 {
        path /data/target0
        size 4G
```

(continues on next page)

[1672] https://www.freebsd.org/cgi/man.cgi?query=ctld&sektion=8&format=html
[1673] https://www.freebsd.org/cgi/man.cgi?query=ctladm&sektion=8&format=html
[1674] https://www.freebsd.org/cgi/man.cgi?query=ctld&sektion=8&format=html
[1675] https://www.freebsd.org/cgi/man.cgi?query=ctld&sektion=8&format=html
[1676] https://www.freebsd.org/cgi/man.cgi?query=ctld&sektion=8&format=html
[1677] https://www.freebsd.org/cgi/man.cgi?query=ctld&sektion=8&format=html
[1678] https://www.freebsd.org/cgi/man.cgi?query=ctl.conf&sektion=5&format=html

(continued from previous page)

```
    }
}
```

该例子创建了一个具有单个 LUN 的单一目标。‘naa.50015178f369f092’ 是一个由 32 个随机十六进制数字组成的设备标识符。‘path’ 行定义了支持 LUN 的文件或 zvol 的完整路径。在启动 ctld(8)[1679] 之前，该文件必须存在。第二行是可选的，指定 LUN 的大小。

要确保 ctld(8)[1680] 守护程序在启动时被启动，在 **/etc/rc.conf** 中添加这一行：

```
ctld_enable="YES"
```

要现在启动 ctld(8)[1681]，运行这个命令：

```
# service ctld start
```

当 ctld(8)[1682] 守护进程被启动时，它读取 **/etc/ctl.conf**。如果该文件在守护进程启动后被编辑，请重新加载修改，使其立即生效：

```
# service ctld reload
```

[1679] https://www.freebsd.org/cgi/man.cgi?query=ctld&sektion=8&format=html
[1680] https://www.freebsd.org/cgi/man.cgi?query=ctld&sektion=8&format=html
[1681] https://www.freebsd.org/cgi/man.cgi?query=ctld&sektion=8&format=html
[1682] https://www.freebsd.org/cgi/man.cgi?query=ctld&sektion=8&format=html

第四部分：网络通讯

FreeBSD 是最广泛部署的高性能网络服务器的操作系统之一。这一部分的章节包括:

- 串行通信
- PPP 和基于以太网的 ppp
- 电子邮件
- 运行网络服务器
- 防火墙
- 其他高级网络话题

这些章节是为了在需要时查阅而设计的。它们不需要按照任何特定的顺序进行阅读，也没有必要在网络环境中使用 FreeBSD 之前阅读所有的章节。

第二十九章串行通信

29.1.概述

UNIX® 素来支持串行通信，因为最早的 UNIX® 机器就依靠串行线进行用户输入和输出。与普通终端由每秒 10 个字符的串行打印机和键盘组成的时代相比，情况已经发生了很大的变化。这一章介绍了在 FreeBSD 上使用串行通信的一些方法。

读完这一章后，你将知道：

- 如何将终端连接到 FreeBSD 系统上。
- 如何使用调制解调器来拨出远程主机。
- 如何允许远程用户用调制解调器登录到 FreeBSD 系统中。
- 如何从串行控制台启动 FreeBSD 系统。

在阅读本章之前，你应该：

- 知道如何配置和安装一个定制内核[1683]。
- 理解 FreeBSD 的权限和进程[1684]。
- 能够获得与 FreeBSD 一起使用的串行硬件的技术手册。

[1683] https://docs.freebsd.org/en/books/handbook/kernelconfig/index.html#kernelconfig
[1684] https://docs.freebsd.org/en/books/handbook/basics/index.html#basics

29.2.串行术语和硬件

以下术语在串行通信中经常使用：

bps

比特每秒（bps）是数据传输的速率。

DTE

数据终端设备（DTE）是串行通信中两个端点之一。例如计算机。

DCE

数据通信设备（DCE）是串行通信中的另一个端点。它通常是调制解调器或串行终端。

RS-232

定义硬件串行通信的原始标准。此后，它被重新命名为 TIA-232。

当提到通信数据率时，本节不使用术语 波特。波特指的是在一段时间内进行的电气状态转换的数量，而 bps 才是正确的术语。

要将串行终端连接到 FreeBSD 系统上，需要在计算机上安装串行端口和适当的电缆来连接到串行设备。已经熟悉串行硬件和电缆的用户可以有把握地跳过这一部分。

29.2.1.串行电缆和端口

有几种不同类型的串行电缆。最常见的两种类型是空调制解调器电缆和标准 RS-232 电缆。硬件的文档应该说明所需的电缆类型。

这两种类型的电缆在电线与连接器的连接方式上有所不同。每条线代表一个信号，RS-232C 信号名称[1685]中总结了定义的信号。一条标准的串行电缆直接传递所有的 RS-232C 信号。例如，电缆一端的“传输数据”引脚会连接到另一端的“传输数据”引脚。这是用于连接调制解调器和 FreeBSD 系统的电缆类型，也适用于某些终端。

零调制解调器电缆将连接器一端的“发送数据”针脚与另一端的“接收数据”针脚进行切换。该连接器可以是 DB-25 或 DB-9。

零调制解调器电缆可以使用 DB-25 到 DB-25 零调制解调器电缆[1686]、DB-9 到 DB-9 零调制解调器电缆[1687]和 DB-9 到 DB-25 零调制解调器电缆[1688]中总结的引脚连接来构建。虽然标准要求 1 号针脚直通 1 号针脚的“保护性接地”线，但它经常被省略。有些终端只使用针脚 2、3 和 7，而其他终端需要不同的配置。如有疑问，请参考硬件的文件。

表 20. RS-232C 信号名称

[1685] https://docs.freebsd.org/en/books/handbook/book/#serialcomms-signal-names

[1686] https://docs.freebsd.org/en/books/handbook/book/#nullmodem-db25

[1687] https://docs.freebsd.org/en/books/handbook/book/#nullmodem-db9

[1688] https://docs.freebsd.org/en/books/handbook/book/#nullmodem-db9-25

简称	全称
RD	收到的数据
TD	转送的数据
DTR	数据终端就绪
DSR	数据集准备就绪
DCD	数据载体检测
SG	信号地线
RTS	请求发送
CTS	清理发送

表 21. DB-25 到 DB-25 零调制解调器电缆

信号	编号#		编号#	信号
SG	7	连接到	7	SG
TD	2	连接到	3	RD
RD	3	连接到	2	TD
RTS	4	连接到	5	CTS
CTS	5	连接到	4	RTS
DTR	20	连接到	6	DSR
DTR	20	连接到	8	DCD
DSR	6	连接到	20	DTR
DCD	8	连接到	20	DTR

表 22. DB-9 到 DB-9 零调制解调器电缆

信号	编号#		编号#	信号
RD	2	连接到	3	TD
TD	3	连接到	2	RD
DTR	4	连接到	6	DSR
DTR	4	连接到	1	DCD
SG	5	连接到	5	SG
DSR	6	连接到	4	DTR
DCD	1	连接到	4	DTR
RTS	7	连接到	8	CTS
CTS	8	连接到	7	RTS

表 23. DB-9 到 DB-25 零调制解调器电缆

信号	编号#		编号#	信号
RD	2	连接到	2	TD
TD	3	连接到	3	RD
DTR	4	连接到	6	DSR
DTR	4	连接到	8	DCD
SG	5	连接到	7	SG
DSR	6	连接到	20	DTR
DCD	1	连接到	20	DTR
RTS	7	连接到	5	CTS
CTS	8	连接到	4	RTS

注意

当一端的一个引脚连接到另一端的一对引脚时，通常是在其连接器中的一对引脚之间用一根短线来实现，用一根长线来连接另一个单引脚。

串行端口是 FreeBSD 主机和终端之间进行数据传输的设备。有几种类型的串行端口。在购买或制作电缆之前，请确保它能够适应终端和 FreeBSD 系统上的端口。

大多数终端有 DB-25 端口。个人计算机可能有 DB-25 或 DB-9 端口。多端口串口卡可能有 RJ-12 或 RJ-45 端口。请参阅硬件附带的文档，了解关于端口种类的规格，或者目测端口的类型。

在 FreeBSD 中，每个串口都可以通过 **/dev** 中的一个条目来访问。有两种不同类型的条目：

- 调入端口被命名为 **/dev/ttyuN**，其中 *N* 是端口号，从 0 开始。如果一个终端被连接到第一个串口（**COM1**），使用 **/dev/ttyu0** 来指代该终端。如果终端在第二个串口（**COM2**）上，使用 **/dev/ttyu1**，以此类推。一般来说，呼入端口是用于终端的。呼入端口要求串行线断言“数据载波检测”信号以正确工作。
- 呼出端口端口在 FreeBSD 8.X 和更高版本中被命名为 **/dev/cuauN**，在 FreeBSD 7.X 和更低版本中被命名为 `/dev/cuadN`。Call-out 端口通常不用于终端，而是用于调制解调器。如果串行电缆或终端不支持“数据载波检测”信号，就可以使用呼出端口。

FreeBSD 还提供了初始化设备（**/dev/ttyuN.init** 和 **/dev/cuauN.init** 或 **/dev/cuadN.init**）和锁定设备（**/dev/ttyuN.lock** 和 **/dev/cuauN.lock** 或 **/dev/cuadN.lock**）。初始化设备用于在每次打开一个端口时初始化通信端口参数，例如用于使用 `RTS/CTS` 信令进行流量控制的调制解调器的 `crtscts`。锁定设备用于锁定端口的标志，以防止用户或程序改变某些参数。请参考 termios(4)[1689]，uart(4)[1690] 和 stty(1)[1691]，分别了解关于终端设置、锁定和初始化设备以及设置终端选项的信息。

[1689] https://www.freebsd.org/cgi/man.cgi?query=termios&sektion=4&format=html

[1690] https://www.freebsd.org/cgi/man.cgi?query=uart&sektion=4&format=html

[1691] https://www.freebsd.org/cgi/man.cgi?query=stty&sektion=1&format=html

29.2.2.串行端口配置

默认情况下，FreeBSD 支持四个串口，即通常所说的 **COM1**、**COM2**、**COM3** 和 **COM4**。FreeBSD 还支持非智能多端口串行接口卡，如 BocaBoard 1008 和 2016，以及更智能的多端口卡，如 Digiboard 制造的那些。然而，默认的内核只寻找标准的 **COM** 端口。

要想知道系统是否能识别串口，可以寻找以 `uart` 开头的系统启动信息：

```
# grep uart /var/run/dmesg.boot
```

如果系统不能识别所有需要的串口，可以在 **/boot/device.hints** 中添加额外的条目。这个文件已经包含了 **COM1** 的 `hint.uart.0.*` 条目和 **COM2** 的 `hint.uart.1.*` 条目。当添加 **COM3** 的端口条目时使用 `0x3E8`，而 **COM4** 使用 `0x2E8`。常见的 IRQ 地址 **COM3** 是 `5` 和 **COM4** 是 `9`。

要确定端口使用的默认终端 I/O 设置，请指定其设备名称。这个例子确定了 **COM2** 上的呼入端口的设置：

```
# stty -a -f /dev/ttyu1
```

串行设备的全系统初始化是由 **/etc/rc.d/serial** 控制的。这个文件会影响到串行设备的默认设置。要改变一个设备的设置，请使用 `stty`。默认情况下，改变后的设置在设备关闭前一直有效，当设备重新打开时，它又回到了默认设置。要永久地改变默认设置，请打开并调整初始化设备的设置。例如，要在 `CLOCAL` 模式、8 位通信和 `XON/XOFF` 流量控制下打开 **ttyu5**，请输入：

```
# stty -f /dev/ttyu5.init clocal cs8 ixon ixoff
```

为了防止某些设置被应用程序改变，对锁定设备进行调整。例如，要把 **ttyu5** 的速度锁定为 57600 bps，请输入：

```
# stty -f /dev/ttyu5.lock 57600
```

现在，任何打开 **ttyu5** 并试图改变端口速度的应用程序将被锁定在 57600 bps。

29.3.终端

终端提供了一种当不在计算机的控制台或连接的网络上时，可以访问 FreeBSD 系统的方便且低成本的方式。这一节介绍了如何在 FreeBSD 中使用终端。

最初的 UNIX® 系统并没有控制台。作为替代，用户通过连接到计算机串口的终端来登录和运行程序。

今天，几乎所有类 UNIX® 操作系统，包括 FreeBSD 都有在串口上建立登录会话的能力。通过使用连接在未使用的串口上的终端，用户可以登录并运行任何通常可以在控制台或 `xterm` 窗口中运行的文本程序。

许多终端可以连接到同一个 FreeBSD 系统上。可以将一台旧的备用计算机用作终端，连接到一台运行 FreeBSD 的更强大的计算机。这可以把本来是单用户的计算机变成一个强大的多用户系统。

FreeBSD 支持三种类型的终端：

哑终端

哑终端是通过串行线与计算机连接的专用硬件。它们被称为“哑巴”，因为它们的计算能力只够显示、发送和接收文本。没有程序可以在这些设备上运行。但是哑终端可以连接到运行所需程序的计算机。

有许多制造商生产的数百种哑终端，几乎所有的哑终端都可以在 FreeBSD 上工作。一些高级终端甚至可以显示图形，但只有某些软件可以利用这些高级功能。

哑终端在工作环境中很受欢迎，因为工人不需要访问图形应用程序。

充当终端的计算机

由于哑终端只有足够的能力来显示、发送和接收文本，任何备用计算机都可以成为哑终端。所需要的只是适当的电缆和一些在计算机上运行的终端仿真软件。

这种配置可能是有用的。例如，如果一个用户正忙于在 FreeBSD 系统的控制台工作，另一个用户可以从一台功率较小的个人计算机上作为终端连接到 FreeBSD 系统上，同时进行一些纯文本的工作。

在 FreeBSD 的基本系统中，至少有两个工具可以用来通过串行连接工作：cu(1)[1692] 和 tip(1)[1693]。

例如，从一个运行 FreeBSD 的客户系统连接到另一个系统的串行连接：

```
# cu -l /dev/cuauN
```

端口的编号从 0 开始。这意味着 **COM1** 是 **/dev/cuau0**。

其他的程序可以通过 ports 获得，例如 comms/minicom[1694]。

X 终端

X 终端是目前最复杂的一种终端。它们通常不连接至串行端口，而是连接到一个像以太网一样的网络。它们可以显示所有的 Xorg 应用程序，而不是局限于纯文本应用程序。

本章不包括 X 终端的设置、配置或使用。

29.3.1.终端配置

这一节说明了如何配置 FreeBSD 系统，使其能够在串行终端上进行登录会话。它假定系统能够识别终端所连接的串口，并且终端是用正确的电缆连接的。

在 FreeBSD 中，`init` 读取 **/etc/ttys** 并在可用终端上启动一个 `getty` 进程。`getty` 进程负责读取登录名并启动登录程序。FreeBSD 系统上允许登录的端口都列在 **/etc/ttys** 中。例如，第一个虚拟控制台，**ttyv0**，在这个文件中有一个条目，允许在该控制台登录(`login`)。这个文件还包含其他虚拟控制台、串行端口和伪 tty

[1692] https://www.freebsd.org/cgi/man.cgi?query=cu&sektion=1&format=html

[1693] https://www.freebsd.org/cgi/man.cgi?query=tip&sektion=1&format=html

[1694] https://cgit.freebsd.org/ports/tree/comms/minicom/pkg-descr

的条目。对于一个硬接线终端，串行端口的 **/dev** 条目被列出，但不包括 **/dev** 部分。例如，**/dev/ttyv0** 被列为 **ttyv0**。

默认的 **/etc/ttys** 配置了对前四个串口的支持，即 **ttyu0** 到 **ttyu3**：

```
ttyu0   "/usr/libexec/getty std.9600"   dialup  off secure
ttyu1   "/usr/libexec/getty std.9600"   dialup  off secure
ttyu2   "/usr/libexec/getty std.9600"   dialup  off secure
ttyu3   "/usr/libexec/getty std.9600"   dialup  off secure
```

当把一个终端连接到这些端口之一时，修改默认条目以设置所需的速度和终端类型，打开(on)设备，如果需要，改变端口的安全(secure)设置。如果终端连接到另一个端口，为该端口添加一个条目。

配置终端条目在 **/etc/ttys** 中配置了两个终端。第一个条目配置了一个连接到 **COM2** 的 Wyse-50。第二个条目配置了一台运行 Procomm 终端软件模拟 VT-100 终端的旧电脑。这台计算机连接到一个多端口串行卡上的第六个串行端口。

例 **43.** 配置终端入口

```
ttyu1  "/usr/libexec/getty std.38400"  wy50   on insecure
ttyu5   "/usr/libexec/getty std.19200"  vt100  on insecure
```

第一个字段指定了串行终端的设备名称。

第二个字段告诉 getty 初始化并打开线路，设置线路速度，提示用户名，然后执行登录（login）程序。可选的 getty 类型（*getty type*）配置了终端线路的特性，如 bps 速率和奇偶性。可用的 getty 类型在 **/etc/gettytab** 中列出。几乎在所有情况下，以 std 开头的 getty 类型都适用于硬接线终端，因为这些条目忽略了奇偶性。从 110 到 115200 的每个 bps 速率都有一个 std 条目。更多信息请参考 gettytab(5)[1695]。当设置 getty 类型时，确保与终端使用的通信设置相匹配。在这个例子中，Wyse-50 使用无奇偶校验，以 38400 bps 连接。计算机使用无奇偶校验，以 19200 bps 连接。

第三个字段是终端的类型。对于拨号端口，通常使用未知（unknown）或拨号（dialup），因为用户几乎可以用任何类型的终端或软件进行拨号。由于硬接线终端的终端类型不会改变，可以从 **/etc/termcap** 中指定一个真实的终端类型。在这个例子中，Wyse-50 使用真实的终端类型，而运行 Procomm 的计算机被设置为模拟 VT-100。

第四个字段指定该端口是否应被启用。要在这个端口上启用登录，这个字段必须设置为开（on）。

最后一个字段用于指定该端口是否安全（secure）。将一个端口标记为安全意味着它被信任到足以允许 root 从该端口登录。不安全的端口不允许 root 登录。在不安全的端口上，用户必须从非特权账户登录，然后使用 su 或类似的机制来获得超级用户的权限，如超级用户账户[1696]中所述。出于安全考虑，建议将此设置改为不安全（inscure）。

在对 **/etc/ttys** 进行任何修改后，向 init 进程发送一个 SIGHUP（挂起）信号，迫使它重新读取其配置文件：

[1695] https://www.freebsd.org/cgi/man.cgi?query=gettytab&sektion=5&format=html

[1696] https://docs.freebsd.org/en/books/handbook/basics/index.html#users-superuser

```
# kill -HUP 1
```

由于 `init` 总是第一个在系统上运行的进程，它的进程 ID 总是 1。

如果一切设置正确，所有电缆都已到位，终端也已通电，现在每个终端上都应该有一个 `getty` 进程在运行，每个终端上都应该有登录提示。

29.3.2.排除连接的故障

即使对细节有最细微的关注，在设置终端时仍可能出错。这里列出了一些常见的症状和一些建议的解决方法。

如果没有出现登录提示，请确保终端已插上插头并接通电源。如果是作为终端的个人电脑，确保它在正确的串行端口上运行终端仿真软件。

确保电缆与终端和 FreeBSD 计算机都连接牢固。确保它是正确的电缆。

确保终端和 FreeBSD 在 bps 速率和奇偶校验设置上一致。如果是视频显示终端，确保对比度和亮度控制被调高。如果是一个打印终端，确保纸张和墨水的供应充足。

使用 `ps` 确保 `getty` 进程正在运行并为终端服务。例如，下面的列表显示一个 `getty` 正在第二个串口 **ttyu1** 上运行，并使用 **/etc/gettytab** 中的 `std.38400` 条目：

```
# ps -axww|grep ttyu
22189  d1  Is+    0:00.03 /usr/libexec/getty std.38400 ttyu1
```

如果没有 `getty` 进程在运行，请确保该端口在 **/etc/ttys** 中被启用。记住在修改 **/etc/ttys** 后运行 `kill -HUP 1`。

如果 `getty` 进程正在运行，但终端仍然不显示登录提示，或者显示提示但不接受键入的输入，终端或电缆可能不支持硬件握手。试着将 **/etc/ttys** 中的条目从 `std.38400` 改为 `3wire.38400`，然后在修改 **/etc/ttys** 后运行 `kill -HUP 1`。`3wire` 条目与 `std` 类似，但忽略了硬件握手。在使用 `3wire` 时，可能还需要降低 bps 或启用软件流控制，以防止缓冲区溢出。

如果出现的是垃圾而不是登录提示，请确保终端和 FreeBSD 在 bps 速率和奇偶校验设置上达成一致。检查 `getty` 进程，确保使用了正确的 `getty` 类型。如果没有，编辑 **/etc/ttys** 并运行 `kill -HUP 1`。

如果字符出现双倍，并且输入密码时出现，请将终端或终端仿真软件从“半双工”（“half duplex”）或“本地回声”（“local echo”）切换到”全双工“（“full duplex”）。

29.4.拨入服务

配置 FreeBSD 系统的拨入服务与配置终端相似，只是使用调制解调器而不是终端设备。FreeBSD 同时支持外部和内部调制解调器。

外部调制解调器更方便，因为它们通常可以通过存储在非易失性内存中的参数进行配置，而且它们通常会提供显示重要 RS-232 信号状态的指示灯，以显示调制解调器是否正常工作。

内部调制解调器通常缺乏非易失性内存，因此其配置可能仅限于设置 DIP 开关。如果内部调制解调器有任何信号指示灯，当系统的盖子放在那里时，它们就很难看到。

当使用外部调制解调器时，需要一个合适的电缆。一条标准的 RS-232C 串行电缆应该足够了。

FreeBSD 需要 RTS 和 CTS 信号来实现 2400bps 以上的流量控制，需要 CD 信号来检测呼叫是否被接听或线路是否被挂断，还需要 DTR 信号来在一个会话结束后重置调制解调器。有些电缆在接线时没有提供所有需要的信号，所以如果登录会话在线路挂断后没有消失，可能是电缆有问题。关于这些信号的更多信息，请参考串行电缆和端口[1697]。

像其他类 UNIX® 操作系统一样，FreeBSD 使用硬件信号来发现呼叫被接听或线路被挂断，并在呼叫后挂断和重置调制解调器。FreeBSD 避免了向调制解调器发送命令或观察调制解调器的状态报告。

FreeBSD 支持基于 NS8250、NS16450、NS16550 和 NS16550A 的 RS-232C（CCITT V.24）通信接口。8250 和 16450 设备有单字符缓冲器。16550 设备提供了 16 个字符的缓冲区，可以获得更好的系统性能。普通的 16550 设备中的错误阻止了 16 字符缓冲器的使用，所以如果可能的话，使用 16550A 设备。由于单字符缓冲器设备比 16 字符缓冲器设备需要操作系统做更多的工作，所以基于 16550A 的串行接口卡是首选。如果系统有许多活动的串口或将有很重的负载，基于 16550A 的卡更适合于低误码率的通信。

本节的其余部分演示了如何配置调制解调器以接收传入的连接，如何与调制解调器通信，并提供了一些故障排除提示。

29.4.1.调制解调器配置

与终端一样，`init` 为每个配置的用于拨入连接的串行端口生成一个 `getty` 进程。当用户拨打调制解调器的线路并且调制解调器连接时，“载波检测”信号由调制解调器报告。内核注意到载波已被检测到，并指示 `getty` 打开端口并以指定的初始线速显示 `login:` 提示。在一个典型的配置中，如果收到垃圾字符，通常是由于调制解调器的连接速度与配置的速度不同，`getty` 会尝试调整线路速度，直到收到合理的字符。在用户输入他们的登录名后，`getty` 执行 `login`，它通过询问用户的密码来完成登录过程，然后启动用户的 shell。

关于拨号调制解调器，有两种方法。第一种配置方法是将调制解调器和系统设置为，无论远程用户以何种速度拨入，拨入的 RS-232 接口都以锁定的速度运行。这种配置的好处是，远程用户总是立即看到系统的登录提示。缺点是系统不知道用户的真实数据速率是多少，所以像 Emacs 这样的全屏程序不会调整它们的屏幕绘制方法，以使它们对较慢的连接有更好的响应。

第二种方法是配置 RS-232 接口，根据远程用户的连接速度来改变其速度。由于 `getty` 不了解任何特定的调制解调器的连接速度报告，它以一个初始速度给出一个 `login:` 消息，并观察响应回来的字符。如果用

[1697] https://docs.freebsd.org/en/books/handbook/book/#term-cables-null

户看到的是垃圾，他们应该按回车键，直到看到一个可识别的提示。如果数据速率不匹配，getty 会把用户输入的任何东西都视为垃圾，尝试下一个速度，并再次给出 login: 提示。这个过程通常只需要按一两个键，用户就能看到一个好的提示。这个登录顺序看起来不像锁定速度的方法那样干净，但是在低速连接上的用户应该从全屏程序中得到更好的交互式响应。

当把调制解调器的数据通信速率锁定在一个特定的速度时，应该不需要改变 **/etc/gettytab**。然而，对于匹配速度的配置，可能需要额外的条目，以定义调制解调器的速度。这个例子配置了一个 14.4Kbps 的调制解调器，最高接口速度为 19.2Kbps，使用 8 位无奇偶校验连接。它将 getty 配置为以 19.2Kbps 开始 V.32bis 连接的通信速率，然后在 9600 bps、2400 bps、1200 bps、300 bps 之间循环，最后回到 19.2Kbps。通信速率循环是通过 nx=（下表）功能实现的。每一行都使用一个 tc=（表的延续）条目来获取特定数据速率的其余设置：

```
#
# 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:
```

对于 28.8Kbps 的调制解调器，或者为了利用 14.4Kbps 调制解调器的压缩优势，请使用较高的通信速率，如本例所示：

```
#
# 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:
```

对于低性能的 CPU 或者没有基于 16550A 的串行端口的重负荷系统，这种配置可能会在 57.6Kbps 的速度下产生 uart “silo” 错误。

/etc/ttys 的配置与配置终端条目相似，但传递给 `getty` 的参数不同，终端类型使用 `dialup`。用 `init` 将在设备上运行的进程替换 *xxx*：

```
ttyu0   "/usr/libexec/getty xxx"   dialup on
```

拨号（`dialup`）终端类型可以被改变。例如，将 `vt102` 设置为默认的终端类型，允许用户在其远程系统上使用 `VT102` 模拟。

对于锁定速度的配置，用 **/etc/gettytab** 中列出的有效类型指定速度。这个例子是针对一个端口速度被锁定在 19.2Kbps 的调制解调器：

```
ttyu0   "/usr/libexec/getty std.19200"   dialup on
```

在一个匹配速度的配置中，该条目需要参考 **/etc/gettytab** 中适当的开头 `auto-baud` 条目。要继续执行匹配速度调制解调器的例子，即从 19.2Kbps 开始，使用这个条目：

```
ttyu0   "/usr/libexec/getty V19200"   dialup on
```

编辑完 **/etc/ttys** 后，等待调制解调器被正确配置和连接，然后向 `init` 发出信号:

```
# kill -HUP 1
```

高速调制解调器，如 V.32、V.32bis 和 V.34 调制解调器，使用硬件（`RTS/CTS`）流量控制。使用 `stty` 来设置调制解调器端口的硬件流控制标志。这个例子在 **COM2** 的拨入和拨出初始化设备上设置 `crtscts` 标志:

```
# stty -f /dev/ttyu1.init crtscts
# stty -f /dev/cuau1.init crtscts
```

29.4.2.故障排除

这一节提供了一些排除拨号调制解调器无法连接到 FreeBSD 系统的故障的提示。

将调制解调器连接到 FreeBSD 系统上并启动系统。如果调制解调器有状态指示灯，观察当系统的控制台出现 `login:` 提示时，调制解调器的 DTR 指示灯是否亮起。如果它亮了，这应该意味着 FreeBSD 已经在相应的通信端口上启动了一个 `getty` 进程，并且正在等待调制解调器接受呼叫。

如果 DTR 指示灯不亮，通过控制台登录到 FreeBSD 系统，并键入 `ps ax` 来查看 FreeBSD 是否在正确的端口上运行 `getty` 进程：

```
114 ??  I      0:00.10 /usr/libexec/getty V19200 ttyu0
```

如果第二列包含 d0 而不是 ??，并且调制解调器还没有接受呼叫，这意味着 getty 已经完成了对通信端口的打开。这可能表明布线有问题或调制解调器配置错误，因为在调制解调器发出载波检测信号之前，getty 不应该能够打开通信端口。

如果没有 getty 进程等待打开该端口，请仔细检查 **/etc/ttys** 中该端口的条目是否正确。另外，检查 **/var/log/messages**，看是否有任何来自 init 或 getty 的日志信息。

接下来，试着拨号进入系统。确保在远程系统上使用 8 位，无奇偶校验，和 1 个停止位。如果没有马上出现提示，或者提示显示为垃圾，试着每秒钟按一次回车键。如果仍然没有 login: 提示，尝试发送 BREAK。当使用高速调制解调器时，在锁定拨号调制解调器的接口速度后再尝试拨号。

如果仍然没有 login: 提示，再次检查 **/etc/gettytab**，仔细检查：

- 在 **/etc/ttys** 中的条目所指定的初始适配器名称与 **/etc/gettytab** 中的适配器名称相符。
- 每个 nx= 条目与另一个 gettytab 适配器名称相匹配。
- 每个 tc= 条目与另一个 gettytab 适配器名称相匹配。

如果 FreeBSD 系统上的调制解调器不响应，请确保调制解调器被配置为在 DTR 被断言时接电话。如果调制解调器似乎配置正确，通过检查调制解调器的指示灯来验证 DTR 线是否被确认。

如果仍然不能工作，试着向 FreeBSD 一般问题邮件列表[1698]发送电子邮件，描述调制解调器的问题。

29.5.拨出服务

以下是一些让主机通过调制解调器连接到另一台计算机的技巧。这适用于与远程主机建立终端会话。

如果使用 PPP 有问题，这种连接对在互联网上获取文件有帮助。如果 PPP 不工作，使用终端会话来通过 FTP 获得所需文件。然后使用 zmodem 将其传输到机器上。

29.5.1.使用内置的 Hayes 调制解调器

tip 中内置了一个通用的 Hayes 拨号器。在 **/etc/remote** 中使用 at=hayes。

驱动 Hayes 不够聪明，不能识别新调制解调器的一些高级功能，如 BUSY、NO DIALTONE 或 CONNECT 115200 等信息。当用 ATX0&W 使用 tip 时，要关闭这些信息。

tip 的拨号超时是 60 秒。调制解调器应使用更少的时间，否则 tip 会认为有通信问题。试试 ATS7=45&W。

[1698] https://lists.freebsd.org/subscription/freebsd-questions

29.5.2.使用 AT 命令

在 **/etc/remote** 中创建一个“direct”条目。例如，如果调制解调器被连接到第一个串口，**/dev/cuau0**，使用下面一行：

```
cuau0:dv=/dev/cuau0:br#19200:pa=none
```

在 br 适配器中使用调制解调器支持的最高 bps 速率。然后，输入 `tip cuau0` 来连接到调制解调器。

或者，以 `root` 身份用以下命令使用 `cu：

```
# cu -lline -sspeed
```

line 是串口，如 **/dev/cuau0**，*speed* 是速度，如 `57600`。当输入完 AT 命令后，输入 `~.` 退出。

29.5.3.@ 符号不起作用

电话号码功能中的 `@` 符号告诉 `tip` 在 **/etc/phones** 中寻找电话号码。但是，`@` 符号在 **/etc/remote** 等配置文件中也是一个特殊字符，所以它需要用反斜杠转义：

```
pn=\@
```

29.5.4.从命令行拨号

在 **/etc/remote** 中添加一个“generic”条目，例如：

```
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:
```

这样就应该能工作了：

```
# tip -115200 5551234
```

喜欢 `cu` 而不是 `tip` 的用户，可以使用一个通用的 `cu` 条目：

```
cu115200|Use cu to dial any number at 115200bps:\
        :dv=/dev/cuau1:br#57600:at=hayes:pa=none:du:
```

然后输入：

```
# cu 5551234 -s 115200
```

29.5.5.设置 bps 速率

添加一项 tip1200 或 cu1200，并将 bps 速率换成更合适的值。tip 的默认值是 1200 bps，也就是为什么会有 tip1200 这条记录的原因。虽然你并不需要使用 1200 bps。

29.5.6.通过终端服务器访问若干主机

与其等到连接后每次都输入 CONNECT 主机，不如使用 tip 的 cm 功能。例如，**/etc/remote** 中的这些条目将让你输入 tip pain 或 tip muffin 来连接到主机 pain 或 muffin，输入tip deep13 来连接到终端服务器：

```
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:
```

29.5.7.使用不止一条线路的 tip

当一所大学只有几条调制解调器线路，以及数千名学生试图使用它们时，这通常是一个问题。

在 **/etc/remote** 中为你的大学添加一个记录，然后为 pn 功能使用 @ 标记：

```
big-university:\
        :pn=\@:tc=dialout
dialout:\
        :dv=/dev/cuau3:br#9600:at=courier:du:pa=none:
```

接着，在 **/etc/phones** 中列出大学的电话号码：

```
big-university 5551111
big-university 5551112
big-university 5551113
big-university 5551114
```

tip 将按顺序试用每一个，然后就停止。如果想继续测试，隔一段时间再运行 tip。

29.5.8.使用强制字符

Ctrl+P 是默认的“强制”字符，用来告诉 tip 下一个字符是字面数据。强制字符可以用 ~s 转义设置为任何其他字符，意思是“设置一个变量”。

输入 ~sforce=single-char，后面加一个换行。如果不输入单字符，那么强制字符就是空字符，可以通过输入 Ctrl + 2 或 Ctrl + 空格键进入。一个相当好的单字符值是 Shift + Ctrl + 6，它只在一些终端服务器上使用。

要改变强制字符，在 ~/.tiprc 中指定以下内容：

```
force=single-char
```

29.5.9.大写字符

这发生在按下 Ctrl+A 的时候，这是 tip 的“大写字符”，专门为有大写锁定键的人设计。使用 ~s 来设置 raisechar 到合理的位置。如果这两个功能都不使用，它可以被设置为与强制字符相同。

这里是为需要输入 Ctrl + 2 和 Ctrl + A 的 Emacs 用户提供的 ~/.tiprc 样本:

```
force=^^
raisechar=^^
```

^^ 是 Shift + Ctrl + 6

29.5.10.用 tip 来传输文件

当与另一个类 UNIX® 操作系统通信时，可以使用 ~p（put）和 ~t（take）发送和接收文件。这些命令在远程系统上运行 cat 和 echo 来接受和发送文件。语法是：~p 本地文件 [远程文件] ~t 远程文件 [本地文件]

它没有错误检查，所以可能应该使用其他协议，如 zmodem。

29.5.11.用 tip 来使用 Zmodem？

要接收文件，在远程端启动发送程序。然后，键入~C rz，开始在本地接收它们。

要发送文件，在远程端启动接收程序。然后，键入 ~C sz 文件，将它们发送到远程系统。

29.6.设置串行控制台

FreeBSD 有能力在启动系统的时候将一个哑终端放在串口上作为控制台。这种配置对于希望在没有键盘或显示器的机器上安装 FreeBSD 的系统管理员，以及希望调试内核或设备驱动程序的开发人员来说非常有用。

正如在FreeBSD 引导过程[1699]中所述，FreeBSD 的引导过程采用了三个阶段。前两个阶段是在引导块代码中，它被存储在引导盘上的 FreeBSD slice 的开头。然后，引导块加载并运行作为第三阶段代码的引导加载器。

为了设置从串行控制台启动，需要配置引导块代码、引导加载器代码和内核。

29.6.1.快速配置串口控制台

本节提供了一个关于设置串行控制台的快速概述。这个程序可以在哑终端连接到 **COM1** 时使用。

在 **COM1** 上配置串行控制台的过程

1. 将串行电缆连接到 **COM1** 和控制终端。
2. 要配置在串行控制台显示的启动信息，以超级用户身份执行执行命令：

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

3. 编辑 **/etc/ttys**，将 **ttyu0** 条目中的 `off` 改为 `on`，`dialup` 改为 `vt100`。否则，通过串行控制台连接时将不需要密码，从而导致一个潜在的安全漏洞。
4. 重启系统来检查这些变更是否起作用

如果需要不同的配置，请看下一节更深入的配置解释。

29.6.2.深入的串行控制台配置

本节将对在 FreeBSD 中设置串行控制台所需的步骤进行更详细的解释。

配置串口控制台的过程

1. 准备一根串口线缆。
   你需要使用一个 null-modem 的线缆或标准的串口线和一个零调试解调器适配器。请参考线缆和端口[1700] 中有关串口线的讨论。
2. 拔掉键盘。
   绝大多数的电脑在开机检测的时候会对键盘进行检测，如果没有检测到键盘，则会出现错误。一些机器会提示缺少键盘，就不会继续引导系统。
   如果你的计算机出现错误，但仍能继续启动，你可以不必理它。
   如果你的计算机没有键盘就拒绝启动，那你需要配置 BIOS 来避免它。请参考你的主板的使用说明了解更多细节。

[1699] https://docs.freebsd.org/en/books/handbook/boot/index.html#boot

[1700] https://docs.freebsd.org/zh-cn/books/handbook/serialcomms/#serial-cables-ports

技巧

尝试在 BIOS 中将键盘设为“Not installed”。这个设置告诉 BIOS 在上电时不要探测键盘，这样如果键盘不在，它就不会发出警告。如果该选项在 BIOS 中不存在，则寻找“Halt on Error”选项。将此设置为“All but Keyboard”或“No Errors”将具有相同的效果。

如果系统有 PS/2 鼠标，如果幸运的话，你也可以象键盘一样把它拔下来，这是因为 PS/2 鼠标与键盘的一些硬件是共享的，你的鼠标插上去，系统会认为键盘也仍然存在。

注意

虽然大多数系统可以在没有键盘的情况下启动，但相当多的系统在没有图形适配器的情况下无法启动。一些系统可以通过将 BIOS 配置中的“graphics adapter”设置改为“Not installed”来配置成在没有图形适配器的情况下启动。其他系统不支持这个选项，如果系统中没有显示硬件，将拒绝启动。对于这些机器，要将显卡的接口接入，即使它只是一块废旧的单声道板。不需要连接显示器。

3. 将一个哑终端、一台带有调制解调器程序的旧电脑或另一个 UNIX® box 上的串口插入串口。
4. 在 **/boot/device.hints** 中为串口添加适当的 `hint.uart.*` 条目。一些多端口卡也需要内核配置选项。请参考 uart[4][1701] 的手册页面来了解每个支持的串口所需的选项和设备提示。
5. 在引导驱动器的 a 分区的根目录中创建 **boot.config**。

   这个文件指示引导块代码如何引导系统。为了激活串行控制台，需要下列一个或多个选项。当使用多个选项时，把它们都放在同一行：

   -h

   在内部控制台和串行控制台之间进行切换。用它来切换控制台设备。例如，要从内部（视频）控制台引导，使用 `-h` 来指示引导加载器和内核使用串行端口作为控制台设备。或者，如果要从串口引导，使用 `-h`来告诉引导加载器和内核使用视频显示器作为控制台。

   -D

   在单控制台和双控制台的配置之间进行切换。在单配置中，控制台将是内部控制台（视频显示器）或串行端口，取决于 `-h` 的状态。在双控制台配置中，无论 `-h` 的状态如何，视频显示器和串口将同时成为控制台。然而，双控制台的配置只在引导块运行时生效。引导加载器得到控制后，由 `-h` 指定的控制台就成为唯一的控制台。

   -P

   使得引导块探测键盘。如果没有发现键盘，`-D` 和 `-h` 选项会自动设置。

   注意

   由于当前版本的引导块的空间限制，`-P` 只能检测到扩展键盘。少于 101 个键的键盘和没有 F11 和 F12 键的键盘可能无法被检测到。由于这个限制，一些笔记本电脑上的键盘可能无法被正确发现。如果是这种情况，请不要使用 `-P`。

   使用 `-P` 来自动选择控制台，或者使用 `-h` 来激活串行控制台。更多细节请参考 boot(8)[1702] 和 boot.config(5)[1703] 的手册页面。

   除了 `-P` 之外，其他的选项都被传递给引导加载器。引导加载器将通过检查 `-h` 的状态来决定是内部视频还是串行端口成为控制台。这意味着如果在 **/boot.config** 中指定了 `-D`，但是没有指定

[1701] https://www.freebsd.org/cgi/man.cgi?query=uart&sektion=4&format=html

[1702] https://www.freebsd.org/cgi/man.cgi?query=boot&sektion=8&format=html

[1703] https://www.freebsd.org/cgi/man.cgi?query=boot.config&sektion=5&format=html

-h，那么串口只能在引导块中作为控制台使用，因为 Boot Loader 会使用内部视频显示作为控制台。

1. 引导机器

   当 FreeBSD 引导时，引导块会将 **/boot.config** 的内容回传到控制台，比如说：

```
/boot.config: -P
Keyboard: no
```

   第二行只有在 **/boot.config** 中的 -P 出现时才会出现，并表明键盘的存在或不存在。这些信息要么发送到串行控制台，要么发送到内部控制台，或者同时发送到两者，这取决于 **/boot.config** 中的选项。

参数	送出消息的设备
none	内部控制台
-h	串口控制台
-D	串口控制台和内部控制台
-Dh	串口控制台和内部控制台
-P，有键盘	内部控制台
-P，无键盘	串口控制台

   出现上面信息后，在引导块加载引导器和更多信息显示到屏幕之前将有一个小小的停顿。在通常情况下，你不需要打断引导进程，但为了确信设置是否正确，你也可以这样做。
   在控制台上按回车键以外的任意键就能打断引导进程。引导块将进入命令行模式。你将看到：

```
>> FreeBSD/i386 BOOT
Default: 0:ad(0,a)/boot/loader
boot:
```

   检验上面出现的信息，可能是串口，或内部控制台，或两个同时，完全取决于你在 **/boot.config** 中的参数。如果信息出现在正确的控制台，按回车键继续引导进程。
   如果你要使用串口控制台，但你没有看到命令行，那可能设置有问题。这时，输入 -h 然后按 回车键或 Return 来告诉引导块（然后是引导加载器和内核）选择串口作为控制台。系统起来了之后，就可以回去检查一下是什么出了问题。

在引导过程的第三阶段，人们仍然可以通过在引导加载器中设置适当的环境变量在内部控制台和串行控制台之间切换。参见 loader(8)[1704] 以了解更多信息。

注意

在 **/boot/loader.conf** 或 **/boot/loader.conf.local** 中的这一行将配置 Boot Loader 和内核将他们的引导信息发送到串行控制台，而不考虑 **/boot.config** 中的选项。

[1704] https://www.freebsd.org/cgi/man.cgi?query=loader&sektion=8&format=html

```
console="comconsole"
```

这一行应该是 **/boot/loader.conf** 的第一行，这样引导信息就会尽早显示在串行控制台。

如果这一行不存在，或者它被设置为 `console="vidconsole"`，那么引导加载器和内核将使用引导块中由 `-h` 指示的控制台。参见 loader.conf(5)[1705] 了解更多信息。

目前，引导加载器在引导块中没有相当于 `-P` 的选项，也没有规定根据键盘的存在自动选择内部控制台和串行控制台。

技巧

虽然这不是必须的，但可以通过串行线提供一个 `login` 提示。要配置这一点，请使用终端配置[1706]中的说明编辑 **/etc/ttys** 中的串口条目。如果串口的速度已经改变，请改变 `std.9600` 以符合新的设置。

29.6.3.设置一个更快的串行端口速度

默认情况下，串行端口设置为 9600 波特、8 位、无奇偶校验和 1 个停止位。要改变默认的控制台速度，请使用下列选项之一：

- 编辑 **/etc/make.conf** 并将 `BOOT_COMCONSOLE_SPEED` 设置为新的控制台速度。然后，重新编译并安装引导块和引导加载器：

```
# cd /sys/boot
# make clean
# make
# make install
```

如果串行控制台是以其他方式配置的，而不是用 `-h` 引导，或者内核使用的串行控制台与引导块使用的不同，可以在定制内核配置文件中加入下列选项，并加上所需的速度，然后编译一个新的内核：

```
options CONSPEED=19200
```

- 在 **/boot.config** 中添加 `-S19200` 引导选项，用要使用的速度代替 19200。
- 在 **/boot/loader.conf** 中添加以下选项。用要使用的速度替换 115200：

```
boot_multicons="YES"
boot_serial="YES"
comconsole_speed="115200"
console="comconsole,vidconsole"
```

[1705] https://www.freebsd.org/cgi/man.cgi?query=loader.conf&sektion=5&format=html

[1706] https://docs.freebsd.org/en/books/handbook/book/#term-config

29.6.4.从串行线进入 DDB 调试器

要配置从串行控制台进入内核调试器的能力，请在定制内核配置文件中加入以下选项，并按照配置 FreeBSD 内核[1707]的说明编译内核。注意，虽然这对远程诊断很有用，但如果在串口上产生虚假的 BREAK，也是很危险的。请参考 ddb(4)[1708] 和 ddb(8)[1709] 以了解更多关于内核调试器的信息：

```
options BREAK_TO_DEBUGGER
options DDB
```

[1707] https://docs.freebsd.org/en/books/handbook/kernelconfig/index.html#kernelconfig

[1708] https://www.freebsd.org/cgi/man.cgi?query=ddb&sektion=4&format=html

[1709] https://www.freebsd.org/cgi/man.cgi?query=ddb&sektion=8&format=html

第三十章 PPP

30.1.概述

FreeBSD 支持点对点协议（PPP），可以使用拨号调制解调器来建立网络或互联网连接。本章说明了如何在 FreeBSD 中配置基于调制解调器的通信服务。

读完本章后，你将知道：

- 如何配置、使用和排除 PPP 连接的故障。
- 如何设置基于以太网的 PPP（PPPoE）。
- 如何设置基于 ATM 的 PPP（PPPoA）。

在阅读本章之前，你应该：

- 熟悉基本的网络术语。
- 理解拨号连接和 PPP 的基本知识和目的。

30.2.配置 PPP

FreeBSD 提供了对使用 ppp(8)[1710] 管理拨号 PPP 连接的内置支持。默认的 FreeBSD 内核提供了对 **tun** 的支持，它被用来与调制解调器硬件交互。配置是通过编辑至少一个配置文件来完成的，并且提供了包含实例的配置文件。最后，`ppp` 被用来启动和管理连接。

为了使用 PPP 连接，需要以下物品：

- 一个互联网服务提供商（ISP）的拨号账户。
- 一个拨号调制解调器。
- ISP 的拨号号码。

[1710] https://www.freebsd.org/cgi/man.cgi?query=ppp&sektion=8&format=html

- 由 ISP 分配的登录名和密码。
- 一个或多个 DNS 服务器的 IP 地址。通常情况下，ISP 会提供这些地址。如果没有，FreeBSD 可以被配置为使用 DNS 协商。

如果缺少任何所需的信息，请联系 ISP。

以下信息可能由 ISP 提供，但不是必须的：

- 默认网关的 IP 地址。如果这个信息未知，ISP 会在连接设置时自动提供正确的值。在 FreeBSD 上配置 PPP 时，这个地址被称为 `HISADDR`。
- 子网掩码。如果 ISP 没有提供，则在 ppp(8)[1711] 配置文件中会使用 `255.255.255.255`。

  如果 ISP 已经分配了一个静态 IP 地址和主机名，它应该被输入到配置文件中。否则，这些信息将在连接设置时自动提供。

本节的其余部分将演示如何为常见的 PPP 连接情况配置 FreeBSD。所需的配置文件是 **/etc/ppp/ppp.conf**，其他文件和例子可以在 **/usr/share/examples/ppp/** 中找到。

> 注意
>
> 在本节中，许多文件的例子都显示行号。这些行号是为了更容易跟上讨论而添加的，并不意味着要放在实际文件中。
>
> 当编辑一个配置文件时，正确的缩进是很重要的。以 ":" 结尾的行从第一列开始（行的开头），而所有其他行应使用空格或制表符缩进，如图所示。

30.2.1.基本配置

为了配置 PPP 连接，首先用 ISP 的拨入信息编辑 **/etc/ppp/ppp.conf**。该文件说明如下。

```
1     default:
2       set log Phase Chat LCP IPCP CCP tun command
3       ident user-ppp VERSION
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 timeout 300
```

(continues on next page)

[1711] https://www.freebsd.org/cgi/man.cgi?query=ppp&sektion=8&format=html

(continued from previous page)

```
16      set ifaddr x.x.x.x/0 y.y.y.y/0 255.255.255.255 0.0.0.0
17      add default HISADDR
```

第 1 行

标明 `default` 条目。该条目中的命令（第 2 行到第 9 行）在 `ppp` 运行时自动执行。

第 2 行

启用用于测试连接的粗略日志参数。若配置工作令人满意，这一行应该被减少为：

```
set log phase tun
```

第 3 行

向连接的另一端运行的 PPP 软件显示 ppp(8)[1712] 的版本。

第 4 行

确定调制解调器所连接的设备，其中 **COM1** 是 **/dev/cuau0**，**COM2** 是 **/dev/cuau1**。

第 5 行

设置连接速度。如果 `115200` 在老式调制解调器上不起作用，可以尝试用 `38400` 代替。

第 6、7 行

写为期望——发送语法的拨号字符串。更多信息请参考 chat(8)[1713]。

注意，为了便于阅读，这个命令会延续到下一行。**ppp.conf** 中的任何命令都可以这样做，如果该行的最后一个字符是 `\`。

第 8 行

设置链接的空闲超时，单位是秒。

第 9 行

指示对等体确认 DNS 设置。如果本地网络正在运行自己的 DNS 服务器，这一行应该被注释掉，在该行的开头加上 `#`，或者删除。

第 10 行

为了便于阅读，这是一个空行。空白行会被 ppp(8)[1714] 忽略。

第 11 行

识别了一个名为 `provider` 的条目。这可以改成 ISP 的名字，这样就可以用 `load` *ISP* 来启动连接。

第 12 行

[1712] https://www.freebsd.org/cgi/man.cgi?query=ppp&sektion=8&format=html

[1713] https://www.freebsd.org/cgi/man.cgi?query=chat&sektion=8&format=html

[1714] https://www.freebsd.org/cgi/man.cgi?query=ppp&sektion=8&format=html

使用 ISP 的电话号码。可以使用冒号（:）或管道字符（|）作为分隔符来指定多个电话号码。要轮流拨打这些号码，请使用冒号。要总是尝试先拨第一个号码，只有在第一个号码失败时才使用其他号码，使用管道字符。始终用引号（“）括住整组电话号码，以防止拨号失败。

第 13、14 行

使用 ISP 的用户名和密码。

第 15 行

设置连接的默认空闲超时，单位为秒。在这个例子中，连接将在 300 秒的不活动后自动关闭。要防止超时，将此值设为零。

第 16 行

设置接口地址。使用的值取决于是否从 ISP 那里获得了静态 IP 地址，或者在连接过程中是否协商了一个动态 IP 地址。

如果 ISP 已经分配了一个静态 IP 地址和默认网关，用静态 IP 地址替换 *x.x.x.x*，用默认网关的 IP 地址替换 *y.y.y.y*。如果 ISP 只提供了一个静态 IP 地址而没有网关地址，则用 `10.0.0.2/0` 替换 *y.y.y.y.y*。

如果每次连接时 IP 地址都会改变，请将这一行改为以下值。这告诉 ppp(8)[1715] 使用 IP 配置协议(IPCP)来协商一个动态 IP 地址：

```
set ifaddr 10.0.0.1/0 10.0.0.2/0 255.255.255.255 0.0.0.0
```

第 17 行

保持这一行不变，因为它为网关添加了一个默认路由。`HISADDR` 将自动被替换成第 16 行指定的网关地址。重要的是这一行出现在第 16 行之后。

根据是手动还是自动启动 ppp(8)[1716]，可能还需要创建一个 **/etc/ppp/ppp.linkup**，其中包含以下几行。在 `-auto` 模式下运行 `ppp` 时，需要这个文件。这个文件在连接建立后使用。在这一点上，IP 地址已经被分配，现在可以添加路由表项。在创建这个文件时，确保 *provider* 与 **ppp.conf** 第 11 行中显示的值一致。

```
provider:
      add default HISADDR
```

在静态 IP 地址配置中，如果默认网关地址是“猜测”的，也需要这个文件。在这种情况下，从 **ppp.conf** 中删除第 17 行，用上述两行创建 **/etc/ppp/ppp.linkup**。这个文件的更多例子可以在 **/usr/share/examples/ppp/** 中找到。

默认情况下，`ppp` 必须以 `root` 身份运行。要改变这个默认值，将运行 `ppp` 的用户的账户添加到 **/etc/group** 中的 `network` 组。

然后，使用 `allow` 让该用户可访问 **/etc/ppp/ppp.conf** 中的一个或多个条目。例如，要给 `fred` 和 `mary` 只访问 `provider:` 条目的权限，在 `provider:` 部分添加这一行：

[1715] https://www.freebsd.org/cgi/man.cgi?query=ppp&sektion=8&format=html

[1716] https://www.freebsd.org/cgi/man.cgi?query=ppp&sektion=8&format=html

```
allow users fred mary
```

要让指定的用户访问所有条目，就把这一行放在 `default` 部分。

30.2.2.高级配置

可以将 PPP 配置为按需提供 DNS 和 NetBIOS 域名服务器地址。

要在 PPP 1.x 版本中启用这些扩展，可以在 **/etc/ppp/ppp.conf** 的相关部分添加以下行：

```
enable msext
set ns 203.14.100.1 203.14.100.2
set nbns 203.14.100.5
```

而对于 PPP 第 2 版及以上：

```
accept dns
set dns 203.14.100.1 203.14.100.2
set nbns 203.14.100.5
```

这将告诉客户主要和次要的域名服务器地址，以及一个 NetBIOS 域名服务器主机。

在第 2 版及以上，如果省略 `set dns` 一行，PPP 将使用在 **/etc/resolv.conf** 中找到的值。

30.2.2.1.PAP 和 CHAP 认证

一些 ISP 将他们的系统设置为使用 PAP 或 CHAP 认证机制完成连接的认证部分。如果是这种情况，ISP 不会在连接时给出 `login:` 提示，而是立即开始 PPP 对话。

PAP 不如 CHAP 安全，但安全在这里通常不是问题，因为密码虽然在 PAP 中以纯文本形式发送，但只在串行线路上传输。破解者没有太多的空间来“窃听”。

必须做以下改动：

```
13      set authname MyUserName
14      set authkey MyPassword
15      set login
```

第 13 行

这一行指定了 PAP/CHAP 用户名。输入 `MyUserName` 的正确值。

第 14 行

这一行指定了 PAP/CHAP 密码。输入 `MyPassword` 的正确值。你可能想增加一行，例如：

```
16      accept PAP
```

或

```
16      accept CHAP
```

来表明这是你的意图，但 PAP 和 CHAP 都是默认接受的。

第 15 行

在使用 PAP 或 CHAP 时，ISP 通常不会要求登录到服务器。因此，禁用“set login”字符串。

30.2.2.2.使用 PPP 网络地址转换能力

PPP 具有使用内部 NAT 的能力，没有内核转移功能。这一功能可以通过 **/etc/ppp/ppp.conf** 中的以下一行启用：

```
nat enable yes
```

或者，可以通过命令行选项 `-nat` 启用 NAT。还有 **/etc/rc.conf** 中名为 `ppp_nat` 的选项，默认是启用的。

当使用这个特性时，可以使用下面的 **/etc/ppp/ppp.conf** 选项来启用传入连接转发：

```
nat port tcp 10.0.0.2:ftp ftp
nat port tcp 10.0.0.2:http http
```

或者完全不信任外部的连接

```
nat deny_incoming yes
```

30.2.3.最终的系统配置

虽然 `ppp` 现在已经配置好了，但仍然需要对 **/etc/rc.conf** 进行一些编辑。

在这个文件中，从上往下工作，确保 `hostname=` 一行被设置：

```
hostname="foo.example.com"
```

如果 ISP 提供了一个静态的 IP 地址和名称，请使用这个名称作为主机名。

找出 `network_interfaces` 变量。要配置系统按需拨号给 ISP，确保 **tun0** 设备被添加到列表中，否则将其删除。

```
network_interfaces="lo0 tun0"
ifconfig_tun0=
```

注意

变量 ifconfig_tun0应该为空，并且应该创建一个名为 **/etc/start_if.tun0** 的文件。这个文件应该包含这一行：

```
ppp -auto mysystem
```

这个脚本在网络配置时被执行，在自动模式下启动 ppp 守护进程。如果这台机器作为一个网关，可以考虑包括 -alias。更多细节请参考手册页面。

确保路由器程序被设置为 NO，在 **/etc/rc.conf** 中加入以下一行：

```
router_enable="NO"
```

重要的是，不要启动 routed 守护进程，因为 routed 倾向于删除 ppp 创建的默认路由表项。

确保 sendmail_flags 一行不包括 -q 选项可能是个好主意，否则 sendmail 会不时地尝试进行网络查询，可能会导致你的机器拨出。你可以试试：

```
sendmail_flags="-bd"
```

缺点是，每当 ppp 链接时，sendmail 都被迫重新检查邮件队列。为了自动处理这个问题，可以在 **ppp.linkup** 中加入 !bg：

```
1     provider:
2       delete ALL
3       add 0 0 HISADDR
4       !bg sendmail -bd -q30m
```

另一种方法是设置一个“dfilter”来阻止 SMTP 流量。更多细节请参考 sample 文件。

30.2.4.使用 ppp

剩下的就是重启机器了。重启后，可以输入：

```
# ppp
```

然后 dial provider 来启动 PPP 会话，或者，要配置 ppp 在有出站流量且 **start_if.tun0** 不存在时自动建立会话，请输入：

```
# ppp -auto provider
```

当 ppp 程序在后台运行时，有可能与它通信，但前提是已经设置了一个合适的诊断端口。要做到这一点，在配置中添加以下一行：

```
set server /var/run/ppp-tun%d DiagnosticPassword 0177
```

这将告诉 PPP 监听指定的 UNIX® 域套接字，在允许访问之前要求客户提供指定的密码。名称中的 %d 被替换为正在使用的 **tun** 设备号。

设置了一个套接字之后，就可以在希望操纵运行中的程序的脚本中使用 pppctl(8)[1717] 程序。

30.2.5.配置拨入服务

“拨号服务”[1718]提供了关于使用 getty(8)[1719] 启用拨号服务的良好说明。

getty 的另一个选择是 comms/mgetty+sendfax port[1720]，这是 getty 的一个更智能的版本，是为拨号线路设计的。

使用 mgetty 的好处是它能主动与调制解调器对话，这意味着如果在 **/etc/ttys** 中关闭端口，那么调制解调器就不会接电话。

mgetty 的后期版本（从 0.99beta 开始）还支持 PPP 流的自动检测，允许客户端无脚本地访问服务器。

关于 mgetty 的更多信息，请参考 http://mgetty.greenie.net/doc/mgetty_toc.html。

默认情况下，comms/mgetty+sendfax[1721] 端口启用了 AUTO_PPP 选项，允许 mgetty 检测 PPP 连接的 LCP 阶段并自动生成 ppp shell。然而，由于默认的登录/密码序列没有发生，所以有必要使用 PAP 或 CHAP 来验证用户。

本节假设用户已经成功编译，并在其系统上安装了 port comms/mgetty+sendfax[1722]。

确保 **/usr/local/etc/mgetty+sendfax/login.config** 有以下内容：

```
/AutoPPP/ -     - /etc/ppp/ppp-pap-dialup
```

这告诉 mgetty 对检测到的 PPP 连接运行 **ppp-pap-dialup**。

创建一个名为 **/etc/ppp/ppp-pap-dialup** 的可执行文件，包含以下内容：

[1717] https://www.freebsd.org/cgi/man.cgi?query=pppctl&sektion=8&format=html
[1718] https://docs.freebsd.org/en/books/handbook/serialcomms/index.html#dialup
[1719] https://www.freebsd.org/cgi/man.cgi?query=getty&sektion=8&format=html
[1720] https://cgit.freebsd.org/ports/tree/comms/mgetty+sendfax/pkg-descr
[1721] https://cgit.freebsd.org/ports/tree/comms/mgetty+sendfax/pkg-descr
[1722] https://cgit.freebsd.org/ports/tree/comms/mgetty+sendfax/pkg-descr

```
#!/bin/sh
exec /usr/sbin/ppp -direct pap$IDENT
```

对于在 **/etc/ttys** 中启用的每一条拨号线路，在 **/etc/ppp/ppp.conf** 中创建一个相应的条目。这将与我们上面创建的定义愉快地共存。

```
pap:
  enable pap
  set ifaddr 203.14.100.1 203.14.100.20-203.14.100.40
  enable proxy
```

每个用这种方法登录的用户都需要在 **/etc/ppp/ppp.secret** 里有一个用户名/密码，或者添加以下选项，从 **/etc/passwd** 里通过 PAP 认证用户。

```
enable passwdauth
```

要给某些用户分配一个静态 IP 号码，在 **/etc/ppp/ppp.secret** 中指定号码作为第三个参数。例子见 **/usr/share/examples/ppp/ppp.secret.sample**。

30.3.PPP 连接的故障排除

本节涉及通过调制解调器连接使用 PPP 时可能出现的几个问题。一些 ISP 给出 `ssword` 提示，而另一些则提示 `password`。如果没有编写相应地 `ppp` 脚本，登录尝试就会失败。调试 `ppp` 连接的最常用方法是通过手动连接，如本节所述。

30.3.1.检查设备节点

当使用定制内核时，确保在内核配置文件中包括以下一行：

```
device uart
```

uart 设备已经包含在 `GENERIC` 内核中，所以在这种情况下无需多余的步骤。只要在 `dmesg` 输出中检查调制解调器设备就可以了：

```
# dmesg | grep uart
```

这应该会显示一些关于 **uart** 设备的相关输出。这些是我们需要的 COM 端口。如果调制解调器像一个标准的串行端口，它应该被列在 **uart1** 或 **COM2** 上。如果是这样，就不需要重建内核了。在匹配时，如果调制解调器在 **uart1** 上，调制解调器设备将是 **/dev/cuau1**。

30.3.2.手动连接

通过手动控制 ppp 连接到互联网是快速、简单的，而且是调试连接或者只是获得 ISP 如何对待 ppp 客户端连接的信息的好方法，。让我们从命令行启动 PPP。注意，在我们所有的例子中，我们将使用 *example* 作为运行 PPP 的机器的主机名。要启动 ppp：

```
# ppp
```

```
ppp ON example> set device /dev/cuau1
```

这第二条命令设置调制解调器设备为 **cuau1**。

```
ppp ON example> set speed 115200
```

这将连接速度设置为 115,200 kbps。

```
ppp ON example> enable dns
```

这告诉 ppp 配置解析器，并在 **/etc/resolv.conf** 中添加 nameserver 行。如果 ppp 不能确定主机名，可以在以后手动设置。

```
ppp ON example> term
```

这将切换到“terminal”模式，以便手动控制调制解调器。

```
deflink: Entering terminal mode on /dev/cuau1
type '~h' for help
```

```
at
OK
atdt123456789
```

使用 at 初始化调制解调器，然后使用 atdt 和 ISP 的号码，开始拨号过程。

```
CONNECT
```

确认连接，如果我们将有任何连接问题，与硬件无关，我们将在这里尝试解决这些问题。

```
ISP Login:myusername
```

在这个提示下，用 ISP 提供的用户名返回提示。

```
ISP Pass:mypassword
```

在这个提示下，用 ISP 提供的密码来回答。就像登录 FreeBSD 一样，密码不会有回显。

```
Shell or PPP:ppp
```

根据 ISP 的不同，这个提示可能不会出现。如果它出现了，它将询问是否在供应商上使用 shell 或启动 ppp。在这个例子中，选择 ppp 是为了建立互联网连接。

```
Ppp ON example>
```

注意，在这个例子中，第一个 P 被大写了。这表明我们已经成功连接到 ISP。

```
Ppp ON example>
```

我们已经成功地与 ISP 进行了认证，正在等待分配的 IP 地址。

```
PPP ON example>
```

我们已就 IP 地址达成协议并成功完成连接。

```
PPP ON example>add default HISADDR
```

在这里我们添加我们的默认路由，我们需要在与外界通信之前先做这个，因为目前唯一建立的连接是与对等者的连接。如果由于现有的路由而导致失败，在 add 的前面加上一个砰的字符 !。或者，在进行实际连接之前设置这个，它将相应地协商一个新的路由。

如果一切顺利，我们现在应该有一个与互联网的活动连接，可以用 CTRL+z 将其扔到后台。如果 PPP 返回 ppp，则连接已经丢失。这是很好的信息，因为它显示了连接状态。大写的 P 代表与 ISP 的连接，小写的 p 表示连接已经丢失。

30.3.3.调试

如果不能建立连接，使用 set ctsrts off 将硬件流 CTS/RTS 转为关闭。这主要是在连接到某些具有 PPP 功能的终端服务器时，PPP 在试图向通信链路写入数据时挂起，并等待可能永远不会出现的 Clear To Send（CTS）信号。当使用这个选项时，加入 set accmap，因为可能需要它来打败依赖于从头到尾传递某些字符的硬件，大多数时候是 XON/XOFF。参考 ppp(8)[1723] 以了解关于这个选项的更多信息以及它的使用方法。

老式调制解调器可能需要 set parity even。默认情况下，奇偶校验被设置为无，但在较旧的调制解调器上，用于流量大增时的错误检查。

[1723] https://www.freebsd.org/cgi/man.cgi?query=ppp&sektion=8&format=html

PPP 可能不会返回到命令模式，这通常是通信错误，ISP 正在等待协商的开始。在这一点上，使用 ~p 将迫使 ppp 开始发送配置信息。

如果从未出现登录提示，很可能需要 PAP 或 CHAP 认证。要使用 PAP 或 CHAP，在进入终端模式之前，在 PPP 中添加以下选项：

```
ppp ON example> set authname myusername
```

其中 *myusername* 应替换为 ISP 分配的用户名。

```
ppp ON example> set authkey mypassword
```

其中 *mypassword* 应替换为 ISP 分配的密码。

如果建立了连接，但似乎找不到任何域名，试着 ping(8)[1724] 一个 IP 地址。如果有百分之百（100%）的数据包丢失，很可能是没有分配默认路由。仔细检查在连接过程中是否设置了 `add default HISADDR`。如果可以连接到一个远程 IP 地址，有可能是没有在 **/etc/resolv.conf** 中添加一个解析地址。这个文件应该是这样的：

```
domain example.com
nameserver x.x.x.x
nameserver y.y.y.y
```

其中 *x.x.x.x* 和 *y.y.y.y* 应该被替换成 ISP 的 DNS 服务器的 IP 地址。

要配置 syslog(3)[1725] 为 PPP 连接提供日志记录，确保在 **/etc/syslog.conf** 中存在这一行：

```
!ppp
*.*     /var/log/ppp.log
```

30.4.使用以太网 PPP（PPPoE）

本节介绍如何设置基于以太网的 PPP（PPPoE）。

下面是一个工作中的 **ppp.conf** 的例子：

```
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:
```

(continues on next page)

[1724] https://www.freebsd.org/cgi/man.cgi?query=ping&sektion=8&format=html

[1725] https://www.freebsd.org/cgi/man.cgi?query=syslog&sektion=3&format=html

(continued from previous page)

```
  set device PPPoE:xl1 # replace xl1 with your Ethernet device
  set authname YOURLOGINNAME
  set authkey YOURPASSWORD
  set dial
  set login
  add default HISADDR
```

作为 `root`，运行：

```
# ppp -ddial name_of_service_provider
```

在 **/etc/rc.conf** 中添加以下内容：

```
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"
```

30.4.1.使用 PPPoE 服务标签

有时需要使用一个服务标签来建立连接。服务标签用于区分连接到特定网络的不同 PPPoE 服务器。

任何所需的服务标签信息应该在 ISP 提供的文件中。

作为最后的手段，人们可以尝试通过二进制包或 port 安装 net/rr-pppoe[1726]。但要记住，这可能会使你的调制解调器失去程序，使其失去作用，所以在这样做之前要三思。简单地安装调制解调器附带的程序。然后，从该程序进入 **System** 菜单。配置文件的名称应该列在那里。它通常是 *ISP*。

配置文件名称（服务标签）将被用于 **ppp.conf** 中的 PPPoE 配置条目，作为 `set device` 的 provider 部分。请参阅 ppp(8)[1727] 以了解完整的细节。它应该看起来像这样：

```
set device PPPoE:xl1:ISP
```

不要忘记把 *xl1* 改成以太网卡的正确设备。

不要忘记将 *ISP* 改为配置文件。

更多信息请参考 Renaud Waldura 写的 在 DSL 上使用 FreeBSD 的更便宜的宽带[1728]。

[1726] https://cgit.freebsd.org/ports/tree/net/rr-pppoe/pkg-descr

[1727] https://www.freebsd.org/cgi/man.cgi?query=ppp&sektion=8&format=html

[1728] http://renaud.waldura.com/doc/freebsd/pppoe/

30.4.2.使用 3Com® HomeConnect™ ADSL 调制解调器双链路的 PPPoE

这个调制解调器并不遵循 RFC 2516[1729] 中定义的 PPPoE 规范。

为了使 FreeBSD 能够与这个设备进行通信，必须设置 sysctl。这可以在启动时通过更新 **/etc/sysctl.conf** 来自动完成。

```
net.graph.nonstandard_pppoe=1
```

或者可以立即用命令来完成：

```
# sysctl net.graph.nonstandard_pppoe=1
```

不幸的是，由于这是一个系统范围内的设置，所以不可能同时与普通的 PPPoE 客户端或服务器和 3Com® HomeConnect™ ADSL 调制解调器通信。

30.5.使用 ATM 上的 PPP (PPPoA)

下面介绍如何设置基于 ATM 的 PPP（PPPoA）。PPPoA 是欧洲 DSL 供应商中的一个普遍的选择。

30.5.1.使用 mpd

mpd 应用程序可以用来连接各种服务，特别是 PPTP 服务。它可以使用二进制包或 port 来安装 net/mpd5[1730]。许多 ADSL 调制解调器要求在调制解调器和计算机之间建立一个 PPTP 隧道。

一旦安装，配置 mpd 以适应供应商的设置。该端口放置了一组样本配置文件，这些文件在 **/usr/local/etc/mpd/** 中有详细的记录。在 **/usr/ports/shared/doc/mpd/** 中有一份 HTML 格式的完整的 mpd 配置指南。下面是一个用 mpd 连接 ADSL 服务的配置样本。该配置分布在两个文件中，首先是 **mpd.conf**：

> 注意
> 这个例子的 **mpd.conf** 仅适用于 mpd 4.x。

```
default:
    load adsl

adsl:
    new -i ng0 adsl adsl
    set bundle authname username  ①
    set bundle password password  ②
    set bundle disable multilink
```

(continues on next page)

[1729] http://www.faqs.org/rfcs/rfc2516.html

[1730] https://cgit.freebsd.org/ports/tree/net/mpd5/pkg-descr

(continued from previous page)

```
    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
```

① 用来与你的 ISP 进行身份验证的用户名。

② 用来与你的 ISP 进行身份验证的密码。

关于要建立的链接的信息可以在 **mpd.links** 中找到。下面给出了一个配合上述例子的 **mpd.links** 的例子：

```
adsl:
    set link type pptp
    set pptp mode active
    set pptp enable originate outcall
    set pptp self 10.0.0.1  ①
    set pptp peer 10.0.0.138  ②
```

① 运行 mpd 的 FreeBSD 计算机的 IP 地址。

②ADSL 调制解调器的 IP 地址。Alcatel SpeedTouch™ Home 默认为 `10.0.0.138`。

可以通过以 `root` 身份执行以下命令来轻松初始化连接：

```
# mpd -b adsl
```

要查看连接的状态：

```
% ifconfig ng0
ng0: flags=88d1<UP,POINTOPOINT,RUNNING,NOARP,SIMPLEX,MULTICAST> mtu 1500
     inet 216.136.204.117 --> 204.152.186.171 netmask 0xffffffff
```

使用 mpd 是使用 FreeBSD 连接 ADSL 服务的推荐方式。

30.5.2.使用 pptpclient

使用 FreeBSD 也可以通过 net/pptpclient[1731] 连接到其他 PPPoA 服务。

要使用 net/pptpclient[1732] 连接到 DSL 服务，需要安装这个二进制包或 port，然后编辑 **/etc/ppp/ppp.conf**。下面给出了 **ppp.conf** 的一个示例部分。关于 **ppp.conf** 选项的进一步信息，请参考 ppp(8)[1733]。

```
adsl:
 set log phase chat lcp ipcp ccp tun command
 set timeout 0
 enable dns
 set authname username  ①
 set authkey password   ②
 set ifaddr 0 0
 add default HISADDR
```

①DSL 供应商的用户名。

② 你账户的密码。

> 警告
>
> 由于账户的密码是以纯文本形式添加到 **ppp.conf** 中的，所以要确保没有人能够读到这个文件的内容：
>
> ```
> # chown root:wheel /etc/ppp/ppp.conf
> # chmod 600 /etc/ppp/ppp.conf
> ```

这将为通往 DSL 路由器的 PPP 会话打开一条隧道。以太网 DSL 调制解调器有一个预先配置好的 LAN IP 地址来连接。就 Alcatel SpeedTouch™ Home 而言，这个地址是 `10.0.0.138`。路由器的文档应该列出设备使用的地址。要打开隧道并开始一个 PPP 会话：

```
# pptp address adsl
```

> 技巧
>
> 如果在这个命令的末尾加上一个记号（“&”），pptp 将返回提示。

将创建虚拟隧道设备 **tun**，用于 pptp 和 ppp 进程之间的交互。一旦提示返回，或者 pptp 进程确认了连接，请检查隧道：

```
% ifconfig tun0
tun0: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST> mtu 1500
```

(continues on next page)

[1731] https://cgit.freebsd.org/ports/tree/net/pptpclient/pkg-descr

[1732] https://cgit.freebsd.org/ports/tree/net/pptpclient/pkg-descr

[1733] https://www.freebsd.org/cgi/man.cgi?query=ppp&sektion=8&format=html

(continued from previous page)

```
        inet 216.136.204.21 --> 204.152.186.171 netmask 0xffffff00
    Opened by PID 918
```

如果连接失败，请检查路由器的配置，通常可以用网络浏览器访问。另外，检查 `pptp` 的输出和日志文件 **/var/log/ppp.log** 的内容，寻找线索。

第三十一章电子邮件

31.1.概述

“电子邮件”，即众所周知的 email，是当今最广泛使用的通信方式之一。这一章提供了在 FreeBSD 上运行邮件服务器的基本介绍，以及使用 FreeBSD 发送和接收电子邮件的说明。如果想了解更完整的内容，请参考参考书目[1734]中列出的书籍。

读完这一章后，你将知道：

- 哪些软件组件参与了电子邮件的发送和接收。
- 在 FreeBSD 中 Sendmail 基本配置文件的位置。
- 远程邮箱和本地邮箱的区别。
- 如何阻止垃圾邮件发送者非法使用邮件服务器作为中转站。
- 如何安装和配置其他的邮件传输代理以取代 Sendmail。
- 如何排除常见的邮件服务器故障。
- 如何将系统设置为仅发送邮件。
- 如何在拨号连接下使用邮件。
- 如何配置 SMTP 认证以增加安全性。
- 如何安装和使用邮件用户代理（如 mutt）来发送和接收邮件。
- 如何从远程 POP 或 IMAP 服务器下载邮件。
- 如何对收到的邮件自动应用过滤器和规则。

在阅读本章之前，你应该：

- 正确地设置网络连接（高级网络[1735]）。

[1734] https://docs.freebsd.org/en/books/handbook/bibliography/index.html#bibliography

[1735] https://docs.freebsd.org/en/books/handbook/advanced-networking/index.html#advanced-networking

- 正确地设置邮件主机的 DNS 信息（网络服务器[1736]）。
- 知道如何安装额外的第三方软件（安装应用程序：软件包和 ports[1737]）。

31.2.邮件组件

在电子邮件交换中涉及到五个主要部分：邮件用户代理（MUA）、邮件传输代理（MTA）、邮件主机、远程或本地邮箱、以及 DNS。本节对这些部分进行了概述。

邮件用户代理（MUA）

邮件用户代理（MUA）是一个用来编写、发送和接收电子邮件的应用程序。这个程序可以是命令行程序，如内置的 `mail` 工具，或来自 Ports 的第三方应用程序，比如 mutt、alpine 或 elm。在 Ports 中也有几十个图形化程序可用，包括 Claws Mail、Evolution 和 Thunderbird。一些组织提供了可通过网络浏览器访问的网络邮件程序。关于在 FreeBSD 上安装和使用 MUA 的更多信息可以在邮件用户代理[1738]中找到。

邮件传输代理（MTA）

邮件传输代理（MTA）负责接收传入的邮件和传递传出的邮件。FreeBSD 默认的 MTA 是 Sendmail，但它也支持许多其他的邮件服务器守护程序，包括 Exim、Postfix 和 qmail。Sendmail 的配置在 Sendmail 配置文件[1739]中说明。如果使用 Ports 安装了其他 MTA，请参考其安装后的信息以了解在 FreeBSD 中特定的配置细节，以及该应用程序的网站以了解更多常用的配置说明。

邮件主机和邮箱

邮件主机是负责为主机或网络传递和接收邮件的服务器。邮件主机收集所有发送到域名的邮件，并将其存储在默认的 **mbox** 或替代的 Maildir 格式中，具体取决于配置。邮件被储存起来后，它既可以使用 MUA 在本地读取，也可以使用 POP 或 IMAP 等协议远程访问和收集。如果在本地读取邮件，无需安装 POP 或 IMAP 服务器。

如果要远程访问邮箱，则需要 POP 或 IMAP 服务器，因为这些协议允许用户从远程位置连接到他们的邮箱。IMAP 比 POP 有几个优点。这些优势包括在下载邮件后在远程服务器上存储一份邮件副本和同步更新的能力。IMAP 在低速链接中很有用，因为它允许用户在不下载邮件的情况下获取邮件的结构。它还可以在服务器上执行搜索等任务，以尽量减少客户和服务器之间的数据传输。

在 Ports 中有几个 POP 和 IMAP 服务器可用。这些服务器包括 mail/qpopper[1740]，mail/imap-uw[1741]、mail/courier-imap[1742] 和 mail/dovecot2[1743]。

警告

[1736] https://docs.freebsd.org/en/books/handbook/network-servers/index.html#network-servers
[1737] https://docs.freebsd.org/en/books/handbook/ports/index.html#ports
[1738] https://docs.freebsd.org/en/books/handbook/mail/#mail-agents
[1739] https://docs.freebsd.org/en/books/handbook/mail/#sendmail
[1740] https://cgit.freebsd.org/ports/tree/mail/qpopper/pkg-descr
[1741] https://cgit.freebsd.org/ports/tree/mail/imap-uw/pkg-descr
[1742] https://cgit.freebsd.org/ports/tree/mail/courier-imap/pkg-descr
[1743] https://cgit.freebsd.org/ports/tree/mail/dovecot2/pkg-descr

值得注意的是，POP 和 IMAP 都以明文方式传输信息，包括用户名和密码凭据。为了保证这些协议的信息传输安全，可以考虑通过 ssh(1)[1744]（SSH 隧道[1745]）或使用 SSL（OpenSSL[1746]）进行隧道会话。

域名系统（DNS）

域名系统（DNS）及其守护程序 named 在电子邮件的传输中发挥着很大的作用。为了将邮件从一个站点传递到另一个站点，MTA 将在 DNS 中查找远程站点，以确定哪个主机将接收目的地的邮件。当邮件从远程主机发送至 MTA 时，同样如此。

除了将主机名映射到 IP 地址外，DNS 还负责存储邮件发送的特定信息，称为 Mail eXchanger MX 记录。MX 记录指定了哪些主机将接收某个特定域的邮件。

要查看一个域名的 MX 记录，需要指定记录的类型。参考 host(1)[1747] 可了解关于这个命令的更多细节。

```
% host -t mx FreeBSD.org
FreeBSD.org mail is handled by 10 mx1.FreeBSD.org
```

关于 DNS 及其配置的更多信息，请参考域名系统（DNS）[1748]。

31.3.Sendmail 配置文件

Sendmail 是 FreeBSD 默认安装的 MTA。它接受来自 MUA 的邮件，并根据其配置定义将其送到相应的邮件主机。Sendmail 也可以接受网络连接，并将邮件发送到本地邮箱或其他程序。

Sendmail 的配置文件位于 **/etc/mail** 中。本节将对这些文件进行详细地说明。

/etc/mail/access

这个访问数据库文件定义了哪些主机或 IP 地址可以访问本地邮件服务器，以及它们有什么样的访问权限。列为 OK 的主机，也就是默认选项，允许向该主机发送邮件，只要邮件的最终目的地是本地机器。列为 REJECT 的主机会拒绝所有的邮件连接。被列为 RELAY 的主机被允许使用该邮件服务器向任何目的地发送邮件。被列为 ERROR 的主机将使他们的邮件以指定的邮件错误返回。如果一个主机被列为 SKIP，Sendmail 将放弃当前对这个条目的搜索而不接受或拒绝邮件。被列为 QUARANTINE 的主机将被保留其邮件，并将收到指定的文字作为保留的原因。

适用于 IPv4 和 IPv6 地址使用这些选项的例子可以在 FreeBSD 样本配置 **/etc/mail/access.sample** 中找到：

```
# $FreeBSD$
#
# Mail relay access control list.  Default is to reject mail unless the
```

(continues on next page)

[1744] https://www.freebsd.org/cgi/man.cgi?query=ssh&sektion=1&format=html
[1745] https://docs.freebsd.org/en/books/handbook/security/index.html#security-ssh-tunneling
[1746] https://docs.freebsd.org/en/books/handbook/security/index.html#openssl
[1747] https://www.freebsd.org/cgi/man.cgi?query=host&sektion=1&format=html
[1748] https://docs.freebsd.org/en/books/handbook/network-servers/index.html#network-dns

(continued from previous page)

```
# destination is local, or listed in /etc/mail/local-host-names
#
## Examples (commented out for safety)
#From:cyberspammer.com          ERROR:"550 We don't accept mail from spammers"
#From:okay.cyberspammer.com     OK
#Connect:sendmail.org           RELAY
#To:sendmail.org                RELAY
#Connect:128.32                 RELAY
#Connect:128.32.2               SKIP
#Connect:IPv6:1:2:3:4:5:6:7     RELAY
#Connect:suspicious.example.com QUARANTINE:Mail from suspicious host
#Connect:[127.0.0.3]            OK
#Connect:[IPv6:1:2:3:4:5:6:7:8] OK
```

要配置访问数据库，请使用样本中所示的格式在 **/etc/mail/access** 中建立条目，但不要在条目前面加上注释符号（#）。为每个需要配置访问权限的主机或网络创建一个条目。匹配表格左边的邮件发送者会受到表格右边的动作的影响。

每当这个文件被更新时，要更新它的数据库并重新启动 Sendmail：

```
# makemap hash /etc/mail/access < /etc/mail/access
# service sendmail restart
```

/etc/mail/aliases

这个数据库文件包含了一个虚拟邮箱的列表，这些虚拟邮箱被扩展到用户、文件、程序或其他别名。下面是几个条目来说明文件的格式：

```
root: localuser
ftp-bugs: joe,eric,paul
bit.bucket:  /dev/null
procmail: "|/usr/local/bin/procmail"
```

冒号左边的邮箱名称被扩展到右边的目标。第一个条目将 `root` 邮箱扩展到 `localuser` 邮箱，然后在 **/etc/mail/aliases** 数据库中进行查找。如果没有找到匹配的，邮件就会被投递到 `localuser`。第二个条目显示了一个邮件列表。给 `ftp-bugs` 的邮件被扩展到三个本地邮箱 `joe`, `eric` 和 `paul`。一个远程邮箱可以被指定为 *user@example.com*。第三个条目显示了如何将邮件写入一个文件，在这个例子中是 **/dev/null**。最后一条演示了如何通过 UNIX® 管道将邮件发送到一个程序，**/usr/local/bin/procmail**。关于这个文件的格式，请参考 aliases(5)[1749] 了解更多信息。

每当这个文件被更新时，运行 `newaliases` 来更新和初始化别名数据库。

/etc/mail/sendmail.cf

[1749] https://www.freebsd.org/cgi/man.cgi?query=aliases&sektion=5&format=html

这是 Sendmail 的主配置文件。它控制了 Sendmail 的整体行为，包括从重写电子邮件地址到打印拒绝信息到远程邮件服务器的一切。因此，这个配置文件是相当复杂的。幸运的是，对于标准的邮件服务器来说，很少需要对这个文件进行修改。

主 Sendmail 配置文件可以由定义 Sendmail 功能和行为的 m4(1)[1750] 宏构建。参考 **/usr/src/contrib/sendmail/cf/README** 以了解其中的一些细节。

每当对这个文件进行修改时，需要重新启动 Sendmail 以使修改生效。

/etc/mail/virtusertable

这个数据库文件将虚拟域和用户的邮件地址映射到真实的邮箱中。这些邮箱可以是本地的、远程的、在 **/etc/mail/aliases** 中定义的别名，或文件。这允许在一台机器上托管多个虚拟域。

FreeBSD 在 **/etc/mail/virtusertable.sample** 中提供了一个配置文件样本来进一步演示其格式。下面的例子演示了如何使用该格式创建自定义条目：

```
root@example.com                root
postmaster@example.com          postmaster@noc.example.net
@example.com                    joe
```

这个文件是以第一匹配顺序处理的。当一个电子邮件地址与左边的地址相匹配时，它就被映射到右边列出的本地邮箱中。本例中第一个条目的格式是将一个特定的电子邮件地址映射到本地邮箱，而第二个条目的格式是将一个特定的电子邮件地址映射到远程邮箱。最后，来自 `example.com` 的任何电子邮件地址，如果没有与前面的任何条目相匹配，将与最后一个映射相匹配，并被发送到本地邮箱 `joe`。当创建自定义条目时，使用这种格式并将其添加到 **/etc/mail/virtusertable**。每当这个文件被编辑时，要更新它的数据库并重新启动 Sendmail：

```
# makemap hash /etc/mail/virtusertable < /etc/mail/virtusertable
# service sendmail restart
```

/etc/mail/relay-domains

在默认的 FreeBSD 系统中，Sendmail 被配置为只从它所运行的主机上发送邮件。例如，如果有一个 POP 服务器，用户将能够从远程位置检查邮件，但他们将不能从外部位置发送邮件。通常情况下，在尝试后的片刻，将从 `MAILER-DAEMON` 发送一封带有 `5.7 Relaying Denied` 信息的电子邮件。

最直接的解决办法是将 ISP 的 FQDN 添加到 **/etc/mail/relay-domains**。如果需要多个地址，可每行添加一个：

```
your.isp.example.com
other.isp.example.net
users-isp.example.org
www.example.org
```

[1750] https://www.freebsd.org/cgi/man.cgi?query=m4&sektion=1&format=html

在创建或编辑这个文件后，用 `service sendmail restart` 重新启动 Sendmail。

现在，这个列表中的任何主机通过系统发送的任何邮件，只要用户在系统上有一个账户，都会成功。这使得用户可以从系统中远程发送邮件，而不至于让系统从互联网上转发垃圾邮件。

31.4.改变邮件传输代理

FreeBSD 已经安装了 Sendmail 作为 MTA，负责发信和收信。然而，系统管理员可以改变系统的 MTA。在 FreeBSD Ports 的 `mail` 类别中，有大量的 MTA 可供选择。

如果安装了新的 MTA，在替换 Sendmail 之前要配置和测试新的软件。关于如何配置软件的信息，请参考新 MTA 的文档。

新的 MTA 开始工作之后，使用本节中的说明来禁用 Sendmail，并配置 FreeBSD 来使用替换的 MTA。

31.4.1.停用 Sendmail

> 警告
>
> 如果 Sendmail 的外发邮件服务被禁用，重要的是要用一个另外的邮件发送系统来代替它。否则，诸如 periodic(8)[1751] 这样的系统功能将无法通过电子邮件来传递它们的结果。系统的许多部分都期望有一个功能性的 MTA。如果应用程序在禁用 Sendmail 之后继续使用 Sendmail 的二进制文件来尝试发送电子邮件，那么邮件可能会进入一个不活跃的 Sendmail 队列而永远不会被投递。

为了完全禁用 Sendmail，在 **/etc/rc.conf** 中添加或编辑以下几行:

```
sendmail_enable="NO"
sendmail_submit_enable="NO"
sendmail_outbound_enable="NO"
sendmail_msp_queue_enable="NO"
```

要只禁用 Sendmail 的入站邮件服务，只需在 **/etc/rc.conf** 中使用这个条目:

```
sendmail_enable="NO"
```

关于 Sendmail 的启动选项的更多信息可以在 rc.sendmail(8)[1752] 中找到。

[1751] https://www.freebsd.org/cgi/man.cgi?query=periodic&sektion=8&format=html

[1752] https://www.freebsd.org/cgi/man.cgi?query=rc.sendmail&sektion=8&format=html

31.4.2.替换默认的 MTA

当使用 Ports 安装新的 MTA 时，它的启动脚本也会被安装，并且在其软件包信息中提到了启动说明。在启动新的 MTA 之前，请停止运行中的 Sendmail 进程。该例子停止所有这些服务，然后启动 Postfix 服务：

```
# service sendmail stop
# service postfix start
```

为了在系统启动时启动替换的 MTA，在 **/etc/rc.conf** 中添加其配置行。这个条目可以启用 Postfix MTA：

```
postfix_enable="YES"
```

一些额外的配置是需要的，因为 Sendmail 被广泛使用，以至于一些软件认为它已经被安装和配置了。检查 **/etc/periodic.conf**，确保这些值被设置为 `NO`。如果这个文件不存在，用这些条目创建它：

```
daily_clean_hoststat_enable="NO"
daily_status_mail_rejects_enable="NO"
daily_status_include_submit_mailq="NO"
daily_submit_queuerun="NO"
```

一些可选的 MTA 提供了他们自己的 Sendmail 命令行接口的兼容实现，以便于使用它们作为 Sendmail 的直接替代。然而，一些 MUA 可能会试图执行标准的 Sendmail 二进制文件，而非新的 MTA 的二进制文件。FreeBSD 使用 **/etc/mail/mailer.conf** 来将预期的 Sendmail 二进制文件映射到新的二进制文件的位置。关于这个映射的更多信息可以在 mailwrapper(8)[1753] 中找到。

默认的 **/etc/mail/mailer.conf** 看起来像这样：

```
# $FreeBSD$
#
# Execute the "real" sendmail program, named /usr/libexec/sendmail/sendmail
#
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
```

当左边列出的任何命令被运行时，系统实际上会执行右边显示的相关命令。该系统使我们很容易改变在调用这些默认二进制文件时执行的二进制文件。

一些 MTA 在使用 Ports 安装时，会提示为新的二进制文件更新这个文件。例如，Postfix 会像这样更新文件：

[1753] https://www.freebsd.org/cgi/man.cgi?query=mailwrapper&sektion=8&format=html

```
#
# Execute the Postfix sendmail program, named /usr/local/sbin/sendmail
#
sendmail        /usr/local/sbin/sendmail
send-mail       /usr/local/sbin/sendmail
mailq           /usr/local/sbin/sendmail
newaliases      /usr/local/sbin/sendmail
```

如果 MTA 在安装中没有自动更新 **/etc/mail/mailer.conf**，请用文本编辑器编辑这个文件，使其指向新的二进制文件。这个例子指向 mail/ssmtp[1754] 所安装的二进制文件：

```
sendmail        /usr/local/sbin/ssmtp
send-mail       /usr/local/sbin/ssmtp
mailq           /usr/local/sbin/ssmtp
newaliases      /usr/local/sbin/ssmtp
hoststat        /usr/bin/true
purgestat       /usr/bin/true
```

一切配置完毕后，建议重新启动系统。重启提供了一个机会，以确保系统被正确配置为在启动时自动启动新的 MTA。

31.5.故障排除

31.5.1.为什么在我的网站上的主机必须使用 FQDN？

主机实际上可能在一个不同的域中。例如，为了让 `foo.bar.edu` 中的主机能够到达 `bar.edu` 域中一个叫 `mumble` 的主机，要用完全限定域名 FQDN，`mumble.bar.edu`，而不能只用 `mumble`。

这是因为 FreeBSD 的 BIND 版本不再为本地域以外的非 FQDN 提供默认缩写。像 `mumble` 这样的非限定主机必须以 `mumble.foo.bar.edu` 的形式出现，否则就会在根域中搜索到它。

在老版本的 BIND 中，搜索继续跨越 `mumble.bar.edu` 和 `mumble.edu`。RFC 1535 详细说明了为什么这被认为是不好的做法甚至是一个安全漏洞。

作为一个很好的解决方法，把这一行放在：

```
search foo.bar.edu bar.edu
```

而不是之前的把：

```
domain foo.bar.edu
```

[1754] https://cgit.freebsd.org/ports/tree/mail/ssmtp/pkg-descr

放入 **/etc/resolv.conf**。然而，要确保搜索顺序不超过 RFC 1535 所说的“地方和公共管理之间的边界”。

31.5.2.如何在拨号的 PPP 主机上运行一个邮件服务器？

连接到 LAN 上的 FreeBSD 邮件网关。这个 PPP 连接是非专用的。

一种方法是让一个全职的互联网服务器为该域名提供二级 MX 服务。在这个例子中，域名是 `example.com`，ISP 配置了 `example.net` 来为该域名提供二级 MX 服务：

```
example.com.          MX        10      example.com.
                      MX        20      example.net.
```

只有一个主机应该被指定为最终收件人。对于 Sendmail，在 **/etc/mail/sendmail.cf** 中的 `example.com` 上添加 `Cw example.com`。

当发送 MTA 试图投递邮件时，它将尝试通过 PPP 链接连接到 `example.com` 这个系统。如果目的地是离线的，这将超时。MTA 将自动把邮件送到互联网服务提供商（ISP）的二级 MX 站点 `example.net`。二级 MX 站点将定期尝试连接到一级 MX 主机example.com`。

使用类似这样的东西作为登录脚本：

```
#!/bin/sh
# Put me in /usr/local/bin/pppmyisp
( sleep 60 ; /usr/sbin/sendmail -q ) &
/usr/sbin/ppp -direct pppmyisp
```

当为用户创建一个单独的登录脚本时，在上面的脚本中改用 `sendmail -qRexample.com`。这将迫使 `example.com` 的队列中的所有邮件被立即处理。

从 FreeBSD 互联网服务提供商的邮件列表[1755]中的这个例子可以看出对这种情况的进一步细化：

```
> 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
```

(continues on next page)

[1755] https://lists.freebsd.org/subscription/freebsd-isp

(continued from previous page)

```
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.
```

31.6.高级主题

本节涵盖了更多的内容，如邮件配置和为整个域设置邮件。

31.6.1.基本配置

只要 **/etc/resolv.conf** 配置好了即可开箱即用，或者网络可以访问配置好的 DNS 服务器，就可以向外部主机发送邮件。要让邮件被送到 FreeBSD 主机上的 MTA，请执行以下操作之一：

- 运行一个域名的 DNS 服务器。
- 让邮件直接投递到机器的 FQDN 上。

为了让邮件直接送到主机上，它必须有一个永久的静态 IP 地址，而非动态 IP 地址。如果系统在防火墙后面，它必须被配置为允许 SMTP 流量。要在主机上直接接收邮件，必须配置以下两个中的一个：

- 确保 DNS 中编号最低的 MX 记录指向该主机的静态 IP 地址。
- 确保在 DNS 中没有该主机的 MX 条目。

上述任何一项都将允许在主机上直接接收邮件。

试试这个：

```
v# hostname
example.FreeBSD.org
# host example.FreeBSD.org
example.FreeBSD.org has address 204.216.27.XX
```

在这个例子中，假设 Sendmail 在 example.FreeBSD.org 上正常运行，直接发送到 your-login@example.FreeBSD.org 的邮件应该没有问题。

对于这个例子：

```
# host example.FreeBSD.org
example.FreeBSD.org has address 204.216.27.XX
example.FreeBSD.org mail is handled (pri=10) by nevdull.FreeBSD.org
```

所有发往 example.FreeBSD.org 的邮件都会被收集到 hub 上的同一个用户名下，而不是直接发往你的主机。

上述信息是由 DNS 服务器处理的。携带邮件路由信息的 DNS 记录是 MX 条目。如果没有 MX 记录，邮件将通过其 IP 地址直接送到主机上。

freefall.FreeBSD.org 的 MX 条目曾经是这样的：

```
# host example.FreeBSD.org
example.FreeBSD.org has address 204.216.27.XX
example.FreeBSD.org mail is handled (pri=10) by nevdull.FreeBSD.org
```

freefall 有许多 MX 条目。最低的 MX 号码是直接接收邮件的主机，如果有的话。如果它由于某种原因不能访问，下一个较低编号的主机将暂时接受邮件，并在较低编号的主机可用时将其传递。

备用 MX 站点应该有独立的互联网连接，以便发挥最大作用。你的 ISP 可以提供这种服务。

31.6.2.域名邮件

当为一个网络配置 MTA 时，任何发送到其域内主机的邮件都应该被转移到 MTA，以便用户可以在主邮件服务器上收到他们的邮件。

为了使生活变得更简单，在 MTA 和带有 MUA 的系统上都应该存在一个具有相同 *username* 的用户账户。使用 adduser(8)[1756] 来创建用户账户。

MTA 必须是网络上每个工作站的指定邮件交换器。这可以在 DNS 配置中通过 MX 记录来完成：

```
example.FreeBSD.org A   204.216.27.XX       ; Workstation
            MX  10 nevdull.FreeBSD.org  ; Mailhost
```

这将使工作站的邮件重定向到 MTA，无论 A 记录指向哪里。邮件会被发送到 MX 主机上。

这必须在 DNS 服务器上进行配置。如果网络没有运行自己的 DNS 服务器，请与 ISP 或 DNS 提供商沟通。

下面是一个虚拟电子邮件托管的例子。假设一个域名为 customer1.org 的客户，customer1.org 的所有邮件都应该被发送到 mail.myhost.com。其 DNS 条目应该是这样的：

[1756] https://www.freebsd.org/cgi/man.cgi?query=adduser&sektion=8&format=html

```
customer1.org       MX  10  mail.myhost.com
```

`customer1.org` 不需要 `A` 记录，以便只处理该域名的电子邮件。然而，除非有 `A` 记录存在，否则对 `customer1.org` 运行 `ping` 将不工作。

告诉 MTA 它应该接受哪些域名和/或主机名的邮件。以下两种方法都适用于 Sendmail：

- 在使用 `FEATURE(use_cw_file)` 时，将主机添加到 **/etc/mail/local-host-names**。
- 在 **/etc/sendmail.cf** 中添加 `Cwyour.host.com` 一行。

31.7.设置为仅发送

在很多情况下，人们可能只想通过中转来发送邮件。一些例子是：

- 计算机是一台台式机，需要使用诸如 mail(1)[1757] 等程序，使用 ISP 的邮件中继。
- 这台计算机是一台服务器，它不在本地处理邮件，而是需要将所有的邮件传递给中继站进行处理。

虽然任何 MTA 都能够填补这个特殊的空缺，但要正确配置一个全功能的 MTA 只是为了处理卸载邮件是很困难的。像 Sendmail 和 Postfix 这样的程序对于这种用途来说是过剩的。

此外，通常的互联网接入服务协议可能禁止个人运行“邮件服务器”。

满足这些需求的最简单方法是通过 ports 安装 mail/ssmtp[1758]：

```
# cd /usr/ports/mail/ssmtp
# make install replace clean
```

安装完毕之后，mail/ssmtp[1759] 可以用 **/usr/local/etc/ssmtp/ssmtp.conf** 来配置：

```
root=yourrealemail@example.com
mailhub=mail.example.com
rewriteDomain=example.com
hostname=_HOSTNAME_
```

使用真实的电子邮件地址作为 `root`。在 `mail.example.com` 的位置输入 ISP 的外发邮件中继站。有些 ISP 称其为“外发邮件服务器”或“SMTP 服务器”。

请确保禁用 Sendmail，包括外发邮件服务。详情见禁用 Sendmail[1760]。

[1757] https://www.freebsd.org/cgi/man.cgi?query=mail&sektion=1&format=html
[1758] https://cgit.freebsd.org/ports/tree/mail/ssmtp/pkg-descr
[1759] https://cgit.freebsd.org/ports/tree/mail/ssmtp/pkg-descr
[1760] https://docs.freebsd.org/en/books/handbook/mail/#mail-disable-sendmail

mail/ssmtp[1761] 还有一些其他可用的选项。请参考 **/usr/local/etc/ssmtp** 中的例子或 ssmtp 的手册页以获得更多信息。

以这种方式设置 ssmtp 允许计算机上任何需要发送邮件的软件正常运行，同时不违反 ISP 的使用规则，也不允许计算机被劫持用于发送垃圾邮件。

31.8.在拨号连接中使用邮件

当使用静态 IP 地址时，应该无需调整默认配置。将主机名设置为指定的互联网名称，剩下的就由 Sendmail 来完成。

当使用一个动态分配的 IP 地址和拨号 PPP 连接到互联网时，人们通常在 ISP 的邮件服务器上有一个邮箱。在这个例子中，ISP 的域名是 `example.net`，用户名是 `user`，主机名是 `bsd.home`，ISP 允许 `relay.example.net` 作为一个邮件中继。

为了从 ISP 的邮箱中检索邮件，请从 ports 安装检索代理。mail/fetchmail[1762] 是一个不错的选择，因为它支持许多不同的协议。通常情况下，ISP 会提供 POP。当使用用户 PPP 时，在 **/etc/ppp/ppp.linkup** 中的以下条目，当互联网连接建立后，可以自动获取电子邮件：

```
MYADDR:
!bg su user -c fetchmail
```

当使用 Sendmail 向非本地账户投递邮件时，配置 Sendmail 在互联网连接建立后立即处理邮件队列。要做到这一点，在 **/etc/ppp/ppp.linkup** 中的上述 `fetchmail` 条目之后添加这一行：

```
!bg su user -c "sendmail -q"
```

在这个例子中，在 `bsd.home` 上有一个 `user` 的账户。在 `bsd.home` 上的用户的主目录下，创建一个包含这一行的 **.fetchmailrc**：

```
poll example.net protocol pop3 fetchall pass MySecret
```

这个文件除了 `user` 之外任何人都不能读，因为它包含密码 `MySecret`。

为了发送带有正确的标头 `from:` 的邮件，配置 Sendmail 使用 user@example.net，而非 user@bsd.home，并且通过 `relay.example.net` 发送所有的邮件，这样可以更快地传输邮件。

下面的 **.mc** 应该足够了：

```
VERSIONID(`bsd.home.mc version 1.0')
OSTYPE(bsd4.4)dnl
```

(continues on next page)

[1761] https://cgit.freebsd.org/ports/tree/mail/ssmtp/pkg-descr

[1762] https://cgit.freebsd.org/ports/tree/mail/fetchmail/pkg-descr

(continued from previous page)

```
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
```

关于如何将这个文件转换成 **sendmail.cf** 格式的细节，请参考上一节。在更新 **sendmail.cf** 之后，不要忘记重新启动 Sendmail。

31.9.SMTP 认证

在 MTA 上配置 SMTP 认证有很多好处。SMTP 认证为 Sendmail 增加了一层安全性，并为切换主机的移动用户提供了使用同一 MTA 的能力，而不需要每次都重新配置他们邮件客户端的设置。

1.从 Ports 中安装 security/cyrus-sasl2[1763]。这个 port 支持一些编译选项。对于本例中演示的 SMTP 认证方法，确保 LOGIN 没有被禁用。

2.安装 security/cyrus-sasl2[1764] 后，编辑 **/usr/local/lib/sasl2/Sendmail.conf**，如果它不存在，则创建它，并添加以下一行：

```
pwcheck_method: saslauthd
```

3.接下来，安装 security/cyrus-sasl2-saslauthd[1765]，并在 **/etc/rc.conf** 中添加以下一行：

```
saslauthd_enable="YES"
```

最后，启动守护程序 saslauthd：

```
# service saslauthd start
```

这个守护进程作为 Sendmail 的代理，对 FreeBSD 的 passwd(5)[1766] 数据库进行认证。这样就省

[1763] https://cgit.freebsd.org/ports/tree/security/cyrus-sasl2/pkg-descr
[1764] https://cgit.freebsd.org/ports/tree/security/cyrus-sasl2/pkg-descr
[1765] https://cgit.freebsd.org/ports/tree/security/cyrus-sasl2-saslauthd/pkg-descr
[1766] https://www.freebsd.org/cgi/man.cgi?query=passwd&sektion=5&format=html

去了为每个需要使用 SMTP 认证的用户创建一套新的用户名和密码的麻烦，并使登录和邮件密码保持一致。

4.接下来，编辑 **/etc/make.conf** 并添加以下几行：

```
SENDMAIL_CFLAGS=-I/usr/local/include/sasl -DSASL
SENDMAIL_LDADD=/usr/local/lib/libsasl2.so
```

这几行为 Sendmail 提供了适当的配置选项，以便在编译时与 cyrus-sasl2[1767] 连接。在重新编译 Sendmail 之前，请确保cyrus-sasl2[1768] 已经安装完毕。

5.通过执行以下命令来重新编译 Sendmail。

```
# 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
```

如果 **/usr/src** 没有大的变化，并且它所需要的共享库是可用的，那么这个编译应该不会有任何问题。

6.在 Sendmail 被编译并重新安装后，编辑 **/etc/mail/freebsd.mc** 或本地的 **.mc**。许多管理员选择使用 hostname(1)[1769] 的输出作为 **.mc** 的名称，以保持其唯一性。添加这些行：

```
dnl set SASL options
TRUST_AUTH_MECH(`GSSAPI DIGEST-MD5 CRAM-MD5 LOGIN')dnl
define(`confAUTH_MECHANISMS', `GSSAPI DIGEST-MD5 CRAM-MD5 LOGIN')dnl
```

这些选项配置了 Sendmail 可用来验证用户的不同方法。要使用 pwcheck 以外的方法，请参考 Sendmail 的文档。

7.最后，在 **/etc/mail** 中运行 make(1)[1770]。这将运行新的 **.mc** 并创建一个 **.cf**，命名为 **freebsd.cf** 或本地 **.mc** 所用的名称。然后，运行`make install restart`，这将把文件复制到 **sendmail.cf**，并正确地重新启动 Sendmail。关于这个过程的更多信息，请参考 **/etc/mail/Makefile**。

为了测试配置，使用 MUA 来发送一个测试信息。为了进一步调查，将 Sendmail 的 `LogLevel` 设置为 `13`，并观察 **/var/log/maillog** 是否有错误。

欲了解更多信息，请参考 SMTP 认证[1771]。

[1767] https://cgit.freebsd.org/ports/tree/cyrus-sasl2/pkg-descr

[1768] https://cgit.freebsd.org/ports/tree/cyrus-sasl2/pkg-descr

[1769] https://www.freebsd.org/cgi/man.cgi?query=hostname&sektion=1&format=html

[1770] https://www.freebsd.org/cgi/man.cgi?query=make&sektion=1&format=html

[1771] http://www.sendmail.org/~ca/email/auth.html

31.10.邮件用户代理

MUA 是一个用于发送和接收电子邮件的应用程序。随着电子邮件的“进化”和变得更加复杂，MUA 也变得越来越强大，并为用户提供了更多的功能和灵活性。FreeBSD 中的 `mail` 类别包含了许多 MUA。其中包括图形化的电子邮件客户端（如 Evolution 或 Balsa）以及基于控制台的客户端（如 mutt 或 alpine）。

31.10.1.mail

mail(1)[1772] 是 FreeBSD 默认安装的 MUA。它是一个基于控制台的 MUA，提供了发送和接收基于文本的电子邮件所需的基本功能。它提供有限的附件支持，并且只能访问本地邮箱。

尽管 `mail` 本身不支持与 POP 或 IMAP 服务器的交互，但这些邮箱可以通过一个诸如 fetchmail 的应用程序下载到本地的 **mbox**。

为了发送和接收电子邮件，请运行 `mail`：

```
% mail
```

用户在 **/var/mail** 中的邮箱内容会被 `mail` 自动读取。如果邮箱是空的，该程序退出时将显示没有找到邮件的信息。如果邮件存在，应用程序界面就会启动，并显示一个邮件列表。邮件是自动编号的，在下面的例子中可以看到：

```
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"
```

现在可以通过输入 `t` 和信息编号来读取信息。这个例子读取了第一封邮件：

```
& 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.
```

[1772] https://www.freebsd.org/cgi/man.cgi?query=mail&sektion=1&format=html

从这个例子中可以看出，消息将以完整的标题显示。要再次显示邮件列表，请按 `h`。

如果该邮件需要回复，请按 `R` 或 `r` `mail` 键。`R` 指示 `mail` 只回复给邮件的发件人，而 `r` 回复给邮件的所有其他收件人。这些命令的后缀可以是要回复的信息的邮件编号。在输入回复后，邮件的结尾应在自己的行中用一个 `.` 来标记。下面是一个例子：

```
& R 1
To: root@localhost
Subject: Re: test

Thank you, I did get your email.
.
EOT
```

为了发送一封新的电子邮件，请按 `m`，然后是收件人的电子邮件地址。可以指定多个收件人，用分隔符 `,` 分隔每个地址。然后可以输入信息的主题，接着是信息内容。信息的结尾应该在自己的行中加上一个 `.` 来指定。

```
& mail root@localhost
Subject: I mastered mail

Now I can send and receive email using mail ... :)
.
EOT
```

在使用 `mail` 的过程中，按 `?` 来随时显示帮助。请参考 mail(1)[1773] 以获得更多关于如何使用 `mail` 的帮助。

> 注意
>
> mail(1)[1774] 并不是为处理附件而设计的，因此对它们的处理很差。新 MUA 以一种更智能的方式处理附件。喜欢使用 `mail` 的用户可能会发现 ports 中的 converters/mpack[1775] 是相当有用的。

31.10.2.Mutt

mutt 是一个强大的 MUA，有很多功能，包括：

- 多线程消息的能力。
- 支持 PGP，可对邮件进行数字签名和加密。
- 支持 MIME。
- 支持 Maildir。

[1773] https://www.freebsd.org/cgi/man.cgi?query=mail&sektion=1&format=html

[1774] https://www.freebsd.org/cgi/man.cgi?query=mail&sektion=1&format=html

[1775] https://cgit.freebsd.org/ports/tree/converters/mpack/pkg-descr

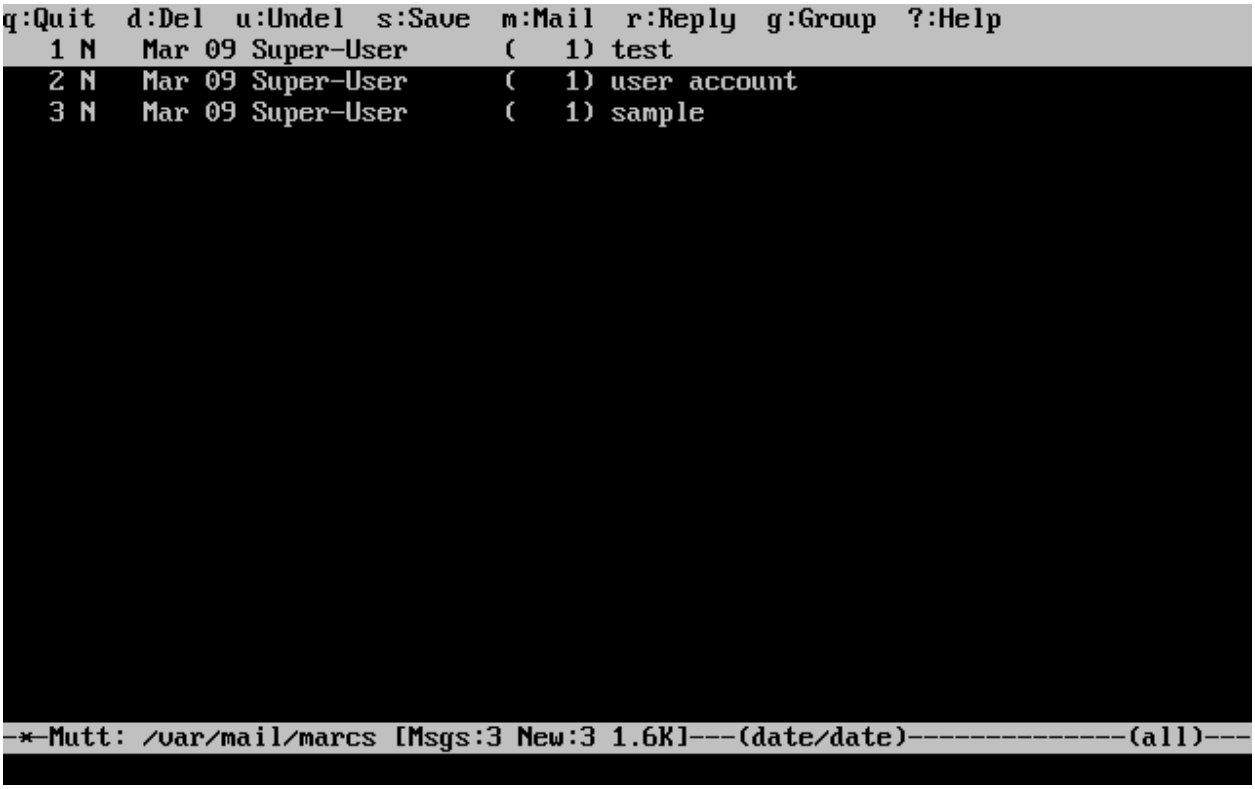
- 高可定制性。

关于 mutt 的更多信息，请参考 http://www.mutt.org。

mutt 可以使用 port mail/mutt[1776] 进行安装。安装完 port 后，可以通过执行以下命令来启动 mutt：

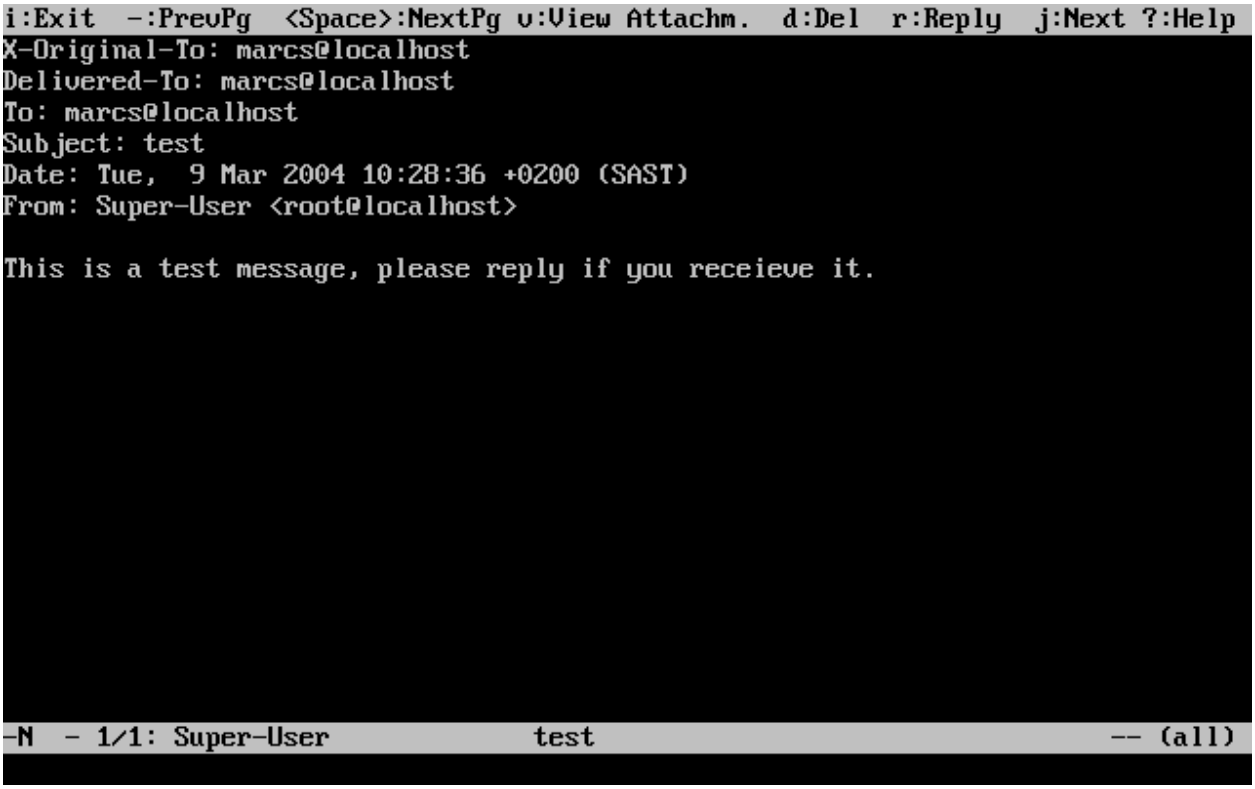
```
% mutt
```

mutt 将自动读取并显示 **/var/mail** 中用户邮箱的内容。如果没有发现邮件，mutt 将等待用户的命令。下面的例子显示 mutt 显示了一个邮件列表：



要阅读一封邮件，用光标键选择它，然后按回车键。下面可以看到 mutt 显示邮件的例子。

[1776] https://cgit.freebsd.org/ports/tree/mail/mutt/pkg-descr

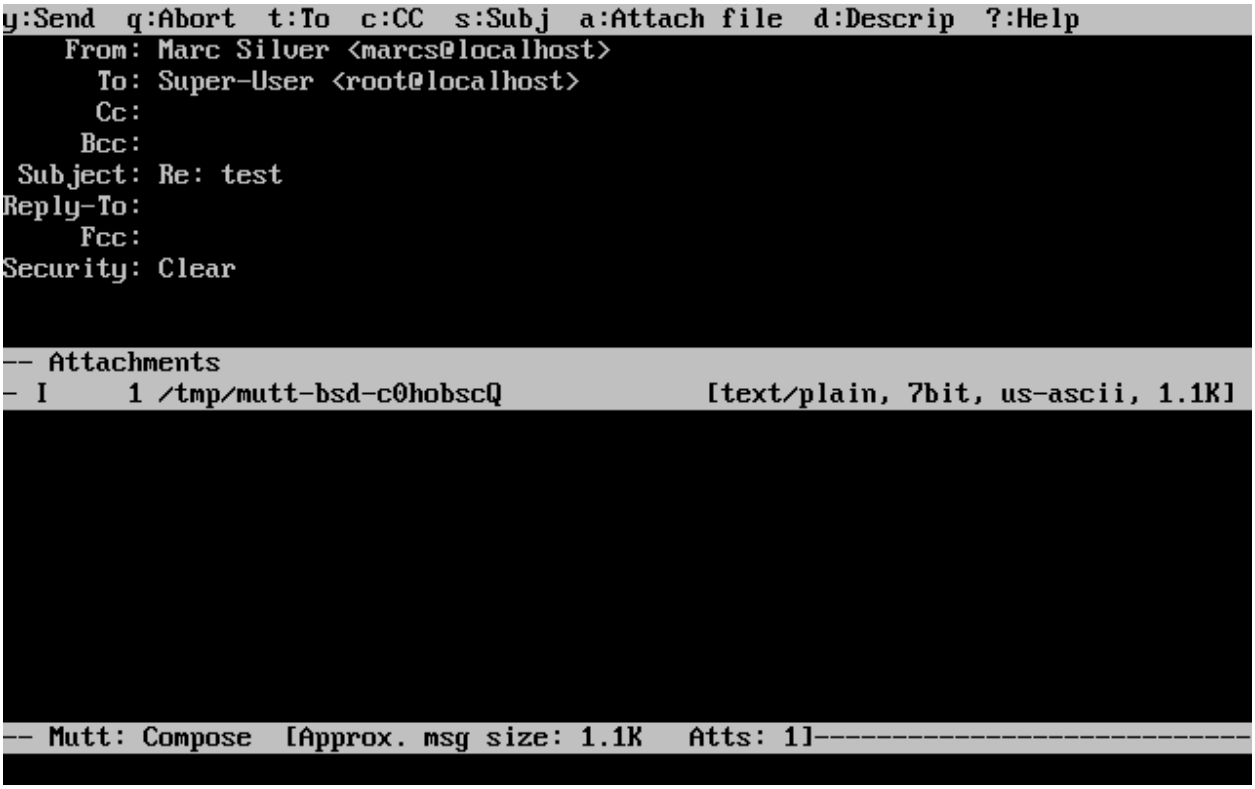


与 mail(1)[1777] 类似，mutt 可以用来只回复邮件的发件人，也可以回复所有收件人。要想只回复邮件的发件人，请按 `r`；要想向原发件人以及所有的邮件收件人发送一个群组回复，请按 `g`。

> 注意
>
> 在默认情况下，mutt 使用 vi(1)[1778] 编辑器来创建和回复邮件。每个用户可以通过在他们的主目录中创建或编辑 **.muttrc** 并设置编辑器变量或设置 `EDITOR` 环境变量来更改。关于配置 mutt 的更多信息，请参考 http://www.mutt.org/。

要编写一个新的邮件，按 `m`。在给出一个有效的主题后，mutt 将启动 vi(1)[1779]，这样邮件就可以被写入。邮件内容完成后，保存并退出 vi，mutt 将恢复，显示将要发送的邮件的摘要屏幕。为了发送邮件，请按 `y`。下面是一个摘要的屏幕例子：

[1777] https://www.freebsd.org/cgi/man.cgi?query=mail&sektion=1&format=html
[1778] https://www.freebsd.org/cgi/man.cgi?query=vi&sektion=1&format=html
[1779] https://www.freebsd.org/cgi/man.cgi?query=vi&sektion=1&format=html



mutt 包含详尽的帮助，可以从大多数菜单中按 ?。顶行还显示了适当的键盘快捷键。

31.10.3.alpine

alpine 是针对初级用户的，但也包括一些高级功能。

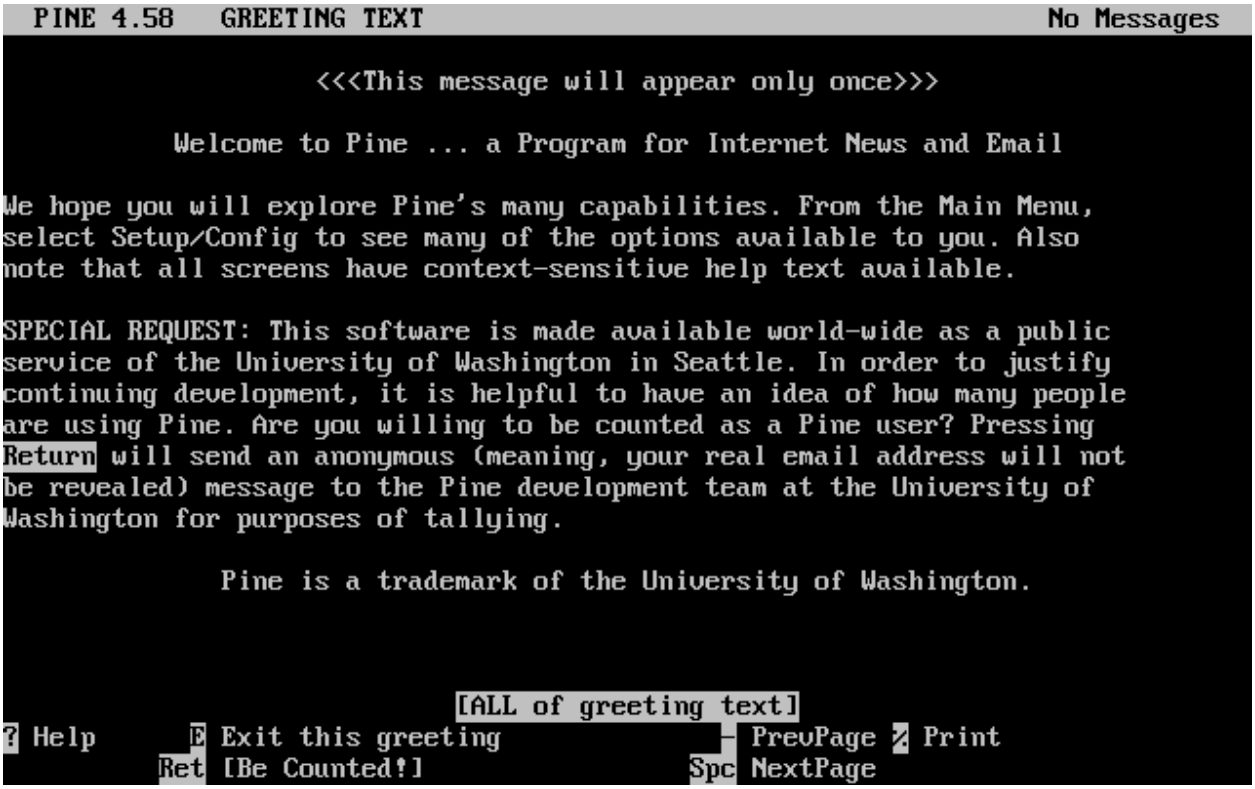
> 警告
>
> alpine 在过去曾发现过几个远程漏洞，这些漏洞允许远程攻击者通过发送特别准备的电子邮件的动作，以本地系统的用户身份执行任意代码。虽然已知的问题已经被修复，但 alpine 的代码是以不安全的风格编写的，FreeBSD 安全长官认为可能还有其他未被发现的漏洞。用户应自行承担安装 alpine 的风险。

当前版本的 alpine 可以通过 port mail/alpine[1780] 来安装。port 安装完毕后，就可以通过执行以下命令来启动 alpine：

```
% alpine
```

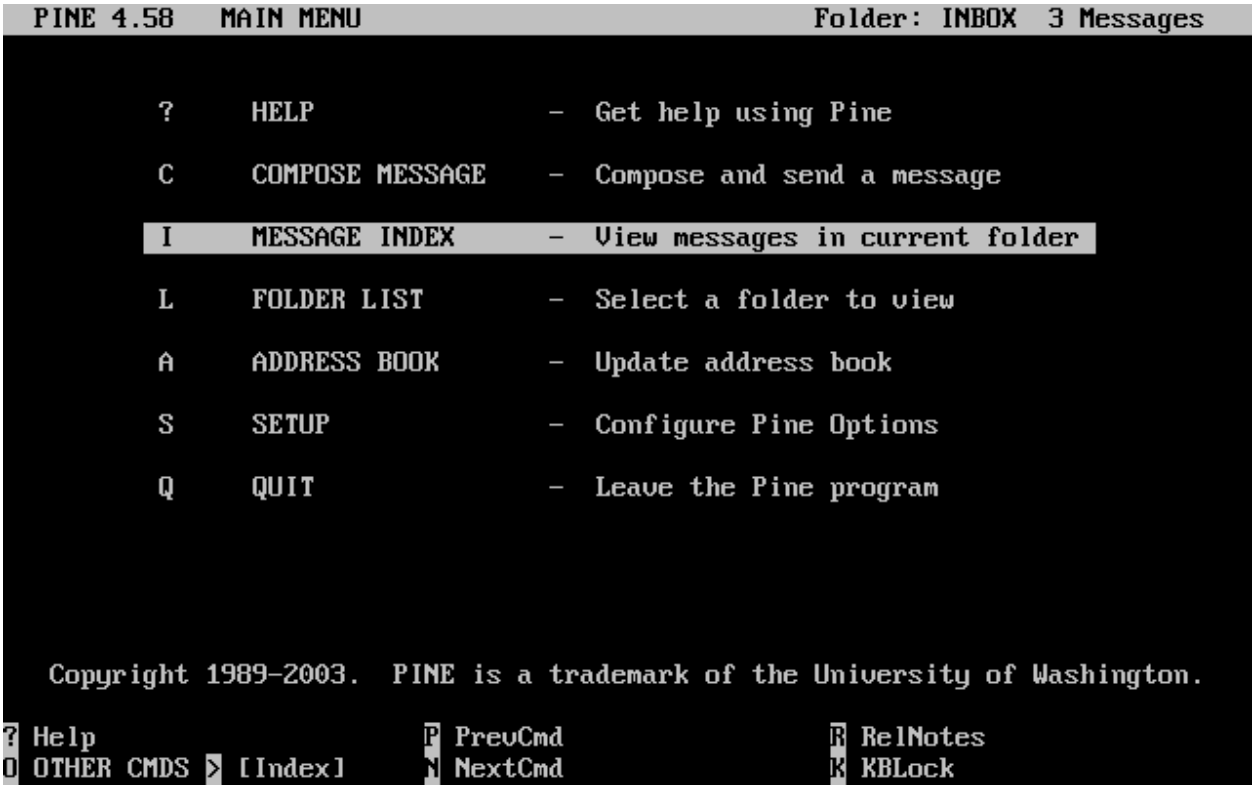
alpine 第一次运行时，会显示一个带有简单介绍的问候页面，以及一个来自 alpine 开发团队的请求，即发送一个匿名的电子邮件，让他们判断有多少用户在使用他们的客户端。要发送这个匿名信息，请按回车键。或者，按 E 键退出问候语而不发送匿名信息。下面是一个问候语页面的例子：

[1780] https://cgit.freebsd.org/ports/tree/mail/alpine/pkg-descr

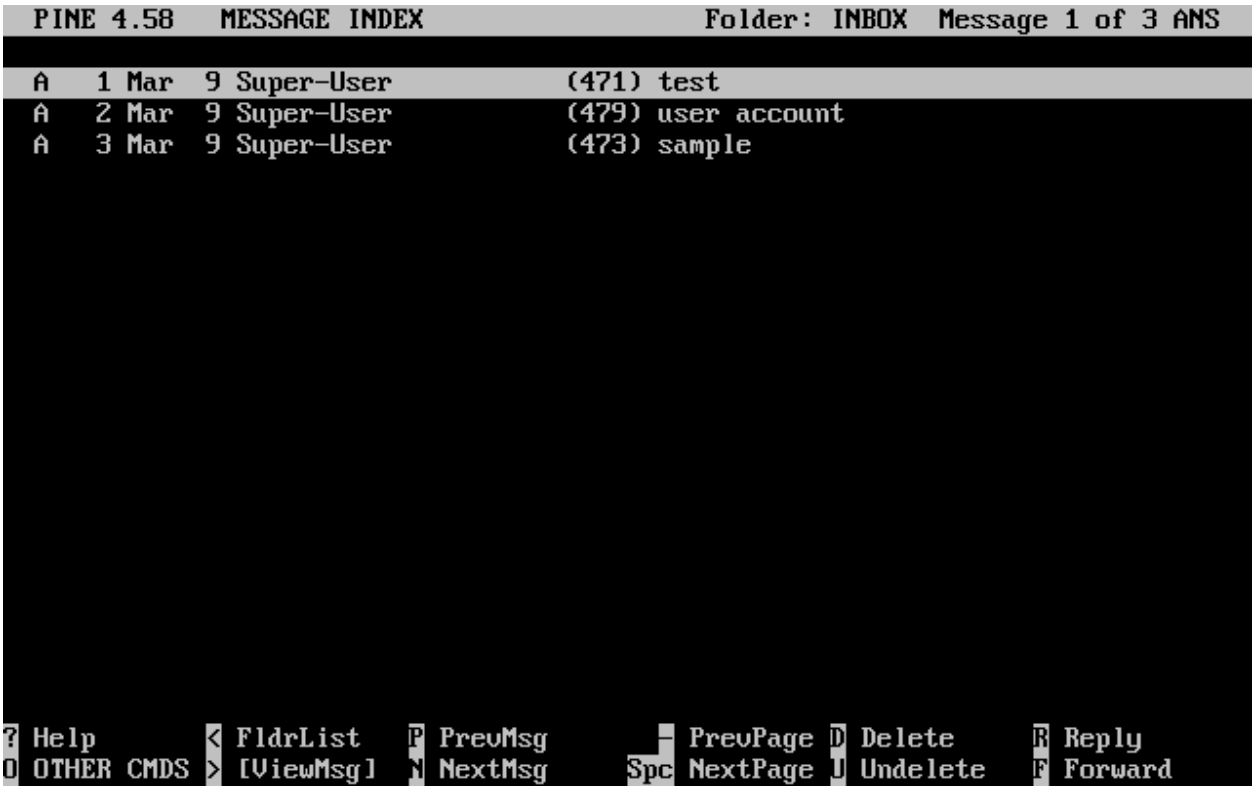


然后出现主菜单，可以用光标键进行导航。这个主菜单为撰写新邮件、浏览邮件目录和管理地址簿条目提供快捷方式。在主菜单下面，显示了相关的键盘快捷键，以执行手头任务的特定功能。

alpine 打开的默认目录是 **inbox**。要查看邮件索引，请按 `I`，或选择下面显示的选项 MESSAGE INDEX：



信息索引显示了当前目录下的信息，可以用光标键进行导航。突出显示的信息可以通过按回车键来阅读。

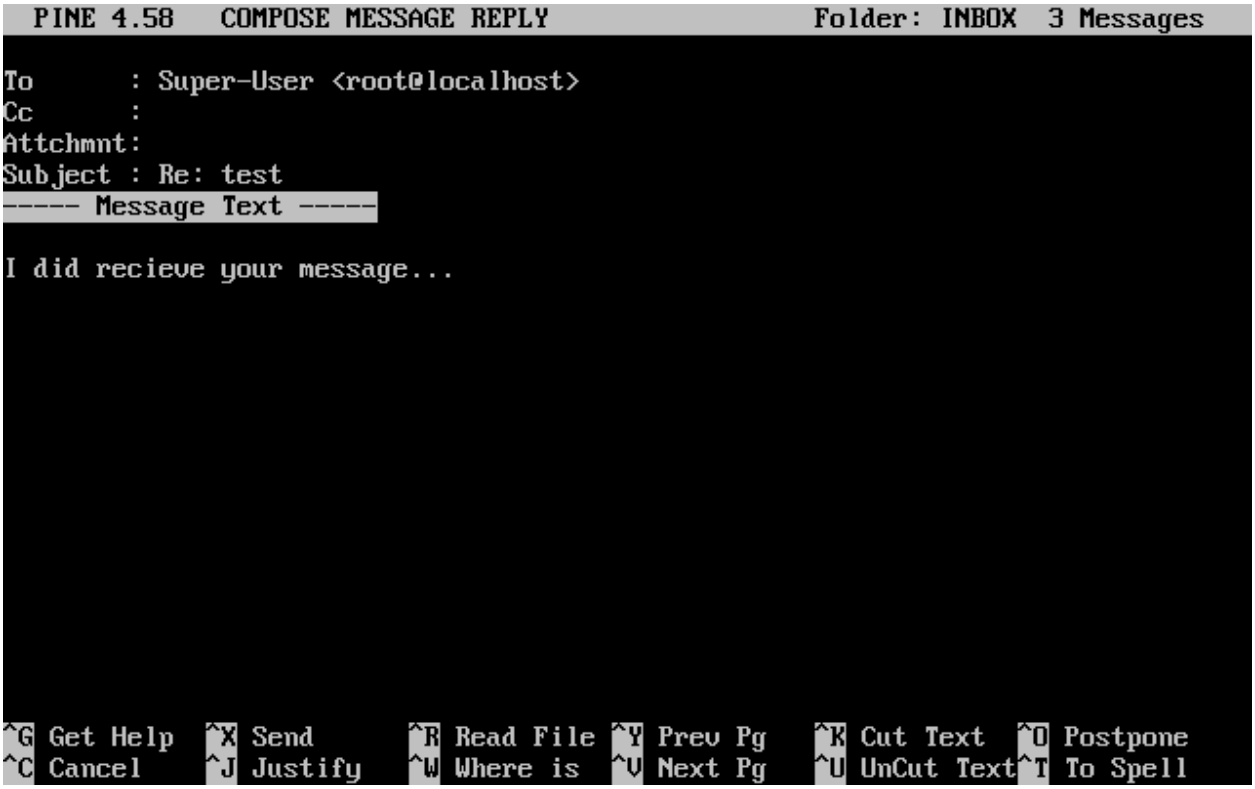


在下面的截图中，alpine 显示的是一个样本消息。上下文的键盘快捷方式显示在屏幕的底部。其中一个快捷键的例子是 `r`，它告诉 MUA 回复正在显示的当前消息。

在 alpine 中回复邮件是使用 pico 编辑器来完成的，这个编辑器默认是和 alpine 一起安装的。pico 使得浏览邮件变得很容易，对新手来说比 vi(1)[1781] 或 mail(1)[1782] 更容易使用。回复完成以后，就可以按 `Ctrl+X` 来发送消息。在发送消息之前，alpine 会要求确认。

[1781] https://www.freebsd.org/cgi/man.cgi?query=vi&sektion=1&format=html

[1782] https://www.freebsd.org/cgi/man.cgi?query=mail&sektion=1&format=html



alpine 可以使用主菜单中的 SETUP 选项进行定制。更多信息请查阅 http://www.washington.edu/alpine/。

31.11.使用 fetchmail

fetchmail 是一个全功能的 IMAP 和 POP 客户端。它允许用户自动从远程 IMAP 和 POP 服务器下载邮件，并将其保存到本地邮箱中，以便更容易地访问。fetchmail 可以通过 port mail/fetchmail[1783] 安装，并提供各种功能，包括：

- 支持 POP3、APOP、KPOP、IMAP、ETRN 和 ODMR 协议。
- 能够使用 SMTP 转发邮件，这使得过滤、转发和别名能够正常运行。
- 可在守护模式下运行，定期检查新邮件。
- 可以检索多个邮箱，并根据配置将其转发给不同的本地用户。

本节解释了 fetchmail 的一些基本功能。这个工具需要在用户的主目录下有一个 **.fetchmailrc** 配置，以便正确运行。这个文件包括服务器信息以及登录凭据。由于该文件内容的敏感性，建议使用以下命令使其只能由用户阅读：

```
% chmod 600 .fetchmailrc
```

[1783] https://cgit.freebsd.org/ports/tree/mail/fetchmail/pkg-descr

下面的 **.fetchmailrc** 是一个使用 POP 下载单个用户邮箱的例子。它告诉 fetchmail 使用用户名 `joesoap` 和密码 `XXX` 连接到 `example.com`。这个例子假设在本地系统中存在 `joesoap` 这个用户。

```
poll example.com protocol pop3 username "joesoap" password "XXX"
```

下一个例子连接到多个 POP 和 IMAP 服务器，并在适用时重定向到不同的本地用户名：

```
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 可以在守护模式下运行，方法是用 `-d` 运行，然后是 fetchmail 应该轮询 **.fetchmailrc** 中列出的服务器的时间间隔（秒）。下面的例子将 fetchmail 配置为每 600 秒轮询一次：

```
% fetchmail -d 600
```

关于 fetchmail 的更多信息可以在 http://www.fetchmail.info/ 查看。

31.12.使用 procmail

procmail 是一个强大的应用程序，用于过滤进入的邮件。它允许用户定义“规则”，这些规则可以与传入的邮件相匹配，以执行特定的功能或将邮件改道到其他邮箱或电子邮件地址。procmail 可以通过 port mail/procmail[1784] 安装。安装后，它可以直接集成到大多数 MTA 中。更多信息请查阅 MTA 文档。另外，也可以通过在用户的主目录下的 **.forward** 中添加以下一行来进行集成 procmail：

```
"|exec /usr/local/bin/procmail || exit 75"
```

下面的部分显示了一些基本的 procmail 规则，以及对它们的作用的摘要。规则必须插入到 **.procmailrc** 中，它必须位于用户的主目录中：

这些规则的大部分可以在 procmailex(5)[1785] 中找到。

要把所有来自 user@example.com 的邮件转发到外部地址 goodmail@example2.com：

```
:0
* ^From.*user@example.com
! goodmail@example2.com
```

要转发所有短于 1000 字节的邮件到外部地址 goodmail@example2.com：

[1784] https://cgit.freebsd.org/ports/tree/mail/procmail/pkg-descr

[1785] https://www.freebsd.org/cgi/man.cgi?query=procmailex&sektion=5&format=html

```
:0
* < 1000
! goodmail@example2.com
```

要将所有发送到 alternate@example.com 的邮件发送到一个叫做 **alternate** 的邮箱：

```
:0
* ^TOalternate@example.com
alternate
```

将所有主题为“Spam”（垃圾邮件）的邮件发送至 **/dev/null**：

```
:0
^Subject:.*Spam
/dev/null
```

一个有用的配置，可以解析进入的 `FreeBSD.org` 邮件列表，并将每个列表放在它自己的邮箱中：

```
:0
* ^Sender:.owner-freebsd-\/[^@]+@FreeBSD.ORG
{
    LISTNAME=${MATCH}
    :0
    * LISTNAME??^\/[^@]+
    FreeBSD-${MATCH}
}
```

第三十二章网络服务器

32.1. 概述

本章介绍了 UNIX® 系统上一些更常用的网络服务。这包括安装、配置、测试和维护许多不同类型的网络服务。本章中包含了示例配置文件以供参考。

在本章结束时，你将知道：

- 如何管理 inetd 守护程序。
- 如何设置网络文件系统（NFS）。
- 如何设置网络信息服务器（NIS）以集中和共享用户帐户。
- 如何将 FreeBSD 设置充当为 LDAP 服务器或客户端。
- 如何使用 DHCP 进行自动网络设置。
- 如何设置域名服务器（DNS）。
- 如何设置 Apache HTTP 服务器。
- 如何设置文件传输协议（FTP）服务器。
- 如何使用 Samba 为 Windows® 客户端设置文件和打印服务器。
- 如何同步时间和日期，并使用网络时间协议（NTP）设置时间服务器。
- 如何设置 iSCSI。

本章假定你具备以下基本知识：

- **/etc/rc** 脚本。
- 网络术语。
- 安装其他第三方软件（安装应用程序：软件包和 port[1786]）。

[1786] https://docs.freebsd.org/en/books/handbook/ports/index.html#ports

32.2. inetd 超级服务器

守护进程 inetd(8)[1787] 有时被称为超级服务器，因为它管理了许多服务的连接。只需启动 inetd 服务，而无需启动多个应用程序。当收到由 inetd 管理的服务的连接时，它会确定连接的目标程序，为该程序生成一个进程，并向该程序委派一个套接字。与在独立模式下单独运行每个守护程序相比，对未大量使用的服务使用 inetd 可以减少系统负载。

inetd 主要用于生成其他守护程序，但有几个简单的协议在内部处理，例如 chargen、auth、time、echo、discard 和 daytime。

本节介绍配置 inetd 的基础知识。

32.2.1. 配置文件

inetd 的配置是通过编辑 **/etc/inetd.conf** 来完成的。此配置文件的每一行都表示一个可由 inetd 启动的应用程序。默认情况下，每行都以注释 (#) 开头，这意味着 inetd 不侦听任何应用程序。要将 inetd 配置为侦听应用程序的连接，请删除该应用程序的行首处的 #。

保存编辑后，通过编辑 **/etc/rc.conf** 将 inetd 配置为在系统启动时启动：

```
inetd_enable="YES"
```

要立即启动 inetd，以便它侦听你配置的服务，请键入：

```
# service inetd start
```

inetd 启动后，每当对 **/etc/inetd.conf** 进行修改时，都需要通知它：

例 **44.**重新加载 **inetd** 配置文件

```
# service inetd reload
```

通常情况下，一个应用程序的默认条目除了删除 # 外，不需要其他编辑。在某些情况下，编辑默认条目可能是合适的。

例如，这是 ftpd(8)[1788] 在 IPv4 上的默认条目：

```
ftp     stream  tcp     nowait  root    /usr/libexec/ftpd       ftpd -l
```

条目中的七列如下所示：

[1787] https://www.freebsd.org/cgi/man.cgi?query=inetd&sektion=8&format=html

[1788] https://www.freebsd.org/cgi/man.cgi?query=ftpd&sektion=8&format=html

```
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
```

此处：

- **service-name**

  要启动的守护程序的服务名称。它必须与 **/etc/services** 中列出的服务相对应。这将确定 inetd 侦听与该服务的传入连接的端口。使用自定义服务时，必须首先将其添加到 **/etc/services**。

- **socket-type**

  无论 `stream`、`dgram`、`raw` 或 `seqpacket`。TCP 连接使用 `stream`，UDP 服务使用 `dgram`。

- **protocol**

  使用以下协议名称之一：

协议名称	解释
tcp 或 tcp4	TCP IPv4
udp 或 udp4	UDP IPv4
tcp6	TCP IPv6
udp6	UDP IPv6
tcp46	TCP IPv4 和 IPv6
udp46	UDP IPv4 和 IPv6

- **{wait|nowait}[/max-child[/max-connections-per-ip-per-minute[/max-child-per-ip]]]**

  在此字段中，必须指定 `wait` 或 `nowait`。`max-child`、`max-connections-per-ip-per-minute` 和 `max-child-per-ip` 是可选的。

  `wait|nowait` 指示服务是否能够处理自己的套接字。`dgram` socket 类型必须使用 `wait` 而 `stream` 守护程序（通常是多线程的）应使用 `nowait`。`wait` 通常将多个套接字交给单个守护程序，同时 `nowait` 为每个新套接字生成一个子守护进程。

  由 `max-child` 设置可生成的最大子守护程序数。例如，要限制守护程序的十个实例，请在 `nowait` 之后放置一个 `/10`。指定 `/0` 允许无限数量的子项。

  `max-connections-per-ip-per-minute` 限制每分钟来自任何特定 IP 地址的连接数。一旦达到限额，来自这个 IP 地址的进一步连接将被放弃，直到一分钟结束。例如，一个 `/10` 的值将限制任何特定的 IP 地址每分钟 10 次连接尝试。`max-child-per-ip` 限制在任何时候可以代表任何单一 IP 地址启动的子进程的数量。这些选项可以限制过度的资源消耗，有助于防止拒绝服务攻击。

在 fingerd(8)[1789] 的默认设置中可以看到一个示例：

```
finger stream tcp nowait/3/10 nobody /usr/libexec/fingerd fingerd -k -s
```

- **user**

  守护进程运行时使用的用户名。守护程序通常以 `root` 、`daemon` 或 `nobody` 运行。

- **server-program**

  守护程序的完整路径。如果守护程序是 inetd 内部提供的服务，请使用 `internal`。

- **server-program-arguments**

  用于指定在调用时要传递给守护程序的任何命令参数。如果守护程序是内部服务，请使用 `internal`。

32.2.2. 命令行选项

与大多数服务器守护程序一样，inetd 具有许多可用于修改其行为的选项。默认情况下，inetd 以 `-wW -C 60` 启动。这些选项为所有服务（包括内部服务）启用 TCP wrapper，并防止任何 IP 地址每分钟请求任何服务超过 60 次。

要更改传递给 inetd 的默认选项，请在 **/etc/rc.conf** 中添加一个 `inetd_flags` 条目。如果 inetd 已在运行，请使用 `service inetd restart` 重新启动它。

可用的速率限制选项包括：

- **-c maximum**

  指定每个服务的默认最大同时调用次数，其中默认值为无限制。可以使用 **/etc/inetd.conf** 的中 `max-child`，在每个服务上覆盖。

- **-C rate**

  指定每分钟可以从单个 IP 地址调用服务的默认最大次数。可以使用 **/etc/inetd.conf** 中的 `max-connections-per-ip-per-minute`，在每个服务上覆盖。

- **-R rate**

  指定一分钟内可以调用服务的最大次数，其中缺省值为 `256`。`rate` 为 `0` 允许无限数量。

- **-s maximum**

  指定在任何时候都可以从单个 IP 地址调用服务的最大次数，其中默认值为无限制。可以使用 **/etc/inetd.conf** 中的 `max-child-per-ip`，在每个服务上覆盖。

其他可用选项请参阅 inetd(8)[1790]，以获取完整的选项列表。

[1789] https://www.freebsd.org/cgi/man.cgi?query=fingerd&sektion=8&format=html

[1790] https://www.freebsd.org/cgi/man.cgi?query=inetd&sektion=8&format=html

32.2.3. 安全注意事项

许多可以由 inetd 管理的守护程序都不具有安全意识。某些守护程序（如 fingerd）可以提供可能对攻击者有用的信息。仅启用所需的服务，并监控系统是否存在过多的连接尝试。`max-connections-per-ip-per-minute`、`max-child` 和 `max-child-per-ip` 可用于限制此类攻击。

默认情况下，TCP wrapper 处于启用状态。请查阅 hosts_access(5)[1791] 以获取有关对各种 inetd 调用的守护程序设置 TCP 限制的更多信息。

32.3. 网络文件系统（NFS）

FreeBSD 支持网络文件系统（NFS），它允许服务器通过网络与客户机共享目录和文件。使用 NFS，用户和程序可以访问远程系统上的文件，它们就像存储在本地一样。

NFS 有许多实际用途。一些较常见的用途包括：

- 将在每个客户端上重复的数据保存在一个位置，并由网络上的客户端访问。
- 多个客户端可能都需要访问目录 **/usr/ports/distfiles**。共享该目录允许快速访问源文件，而无需将它们下载到每个客户端。
- 在大型网络上，配置存储所有用户主目录的中央 NFS 服务器通常更方便。用户可以登录到网络上任何位置的客户端，并可以访问其主目录。
- NFS 导出的管理得到简化。例如，只有一个文件系统必须设置安全或备份策略。
- 可移动媒体存储设备可由网络上的其他计算机使用。这减少了整个网络中的设备数量，并提供一个集中的位置来管理其安全性。从集中安装设备在多台计算机上安装软件通常更方便。

NFS 由一台服务器和一个或多个客户端组成。客户端远程访问存储在服务器计算机上的数据。为了使其正常运行，必须配置和运行一些进程。

这些守护程序必须在服务器上运行：

守护进程	说明
nfsd	NFS 守护程序，用于处理来自 NFS 客户端的请求。
mountd	NFS 挂载守护程序，它执行从 nfsd 收到的请求。
rpcbind	此守护程序允许 NFS 客户端发现 NFS 服务器正在使用的端口。

在客户端上运行 nfsiod(8)[1792] 可以提高性能，但非必需。

[1791] https://www.freebsd.org/cgi/man.cgi?query=hosts_access&sektion=5&format=html

[1792] https://www.freebsd.org/cgi/man.cgi?query=nfsiod&sektion=8&format=html

32.3.1. 配置服务器

NFS 服务器在 **/etc/exports** 中指定共享的文件系统。此文件中的每一行都指定要导出的文件系统、哪些客户端有权访问该文件系统以及任何访问选项。向此文件添加条目时，每个导出的文件系统、其属性和允许的主机必须位于一行上。如果该条目中未列出任何客户机，则网络上的任何客户机都可以挂载该文件系统。

以下 **/etc/export** 条目演示了如何导出文件系统。可以修改这些示例以匹配读取器网络上的文件系统和客户端名称。在此文件中可以使用许多选项，但此处仅提及少数几个选项。有关选项的完整列表见 exports(5)[1793]。

此示例演示如何将 **/cdrom** 导出到三个名为 *alpha*、*bravo* 和 *charlie* 的主机：

```
/cdrom -ro alpha bravo charlie
```

`-ro` 参数使文件系统为只读，防止客户端对导出的文件系统进行任何更改。此示例假定主机名位于 DNS 或 **/etc/hosts** 中。如果网络没有 DNS 服务器，请参考 hosts(5)[1794]。

下一个示例按 IP 地址将 **/home** 导出到三个客户端。这对于没有 DNS 或 **/etc/hosts** 条目的网络非常有用。`-alldirs` 标志允许子目录成为挂载点。换句话说，它不会自动挂载子目录，但会允许客户端按需挂载所需的目录。

```
/usr/home  -alldirs  10.0.0.2 10.0.0.3 10.0.0.4
```

下面这个例子导出 **/a**，这样来自不同领域的两个客户可以访问该文件系统。`-maproot=roo` 允许远程系统的 `root` 以 `root` 身份在导出的文件系统上写数据。如果没有指定 `-maproot=root`，客户端的 `root` 用户将被映射到服务器的 `nobody` 账户上，并将受到为 `nobody` 定义的访问限制。

```
/a  -maproot=root  host.example.com box.example.org
```

每个文件系统一次只能指定一个客户机。例如，如果 **/usr** 是单个文件系统，则这些条目无效，因为这两个条目指定相同的主机：

```
# Invalid when /usr is one file system
/usr/src   client
/usr/ports client
```

这种情况的正确格式是使用一个条目：

```
/usr/src /usr/ports  client
```

以下是有效导出列表的示例，其中 **/usr** 和 **/export** 是本地文件系统：

[1793] https://www.freebsd.org/cgi/man.cgi?query=exports&sektion=5&format=html
[1794] https://www.freebsd.org/cgi/man.cgi?query=hosts&sektion=5&format=html

```
# 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
```

要在引导时启用 NFS 服务器所需的进程，请将以下选项添加到 **/etc/rc.conf**：

```
rpcbind_enable="YES"
nfs_server_enable="YES"
mountd_enable="YES"
```

现在可以通过运行以下命令启动服务器：

```
# service nfsd start
```

每当 NFS 服务器启动时，mountd 也会自动启动。但是，mountd 仅在启动时读取 **/etc/export**。要使后续的 **/etc/export** 编辑立即生效，请强制重载以重新读取它：

```
# service mountd reload
```

请参考 nfsv4(4)[1795] 以了解对 NFS 第 4 版配置的说明。

32.3.2. 配置客户端

要启用 NFS 客户端，请在每个客户端的 **/etc/rc.conf** 中设置此选项：

```
nfs_client_enable="YES"
```

然后，在每个 NFS 客户端上运行以下命令：

```
# service nfsclient start
```

现在客户端拥有挂载远程文件系统所需的一切。在这些示例中，服务器的名称为 `server`，客户端的名称为 `client`。要将 `server` 上的 **/home** 目录挂载到 `client` 的挂载点 **/mnt**：

```
# mount server:/home /mnt
```

[1795] https://www.freebsd.org/cgi/man.cgi?query=nfsv4&sektion=4&format=html

现在，**/home** 中的文件和目录将在 `client` 的 **/mnt** 目录中可用。

要在每次客户端引导时挂载远程文件系统，请将其添加到 **/etc/fstab**：

```
server:/home    /mnt    nfs rw  0   0
```

请参考 fstab(5)[1796] 获取所有可用选项的说明。

32.3.3. 锁定

某些应用程序需要文件锁定才能正常运行。要启用锁定，请将以下行添加到客户端和服务器上的 **/etc/rc.conf** 中：

```
rpc_lockd_enable="YES"
rpc_statd_enable="YES"
```

然后启动应用程序：

```
# service lockd start
# service statd start
```

如果服务器上不需要锁定，则可以将 NFS 客户端配置为通过在运行挂载时包含 `-L` 来在本地锁定。有关详细信息，请参阅 mount_nfs(8)[1797]。

32.3.4. 使用 autofs(5)[Page 730, 1798] 自动挂载

> 注意
>
> 自动挂载工具 autofs(5)[1799] 从 FreeBSD 10.1-RELEASE 开始受支持。要在旧版本的 FreeBSD 中使用自动挂载器功能，请改用 amd(8)[1800]。本章仅介绍自动挂载器 autofs(5)[1801]。

autofs(5)[1802] 工具是多个组件的通用名称，每当访问远程和本地文件系统中的文件或目录时，这些组件一起允许自动挂载远程和本地文件系统。它由内核组件 autofs(5)[1803] 和几个用户空间应用程序组成：automount(8)[1804]、automountd(8)[1805] 和 autounupd(8)[1806]。它作为旧的 FreeBSD release 中 amd(8)[1807] 的替代方案。

[1796] https://www.freebsd.org/cgi/man.cgi?query=fstab&sektion=5&format=html
[1797] https://www.freebsd.org/cgi/man.cgi?query=mount_nfs&sektion=8&format=html
[1798] https://www.freebsd.org/cgi/man.cgi?query=autofs&sektion=5&format=html
[1799] https://www.freebsd.org/cgi/man.cgi?query=autofs&sektion=5&format=html
[1800] https://www.freebsd.org/cgi/man.cgi?query=amd&sektion=8&format=html
[1801] https://www.freebsd.org/cgi/man.cgi?query=autofs&sektion=5&format=html
[1802] https://www.freebsd.org/cgi/man.cgi?query=autofs&sektion=5&format=html
[1803] https://www.freebsd.org/cgi/man.cgi?query=autofs&sektion=5&format=html
[1804] https://www.freebsd.org/cgi/man.cgi?query=automount&sektion=8&format=html
[1805] https://www.freebsd.org/cgi/man.cgi?query=automountd&sektion=8&format=html
[1806] https://www.freebsd.org/cgi/man.cgi?query=autounmountd&sektion=8&format=html
[1807] https://www.freebsd.org/cgi/man.cgi?query=amd&sektion=8&format=html

amd 仍然用于向后兼容目的，因为两者使用不同的映射格式；autofs 使用的自动加载器与其他 SVR4 自动加载器相同，例如 Solaris、MacOS X 和 Linux 中的自动加载器。

autofs(5)[1808] 虚拟文件系统通过 automount(8)[1809] 挂载在指定的挂载点上，通常在引导期间调用。

每当进程尝试访问 autofs(5)[1810] 挂载点内的文件时，内核都会通知守护程序 automountd(8)[1811] 并暂停触发进程。守护程序 automountd(8)[1812] 将处理内核请求，通过找到正确的映射并据此挂载文件系统，然后向内核发出释放被阻止进程的信号。守护程序 autounmountd(8)[1813] 会在一段时间后自动卸载自动挂载的文件系统，除非它们仍在使用中。

autofs 主配置文件是 **/etc/auto_master**。它将单个映射分配给顶级挂载。有关 **auto_master** 和映射语法的说明，请参阅 auto_master(5)[1814]。

在 **/net** 上有一个特殊的自动挂载器映射。当一个文件在这个目录中被访问时，autofs(5)[1815] 会查找相应的远程挂载并自动挂载它。例如，尝试访问 **/net/foobar/usr** 中的文件会告诉 automountd(8)[1816] 从主机 `foobar` 挂载的 **/usr** 导出。

例 **45.**使用 autofs(5)[1817] 挂载导出

在此示例中，`showmount -e` 显示可以从 `foobar` NFS 服务器挂载的导出文件系统：

```
% showmount -e foobar
Exports list on foobar:
/usr                                10.10.10.0
/a                                  10.10.10.0
% cd /net/foobar/usr
```

`showmount` 的输出显示 **/usr** 是一个导出目录。当改变目录到 **/host/foobar/usr** 时，automountd(8)[1818] 会截获请求并尝试解析主机名 `foobar`。如果成功，automountd(8)[1819] 会自动挂载源导出。

要在引导时启用 autofs(5)[1820]，请将以下行添加到 **/etc/rc.conf**：

```
autofs_enable="YES"
```

autofs(5)[1821] 可以通过运行来启动：

[1808] https://www.freebsd.org/cgi/man.cgi?query=autofs&sektion=5&format=html
[1809] https://www.freebsd.org/cgi/man.cgi?query=automount&sektion=8&format=html
[1810] https://www.freebsd.org/cgi/man.cgi?query=autofs&sektion=5&format=html
[1811] https://www.freebsd.org/cgi/man.cgi?query=automountd&sektion=8&format=html
[1812] https://www.freebsd.org/cgi/man.cgi?query=automountd&sektion=8&format=html
[1813] https://www.freebsd.org/cgi/man.cgi?query=autounmountd&sektion=8&format=html
[1814] https://www.freebsd.org/cgi/man.cgi?query=auto_master&sektion=5&format=html
[1815] https://www.freebsd.org/cgi/man.cgi?query=autofs&sektion=5&format=html
[1816] https://www.freebsd.org/cgi/man.cgi?query=automountd&sektion=8&format=html
[1817] https://www.freebsd.org/cgi/man.cgi?query=autofs&sektion=5&format=html
[1818] https://www.freebsd.org/cgi/man.cgi?query=automountd&sektion=8&format=html
[1819] https://www.freebsd.org/cgi/man.cgi?query=automountd&sektion=8&format=html
[1820] https://www.freebsd.org/cgi/man.cgi?query=autofs&sektion=5&format=html
[1821] https://www.freebsd.org/cgi/man.cgi?query=autofs&sektion=5&format=html

```
# service automount start
# service automountd start
# service autounmountd start
```

autofs(5)[1822] 映射格式与其他操作系统相同。来自其他来源的有关此格式的信息可能很有用，例如 Mac OS X 文档[1823]。

有关详细信息，请参阅 automount(8)[1824] 、automountd(8)[1825]、autounmountd(8)[1826] 和 auto_master(5)[1827] 手册页。

32.4. 网络信息系统（NIS）

网络信息系统（NIS）旨在集中管理类 UNIX® 系统，如 Solaris™、HP-UX、AIX®、Linux、NetBSD、OpenBSD 和 FreeBSD。NIS 最初被称为黄页（Yellow Pages），但由于商标问题，名称已更改。这就是 NIS 命令以 `yp` 开头的原因。

NIS 是基于远程过程调用（RPC）的客户端/服务器系统，它允许 NIS 域中的一组计算机共享一组通用的配置文件。这允许系统管理员仅使用最少的配置数据设置 NIS 客户端系统，并从单个位置添加、删除或修改配置数据。

FreeBSD 使用第 2 版 NIS 协议。

[1822] https://www.freebsd.org/cgi/man.cgi?query=autofs&sektion=5&format=html
[1823] http://web.archive.org/web/20160813071113/http://images.apple.com/business/docs/Autofs.pdf
[1824] https://www.freebsd.org/cgi/man.cgi?query=automount&sektion=8&format=html
[1825] https://www.freebsd.org/cgi/man.cgi?query=automountd&sektion=8&format=html
[1826] https://www.freebsd.org/cgi/man.cgi?query=autounmountd&sektion=8&format=html
[1827] https://www.freebsd.org/cgi/man.cgi?query=auto_master&sektion=5&format=html

32.4.1. NIS 术语和流程

表 28.1 总结了 NIS 使用的术语和重要过程：

表 24. NIS 术语

术语	说明
NIS 域名	NIS 服务器和客户端共享一个 NIS 域名。通常，此名称与 DNS 没有任何关系。
rpcbind(8)[1828]	此服务启用 RPC，并且必须正在运行才能运行 NIS 服务器或充当 NIS 客户端。
ypbind(8)[1829]	此服务将 NIS 客户端绑定到其 NIS 服务器。它将采用 NIS 域名并使用 RPC 连接到服务器。它是 NIS 环境中客户端/服务器通信的核心。如果此服务未在客户端计算机上运行，则它将无法访问 NIS 服务器。
ypserv(8)[1830]	这是 NIS 服务器的过程。如果此服务停止运行，服务器将不再能够响应 NIS 请求，因此希望有一个从属服务器可以接管。一些非 FreeBSD 客户端不会尝试使用从属服务器重新连接，ypbind 进程可能需要在这些客户端上重新启动。
rpc.yppasswdd(8)[1831]	此过程仅在 NIS 主服务器上运行。此守护程序允许 NIS 客户端更改其 NIS 密码。如果此守护程序未运行，则用户必须登录到 NIS 主服务器并在那里更改其密码。

32.4.2. 机器类型

NIS 环境中有三种类型的主机：

- NIS 主服务器

  此服务器充当主机配置信息的中央存储库，并维护所有 NIS 客户机使用的文件的权威副本。NIS 客户端使用的 **passwd**、**group** 和其他各种文件存储在主服务器上。虽然一台计算机可能是多个 NIS 域的 NIS 主服务器，但本章不介绍这种类型的配置，因为它假定的是一个相对较小的 NIS 环境。

- NIS 从属服务器

  NIS 从属服务器维护 NIS 主服务器数据文件的副本，以提供冗余。从属服务器还有助于平衡主服务器的负载，因为 NIS 客户端始终连接到首先响应的 NIS 服务器。

- NIS 客户端

  NIS 客户端在登录期间对 NIS 服务器进行身份验证。

可以使用 NIS 共享许多文件中的信息。通常通过 NIS 共享 **master.passwd**、**group** 和 **hosts** 文件。每当客户端上的进程需要通常在本地这些文件中找到的信息时，它就会向绑定到的 NIS 服务器发出查询。

[1828] https://www.freebsd.org/cgi/man.cgi?query=rpcbind&sektion=8&format=html

[1829] https://www.freebsd.org/cgi/man.cgi?query=ypbind&sektion=8&format=html

[1830] https://www.freebsd.org/cgi/man.cgi?query=ypserv&sektion=8&format=html

[1831] https://www.freebsd.org/cgi/man.cgi?query=rpc.yppasswdd&sektion=8&format=html

32.4.3. 规划注意事项

本节介绍一个 NIS 示例环境，它由 15 台 FreeBSD 机器组成，没有集中的管理点。每台机器都有自己的 **/etc/passwd** 和 **/etc/master.passwd**。这些文件仅通过手动干预保持彼此同步。目前，将用户添加到实验室时，必须在所有 15 台计算机上重复该过程。

实验室的配置如下：

计算机名称	IP 地址	计算机角色
`ellington`	`10.0.0.2`	NIS 主服务器
`coltrane`	`10.0.0.3`	NIS 从属服务器
`basie`	`10.0.0.4`	教师工作站
`bird`	`10.0.0.5`	客户端
`cli[1-11]`	`10.0.0.[6-17]`	其他客户端

如果这是第一次开发 NIS 方案，则应提前进行彻底规划。无论网络规模如何，作为规划过程的一部分，都需要做出一些决策。

32.4.3.1. 选择 NIS 域名

当客户端广播其信息请求时，它包括它所属的 NIS 域的名称。这就是一个网络上的多个服务器如何判断哪个服务器应该回答哪个请求。将 NIS 域名视为一组主机的名称。

某些组织选择使用其互联网域名作为其 NIS 域名。不建议这样做，因为在尝试调试网络问题时，这可能会导致混淆。NIS 域名在网络中应该是唯一的，如果它描述了它所代表的计算机组，则会很有帮助。例如 Acme Inc. 的艺术部门可能属于 NIS 域 “acme-art”。此示例将使用域名 `test-domain`。

但是，一些非 FreeBSD 操作系统要求 NIS 域名与互联网域名相同。如果网络上的一台或多台计算机具有此限制，则必须将互联网域名用作 NIS 域名。

32.4.3.2. 物理服务器要求

在选择要用作 NIS 服务器的计算机时，需要记住几件事。由于 NIS 客户端取决于服务器的可用性，因此请选择不经常重新启动的计算机。理想情况下，NIS 服务器应该是一台独立计算机，其唯一目的是成为 NIS 服务器。如果网络没有被大量使用，则可以将 NIS 服务器放在运行其他服务的计算机上。但是，如果 NIS 服务器变得不可用，它将对所有 NIS 客户端产生负面影响。

32.4.4. 配置 NIS 主服务器

所有 NIS 文件的规范副本都存储在主服务器上。用于存储信息的数据库称为 NIS 映射。在 FreeBSD 中，这些映射存储在 **/var/yp/[domainname]** 中，其中 **[domainname]** 是 NIS 域的名称。由于支持多个域，因此可以有多个目录，每个域一个目录。每个域都有自己独立的映射集。

NIS 主服务器和从属服务器通过 ypserv(8)[1832] 处理所有 NIS 请求。此守护程序负责接收来自 NIS 客户端的传入请求，将请求的域和映射名称转换为相应数据库文件的路径，以及将数据从数据库传输回客户端。

根据环境需要，设置主 NIS 服务器可能相对简单。由于 FreeBSD 提供了内置的 NIS 支持，因此只需通过向 **/etc/rc.conf** 添加以下行来启用它：

```
nisdomainname="test-domain"    ①
nis_server_enable="YES"        ②
nis_yppasswdd_enable="YES"     ③
```

① 这一行将 NIS 域名设置为 `test-domain`。

② 在系统引导时自动启动 NIS 服务器进程。

③ 启用守护程序 rpc.yppasswdd(8)[1833]，以便用户可以从客户端计算机更改其 NIS 密码。

在服务器计算机也是 NIS 客户端的多服务器域中必须小心。通常，强制服务器绑定到自身，而不是允许它们广播绑定请求并可能彼此绑定，这是一个好主意。如果一台服务器出现故障，而其他服务器依赖于它，则可能会导致奇怪的故障模式。最终，所有客户端都将超时并尝试绑定到其他服务器，但所涉及的延迟可能相当大，并且故障模式仍然存在，因为服务器可能会再次相互绑定。

也可以是客户端的服务器通过向 **/etc/rc.conf** 添加以下附加行来强制绑定到特定服务器：

```
nis_client_enable="YES"    ①
nis_client_flags="-S test-domain,server"    ②
```

① 这也使运行客户端内容成为可能。

② 此行将 NIS 域名设置为 `test-domain` 并绑定到自身。

保存编辑后，键入 `/etc/netstart` 以重新启动网络并应用 **/etc/rc.conf** 中定义的值。在初始化 NIS 映射之前，启动 ypserv(8)[1834]：

```
# service ypserv start
```

[1832] https://www.freebsd.org/cgi/man.cgi?query=ypserv&sektion=8&format=html

[1833] https://www.freebsd.org/cgi/man.cgi?query=rpc.yppasswdd&sektion=8&format=html

[1834] https://www.freebsd.org/cgi/man.cgi?query=ypserv&sektion=8&format=html

32.4.4.1. 初始化 NIS 映射

NIS 映射是从 NIS 主服务器上 **/etc** 中的配置文件生成的，但有一个例外：**/etc/master.passwd**。这是为了防止将密码传播到 NIS 域中的所有服务器。因此，在初始化 NIS 映射之前，请配置主密码文件：

```
# cp /etc/master.passwd /var/yp/master.passwd
# cd /var/yp
# vi master.passwd
```

建议删除系统帐户的所有条目以及不需要传播到 NIS 客户端的任何用户帐户，例如 `root` 和任何其他管理帐户。

> 注意
> 通过将 **/var/yp/master.passwd** 的权限设置为 `600`，确保组和其它用户对其不可读。

完成此任务后，初始化 NIS 映射。FreeBSD 用 ypinit(8)[1835] 脚本来完成这个任务。为主服务器生成映射时，请包括 `-m` 并指定 NIS 域名：

```
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 not, 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.
```

这将从 **/var/yp/Makefile.dist** 创建 **/var/yp/Makefile**。缺省情况下，此文件假定环境具有一个只有 FreeBSD 客户端的 NIS 服务器。由于`test-domain` 有一个从属服务器，请在 **/var/yp/Makefile** 中编辑此行，使其以注

[1835] https://www.freebsd.org/cgi/man.cgi?query=ypinit&sektion=8&format=html

释（#）开头：

```
NOPUSH = "True"
```

32.4.4.2. 添加新用户

每次创建新用户时，都必须将用户帐户添加到主 NIS 服务器并重新生成 NIS 映射。在此之前，新用户将无法登录除 NIS 主服务器以外的任何位置。例如，若要将新用户 `jsmith` 添加到 `test-domain` 域中，请在主服务器上运行以下命令：

```
# pw useradd jsmith
# cd /var/yp
# make test-domain
```

也可以使用 `adduser jsmith` 代替 `pw useradd smith` 来添加用户。

32.4.5. 设置 NIS 从属服务器

要设置 NIS 从属服务器，请登录到从属服务器并编辑 **/etc/rc.conf** 作为主服务器。不要生成任何 NIS 映射，因为主服务器上已存在这些映射。在从属服务器上运行 `ypinit` 时，请使用 `-s`（对于从属服务器）而不是 `-m`（对于主服务器）。除了域名之外，此选项还需要 NIS 主机的名称，如以下示例所示：

```
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 not, 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...
```

(continues on next page)

(continued from previous page)

```
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.
Remember to update map ypservers on ellington.
```

这将在从属服务器上生成一个名为 **/var/yp/test-domain** 的目录，其中包含 NIS 主服务器映射的副本。在每个从属服务器上添加这些 **/etc/crontab** 条目将强制从属服务器将其映射与主服务器上的映射同步：

```
20      *       *       *       *       root    /usr/libexec/ypxfr passwd.byname
21      *       *       *       *       root    /usr/libexec/ypxfr passwd.byuid
```

这些条目不是必需的，因为主服务器会自动尝试将任何映射更改推送到其从属服务器。但是，由于客户端可能依赖于从属服务器来提供正确的密码信息，因此建议强制频繁更新密码映射。这在映射更新可能并不总是完成的繁忙网络中尤其重要。

要完成配置，请在从属服务器上运行 `/etc/netstart` 以启动 NIS 服务。

32.4.6. 设置 NIS 客户端

NIS 客户机使用 ypbind(8)[1836] 绑定到 NIS 服务器。此守护程序在本地网络上广播 RPC 请求。这些请求指定在客户端上配置的域名。如果同一域中的 NIS 服务器收到其中一个广播，它将响应 ypbind，这将记录服务器的地址。如果有多个服务器可用，则客户端将使用第一个服务器的地址进行响应，并将其所有 NIS 请求定向到该服务器。客户端将定期自动 ping 服务器，以确保它仍然可用。如果它未能在合理的时间内收到回复，ypbind 会将域标记为未绑定，并再次开始广播，希望找到另一台服务器。

要将 FreeBSD 机器配置为 NIS 客户端：

1. 编辑 **/etc/rc.conf** 并添加以下行，以便在网络启动期间设置 NIS 域名并开始 ypbind(8)[1837]：

```
nisdomainname="test-domain"
nis_client_enable="YES"
```

1. 要从 NIS 服务器导入所有可能的密码条目，请使用 `vipw` 从 **/etc/master.passwd** 中删除除一个用户帐户之外的所有用户帐户。删除帐户时，请记住，至少应保留一个本地帐户，并且此帐户应是 `wheel` 的成员。如果 NIS 存在问题，则可以使用此本地帐户远程登录、成为超级用户并修复问题。在保存编辑之前，请将以下行添加到文件末尾：

```
+::::::::::
```

此行将客户端配置为在 NIS 服务器的密码中为任何人提供有效帐户，以映射客户端上的帐户。通过修改此行，可以通过多种方式配置 NIS 客户端。使用网络组[1838] 中介绍了一种方法。要了解更多详细信息，请参阅 O'Reilly Media 出版的 `Managing NFS and NIS`。

1. 要从 NIS 服务器导入所有可能的组条目，请将此行添加到 **/etc/group**：

```
+:*::
```

要立即启动 NIS 客户端，请以超级用户身份执行以下命令：

```
# /etc/netstart
# service ypbind start
```

完成这些步骤后，在客户端上运行 `ypcat passwd` 应显示服务器的 **passwd** 映射。

[1836] https://www.freebsd.org/cgi/man.cgi?query=ypbind&sektion=8&format=html

[1837] https://www.freebsd.org/cgi/man.cgi?query=ypbind&sektion=8&format=html

[1838] https://docs.freebsd.org/en/books/handbook/network-servers/#network-netgroups

32.4.7. NIS 安全

由于 RPC 是一种基于广播的服务，在同一域内运行 `ypbind` 的任何系统都可以检索 NIS 映射的内容。为了防止未经授权的交换，ypserv(8)[1839] 支持一个叫做 `securenets` 的功能，它可以用来限制对一组特定主机的访问。默认情况下，这些信息存储在 **/var/yp/securenets** 中，除非 ypserv(8)[1840] 是以 `-p` 和另一个路径启动的。这个文件包含的条目由一个网络规范和一个网络掩码组成，中间用空格隔开。以 `"#"` 开头的行被认为是注释。一个 `[.filename]#securenets` 的样本可能看起来像这样：

```
# 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
```

如果 ypserv(8)[1841] 收到来自与这些规则之一匹配的地址的请求，它将正常处理该请求。如果地址与规则不匹配，则将忽略该请求并记录警告消息。如果 **securenets** 不存在，`ypserv` 将允许来自任何主机的连接。

TCP wrapper[1842] 是一种替代机制，用于提供访问控制而不是 **securenets**。虽然这两种访问控制机制都增加一些安全性，但它们都容易受到“IP 欺骗”攻击。所有与 NIS 相关的流量都应在防火墙处阻止。

使用 **securenets** 的服务器可能无法为具有过时 TCP/IP 实现的合法 NIS 客户端提供服务。其中一些实现在执行广播时将所有主机位设置为零，或者在计算广播地址时无法观察到子网掩码。虽然其中一些问题可以通过更改客户端配置来解决，但其他问题可能会强制停用这些客户端系统或放弃 **securenets**。

使用 TCP 包装会增加 NIS 服务器的延迟。额外的延迟可能足够长，会导致客户端程序超时，尤其是在具有慢速 NIS 服务器的繁忙网络中。如果一个或多个客户端存在延迟，请将这些客户端转换为 NIS 从属服务器，并强制它们绑定到自身。

32.4.7.1. 禁止某些用户

在此示例中，`basie` 系统是 NIS 域中的教师工作站。主 NIS 服务器上的 **passwd** 映射包含教职员工和学生的帐户。本节演示如何在拒绝学生登录的同时允许教师在此系统上登录。

要阻止指定的用户登录系统，即使他们存在于 NIS 数据库中，使用 `vipw` 在客户机上的 **/etc/master.passwd** 末尾添加 `-username` 和正确数量的冒号，其中 `username` 是要禁止登录的用户的用户名。含有被禁止的用户的那一行必须在允许 NIS 用户的 + 行之前。在这个例子中，`bill` 被禁止登录到 `basie`。

[1839] https://www.freebsd.org/cgi/man.cgi?query=ypserv&sektion=8&format=html
[1840] https://www.freebsd.org/cgi/man.cgi?query=ypserv&sektion=8&format=html
[1841] https://www.freebsd.org/cgi/man.cgi?query=ypserv&sektion=8&format=html
[1842] https://docs.freebsd.org/en/books/handbook/security/index.html#tcpwrappers

```
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:/usr/sbin/nologin
operator:*:2:5::0:0:System &:/:/usr/sbin/nologin
bin:*:3:7::0:0:Binaries Commands and Source,,,:/:/usr/sbin/nologin
tty:*:4:65533::0:0:Tty Sandbox:/:/usr/sbin/nologin
kmem:*:5:65533::0:0:KMem Sandbox:/:/usr/sbin/nologin
games:*:7:13::0:0:Games pseudo-user:/usr/games:/usr/sbin/nologin
news:*:8:8::0:0:News Subsystem:/:/usr/sbin/nologin
man:*:9:9::0:0:Mister Man Pages:/usr/share/man:/usr/sbin/nologin
bind:*:53:53::0:0:Bind Sandbox:/:/usr/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:/usr/sbin/nologin
pop:*:68:6::0:0:Post Office Owner:/nonexistent:/usr/sbin/nologin
nobody:*:65534:65534::0:0:Unprivileged user:/nonexistent:/usr/sbin/nologin
-bill:::::::::
+:::::::::

basie#
```

32.4.8. 使用网络组

禁止指定用户登录到单个系统在较大的网络上变得无法扩展，并很快失去 NIS 的主要好处：集中管理。

开发网络组是为了处理具有数百个用户和计算机的大型复杂网络。它们的使用与 UNIX® 组相当，其中的主要区别在于缺少数字 ID 以及通过包括用户帐户和其他网络组来定义网络组的能力。

为了扩展本章中使用的示例，将扩展 NIS 域以添加表 28.2 和 28.3 中所示的用户和系统：

表 25. 其他用户

用户名	说明
alpha,beta	IT 部门员工
charlie,delta	IT 部门学徒
echo，foxtrott，golf …	员工
able,baker，…	实习生

表 26. 其他系统

计算机名称	说明
war, death, famine, pollution	只有 IT 员工才能登录到这些服务器。
pride , greed, envy, wrath, lust, sloth	允许 IT 部门的所有成员登录到这些服务器。
one, two, three, four , ···	员工使用的普通工作站。
trashcan	一台非常旧的机器，没有任何关键数据。甚至实习生也可以使用这个系统。

当使用网络组配置此方案时，会将每个用户分配到一个或多个网络组，然后允许或禁止网络组的所有成员登录。添加新计算机时，必须为所有网络组定义登录限制。添加新用户时，必须将该帐户添加到一个或多个网络组。如果仔细规划了 NIS 设置，则只需修改一个中央配置文件即可授予或拒绝对计算机的访问权限。

第一步是初始化 NIS ‘netgroup’ 映射。在 FreeBSD 中，默认情况下不创建此映射。在 NIS 主服务器上，使用编辑器创建名为 **/var/yp/netgroup** 的映射。

此示例创建四个网络组来表示 IT 员工、IT 学徒、员工和实习生：

```
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)
```

每个条目配置一个网络组。条目中的第一列是网络组的名称。每组括号表示一组由一个或多个组成的用户或另一个网络组的名称。指定用户时，每个组中以逗号分隔的三个字段表示：

1. 表示用户的其他字段有效的主机的名称。如果未指定主机名，则该条目在所有主机上都有效。
2. 属于此网络组的帐户的名称。
3. 帐户的 NIS 域。帐户可以从其他 NIS 域导入到网络组中。

如果组包含多个用户，请用空格分隔每个用户。此外，每个字段可能包含通配符。有关详细信息，请参见 netgroup(5)[1843]。

不应使用长度超过 8 个字符的网络组名称。名称区分大小写，对网络组名称使用大写字母是区分用户、计算机和网络组名称的简单方法。

一些非 FreeBSD NIS 客户端不能处理包含超过 15 个条目的网络组。通过创建多个具有 15 个或更少用户的子网络组以及由子网络组组成的真实网络组，可以规避此限制，如以下示例所示：

```
BIGGRP1  (,joe1,domain)  (,joe2,domain)  (,joe3,domain) [...]
BIGGRP2  (,joe16,domain)  (,joe17,domain) [...]
```

(continues on next page)

[1843] https://www.freebsd.org/cgi/man.cgi?query=netgroup&sektion=5&format=html

(continued from previous page)

```
BIGGRP3  (,joe31,domain)  (,joe32,domain)
BIGGROUP  BIGGRP1 BIGGRP2 BIGGRP3
```

如果单个网络组中存在超过 225 个（15 x 15）个用户，请重复此过程。

要激活并分发新的 NIS 映射，请执行以下操作：

```
ellington# cd /var/yp
ellington# make
```

这将生成三个 NIS 映射 **netgroup**、**netgroup.byhost** 和 **netgroup.byuser**。使用 ypcat(1)[1844] 的映射键选项来检查新的 NIS 映射是否可用：

```
ellington% ypcat -k netgroup
ellington% ypcat -k netgroup.byhost
ellington% ypcat -k netgroup.byuser
```

第一个命令的输出应类似于 **/var/yp/netgroup** 的内容。第二个命令仅在创建特定于主机的网络组时生成输出。第三个命令用于获取用户的网络组列表。

要配置客户端，请使用 vipw(8)[1845] 指定网络组的名称。例如，在名为 `war` 的服务器上，替换以下行：

```
+:::::::::
```

跟

```
+@IT_EMP:::::::::
```

这指定只有在网络组 `IT_EMP` 中定义的用户将被导入到这个系统的密码数据库，并且只有这些用户被允许登录到这个系统。

这个配置也适用于 shell 的 ~ 函数和所有在用户名和数字用户 ID 之间转换的程序。换句话说，`cd ~user` 将不工作，`ls -l` 将显示数字 ID 而不是用户名，`find . -user joe -print` 将会失败，提示 `No such user`。要解决这个问题，请导入所有用户条目，而不允许他们登录到服务器。这可以通过添加一个额外的行来实现：

```
+::::::::::/usr/sbin/nologin
```

此行将客户端配置为导入所有条目，但将这些条目中的 shell 替换为 **/usr/sbin/nologin**。

确保在 `+@IT_EMP:::::::::` 之后放置了多行。否则，从 NIS 导入的所有用户帐户都将使用 **/usr/sbin/nologin** 作为其登录 shell，并且没有人能够登录到系统。

[1844] https://www.freebsd.org/cgi/man.cgi?query=ypcat&sektion=1&format=html
[1845] https://www.freebsd.org/cgi/man.cgi?query=vipw&sektion=8&format=html

要配置不太重要的服务器，请将服务器上的 `+::::::::::` 替换为以下行：

```
+@IT_EMP:::::::::
+@IT_APP:::::::::
+::::::::::/usr/sbin/nologin
```

工作站的相应行是：

```
+@IT_EMP:::::::::
+@USERS:::::::::
+::::::::::/usr/sbin/nologin
```

NIS 支持从其他网组创建网组，这在有关用户访问的政策发生变化时很有用。一种可能性是创建基于角色的网组。例如，我们可以创建一个叫做 `BIGSRV` 的网组来定义重要服务器的登录限制，另一个叫做 `SMALLSRV` 的网组用于不太重要的服务器，第三个网组叫做 `USERBOX` 用于工作站。这些网组中的每一个都包含允许登录到这些机器上的网组。NIS `netgroup` 映射的新条目看起来是这样的。

```
BIGSRV    IT_EMP  IT_APP
SMALLSRV  IT_EMP  IT_APP  ITINTERN
USERBOX   IT_EMP  ITINTERN USERS
```

当可以定义具有相同限制的计算机组时，这种定义登录限制的方法相当有效。不幸的是，这是例外，而不是规则。大多数情况下，需要能够基于每台计算机定义登录限制。

特定于计算机的网络组定义是处理策略更改的另一种可能性。在这种情况下，每个系统的 **/etc/master.passwd** 包含两行以“+”开头的行。第一行添加一个网络组，其中包含允许登录到此计算机的帐户，第二行添加所有其他帐户，其中 **/usr/sbin/nologin** 作为 shell。建议使用主机名的“ALL-CAPS”版本作为网络组的名称：

```
+@BOXNAME:::::::::
+::::::::::/usr/sbin/nologin
```

在所有计算机上完成此任务后，不再需要再次修改 **/etc/master.passwd** 的本地版本。所有进一步的更改都可以通过修改 NIS 映射来处理。下面是此方案的 `netgroup` 映射示例：

```
# 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)
```

(continues on next page)

(continued from previous page)

```
#
# 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]
```

使用基于计算机的网络组可能并不总是可取的。部署几十个或数百个系统时，可以使用基于角色的网络组而不是基于计算机的网络组，以将 NIS 映射的大小保持在合理的限制范围内。

32.4.9. 密码格式

NIS 要求 NIS 域中的所有主机使用相同的格式来加密密码。如果用户在 NIS 客户端上无法进行身份验证，则可能是由于密码格式不同。在异构网络中，所有操作系统都必须支持该格式，其中 DES 是最低的通用标准。

要检查服务器或客户端使用的格式，请查看 **/etc/login.conf** 的此部分：

```
default:\
    :passwd_format=des:\
    :copyright=/etc/COPYRIGHT:\
    [Further entries elided]
```

在这个例子中，系统使用 DES 格式进行密码散列。其他可能的值包括 `blowfish` 的 blf，MD5 的 `md5`，SHA-256 和 SHA-512 的 `sha256` 和 `sha512`。关于更多的信息和系统上可用的最新列表，请查阅 crypt(3)[1846] 手册页。

如果需要编辑主机上的格式以匹配 NIS 域中使用的格式，则必须在保存更改后重建登录功能数据库：

```
# cap_mkdb /etc/login.conf
```

> 注意
> 在重建登录功能数据库后，每个用户在更改其密码之前，不会更新现有用户帐户的密码格式。

32.5. 轻型目录访问协议（LDAP）

轻型目录访问协议（LDAP）是一种应用层协议，用于使用分布式目录信息服务访问、修改和验证对象。可以把它想象成一本电话或唱片簿，它存储了几个层次的分层、同质信息。它用于 Active Directory 和 OpenLDAP 网络，允许用户使用单个帐户访问多个级别的内部信息。例如，电子邮件身份验证、提取员工联系信息和内部网站身份验证都可能使用 LDAP 服务器记录库中的单个用户帐户。

本节提供在 FreeBSD 系统上配置 LDAP 服务器的快速入门指南。它假定管理员已经有一个设计计划，其中包括要存储的信息类型、该信息将用于什么目的、哪些用户应有权访问该信息，以及如何防止未经授权的访问。

32.5.1. LDAP 术语和结构

LDAP 使用几个术语，在开始配置之前应了解这些术语。所有目录条目都由一组 *属性* 组成。这些属性集中的每一个都包含一个称为 *可分辨名称*（DN）的唯一标识符，该标识符通常由其他几个属性（如公用或 *相对可分辨名称*（RDN））构建。与目录具有绝对路径和相对路径的方式类似，将 DN 视为绝对路径，将 RDN 视为相对路径。

示例 LDAP 条目如下所示。本示例搜索指定用户帐户（`uid`）、组织单位（`ou`）和组织（`o`）的条目：

```
% 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
```

(continues on next page)

[1846] https://www.freebsd.org/cgi/man.cgi?query=crypt&sektion=3&format=html

(continued from previous page)

```
dn: uid=trhodes,ou=users,o=example.com
mail: trhodes@example.com
cn: Tom Rhodes
uid: trhodes
telephoneNumber: (123) 456-7890

# search result
search: 2
result: 0 Success

# numResponses: 2
# numEntries: 1
```

此示例条目显示 `dn`、`mail`、`cn`、`uid` 和 `telephoneNumber` 属性的值。cn 的属性是 RDN。

有关 LDAP 及其术语的更多信息，请参阅 http://www.openldap.org/doc/admin24/intro.html。

32.5.2. 配置 LDAP 服务器

FreeBSD 不提供内置的 LDAP 服务器。通过软件包或 port net/openldap[1847] 来安装服务器软件以开始配置：

```
# pkg install openldap-server
```

软件包[1848]中启用了大量默认选项。可通过运行 `pkg info openldap-server` 来查看它们。如果它们还不够（例如，如果需要支持 SQL），请考虑使用适当的框架[1849]重新编译 port。

安装中将创建目录 **/var/db/openldap-data** 来保存数据。必须创建用于存储证书的目录：

```
# mkdir /usr/local/etc/openldap/private
```

下一阶段是配置证书颁发机构。以下命令必须从 **/usr/local/etc/openldap/private** 执行。这一点很重要，因为文件权限需要受到限制，并且用户不应有权访问这些文件。有关证书及其参数的更多详细信息，请参阅 OpenSSL[1850]。要创建证书颁发机构，请从以下命令开始，然后按照提示操作：

```
# openssl req -days 365 -nodes -new -x509 -keyout ca.key -out ../ca.crt
```

提示的条目可能是通用的，但 `Common Name` 条目必须与系统主机名 *不同*。如果这是自签名证书，请在主机名前面加上“证书颁发机构”前缀 `CA`。

下一个任务是创建证书签名请求和私钥。输入以下命令并按照提示操作：

[1847] https://cgit.freebsd.org/ports/tree/net/openldap-server/pkg-descr

[1848] https://docs.freebsd.org/en/articles/linux-users/#software

[1849] https://docs.freebsd.org/en/books/handbook/ports/index.html#ports-using

[1850] https://docs.freebsd.org/en/books/handbook/security/index.html#openssl

```
# openssl req -days 365 -nodes -new -keyout server.key -out server.csr
```

在证书生成过程中，请确保正确设置 Common Name 属性。证书签名请求必须使用证书颁发机构进行签名，才能用作有效证书：

```
# openssl x509 -req -days 365 -in server.csr -out ../server.crt -CA ../ca.crt -CAkey␣
↪ca.key -CAcreateserial
```

证书生成过程的最后一部分是生成客户端证书并对其进行签名：

```
# 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
```

请记住在出现提示时使用相同的 Common Name 属性。完成后，确保通过一连串命令总共生成了 8 个新文件。

运行 OpenLDAP 服务器的守护程序已 **过期**。它的配置是通过 **slapd.ldif** 执行的：旧的 **slapd.conf** 已被 OpenLDAP 弃用。

slapd.ldif 的 配置示例[1851]可用，也可以在 **/usr/local/etc/openldap/slapd.ldif.sample** 中找到。选项记录在 slapd-config（5）中。与所有其他 LDAP 属性集一样，**slapd.ldif** 的每个部分都通过 DN 进行唯一标识。请确保在语句和该部分的所需末尾之间不留任何空行。在以下示例中，TLS 将用于实现安全通道。第一部分是全局配置：

```
#
# See slapd-config(5) for details on configuration options.
# This file should NOT be world readable.
#
dn: cn=config
objectClass: olcGlobal
cn: config
#
#
# Define global ACLs to disable default read access.
#
olcArgsFile: /var/run/openldap/slapd.args
olcPidFile: /var/run/openldap/slapd.pid
olcTLSCertificateFile: /usr/local/etc/openldap/server.crt
olcTLSCertificateKeyFile: /usr/local/etc/openldap/private/server.key
olcTLSCACertificateFile: /usr/local/etc/openldap/ca.crt
#olcTLSCipherSuite: HIGH
```

(continues on next page)

[1851] http://www.openldap.org/doc/admin24/slapdconf2.html

(continued from previous page)

```
olcTLSProtocolMin: 3.1
olcTLSVerifyClient: never
```

必须在这里指定证书机构、服务器证书和服务器私钥文件。建议让客户选择安全密码，省略选项 `olcTLSCipherSuite`（与 **openssl** 以外的 TLS 客户不兼容）。选项 `olcTLSProtocolMin` 让服务器要求一个最低的安全级别：建议使用。虽然验证对服务器来说是强制性的，但对客户端来说却不是必需的：`olcTLSVerifyClient: never`。

第二部分是关于后端模块的，可以按如下方式配置：

```
#
# Load dynamic backend modules:
#
dn: cn=module,cn=config
objectClass: olcModuleList
cn: module
olcModulepath:  /usr/local/libexec/openldap
olcModuleload:  back_mdb.la
#olcModuleload: back_bdb.la
#olcModuleload: back_hdb.la
#olcModuleload: back_ldap.la
#olcModuleload: back_passwd.la
#olcModuleload: back_shell.la
```

第三部分专门用于加载数据库使用的所需 `ldif` 架构：它们是必不可少的。

```
dn: cn=schema,cn=config
objectClass: olcSchemaConfig
cn: schema

include: file:///usr/local/etc/openldap/schema/core.ldif
include: file:///usr/local/etc/openldap/schema/cosine.ldif
include: file:///usr/local/etc/openldap/schema/inetorgperson.ldif
include: file:///usr/local/etc/openldap/schema/nis.ldif
```

接下来，前端配置部分：

```
# Frontend settings
#
dn: olcDatabase={-1}frontend,cn=config
objectClass: olcDatabaseConfig
objectClass: olcFrontendConfig
olcDatabase: {-1}frontend
```

(continues on next page)

(continued from previous page)

```
olcAccess: to * by * read
#
# Sample global access control policy:
#   Root DSE: allow anyone to read it
#   Subschema (sub)entry DSE: allow anyone to read it
#   Other DSEs:
#       Allow self write access
#       Allow authenticated users read access
#       Allow anonymous users to authenticate
#
#olcAccess: to dn.base="" by * read
#olcAccess: to dn.base="cn=Subschema" by * read
#olcAccess: to *
#   by self write
#   by users read
#   by anonymous auth
#
# if no access controls are present, the default policy
# allows anyone and everyone to read anything but restricts
# updates to rootdn.  (e.g., "access to * by * read")
#
# rootdn can always read and write EVERYTHING!
#
olcPasswordHash: {SSHA}
# {SSHA} is already the default for olcPasswordHash
```

另一部分专门介绍 *配置后端*，以后访问 OpenLDAP 服务器配置的唯一方法是作为全局超级用户。

```
dn: olcDatabase={0}config,cn=config
objectClass: olcDatabaseConfig
olcDatabase: {0}config
olcAccess: to * by * none
olcRootPW: {SSHA}iae+lrQZILpiUdf16Z9KmDmSwT77Dj4U
```

缺省管理员用户名为 `cn=config`。在 shell 中键入 **slappasswd**，选择一个密码并在 `olcRootPW` 中使用其哈希值。如果现在未指定此选项，则在导入 **slapd.ldif** 之前，以后任何人都无法修改 *全局配置*部分。

最后一部分是关于数据库后端的：

```
#######################################################################
# LMDB database definitions
#######################################################################
#
```

(continues on next page)

(continued from previous page)

```
dn: olcDatabase=mdb,cn=config
objectClass: olcDatabaseConfig
objectClass: olcMdbConfig
olcDatabase: mdb
olcDbMaxSize: 1073741824
olcSuffix: dc=domain,dc=example
olcRootDN: cn=mdbadmin,dc=domain,dc=example
# Cleartext passwords, especially for the rootdn, should
# be avoided.  See slappasswd(8) and slapd-config(5) for details.
# Use of strong authentication encouraged.
olcRootPW: {SSHA}X2wHvIWDk6G76CQyCMS1vDCvtICWgn0+
# The database directory MUST exist prior to running slapd AND
# should only be accessible by the slapd and slap tools.
# Mode 700 recommended.
olcDbDirectory: /var/db/openldap-data
# Indices to maintain
olcDbIndex: objectClass eq
```

这个数据库承载了 LDAP 目录的 实际内容。除了 mdb 之外，其他类型的数据库都可以使用。它的超级用户，不要和全局用户混淆，在这里配置：olcRootDN 中一个（可能是自定义的）用户名和 olcRootPW 中的密码散列；**slappasswd** 可以像以前一样使用。

此仓库[1852]包含 **slapd.ldif** 的四个示例。要将现有的 **slapd.conf** 转换为 **slapd.ldif**，请参阅此页面[1853]（请注意，这可能会引入一些无用的选项）。

配置完成后，必须将 **slapd.ldif** 放在空目录中。建议将其创建为：

```
# mkdir /usr/local/etc/openldap/slapd.d/
```

导入配置数据库：

```
# /usr/local/sbin/slapadd -n0 -F /usr/local/etc/openldap/slapd.d/ -l /usr/local/etc/
↪openldap/slapd.ldif
```

启动 **slapd** 守护程序：

```
# /usr/local/libexec/slapd -F /usr/local/etc/openldap/slapd.d/
```

-d 选项可用于调试，如 slapd（8）中所述。要验证服务器是否正在运行且工作正常，请执行以下操作：

[1852] http://www.openldap.org/devel/gitweb.cgi?p=openldap.git;a=tree;f=tests/data/regressions/its8444;h=8a5e808e63b0de3d2bdaf2cf34fecca8577ca7fd;hb=HEAD

[1853] http://www.openldap.org/doc/admin24/slapdconf2.html

```
# ldapsearch -x -b '' -s base '(objectclass=*)' namingContexts
# extended LDIF
#
# LDAPv3
# base <> with scope baseObject
# filter: (objectclass=*)
# requesting: namingContexts
#

#
dn:
namingContexts: dc=domain,dc=example

# search result
search: 2
result: 0 Success

# numResponses: 2
# numEntries: 1
```

服务器仍必须受信任。如果以前从未这样做过，请按照以下说明进行操作。使用软件包或 port 安装 OpenSSL：

```
# pkg install openssl
```

从存储 **ca.crt** 的目录（在此示例中为 **/usr/local/etc/openldap**），运行：

```
# c_rehash .
```

CA 和服务器证书现在都可以在各自的角色中正确识别。若要验证这一点，请从 **server.crt** 目录运行以下命令：

```
# openssl verify -verbose -CApath . server.crt
```

如果 **slapd** 正在运行，请重新启动它。如 **/usr/local/etc/rc.d/slapd** 中所述，要在引导时正确运行 **slapd**，必须在 **/etc/rc.conf** 中添加以下行：

```
slapd_enable="YES"
slapd_flags='-h "ldapi://%2fvar%2frun%2fopenldap%2fldapi/
ldap://0.0.0.0/"'
slapd_sockets="/var/run/openldap/ldapi"
slapd_cn_config="YES"
```

slapd 不提供启动时的调试。为此，请检查 **/var/log/debug.log**、**dmesg -a** 和 **/var/log/messages**。

下面的例子将组 `team` 和用户 `john` 添加到 `domain.example` LDAP 数据库中，该数据库仍然是空的。首先，创建文件 **domain.ldif**。

```
# cat domain.ldif
dn: dc=domain,dc=example
objectClass: dcObject
objectClass: organization
o: domain.example
dc: domain

dn: ou=groups,dc=domain,dc=example
objectClass: top
objectClass: organizationalunit
ou: groups

dn: ou=users,dc=domain,dc=example
objectClass: top
objectClass: organizationalunit
ou: users

dn: cn=team,ou=groups,dc=domain,dc=example
objectClass: top
objectClass: posixGroup
cn: team
gidNumber: 10001

dn: uid=john,ou=users,dc=domain,dc=example
objectClass: top
objectClass: account
objectClass: posixAccount
objectClass: shadowAccount
cn: John McUser
uid: john
uidNumber: 10001
gidNumber: 10001
homeDirectory: /home/john/
loginShell: /usr/bin/bash
userPassword: secret
```

更多细节请参见 **OpenLDAP** 的文档。使用 **slappasswd** 将纯文本密码 `secret` 替换为 `userPassword` 中的哈希值。作为 `loginShell` 指定的路径必须存在于所有允许 `john` 登录的系统中。最后，使用 `mdb` 管理员来修改数据库。

```
# ldapadd -W -D "cn=mdbadmin,dc=domain,dc=example" -f domain.ldif
```

对 全局配置部分的修改只能由全局超级用户执行。例如，假定最初指定了选项 olcTLSCipherSuite: HIGH:MEDIUM:SSLv3，现在必须将其删除。首先，创建一个包含以下内容的文件：

```
# cat global_mod
dn: cn=config
changetype: modify
delete: olcTLSCipherSuite
```

然后，应用修改：

```
# ldapmodify -f global_mod -x -D "cn=config" -W
```

当被要求时，提供在 配置后端部分选择的密码。用户名不是必需的：在这里，cn=config 代表要修改的数据库部分的 DN。另外，使用 ldapmodify 来删除数据库中的某一行，ldapdelete 来删除整个条目。

如果出现问题，或者全局超级用户无法访问配置后端，则可以删除并重新写入整个配置：

```
# rm -rf /usr/local/etc/openldap/slapd.d/
```

然后可以编辑并再次导入 **slapd.ldif**。请仅在没有其他可用解决方案时才执行此过程。

这只是服务器的配置。同一台计算机还可以托管 LDAP 客户端，并具有自己单独的配置。

32.6. 动态主机设置协议（DHCP）

动态主机配置协议（DHCP）允许系统连接到网络，以便为该网络上的通信分配必要的寻址信息。FreeBSD 包含 OpenBSD 版本的 dhclient，使用它来获取寻址信息。FreeBSD 不安装 DHCP 服务器，但是在 FreeBSD ports 中有好几个服务器可用。DHCP 协议在 RFC 2131[1854] 中进行了全面描述。isc.org/downloads/dhcp/[1855] 也提供信息资源。

本节介绍了如何使用内置 DHCP 客户端。然后，它说明了如何安装和配置 DHCP 服务器。

> 注意
>
> 在 FreeBSD 中，DHCP 服务器和 DHCP 客户端都需要 bpf(4)[1856] 设备。这个设备被包含在随 FreeBSD 安装的 **GENERIC** 内核中。喜欢创建定制内核的用户在使用 DHCP 时需要保留这个设备。
>
> 应该注意的是，**bpf** 也允许有特权的用户在该系统上运行网络数据包嗅探器。

[1854] http://www.freesoft.org/CIE/RFC/2131/

[1855] http://www.isc.org/downloads/dhcp/

[1856] https://www.freebsd.org/cgi/man.cgi?query=bpf&sektion=4&format=html

32.6.1. 配置 DHCP 客户端

DHCP 客户端支持包含在 FreeBSD 安装程序中，使得配置新安装的系统很容易，以自动从现有的 DHCP 服务器接收其网络寻址信息。有关网络配置的示例请参阅帐户、时区、服务和加固[1857]。

在客户端计算机上执行时，它将开始广播配置信息请求。默认情况下，这些请求使用 UDP 端口 68。服务器在 UDP 端口 67 上进行回复，为客户端提供 IP 地址和其他相关网络信息，如子网掩码、默认网关和 DNS 服务器地址。此信息采用 DHCP “租约” 的形式，在可配置的时间内有效。这允许自动重用不再连接到网络的客户端的陈旧 IP 地址。DHCP 客户端可以从服务器获取大量信息。详尽的列表可以在 dhcp-options(5)[1858] 中找到。

缺省情况下，当 FreeBSD 系统引导时，它的 DHCP 客户端在后台运行，或者 异步运行。在 DHCP 进程完成时，其他启动脚本将继续运行，从而加快系统启动速度。

当 DHCP 服务器快速响应客户端的请求时，后台 DHCP 可以很好地工作。但是，DHCP 在某些系统上可能需要很长时间才能完成。如果网络服务尝试在 DHCP 分配网络寻址信息之前运行，则它们将失败。在 同步模式下使用 DHCP 可防止此问题，因为它会暂停启动，直到 DHCP 配置完成。

/etc/rc.conf 中的这一行用于配置后台或异步模式：

```
ifconfig_fxp0="DHCP"
```

如果系统配置为在安装过程中使用 DHCP，则此行可能已存在。如 “设置网卡[1859]” 中所述，将这些示例中所示的 *fxp0* 替换为要动态配置的接口的名称。

要改为将系统配置为使用同步模式，并在 DHCP 完成时在启动期间暂停，请使用 “SYNCDHCP”：

```
ifconfig_fxp0="SYNCDHCP"
```

有其他的客户端选项可用。在 rc.conf(5)[1860] 中搜索 `dhclient`。

DHCP 客户端使用以下文件：

- **/etc/dhclient.conf**

  `dhclient` 使用的配置文件。通常，此文件仅包含注释，因为默认值适用于大多数客户端。此配置文件在 dhclient.conf(5)[1861] 中进行了说明。

- **/sbin/dhclient**

  有关命令本身的更多信息可以在 dhclient(8)[1862] 中找到。

- **/sbin/dhclient-script**

[1857] https://docs.freebsd.org/en/books/handbook/bsdinstall/index.html#bsdinstall-post
[1858] https://www.freebsd.org/cgi/man.cgi?query=dhcp-options&sektion=5&format=html
[1859] https://docs.freebsd.org/en/books/handbook/config/index.html#config-network-setup
[1860] https://www.freebsd.org/cgi/man.cgi?query=rc.conf&sektion=5&format=html
[1861] https://www.freebsd.org/cgi/man.cgi?query=dhclient.conf&sektion=5&format=html
[1862] https://www.freebsd.org/cgi/man.cgi?query=dhclient&sektion=8&format=html

FreeBSD 特定的 DHCP 客户端配置脚本。它在 dhclient-script(8)[1863] 中进行了说明，但不需要任何用户修改即可正常运行。

- **/var/db/dhclient.leases.interface**

  DHCP 客户端在此文件中保留一个有效租约数据库，该文件以日志形式写入，并在 dhclient.leases(5)[1864] 中进行了说明。

32.6.2. 安装和配置 DHCP 服务器

本节演示如何使用 DHCP 服务器的互联网系统联盟（ISC）实现来配置 FreeBSD 系统以充当 DHCP 服务器。可以使用软件包或 port net/isc-dhcp44-server[1865] 安装此实现及其文档。

net/isc-dhcp44-server[1866] 的安装将放置一个示例配置文件。将 **/usr/local/etc/dhcpd.conf.example** 复制到 **/usr/local/etc/dhcpd.conf**，并对此新文件进行任何编辑。

配置文件由子网和主机的声明组成，这些声明定义了提供给 DHCP 客户端的信息。例如，这些行配置以下内容：

```
option domain-name "example.org";   ①
option domain-name-servers ns1.example.org;   ②
option subnet-mask 255.255.255.0;   ③

default-lease-time 600;    ④
max-lease-time 72400;    ⑤
ddns-update-style none;    ⑥

subnet 10.254.239.0 netmask 255.255.255.224 {
  range 10.254.239.10 10.254.239.20;    ⑦
  option routers rtr-239-0-1.example.org, rtr-239-0-2.example.org;    ⑧
}

host fantasia {
  hardware ethernet 08:00:07:26:c0:a5;    ⑨
  fixed-address fantasia.fugue.com;    ⑩
}
```

① 此选项指定将提供给客户端的默认搜索域。有关详细信息，请参阅 resolv.conf(5)[1867]。

② 此选项指定客户端应使用的以逗号分隔的 DNS 服务器列表。它们可以按其完全限定域名（FQDN）列出（如示例中所示）或按其 IP 地址列出。

[1863] https://www.freebsd.org/cgi/man.cgi?query=dhclient-script&sektion=8&format=html
[1864] https://www.freebsd.org/cgi/man.cgi?query=dhclient.leases&sektion=5&format=html
[1865] https://cgit.freebsd.org/ports/tree/net/isc-dhcp44-server/pkg-descr
[1866] https://cgit.freebsd.org/ports/tree/net/isc-dhcp44-server/pkg-descr
[1867] https://www.freebsd.org/cgi/man.cgi?query=resolv.conf&sektion=5&format=html

③ 将提供给客户端的子网掩码。

④ 默认租约到期时间（以秒为单位）。可以将客户端配置为覆盖此值。

⑤ 租约允许的最大时间长度（以秒为单位）。如果客户端请求更长的租约，仍会发出租约，但该租约仅对 `max-lease-time` 有效。

⑥ 默认 `none` 禁用动态 DNS 更新。将此更改为 `interim` 将 DHCP 服务器配置为在发出租约时更新 DNS 服务器，以便 DNS 服务器知道哪些 IP 地址与网络中的哪些计算机相关联。除非已将 DNS 服务器配置为支持动态 DNS，否则不要更改默认设置。

⑦ 此行创建一个可用 IP 地址池，这些 IP 地址保留用于分配给 DHCP 客户端。地址范围必须对上一行中指定的网络或子网有效。

⑧ 声明对 { 左括号前指定的网络或子网有效的默认网关。

⑨ 指定客户端的硬件 MAC 地址，以便 DHCP 服务器在发出请求时可以识别客户端。

⑩ 指定应始终为此主机提供相同的 IP 地址。使用主机名是正确的，因为 DHCP 服务器将在返回租约信息之前解析主机名。

此配置文件支持更多选项。有关详细信息和示例，请参阅随服务器一起安装的 dhcpd.conf(5)。

dhcpd.conf 的配置完成后，在 **/etc/rc.conf** 中启用 DHCP 服务器：

```
dhcpd_enable="YES"
dhcpd_ifaces="dc0"
```

替换 `dc0` 为 DHCP 服务器应侦听 DHCP 客户端请求的接口（或多个接口，用空格分隔）。

通过执行以下命令启动服务器：

```
# service isc-dhcpd start
```

将来对服务器配置的任何更改都需要停止 dhcpd 服务，然后使用 service(8)[1868] 启动。

DHCP 服务器使用以下文件。请注意，手册页随服务器软件一起安装。

- **/usr/local/sbin/dhcpd**

  有关 dhcpd 服务器的更多信息可以在 dhcpd(8) 中找到。

- **/usr/local/etc/dhcpd.conf**

  服务器配置文件需要包含应提供给客户端的所有信息，以及有关服务器操作的信息。此配置文件在 dhcpd.conf(5) 中进行了概述。

- **/var/db/dhcpd.leases**

  DHCP 服务器在此文件中保留它已颁发的租约的数据库，该文件以日志形式写入。请参阅 dhcpd.leases(5)，它给出了详细的解释。

[1868] https://www.freebsd.org/cgi/man.cgi?query=service&sektion=8&format=html

- **/usr/local/sbin/dhcrelay**

  此守护程序用于高级环境，其中一个 DHCP 服务器将请求从客户端转发到单独网络上的另一个 DHCP 服务器。如果需要此功能，请通过软件包或 port 安装 net/isc-dhcp44-relay[1869]。安装包括 dhcrelay(8)，它提供了更多细节。

32.7. 域名系统（DNS）

域名系统（DNS）是将域名映射到 IP 地址的协议，反之亦然。DNS 通过一个由权威根、顶级域（TLD）和其他规模较小的名称服务器组成的复杂系统在互联网上进行协调，这些服务器托管和缓存各个域信息。无需运行名称服务器即可在系统上执行 DNS 查找。

下表介绍了与 DNS 关联的一些术语：

表 4. DNS 术语

术语	定义
Forward DNS	主机名到 IP 地址的映射。
Origin	引用特定区域文件中涵盖的域。
Resolver	一个系统进程，计算机通过该进程向名称服务器查询区域信息。
Reverse DNS	将 IP 地址映射到主机名。
Root zone	互联网区域层次结构的开头。所有区域都属于根区域，类似于文件系统中的所有文件都属于根目录。
Zone	由同一颁发机构管理的单个域、子域或 DNS 的一部分。

区域示例：

- `.` 是文档中通常引用根区域的方式。
- `org.` 是根区域下的顶级域（TLD）。
- `example.org.` 是 `org.`TLD 下的一个区域。
- `1.168.192.in-addr.arpa` 是引用属于‘192.168.1.*’IP 地址空间的所有 IP 地址的区域。

正如人们所看到的，主机名中更具体的部分出现在其左边。例如，`example.org.`比 `org.`更具体，因为 `org.` 比根区更具体。主机名的每个部分的布局很像一个文件系统：目录 **/dev** 属于根目录，以此类推。

[1869] https://cgit.freebsd.org/ports/tree/net/isc-dhcp44-relay/pkg-descr

32.7.1. 运行域名服务器的原因

域名服务器通常有两种形式：权威域名服务器和缓存（也称为解析）域名服务器。

在以下情况下，需要权威域名服务器：

- 人们希望向全世界提供 DNS 信息，权威地回复查询。
- 域（如 `example.org`）已注册，并且需要为其下的主机名分配 IP 地址。
- IP 地址阻止需要反向 DNS 条目（IP 到主机名）。
- 备份或辅助域名服务器（称为从属服务器）将回复查询。

在以下情况下需要缓存域名服务器：

- 本地 DNS 服务器可能比查询外部域名服务器更快地缓存和响应。

当查询 `www.FreeBSD.org` 时，解析器通常会查询上行链路 ISP 的域名服务器，并检索回复。使用本地缓存 DNS 服务器时，只需由缓存 DNS 服务器向外界进行一次查询。其他查询不必进入本地网络之外，因为信息缓存在本地。

32.7.2. DNS 服务器配置

FreeBSD 基本系统中提供了 Unbound。在默认情况下，它将仅向本地计算机提供 DNS 解析。虽然基本系统包可以配置为提供本地机器之外的解析服务，但建议通过从 FreeBSD ports 安装 Unbound 来满足这些要求。

要启用 Unbound，请将以下内容添加到 **/etc/rc.conf**：

```
local_unbound_enable="YES"
```

/etc/resolv.conf 中的任何现有域名服务器都将在新的 Unbound 配置中配置为转发器。

> 注意
> 如果列出的任何域名服务器不支持 DNSSEC，则本地的 DNS 解析将失败。请务必测试每个域名服务器，并删除任何未通过测试的域名服务器。以下命令将显示信任树或运行的域名服务器的故障：`192.168.1.1`：
>
> ```
> % drill -S FreeBSD.org @192.168.1.1
> ```

确认每个域名服务器支持 DNSSEC 后，启动 Unbound：

```
# service local_unbound onestart
```

这将负责更新 **/etc/resolv.conf**，以便 DNSSEC 安全域的查询现在正常工作。例如，运行以下命令以验证 FreeBSD.org DNSSEC 信任树：

```
% drill -S FreeBSD.org
;; Number of trusted keys: 1
;; Chasing: freebsd.org. A

DNSSEC Trust tree:
freebsd.org. (A)
|---freebsd.org. (DNSKEY keytag: 36786 alg: 8 flags: 256)
    |---freebsd.org. (DNSKEY keytag: 32659 alg: 8 flags: 257)
    |---freebsd.org. (DS keytag: 32659 digest type: 2)
        |---org. (DNSKEY keytag: 49587 alg: 7 flags: 256)
            |---org. (DNSKEY keytag: 9795 alg: 7 flags: 257)
            |---org. (DNSKEY keytag: 21366 alg: 7 flags: 257)
            |---org. (DS keytag: 21366 digest type: 1)
            |   |---. (DNSKEY keytag: 40926 alg: 8 flags: 256)
            |       |---. (DNSKEY keytag: 19036 alg: 8 flags: 257)
            |---org. (DS keytag: 21366 digest type: 2)
                |---. (DNSKEY keytag: 40926 alg: 8 flags: 256)
                    |---. (DNSKEY keytag: 19036 alg: 8 flags: 257)
;; Chase successful
```

32.8. Apache HTTP 服务器

开源的 Apache HTTP 服务器是使用最广泛的 Web 服务器。FreeBSD 在默认情况下不安装这个 Web 服务器，但它可以通过软件包或 port www/apache24[1870] 安装。

本节介绍了如何在 FreeBSD 上配置和启动 Apache HTTP 服务器的 2. *x* 版本。有关 Apache 2.X 及其配置指令的更多详细信息，请参阅 httpd.apache.org[1871]。

32.8.1. 配置和启动 Apache

在 FreeBSD 中，Apache HTTP 服务器的主配置文件安装为 **/usr/local/etc/apache2x/httpd.conf**，其中 *x* 表示版本号。这个 ASCII 文本文件以 # 开始注释行。最常修改的指令是：

- **ServerRoot "/usr/local"**

  指定 Apache 安装的默认目录层次结构。二进制文件存储在服务器根目录的 **bin** 和 **sbin** 子目录中，配置文件存储在 **etc/apache2x** 子目录中。

- **ServerAdmin you@example.com**

  将其更改为电子邮件地址以接收服务器问题。此地址还显示在某些服务器生成的页面上，例如错误文档。

[1870] https://cgit.freebsd.org/ports/tree/www/apache24/pkg-descr
[1871] http://httpd.apache.org

- **ServerName www.example.com:80**

  允许管理员设置发送回客户端的服务器主机名。例如，可以使用 `www` 代替实际的主机名。如果系统没有注册 DNS 名称，请填写 IP 地址。如果服务器将侦听备用报告，请更改 `80` 为备用端口号。

- **DocumentRoot "/usr/local/www/apache2_x_/data"**

  将从中提供文档的目录。默认情况下，所有请求都取自此目录，但符号链接和别名可用于指向其他位置。

在进行修改之前，对默认的 Apache 配置文件进行备份是一个好主意。当配置完成 Apache 后，保存该文件并使用 `apachectl` 验证该配置。运行 `apachectl configtest` 应该返回 `Syntax OK`。

要在系统启动时启动 Apache，请将以下行添加到 **/etc/rc.conf**：

```
apache24_enable="YES"
```

如果 Apache 要以非默认选项启动，则可以将以下行添加到 **/etc/rc.conf** 以指定所需的参数：

```
apache24_flags=""
```

如果 apachectl 未报告配置错误，现在启动 `httpd`:

```
# service apache24 start
```

httpd 服务可以通过在网页浏览器中输入 `http://localhost`，用运行 `httpd` 的机器的完全限定域名替换 `localhost` 来测试。显示的默认网页是 **/usr/local/www/apache24/data/index.html**。

在 `httpd` 运行时，对 Apache 配置进行后续修改后，可以用以下命令测试 Apache 配置是否出错：

```
# service apache24 configtest
```

> 注意
> 需要注意的是，`configtest` 不是 rc(8)[1872] 的标准，并且不应该期望它适用于所有启动脚本。

32.8.2. 虚拟主机

虚拟主机允许多个网站在一台 Apache 服务器上运行。虚拟主机可以 *基于 IP*，也可以 *基于名称*。基于 IP 的虚拟主机为每个网站使用不同的 IP 地址。基于名称的虚拟主机使用客户端 HTTP/1.1 标头来确定主机名，这允许网站共享相同的 IP 地址。

要设置 Apache 使用基于名字的虚拟主机，为每个网站添加一个 `VirtualHost` 块。例如，对于名为 `www.domain.tld` 的网站服务器，其虚拟域为 `www.someotherdomain.tld`，在 **httpd.conf** 中添加以下条目：

[1872] https://www.freebsd.org/cgi/man.cgi?query=rc&sektion=8&format=html

```
<VirtualHost *>
    ServerName www.domain.tld
    DocumentRoot /www/domain.tld
</VirtualHost>

<VirtualHost *>
    ServerName www.someotherdomain.tld
    DocumentRoot /www/someotherdomain.tld
</VirtualHost>
```

对于每个虚拟主机，将 `ServerName` 和 `DocumentRoot` 的值替换为要使用的值。

有关设置虚拟主机的更多信息，请参阅官方 Apache 文档：http://httpd.apache.org/docs/vhosts/。

32.8.3. Apache 模块

Apache 使用模块来增强基本服务器提供的功能。有关可用模块的完整列表和配置详细信息，请参阅 http://httpd.apache.org/docs/current/mod/。

在 FreeBSD 中，一些模块可以使用 www/apache24[1873] port 进行编译。在 **/usr/ports/www/apache24** 中键入内容，以查看哪些模块可用，哪些模块默认处于启用状态。如果模块不是用 ports 编译的，FreeBSD ports 提供了一种安装许多模块的简单方法。本节介绍三个最常用的模块。

32.8.3.1. SSL 支持

在某个时间点，对 Apache 内部 SSL 的支持需要一个名为 **mod_ssl** 的辅助模块。现在情况已不再如此，Apache 的默认安装带有内置于 Web 服务器中的 SSL。有关如何启用对 SSL 网站支持的示例，请参阅已安装的文件 **httpd-ssl.conf** 中的 **/usr/local/etc/apache24/extra** 目录，在此目录中还有一个名为 **ssl.conf-sample** 的示例文件。建议对这两个文件进行评估，以便在 Apache Web 服务器中正确设置安全网站。

SSL 配置完成后，必须在主 **http.conf** 中取消对以下行的注释，以便在下次重新启动或重新加载 Apache 时激活更改：

```
#Include etc/apache24/extra/httpd-ssl.conf
```

> 警告
>
> SSL 版本 2 和版本 3 存在已知的漏洞问题。强烈建议启用 TLS 版本 1.2 和 1.3 以代替较旧的 SSL 选项。这可以通过在 **ssl.conf** 中设置以下选项来完成：

```
SSLProtocol all -SSLv3 -SSLv2 +TLSv1.2 +TLSv1.3
SSLProxyProtocol all -SSLv2 -SSLv3 -TLSv1 -TLSv1.1
```

[1873] https://cgit.freebsd.org/ports/tree/www/apache24/pkg-descr

要在 Web 服务器中完成 SSL 的配置，请取消注释以下行，以确保在重新启动或重新加载期间将配置拉入 Apache：

```
# Secure (SSL/TLS) connections
Include etc/apache24/extra/httpd-ssl.conf
```

也必须在 **httpd.conf** 中取消以下行的注释，才能完全支持 Apache 中的 SSL：

```
LoadModule authn_socache_module libexec/apache24/mod_authn_socache.so
LoadModule socache_shmcb_module libexec/apache24/mod_socache_shmcb.so
LoadModule ssl_module libexec/apache24/mod_ssl.so
```

下一步是与证书颁发机构合作，以便在系统上安装相应的证书。这将为站点设置信任链，并防止出现任何自签名证书警告。

32.8.3.2. mod_perl

mod_perl 模块使得在 Perl 中编写 Apache 模块成为可能。此外，服务器中嵌入的持久性解释器避免了启动外部解释器的开销和 Perl 启动时间的损失。

可以使用软件包或 port www/mod_perl2[1874] 安装 `mod_perl`。有关使用此模块的文档，请参见 http://perl.apache.org/docs/2.0/index.html。

32.8.3.3. mod_php

PHP：超文本预处理器（PHP）是一种通用的脚本语言，特别适合 Web 开发。它能够嵌入到 HTML 中，其语法借鉴了 C、Java™ 和 Perl，旨在允许 Web 开发人员快速编写动态生成的网页。

通过安装适当的 port，可以添加对 Apache 的 PHP 和使用该语言编写的任何其他特性的支持。

对于所有受支持的版本，请使用 `pkg` 命令搜索软件包数据库：

```
# pkg search php
```

将显示一个列表，包括它们提供的版本和其他功能。这些组件是完全模块化的，这意味着通过安装适当的 port 来启用功能。要安装适用于 Apache 的 PHP 版本 7.4，请执行以下命令：

```
# pkg install mod_php74
```

如果需要安装任何依赖包，它们也将被安装。

默认情况下，将不启用 PHP。需要将以下行添加到位于 **/usr/local/etc/apache24** 中的 Apache 配置文件中，以使其处于活动状态：

[1874] https://cgit.freebsd.org/ports/tree/www/mod_perl2/pkg-descr

```
<FilesMatch "\.php$">
    SetHandler application/x-httpd-php
</FilesMatch>
<FilesMatch "\.phps$">
    SetHandler application/x-httpd-php-source
</FilesMatch>
```

此外，配置文件中的 `DirectoryIndex` 配置文件也需要更新，Apache 需要重新启动或重新加载才能使更改生效。

也可以使用 `pkg` 安装对许多 PHP 功能的支持。例如，要安装对 XML 或 SSL 的支持，请安装其各自的 port：

```
# pkg install php74-xml php74-openssl
```

与上文一样，需要重新加载 Apache 配置才能使更改生效，即使它只是模块安装也是如此。

要执行正常重新启动以重新加载配置，请执行以下命令：

```
# apachectl graceful
```

安装完成后，有两种方法可以获取已安装的 PHP 支持模块和构建的环境信息。首先是安装完整的 PHP 二进制文件并运行命令以获取信息：

```
# pkg install php74
# php -i |less
```

有必要将输出传递给分页器，例如 `more` 或 `less` 为了更容易地控制输出量。

最后，要对 PHP 的全局配置进行任何更改，有一个记录良好的文件安装在 **/usr/local/etc/php.ini** 中。在安装时，此文件将不存在，因为有两个版本可供选择，一个是 **php.ini-development**，另一个是 **php.ini-production**。这些是帮助管理员进行部署的起点。

32.8.3.4. HTTP2 支持

默认情况下，在使用 `pkg` 安装 port 时，包括了对 HTTP2 协议的 Apache 支持。新版本的 HTTP 比以前的版本有许多改进，包括利用与网站的单个连接，减少 TCP 连接的整体往返。此外，数据包标头数据被压缩，默认情况下 HTTP2 需要加密。

当 Apache 配置为仅使用 HTTP2 时，Web 浏览器将需要安全、加密的 HTTPS 连接。当 Apache 配置为同时使用这两个版本时，如果在连接过程中出现任何问题，HTTP1.1 将被视为回退选项。

虽然此更改确实需要管理员进行更改，但它们是积极的，相当于为每个人提供更安全的互联网。仅当前未实现 SSL 和 TLS 的站点需要进行这些更改。

注意

更多操作此配置取决于前面的部分，包括 TLS 支持。建议在继续此配置之前遵循这些说明。

通过取消注释 **/usr/local/etc/apache24/httpd.conf** 中的行来启用 http2 模块，然后用 mpm_event 替换 mpm_prefork 模块，因为后者不支持 HTTP2，从而启动该过程。

```
LoadModule http2_module libexec/apache24/mod_http2.so
LoadModule mpm_event_module libexec/apache24/mod_mpm_event.so
```

注意

有一个单独的 port **mod_http2** 可用。它的存在是为了提供比随 port **apache24** 捆绑安装的模块更快的安全性和错误修复。它不是 HTTP2 支持所必需的，但可用。安装后，应在 Apache 配置中使用 **mod_h2.so** 代替 **mod_http2.so**。

在 Apache 中实现 HTTP2 有两种方法;一种方法是针对系统上运行的所有站点和每个 VirtualHost 的全局方法。若要全局启用 HTTP2，请在 ServerName 指令下添加以下行：

```
Protocols h2 http/1.1
```

注意

要通过纯文本启用 HTTP2，请在 **httpd.conf** 中使用 h2h2chttp/1.1。

此处使用 h2c 将允许明文 HTTP2 数据在系统上传递，但不建议这样做。此外，如果系统需要，在此处使用 http/1.1 将允许回退到协议的 HTTP1.1 版本。

要为单个虚拟主机启用 HTTP2，请在 **httpd.conf** 或 **httpd-ssl.conf** 的 VirtualHost 指令中添加相同的行。

使用 `apachectl` 命令重新加载配置，并在访问其中一个托管页面后使用以下任一方法测试配置：

```
# grep "HTTP/2.0" /var/log/httpd-access.log
```

这将返回类似于以下内容的内容：

```
192.168.1.205 - - [18/Oct/2020:18:34:36 -0400] "GET / HTTP/2.0" 304 -
192.0.2.205 - - [18/Oct/2020:19:19:57 -0400] "GET / HTTP/2.0" 304 -
192.0.0.205 - - [18/Oct/2020:19:20:52 -0400] "GET / HTTP/2.0" 304 -
192.0.2.205 - - [18/Oct/2020:19:23:10 -0400] "GET / HTTP/2.0" 304 -
```

另一种方法是使用 Web 浏览器的内置站点调试器或 `tcpdump`;但是，使用任一方法都超出了本文档的讨论范围。

通过使用 **mod_proxy_http2.so** 模块支持 HTTP2 反向代理连接。配置 ProxyPass 或 RewriteRules [P] 语句时，它们应使用 h2:// 进行连接。

32.8.4. 动态网站

除了 mod_perl 和 mod_php 外，还有其他语言可用于创建动态 Web 内容。其中包括 Django 和 Ruby on Rails。

Django 是一个 BSD 许可的框架，旨在允许开发人员快速编写高性能，优雅的 Web 应用程序。它提供了一个对象关系映射器，以便将数据类型开发为 Python 对象。为这些对象提供了丰富的动态数据库访问 API，开发人员无需编写 SQL。它还提供了一个可扩展的模板系统，以便将应用程序的逻辑与 HTML 表示形式分开。

Django 依赖于 **mod_python** 和 SQL 数据库引擎。在 FreeBSD 中，www/py-django[1875] port 会自动安装 **mod_python** 并支持 PostgreSQL、MySQL 或 SQLite 数据库，默认值为 SQLite。要更改数据库引擎，请在 **/usr/ports/www/py-django** 中键入 `make config` 内容，然后安装 port。

安装 Django 后，应用程序将需要一个项目目录以及 Apache 配置才能使用嵌入式 Python 解释器。此解释器用于为站点上的特定 URL 调用应用程序。

若要将 Apache 配置为将某些 URL 的请求传递到 Web 应用程序，请将以下内容添加到 **httpd.conf**，并指定项目目录的完整路径：

```
<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>
```

有关如何使用 Django 的更多信息，请参阅 https://docs.djangoproject.com。

Ruby on Rails 是另一个开源的 Web 框架，它提供了一个完整的开发堆栈。它经过优化，使 Web 开发人员更有效率，并能够快速编写功能强大的应用程序。在 FreeBSD 上，它可以使用软件包或 port www/rubygem-rails[1876] 进行安装。

有关如何使用 Ruby on Rails 的更多信息，请参阅 http://guides.rubyonrails.org。

[1875] https://cgit.freebsd.org/ports/tree/www/py-django/pkg-descr

[1876] https://cgit.freebsd.org/ports/tree/www/rubygem-rails/pkg-descr

32.9. 文件传输协议（FTP）

文件传输协议（FTP）为用户提供了一种在 FTP 服务器之间传输文件的简单方法。FreeBSD 在基本系统中包括 FTP 服务器软件——ftpd。

FreeBSD 提供了几个配置文件来控制对 FTP 服务器的访问。本节总结了这些文件。有关内置 FTP 服务器的更多详细信息，请参阅 ftpd(8)[1877]。

32.9.1. 配置

最重要的配置步骤是确定允许哪些帐户访问 FTP 服务器。FreeBSD 系统有许多系统帐户，这些帐户不应该被允许 FTP 访问。禁止任何 FTP 访问的用户列表可以在 **/etc/ftpusers** 中找到。默认情况下，它包括系统帐户。可以添加不应允许其访问 FTP 的其他用户。

在某些情况下，可能需要限制某些用户的访问，而不完全阻止他们使用 FTP。如 ftpchroot(5)[1878] 中所述，这可以通过创建 **/etc/ftpchroot** 来完成。此文件列出了受 FTP 访问限制的用户和组。

要实现对服务器的匿名 FTP 访问，在 FreeBSD 系统上创建一个名为 `ftp` 的用户。然后，用户将能够以 `ftp` 或 `anonymous` 的用户名登录 FTP 服务器。当提示输入密码时，任何输入都会被接受，但按照惯例，应该使用电子邮件地址作为密码。当一个匿名用户登录时，FTP 服务器将调用 chroot(2)[1879] 以限制对 `ftp` 用户主目录的访问。

可以创建两个文本文件来指定要向 FTP 客户端显示的欢迎消息。**/etc/ftpwelcome** 的内容将在用户到达登录提示之前显示给用户。成功登录后，将显示 **/etc/ftpmotd** 的内容。请注意，此文件的路径是相对于登录环境的，因此将为匿名用户显示 **~ftp/etc/ftpmotd** 的内容。

配置 FTP 服务器后，在 **/etc/rc.conf** 中设置适当的变量以在引导期间启动服务：

```
ftpd_enable="YES"
```

要立即启动该服务，请执行以下操作：

```
# service ftpd start
```

通过键入以下内容来测试与 FTP 服务器的连接：

```
% ftp localhost
```

ftpd 守护程序使用 syslog(3)[1880] 来记录消息。默认情况下，系统日志守护程序将在 **/var/log/xferlog** 中写入与 FTP 相关的消息。FTP 日志的位置可以通过更改 **/etc/syslog.conf** 中的以下行来修改：

[1877] https://www.freebsd.org/cgi/man.cgi?query=ftpd&sektion=8&format=html
[1878] https://www.freebsd.org/cgi/man.cgi?query=ftpchroot&sektion=5&format=html
[1879] https://www.freebsd.org/cgi/man.cgi?query=chroot&sektion=2&format=html
[1880] https://www.freebsd.org/cgi/man.cgi?query=syslog&sektion=3&format=html

```
ftp.info      /var/log/xferlog
```

注意

请注意运行匿名 FTP 服务器所涉及的潜在问题。特别是在允许匿名用户上传文件方面，请三思而后行。事实证明，FTP 站点可能会成为未经许可的商业软件交易的论坛，甚至更糟糕。如果需要匿名 FTP 上载，请验证权限，以便在管理员审阅这些文件之前，使其他匿名用户无法读取这些文件。

32.10. 用于 Microsoft® Windows® 客户端的文件和打印服务（Samba）

Samba 是一个流行的开源软件包，它使用 SMB / CIFS 协议提供文件和打印服务。该协议内置于 Microsoft® Windows® 系统中。可以通过安装 Samba 客户端库将其添加到非 Microsoft® Windows® 系统中。该协议允许客户端访问共享数据和打印机。这些共享可以映射为本地磁盘驱动器，共享打印机可以像使用本地打印机一样使用。

在 FreeBSD 上，Samba 客户端库可以使用软件包或 port 来安装 net/samba413[1881]。客户端为 FreeBSD 系统提供了访问 Microsoft® Windows® 网络中的 SMB/CIFS 共享的功能。

FreeBSD 系统也可以通过安装相同的软件包或 port net/samba413[1882] 来配置充当为 Samba 服务器。这允许管理员在 FreeBSD 系统上创建共享 SMB/CIFS，运行 Microsoft® Windows® 或 Samba 客户端库的客户端可以访问这些共享。

32.10.1. 服务器配置

Samba 在 **/usr/local/etc/smb4.conf** 中进行配置。必须先创建此文件，然后才能使用 Samba。

此处显示了一个简单的 **smb4.conf**，用于与工作组中的 Windows® 客户端共享目录和打印机。对于涉及 LDAP 或 Active Directory 的更复杂的设置，使用 samba-tool(8)[1883] 创建初始 **smb4.conf** 会更容易。

```
[global]
workgroup = WORKGROUP
server string = Samba Server Version %v
netbios name = ExampleMachine
wins support = Yes
security = user
passdb backend = tdbsam

# Example: share /usr/src accessible only to 'developer' user
[src]
```

(continues on next page)

[1881] https://cgit.freebsd.org/ports/tree/net/samba413/pkg-descr

[1882] https://cgit.freebsd.org/ports/tree/net/samba413/pkg-descr

[1883] https://www.freebsd.org/cgi/man.cgi?query=samba-tool&sektion=8&format=html

(continued from previous page)

```
path = /usr/src
valid users = developer
writable  = yes
browsable = yes
read only = no
guest ok = no
public = no
create mask = 0666
directory mask = 0755
```

32.10.1.1. 全局设置

描述网络的设置添加到 **/usr/local/etc/smb4.conf** 中：

- **``workgroup``**

  要提供服务的工作组的名称。

- **``netbios name``**

  已知 Samba 服务器的 NetBIOS 名称。默认情况下，它与主机 DNS 名称的第一个组件相同。

- **``server string``**

  将在 `net view` 的输出中显示的字符串，以及一些其他网络工具，这些工具试图显示有关服务器的描述性文本。

- **``wins support``**

  Samba 是否将充当 WINS 服务器。不要在网络上的多台服务器上启用对 WINS 的支持。

32.10.1.2. 安全设置

/usr/local/etc/smb4.conf 中最重要的设置是安全模型和后端密码格式。这些指令控制选项：

- **``security``**

  最常见的设置是 `security = share` 和 `security = user`。如果客户端使用的用户名与它们在 FreeBSD 机器上的用户名相同，则应使用用户级安全性。这是默认的安全策略，它要求客户端先登录才能访问共享资源。在共享级别安全性中，客户端在尝试连接到共享资源之前，不需要使用有效的用户名和密码登录到服务器。这是旧版 Samba 的默认安全模型。

- **``passdb backend``**

  Samba 有几种不同的后端认证模式。客户端可以通过 LDAP、NIS+、SQL 数据库或修改后的密码文件进行认证。推荐的身份验证方法 `tdbsam` 非常适合于简单的网络，本文将对此进行介绍。对于更大或更复杂的网络，建议使用 `ldapsam`。`smbpasswd` 是以前的默认值，现在已经过时了。

32.10.1.3. Samba 用户

FreeBSD 用户帐户必须映射到 `SambaSAMAccount` 数据库，Windows® 客户端才能访问共享。使用 pdbedit(8)[1884] 可映射现有的 FreeBSD 用户帐户：

```
# pdbedit -a -u username
```

本节仅提及最常用的设置。有关可用配置选项的其他信息，请参阅 Samba 官方 Wiki[1885]。

32.10.2. 启动 Samba

要在引导时启用 Samba，请将以下行添加到 **/etc/rc.conf**：

```
samba_server_enable="YES"
```

现在就启动 Samba：

```
# service samba_server start
Performing sanity check on Samba configuration: OK
Starting nmbd.
Starting smbd.
```

Samba 由三个独立的守护进程组成。守护程序 nmbd 和 smbd 都是由`samba_enable` 启动的。如果还需要域名解析 winbind，请设置：

```
winbindd_enable="YES"
```

可以随时通过键入以下内容来停止 Samba：

```
# service samba_server stop
```

Samba 是一个复杂的软件套件，其功能允许与 Microsoft® Windows® 网络广泛集成。有关此处说明的基本配置之外的功能的详细信息，请参阅 https://www.samba.org[1886]。

[1884] https://www.freebsd.org/cgi/man.cgi?query=pdbedit&sektion=8&format=html
[1885] https://wiki.samba.org/
[1886] https://www.samba.org/

32.11. 用 NTP 进行时钟同步

随着时间的推移，计算机的时钟容易漂移。这是有问题的，因为许多网络服务要求网络上的计算机共享相同的准确时间。还需要准确的时间来确保文件时间戳保持一致。网络时间协议（NTP）是在网络中提供时钟精度的一种方法。

FreeBSD 包含 ntpd(8)[1887]，它可以被配置为查询其他 NTP 服务器以同步该机器上的时钟或为网络中的其他计算机提供时间服务。

本节介绍如何在 FreeBSD 上配置 ntpd。更多文档可以在 **/usr/share/doc/ntp/** 中找到，是 HTML 格式。

32.11.1. NTP 配置

在 FreeBSD 上，内置的 ntpd 可以用来同步系统的时钟。Ntpd 是使用 rc.conf(5)[1888] 变量和 **/etc/ntp.conf** 配置的，详见以下各节。

Ntpd 使用 UDP 数据包与其网络对等体进行通信。你的计算机与其 NTP 对等体之间的任何防火墙都必须配置为允许 UDP 数据包在端口 123 上传入和传出。

32.11.1.1. /etc/ntp.conf 文件

Ntpd 读取 **/etc/ntp.conf** 来决定要查询哪些 NTP 服务器。建议选择多个 NTP 服务器，以防止其中一个服务器无法到达或其时钟不可靠。当 ntpd 收到响应时，它倾向于选择可靠的服务器而不是不太可靠的。被查询的服务器可以是网络中的本地服务器，由 ISP 提供，也可以是从可公开访问的 NTP 服务器的联机列表[1889]中选择的。选择公共 NTP 服务器时，请选择地理位置接近的服务器并查看其使用策略。配置关键字从服务器池中选择一个或多个服务器。提供可公开访问的 NTP 池的在线列表[1890]，按地理区域进行组织。此外，FreeBSD 还提供了一个项目赞助的网站 `0.freebsd.pool.ntp.org`。

例 **46.**示例 **/etc/ntp.conf**

这是 **ntp.conf** 文件的一个简单示例。它可以安全地按原样使用 `restrict`;它包含了在可公开访问的网络连接上操作的建议选项。

```
# Disallow ntpq control/query access.  Allow peers to be added only
# based on pool and server statements in this file.
restrict default limited kod nomodify notrap noquery nopeer
restrict source  limited kod nomodify notrap noquery

# Allow unrestricted access from localhost for queries and control.
restrict 127.0.0.1
```

(continues on next page)

[1887] https://www.freebsd.org/cgi/man.cgi?query=ntpd&sektion=8&format=html
[1888] https://www.freebsd.org/cgi/man.cgi?query=rc.conf&sektion=5&format=html
[1889] http://support.ntp.org/bin/view/Servers/WebHome
[1890] http://support.ntp.org/bin/view/Servers/NTPPoolServers

(continued from previous page)

```
restrict ::1

# Add a specific server.
server ntplocal.example.com iburst

# Add FreeBSD pool servers until 3-6 good servers are available.
tos minclock 3 maxclock 6
pool 0.freebsd.pool.ntp.org iburst

# Use a local leap-seconds file.
leapfile "/var/db/ntpd.leap-seconds.list"
```

此文件的格式在 ntp.conf(5)[1891] 中有说明。下面的说明仅提供了上述示例文件中使用的关键字的快速概述。

默认情况下，任何网络主机都可以访问 NTP 服务器。关键字 `restrict` 控制哪些系统可以访问该服务器。支持多个限制项，每个限制项都能完善前面语句中给出的限制。例子中显示的值授予本地系统完全的查询和控制权限，而只允许远程系统查询时间的能力。更多细节，请参考 ntp.conf(5)[1892] 的 `Access Control Support` 一节。

关键字 `server` 指定要查询的单一服务器。该文件可以包含多个服务器关键字，每行列出一个服务器。`pool` 关键字指定了一个服务器池。Ntpd 将根据需要从这个池中添加一个或多个服务器，以达到使用 `tos minclock` 值指定的对等体的数量。`iburst` 关键字指示 `ntpd` 在第一次建立联系时与一个服务器执行八次快速数据包交换的突发，以帮助快速同步系统时间。

关键字 `leapfile` 指定了一个包含闰秒信息的文件的位置。该文件由 periodic(8)[1893] 自动更新。此关键字指定的文件位置必须与 **/etc/rc.conf** 中的 `ntp_db_leapfile` 变量中设置的位置匹配。

32.11.1.2. /etc/rc.conf 中的 NTP 条目

设置 `ntpd_enable=YES` 为在启动时启动 ntpd。把 `ntpd_enable=YES` 添加到 **/etc/rc.conf** 后，就可以立即启动 ntpd 而无需通过键入以下内容重新启动系统：

```
# service ntpd start
```

仅必须设置 `ntpd_enable` 为使用 ntpd。下面列出的 **rc.conf** 变量也可以根据需要进行设置。

设置 `ntpd_sync_on_start=YES` 为允许 ntpd 在启动时对时钟进行任意量的步进。通常，ntpd 将记录一条错误消息，如果时钟关闭超过 1000 秒，则退出。此选项在没有电池供电的实时时钟的系统上特别有用。

设置 `ntpd_oomprotect=YES` 防止 ntpd 守护程序被尝试从内存不足（OOM）状态恢复的系统杀死。

[1891] https://www.freebsd.org/cgi/man.cgi?query=ntp.conf&sektion=5&format=html
[1892] https://www.freebsd.org/cgi/man.cgi?query=ntp.conf&sektion=5&format=html
[1893] https://www.freebsd.org/cgi/man.cgi?query=periodic&sektion=8&format=html

设置 ntpd_config= 为备用 **ntp.conf** 文件的位置。

根据需要设置 ntpd_flags= 为包含任何其他 ntpd 参数，但避免使用这些由 **/etc/rc.d/ntpd** 内部管理的参数：

- -p（pid 文件的位置）
- -c（设置 ntpd_config= 代替）

32.11.1.3. Ntpd 和未授权的 ntpd 用户

FreeBSD 上的 Ntpd 可以以非特权用户的身份启动和运行。这样做需要 mac_ntpd(4)[1894] 策略模块。**/etc/rc.d/ntpd** 启动脚本首先检查 NTP 配置。如果可能，它会加载 mac_ntpd 模块，然后以非授权用户 ntpd（用户 ID 123）的身份启动 ntpd。为了避免文件和目录访问的问题，当配置中包含任何与文件有关的选项时，启动脚本不会自动以 ntpd 身份启动 ntpd。

ntpd_flags 中存在以下任何一项都需要手动配置，如下所述才能以 ntpd 用户身份运行：

- -f 或–driftfile
- -i 或–jaildir
- -k 或–keyfile
- -l 或–logfile
- -s 或–statsdir

ntp.conf 中存在以下任何关键字都需要手动配置，如下所述才能以 ntpd 用户身份运行：

- crypto
- driftfile
- key
- logdir
- statsdir

要手动将 ntpd 配置为以 ntpd 用户身份运行，你必须：

- 确保 ntpd 用户有权访问配置中指定的所有文件和目录。
- 安排 mac_ntpd 模块加载或编译到内核中。有关详细信息，请参见 mac_ntpd(4)[1895]。
- 在 **/etc/rc.conf** 中设置 ntpd_user="ntpd"

[1894] https://www.freebsd.org/cgi/man.cgi?query=mac_ntpd&sektion=4&format=html
[1895] https://www.freebsd.org/cgi/man.cgi?query=mac_ntpd&sektion=4&format=html

32.11.2. 将 NTP 与 PPP 连接结合使用

ntpd 不需要永久连接到互联网即可正常运行。然而，如果 PPP 连接被配置为按需拨号，则应该阻止 NTP 流量触发拨号或保持连接处于活动状态。可以在 **/etc/ppp/ppp.conf** 中使用 `filter` 指令配置。例如：

```
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
```

要了解更多的细节，请参考 ppp(8)[1896] 中的 `PACKET FILTERING` 部分和 **/usr/share/examples/ppp/** 中的例子。

> 注意
> 一些互联网接入提供商会封锁低编号的端口，因为应答永远无法到达计算机，会阻止 NTP 正常运行。

32.12. iSCSI target 和 initiator 的配置

iSCSI 是一种通过网络共享存储的方法。与在文件系统级别工作的 NFS 不同，iSCSI 在块设备级别工作。

在 iSCSI 术语中，共享存储的系统称为 *target*。存储可以是物理磁盘，也可以是表示多个磁盘的区域或物理磁盘的一部分。例如，如果磁盘是使用 ZFS 格式化的，则可以创建一个 zvol 以用作 iSCSI 存储。

访问 iSCSI 存储的客户端称为 *initiator*。对于 initiator，通过 iSCSI 提供的存储显示为原始的、未格式化的磁盘，称为 LUN。磁盘的设备节点显示在 **/dev/** 中，并且必须单独格式化和装载设备。

FreeBSD 提供了一个原生的、基于内核的 iSCSI target 和 initiator。本节说明如何将 FreeBSD 系统配置为 target 或 initiator。

32.12.1. 配置 iSCSI target

要配置 iSCSI target，请创建 **/etc/ctl.conf** 配置文件，向 **/etc/rc.conf** 添加一行以确保 ctld(8)[1897] 守护程序在引导时自动启动，然后启动守护程序。

下面是一个简单的 **/etc/ctl.conf** 配置文件的示例。有关此文件可用选项的更完整说明，请参阅 ctl.conf(5)[1898]。

[1896] https://www.freebsd.org/cgi/man.cgi?query=ppp&sektion=8&format=html
[1897] https://www.freebsd.org/cgi/man.cgi?query=ctld&sektion=8&format=html
[1898] https://www.freebsd.org/cgi/man.cgi?query=ctl.conf&sektion=5&format=html

```
portal-group pg0 {
    discovery-auth-group no-authentication
    listen 0.0.0.0
    listen [::]
}

target iqn.2012-06.com.example:target0 {
    auth-group no-authentication
    portal-group pg0

    lun 0 {
        path /data/target0-0
        size 4G
    }
}
```

第一个条目定义了 `pg0` 门户组。门户组定义了 ctld(8)[1899] 守护程序将监听的网络地址。`discovery-auth-group no-authentication` 条目表示允许任何 initiator 执行 iSCSI target 发现，而无需验证。第三行和第四行配置 ctld(8)[1900] 在所有 IPv4（`listen 0.0.0.0`）和 IPv6（`listen [:]`）地址上监听，默认端口为 `3260`。

没有必要定义一个门户组，因为有一个内置的门户组叫 `default`。在这种情况下，`default` 和 `pg0` 的区别在于，对于 `default`，target 发现总是被拒绝，而对于`pg0`，总是被允许。

第二个条目定义了一个单一的 target。target 有两种可能的含义：一台为 iSCSI 服务的机器或一个命名的 LUN 组。这个例子使用了后者的含义，其中 `iqn.2012-06.com.example:target0` 是 target 名称。这个 target 名称适合用于测试目的。实际使用时，请将 `com.example` 修改为真实域名的倒写。2012-06 代表获得该域名控制权的年份和月份，而 `target0` 可以是任何值。在这个配置文件中可以定义任何数量的 target。

`auth-group no-authentication` 行允许所有启动程序连接到指定的 target，`portal-group pg0` 使 target 可以通过 pg0 门户组到达。

下一节定义了 LUN。对启动程序来说，每个 LUN 都将作为一个单独的磁盘设备可见。可以为每个 target 定义多个 LUN。每个 LUN 由一个数字标识，其中 LUN 0 是必须的。`path /data/target0-0` 行定义了支持 LUN 的文件或 `Zvol` 的完整路径。在启动 ctld(8)[1901] 之前，该路径必须存在。第二行是可选的，指定 LUN 的大小。

接下来，要确保 ctld(8)[1902] 守护程序在引导时启动，请将以下行添加到 **/etc/rc.conf**：

```
ctld_enable="YES"
```

[1899] https://www.freebsd.org/cgi/man.cgi?query=ctld&sektion=8&format=html
[1900] https://www.freebsd.org/cgi/man.cgi?query=ctld&sektion=8&format=html
[1901] https://www.freebsd.org/cgi/man.cgi?query=ctld&sektion=8&format=html
[1902] https://www.freebsd.org/cgi/man.cgi?query=ctld&sektion=8&format=html

要立即启动 ctld(8)[1903]，请运行以下命令：

```
# service ctld start
```

当 ctld(8)[1904] 守护程序启动时，它读取 **/etc/ctl.conf**。如果在守护程序启动后编辑此文件，请使用以下命令，以便更改立即生效：

```
# service ctld reload
```

32.12.1.1. 身份验证

前面的示例本质上是不安全的，因为它不使用身份验证，而是授予任何人对所有 target 的完全访问权限。要要求用户名和密码才能访问 target，请按如下方式修改配置：

```
auth-group ag0 {
    chap username1 secretsecret
    chap username2 anothersecret
}

portal-group pg0 {
    discovery-auth-group no-authentication
    listen 0.0.0.0
    listen [::]
}

target iqn.2012-06.com.example:target0 {
    auth-group ag0
    portal-group pg0
    lun 0 {
        path /data/target0-0
        size 4G
    }
}
```

`auth-group` 部分定义了用户名和密码对。试图连接到 `iqn.2012-06.com.example:target0` 的发起人必须首先指定一个定义的用户名和密码。但是，在没有身份验证的情况下，仍然允许 target 发现。要要求 target 发现身份验证，请将 `discovery-auth-group` 设置为已定义的 `auth-group` 名称，而不是 `no-authentication`。

通常为每个 initiator 定义一个导出的 target。作为上述语法的简写，可以直接在 target 条目中指定用户名和密码：

[1903] https://www.freebsd.org/cgi/man.cgi?query=ctld&sektion=8&format=html
[1904] https://www.freebsd.org/cgi/man.cgi?query=ctld&sektion=8&format=html

```
target iqn.2012-06.com.example:target0 {
    portal-group pg0
    chap username1 secretsecret

    lun 0 {
        path /data/target0-0
        size 4G
    }
}
```

32.12.2. 配置 iSCSI 启动程序

> 注意
>
> 本节中说明的 iSCSI 启动程序从 FreeBSD 10.0-RELEASE 开始受支持。要在旧版本中使用可用的 iSCSI initiator，请参阅 iscontrol(8)[1905]。

iSCSI 发起程序要求 iscsid(8)[1906] 守护程序正在运行。此守护程序不使用配置文件。要在启动时自动启动它，请将以下行添加到 **/etc/rc.conf**：

```
iscsid_enable="YES"
```

要立即启动 iscsid(8)[1907]，请运行以下命令：

```
# service iscsid start
```

无论是否使用 **/etc/iscsi.conf** 配置文件，都可以连接到 target。本节演示这两种类型的连接。

32.12.2.1. 在没有配置文件的情况下连接到 target

要将 initiator 连接到单个 target，请指定门户的 IP 地址和 target 的名称：

```
# iscsictl -A -p 10.10.10.10 -t iqn.2012-06.com.example:target0
```

要验证连接是否成功，请运行不带任何参数的 `iscsictl`。输出应该如下所示：

```
Target name                                    Target portal    State
iqn.2012-06.com.example:target0                10.10.10.10      Connected: da0
```

[1905] https://www.freebsd.org/cgi/man.cgi?query=iscontrol&sektion=8&format=html
[1906] https://www.freebsd.org/cgi/man.cgi?query=iscsid&sektion=8&format=html
[1907] https://www.freebsd.org/cgi/man.cgi?query=iscsid&sektion=8&format=html

在这个例子中，iSCSI 会话被成功建立，**/dev/da0** 代表连接的 LUN。如果 `iqn.2012-06.com.example:target0` target 导出了一个以上的 LUN，多个设备节点将显示在输出的那个部分。

```
Connected: da0 da1 da2.
```

任何错误都将在输出以及系统日志中报告。例如，此消息通常意味着 iscsid(8)[1908] 守护程序未运行：

```
Target name                                     Target portal   State
iqn.2012-06.com.example:target0                 10.10.10.10     Waiting for iscsid(8)
```

以下消息提示存在网络问题，例如错误的 IP 地址或端口：

```
Target name                                     Target portal   State
iqn.2012-06.com.example:target0                 10.10.10.11     Connection refused
```

此消息表示指定的 target 名称有误：

```
Target name                                     Target portal   State
iqn.2012-06.com.example:target0                 10.10.10.10     Not found
```

此消息表示 target 需要身份验证：

```
Target name                                     Target portal   State
iqn.2012-06.com.example:target0                 10.10.10.10     Authentication failed
```

若要指定 CHAP 用户名和密码，请使用以下语法：

```
# iscsictl -A -p 10.10.10.10 -t iqn.2012-06.com.example:target0 -u user -s␣
↪secretsecret
```

32.12.2.2. 使用配置文件连接到 target

要使用配置文件进行连接，请使用如下内容创建 **/etc/iscsi.conf**：

```
t0 {
    TargetAddress   = 10.10.10.10
    TargetName      = iqn.2012-06.com.example:target0
    AuthMethod      = CHAP
    chapIName       = user
    chapSecret      = secretsecret
}
```

[1908] https://www.freebsd.org/cgi/man.cgi?query=iscsid&sektion=8&format=html

`t0` 指定了配置文件部分的昵称。initiator 将使用它来指定要使用的配置。其他行指定了在连接期间要使用的参数：`TargetAddress` 和 `TargetName` 是必需的，而其他选项是可选的。在此示例中，将显示 CHAP 用户名和密码。

要连接到定义的 target，请指定昵称：

```
# iscsictl -An t0
```

或者，要连接到配置文件中定义的所有 target，请使用：

```
# iscsictl -Aa
```

要使 initiator 自动连接到 **/etc/iscsi.conf** 中的所有 target，请将以下内容添加到 **/etc/rc.conf**：

```
iscsictl_enable="YES"
iscsictl_flags="-Aa"
```

第三十三章防火墙

33.1.概述

防火墙可以过滤经过系统的出入站流量。它可以使用一组或多组“规则”来检查网络连接中的出入流量并决定通过或阻止。防火墙的规则可以检查数据包的单个或多个特征，例如协议类型、来源主机或目标主机的地址以及来源主机或目标主机的端口。

防火墙可以增强主机或网络的安全性。它们可用于执行下列一项或多项操作：

- 保护内部网络中的应用程序、服务和计算机并隔离来自互联网中的有害流量。
- 限制或禁止内部网络的主机访问互联网的服务。
- 支持网络地址转换（NAT），它允许内部网络使用私有 IP 地址，并使用单个 IP 地址或自动分配的公用地址的共享池来共享与公用互联网的单个连接。

FreeBSD 在基本系统中内置了三个防火墙：PF、IPFW 和 IPFILTER（也称为 IPF）。FreeBSD 还提供了两个流量整形器来控制带宽的使用：altq(4)[1909] 和 dummynet(4)[1910]。在传统上 ALTQ 与 PF 紧密相连，而 IPFW 则与 dummynet 紧密联系。每个防火墙都使用规则来控制进出 FreeBSD 系统的数据包的访问，尽管它们以不同的方式实现，各自也都有不同的规则语法。

FreeBSD 提供了多个防火墙，以满足各种用户的不同需求和偏好。每个用户都应评估哪种防火墙最能满足他们的需求。

读完本章，你就会知道：

- 如何自定义数据包过滤规则。
- FreeBSD 内置防火墙之间的差异。
- 如何使用和配置 PF 防火墙。
- 如何使用和配置 IPFW 防火墙。

[1909] https://www.freebsd.org/cgi/man.cgi?query=altq&sektion=4&format=html

[1910] https://www.freebsd.org/cgi/man.cgi?query=dummynet&sektion=4&format=html

- 如何使用和配置 IPFILTER 防火墙。

在阅读本章之前，你应该：

- 了解 FreeBSD 基础和互联网概念。

> 注意
>
> 由于所有防火墙都基于检查所选数据包控制字段的值，因此防火墙规则集的创建者必须了解 TCP/IP 的工作原理、数据包控制字段中的不同值是什么，以及在一般的会话中这些值的作用。有关的较好的介绍，请参阅 Daryl 的 TCP/IP 入门[1911]。

33.2.防火墙的概念

规则集包含一组规则，这些规则根据数据包中包含的值传递或阻止数据包。主机之间的双向数据包交换包括会话会话。防火墙规则集既处理来自互联网的数据包，也处理系统生成的数据包作为对它们的响应。每个 TCP/IP 服务都由其协议和侦听端口预定义。发往特定服务的数据包使用非特权端口从源地址发出，并以目标地址上的特定服务端口为目标。以上所有参数都可以用作选择标准，以创建将传递或阻止服务的规则。

要查找未知的端口号，请参阅 **/etc/services**。或者，访问 http://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers 并执行端口号查找以查找特定端口号的用途。

查看此链接，了解特洛伊木马使用的端口号[1912]。

FTP 有两种模式：主动模式和被动模式。不同之处在于如何获取数据通道。被动模式更安全，因为数据通道由序号 ftp 会话请求者获取。有关 FTP 和不同模式的良好说明，请参阅 http://www.slacksite.com/other/ftp.html。

防火墙规则集可以是“exclusive”（明示拒绝）或“inclusive”（明示允许）。明示拒绝防火墙允许除与规则集匹配的流量之外的所有流量通过。明示允许防火墙执行相反操作，因为它只允许与规则匹配的流量通过并阻止其他所有内容。

明示允许防火墙可以更好地控制传出流量，使其成为向互联网提供服务的系统的最佳选择。它还控制来自互联网的流量类型，这些流量可以访问专用网络。所有与规则不匹配的流量都将被阻止并记录。明示允许防火墙通常比明示拒绝防火墙更安全，因为它们可显著降低允许不需要的流量的风险。

> 注意
>
> 除非另有说明，否则本章中的所有配置和示例规则集都将创建明示允许防火墙的规则集。

使用“状态防火墙”可以进一步加强安全性。这种类型的防火墙跟踪打开的连接，并且仅允许与现有连接匹配或打开新的允许连接的流量。

有状态筛选将流量视为包含会话的数据包的双向交换。在匹配规则上指定状态时，防火墙会为会话期间交换的每个预期数据包动态生成内部规则。它具有足够的匹配功能，可以确定数据包是否对会话有效。任何不符合会话模板的数据包都将自动被拒绝。

会话完成后，将从动态状态表中删除该会话。

[1911] http://www.ipprimer.com/

[1912] http://web.archive.org/web/20150803024617/http://www.sans.org/security-resources/idfaq/oddports.php

有状态筛选允许人们专注于阻止/传递新会话。如果新会话已通过，则会自动允许其所有后续数据包，并自动拒绝任何冒名顶替者数据包。如果新会话被阻止，则不允许其后续数据包。有状态筛选提供高级匹配功能，能够抵御攻击者使用的不同攻击方法的泛滥。

NAT 代表 网络地址转换。NAT 功能使防火墙后面的专用 LAN 能够共享单个 ISP 分配的 IP 地址，即使该地址是动态分配的。NAT 允许局域网中的每台计算机都可以访问互联网，而无需为多个互联网帐户或 IP 地址向 ISP 付费。

NAT 会在数据包离开绑定到互联网的防火墙时，将自动将 LAN 上每个系统的专用 LAN IP 地址转换为单个公共 IP 地址。它还对返回的数据包执行反向转换。

根据 RFC 1918，以下 IP 地址范围是为永远不会直接路由到互联网的专用网络保留的，因此可用于 NAT：

- `10.0.0.0/8`.
- `172.16.0.0/12`.
- `192.168.0.0/16`.

> 警告
>
> 使用防火墙规则时，务必万分慎重。某些配置 可以将管理员锁定在服务器之 外。为了安全起见，请考虑从本地控制台执行初始防火墙配置，而不是通过 ssh 远程执行。

33.3. PF

从 FreeBSD 5.3 开始，OpenBSD 的 PF 防火墙的移植版本就作为基本系统的集成部分被包含进来了。PF 是一个功能齐全的完整防火墙，具有对 ALTQ（交错队列）的可选支持，后者提供服务质量（QoS）。

OpenBSD 项目在 PF FAQ 中维护了 PF[1913] 的权威参考。Peter Hansteen 在 http://home.nuug.no/~peter/pf/ 维护着一个全面的 PF 教程。

> 警告
>
> 在阅读 PF FAQ[1914] 时，请记住，多年以来，FreeBSD 的 PF 版本与上游的 OpenBSD 版本有很大的不同。并非所有功能在 FreeBSD 上的工作方式都与在 OpenBSD 中的工作方式相同，反之亦然。

FreeBSD 数据包过滤器邮件列表[1915] 是询问有关配置和运行 PF 防火墙的问题的好地方。在提出问题之前，请检查邮件列表存档，因为它可能已经得到解答。

手册的这一部分重点介绍与 FreeBSD 相关的 PF。它演示了如何启用 PF 和 ALTQ。它还提供了几个在 FreeBSD 系统上创建规则集的示例。

[1913] http://www.openbsd.org/faq/pf/

[1914] http://www.openbsd.org/faq/pf/

[1915] https://lists.freebsd.org/subscription/freebsd-pf

33.3.1.启用 PF

要使用 PF，必须首先加载其内核模块。本节介绍可以添加到 **/etc/rc.conf** 以启用 PF 的条目。

首先添加 `pf_enable=yes` 到 **/etc/rc.conf**：

```
# sysrc pf_enable=yes
```

pfctl(8)[1916] 中提及的其他选项可以在启动时传递给 PF。在 **/etc/rc.conf** 中添加或更改此条目，并在两个引号（""）之间指定任何必需的参数：

```
pf_flags=""                     # additional flags for pfctl startup
```

如果 PF 找不到其规则集配置文件，则 PF 将不会启动。默认情况下，FreeBSD 不附带规则集，并且没有 **/etc/pf.conf**。可以在 **/usr/share/examples/pf/** 中找到示例规则集。如果自定义规则集已保存在其他位置，请在 **/etc/rc.conf** 中添加一行，指定文件的完整路径：

```
pf_rules="/path/to/pf.conf"
```

由 pflog(4)[1917] 提供 PF 的日志记录支持。要启用日志记录支持，请添加 `pflog_enable=yes` 到 **/etc/rc.conf**：

```
# sysrc pflog_enable=yes
```

还可以添加以下行来更改日志文件的默认位置，或指定在启动时传递给 pflog(4)[1918] 的任何其他参数：

```
pflog_logfile="/var/log/pflog"  # where pflogd should store the logfile
pflog_flags=""                  # additional flags for pflogd startup
```

最后，如果防火墙后面有 LAN，并且需要为 LAN 上的计算机转发数据包，或者需要 NAT，请启用以下选项：

```
gateway_enable="YES"            # Enable as LAN gateway
```

保存所需的编辑后，可以通过键入以下内容来启动 PF 并具有日志记录支持：

```
# service pf start
# service pflog start
```

[1916] https://www.freebsd.org/cgi/man.cgi?query=pfctl&sektion=8&format=html
[1917] https://www.freebsd.org/cgi/man.cgi?query=pflog&sektion=4&format=html
[1918] https://www.freebsd.org/cgi/man.cgi?query=pflog&sektion=4&format=html

默认情况下，PF 从 **/etc/pf.conf** 读取其配置规则，并根据此文件中指定的规则或定义修改、丢弃或传递数据包。FreeBSD 安装包括几个位于 **/usr/share/examples/pf/** 中的样本文件。有关 PF 规则集的完整覆盖范围，请参阅 PF 常见问题解答[1919]。

要控制 PF，请使用 `pfctl`。有用的 pfctl[1920] 选项总结了此命令的一些有用选项。有关所有可用选项的说明，请参阅 pfctl(8)[1921]：

表 28. 有用的 pfctl 选项

命令	目的
`pfctl -e`	启用 PF。
`pfctl -d`	禁用 PF。
`pfctl -F all -f /etc/pf.conf`	刷新所有 NAT、筛选器、状态和表规则，然后重新加载 **/etc/pf.conf**。
`pfctl -s [ rules | nat | states ]`	对过滤规则、NAT 规则或状态表进行报告。
`pfctl -vnf /etc/pf.conf`	检查 **/etc/pf.conf** 是否存在错误，但不加载规则集。

> 技巧
>
> security/sudo[1922] 对于运行 `pfctl` 需要提升权限的命令非常有用。它可以从 ports 安装。

若要监视通过 PF 防火墙的流量，请考虑安装软件包或 port sysutils/pftop[1923]。安装后，可以运行 pftop 以类似于 top(1)[1924] 的格式查看正在运行的流量快照。

33.3.2. PF 规则集

本节演示了如何创建自定义规则集。它从最简单的规则集开始，并使用几个示例在其概念的基础上，以演示 PF 的许多功能的实际使用情况。

最简单的规则集适用于不运行任何服务且需要访问单个网络（可能是互联网）的单个计算机。要创建此最简规则集，请编辑 **/etc/pf.conf**，使其如下所示：

```
block in all
pass out all keep state
```

默认情况下，第一个规则拒绝所有传入流量。第二条规则允许此系统创建的连接传出，同时保留这些连接的状态信息。此状态信息允许这些连接的返回流量回传，并且应仅在可信任的计算机上使用。可以使用以下命令加载该规则集：

[1919] http://www.openbsd.org/faq/pf/

[1920] https://docs.freebsd.org/en/books/handbook/firewalls/#pfctl

[1921] https://www.freebsd.org/cgi/man.cgi?query=pfctl&sektion=8&format=html

[1922] https://cgit.freebsd.org/ports/tree/security/sudo/pkg-descr

[1923] https://cgit.freebsd.org/ports/tree/sysutils/pftop/pkg-descr

[1924] https://www.freebsd.org/cgi/man.cgi?query=top&sektion=1&format=html

```
# pfctl -e ; pfctl -f /etc/pf.conf
```

除了保持状态之外，PF 还提供列表和宏，可以定义这些 列表和 宏以用于创建规则。宏可以包含列表，需要在使用前进行定义。例如，在规则集的最顶部插入以下行：

```
tcp_services = "{ ssh, smtp, domain, www, pop3, auth, pop3s }"
udp_services = "{ domain }"
```

PF 可理解端口名称和端口号，只要这些名称列在 **/etc/services** 中即可。此示例创建了两个宏。第一个是包含七个 TCP 端口名的列表，第二个是一个 UDP 端口名。定义后，可以在规则中使用宏。在此示例中，除了此系统为七个指定的 TCP 服务和一个指定的 UDP 服务启动的连接之外，所有流量都被阻止：

```
tcp_services = "{ ssh, smtp, domain, www, pop3, auth, pop3s }"
udp_services = "{ domain }"
block all
pass out proto tcp to any port $tcp_services keep state
pass proto udp to any port $udp_services keep state
```

尽管 UDP 被认为是无状态协议，但 PF 能够跟踪一些状态信息。例如，当传递一个 UDP 请求，询问域名服务器有关域名的信息时，PF 将监视响应以将其传递回去。

每当对规则集进行编辑后，都必须加载新规则，以便可以使用它们：

```
# pfctl -f /etc/pf.conf
```

如果没有语法错误，`pfctl` 则不会在规则加载期间输出任何消息。在尝试加载规则之前，还可以对其进行测试：

```
# pfctl -nf /etc/pf.conf
```

包含 `-n` 会导致规则仅被解释，但不会加载。这提供了更正任何错误的机会。在任何时候，都将强制执行加载的最后一个有效规则集，直到禁用 PF 或加载新规则集。

> 技巧
>
> 在 pfctl 规则集验证或加载中添加 -v，将显示完全解析过的规则，与它们将被加载的方式完全一致。这在调试规则时是非常有用的。

33.3.2.1.具有 NAT 的简单网关

本节演示如何配置运行 PF 的 FreeBSD 系统作为至少一台其他机器的网关。网关至少需要两个网络接口，每个接口连接到单独的网络。在此示例中，**xl0** 连接到互联网，**xl1** 连接到内部网络。

首先，启用网关，让计算机将其在一个接口上收到的网络流量转发到另一个接口。此系统堆栈设置将转发 IPv4 数据包：

```
# sysctl net.inet.ip.forwarding=1
```

要转发 IPv6 流量，请使用：

```
# sysctl net.inet6.ip6.forwarding=1
```

要在系统启动时启用这些设置，请使用 sysrc(8)[1925] 将它们添加到 **/etc/rc.conf**：

```
# sysrc gateway_enable=yes
# sysrc ipv6_gateway_enable=yes
```

命令 `ifconfig` 可以验证两个接口是否已启动并正在运行。

接下来，创建 PF 规则以允许网关传递流量。虽然以下规则允许来自内部网络主机的有状态流量传递到网关，但 `to` 关键字不保证从源到目标的一直传递：

```
pass in on xl1 from xl1:network to xl0:network port $ports keep state
```

该规则仅允许流量传入内部接口上的网关。要让数据包走得更远，需要一个匹配的规则：

```
pass out on xl0 from xl1:network to xl0:network port $ports keep state
```

虽然这两个规则将起作用，但很少需要这种特定的规则。对于繁忙的网络管理员，可读的规则集是更安全的规则集。本节的其余部分演示如何使规则尽可能简单，以提高可读性。例如，这两个规则可以替换为一个规则：

```
pass from xl1:network to any port $ports keep state
```

`interface:network` 符号可以用一个宏来代替，使规则集更易读。例如，`$localnet` 宏可以被定义为直接连接到内部接口的网络（`$xl1:network`）。另外，`$localnet` 的定义可以改为 *IP 地址/网络掩码*符号来表示一个网络，比如 `192.168.100.1/24` 表示一个私有地址的子网。

如果需要，`$localnet` 甚至可以被定义为一个网络的列表。不管具体的需求是什么，一个合理的 `$localnet` 定义可以在一个典型的通行规则中使用，如下：

[1925] https://www.freebsd.org/cgi/man.cgi?query=sysrc&sektion=8&format=html

```
pass from $localnet to any port $ports keep state
```

以下示例规则集允许由内部网络上的计算机发起的所有流量。它首先定义两个宏来表示网关的外部和内部 3COM 接口。

> 注意
>
> 对于拨号用户，外部接口将使用 **tun0**。对于 ADSL 连接，特别是那些使用以太网 PPP（PPPoE）的连接，正确的外部接口是 **tun0**，而不是以太网物理接口。

```
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
```

这个规则集引入了 `nat` 规则，用来处理从内部网络的不可路由地址到分配给外部接口的 IP 地址的网络地址转换。当外部接口的 IP 地址被动态分配时，`nat` 规则的最后一部分（`$ext_if`）周围的括号被包括在内。它可以确保即使外部 IP 地址发生变化，网络流量的运行也不会出现严重中断。

请注意，此规则集可能允许从网络中传出的流量超过所需的流量。一个合理的设置可以创建此宏：

```
client_out = "{ ftp-data, ftp, ssh, domain, pop3, auth, nntp, http, \
    https, cvspserver, 2628, 5999, 8000, 8080 }"
```

在主传递规则中使用：

```
pass inet proto tcp from $localnet to any port $client_out \
    flags S/SA keep state
```

可能需要一些其他的通过规则。这个在外部接口上启用 SSH：

```
pass in inet proto tcp to $ext_if port ssh
```

此宏定义和规则允许内部客户端使用 DNS 和 NTP：

```
udp_services = "{ domain, ntp }"
pass quick inet proto { tcp, udp } to any port $udp_services keep state
```

注意这个规则中的 `quick` 关键字。由于规则集由多个规则组成，因此了解规则集中规则之间的关系很重要。规则从上到下，按照它们的编写顺序进行评估。对于 PF 评估的每个数据包或连接，规则集中 *最后一个匹配的规则*是被应用的。然而，当一个数据包与包含 `quick` 关键字的规则相匹配时，规则处理就会停止，数据包将按照该规则处理。当需要对一般规则进行例外处理时，这非常有用。

33.3.2.2.创建 FTP 代理

由于 FTP 协议的性质，配置有效的 FTP 规则可能会有问题。FTP 比防火墙早了几十年，并且在设计上是不安全的。最常见的反对使用 FTP 的观点有：

- 密码以明文形式传输。
- 该协议要求在单独的端口上使用至少两个 TCP 连接（控制和数据）。
- 建立会话后，使用随机选择的端口进行数据通信。

所有这些要点都带来了安全挑战，甚至在考虑客户端或服务器软件中的任何潜在安全漏洞之前也是如此。存在更安全的文件传输替代方案，例如 sftp(1)[1926] 或 scp(1)[1927]，它们都具有身份验证和通过加密连接传输数据的功能。

对于需要 FTP 的情况，PF 提供了将 FTP 的流量重定向到一个名为 ftp-proxy(8)[1928] 的小型代理程序，它包含在 FreeBSD 基本系统中。代理的作用是使用一组锚点在规则集中动态插入和删除规则，以正确处理 FTP 流量。

要启用 FTP 代理，请将此行添加到 **/etc/rc.conf**：

```
ftppproxy_enable="YES"
```

然后启动代理，通过运行：

```
# service ftp-proxy start
```

对于基本配置，需要将三个元素添加到 **/etc/pf.conf** 中。首先，代理将用于插入它为 FTP 会话生成的规则的锚点：

```
nat-anchor "ftp-proxy/*"
rdr-anchor "ftp-proxy/*"
```

其次，需要传递规则来允许 FTP 流量进入代理。

第三，重定向和 NAT 规则需要在过滤规则之前定义。在 NAT 规则之后立即插入这个 `rdr` 规则：

```
rdr pass on $int_if proto tcp from any to any port ftp -> 127.0.0.1 port 8021
```

最后，允许重定向的流量通过：

```
pass out proto tcp from $proxy to any port ftp
```

[1926] https://www.freebsd.org/cgi/man.cgi?query=sftp&sektion=1&format=html
[1927] https://www.freebsd.org/cgi/man.cgi?query=scp&sektion=1&format=html
[1928] https://www.freebsd.org/cgi/man.cgi?query=ftp-proxy&sektion=8&format=html

其中，`$proxy` 扩展到代理守护程序绑定到的地址。

保存 **/etc/pf.conf**，加载新规则，然后在客户端验证 FTP 连接是否正常工作：

```
# pfctl -f /etc/pf.conf
```

此示例介绍了一个基本设置，其中本地网络中的客户端需要联系其他位置的 FTP 服务器。此基本配置应该适用于 FTP 客户端和服务器的大多数组合。如 ftp-proxy（8）[1929] 所示，通过向行中添加选项 `ftp-proxy_flags=`，可以通过多种方式更改代理的行为。部分客户端或服务器可能有特定设置，必须单独为其增加特定配置。或者需要通过特定方式来集成代理，如将 FTP 流量分配到特定队列。

要运行受 PF 和 ftp-proxy（8）[1930] 保护的 FTP 服务器，请在具有自己的重定向传递规则的单独端口上使用 `-R` 以反向模式配置单独的 FTP 服务器。

33.3.2.3.管理 ICMP

用于调试或排除 TCP/IP 网络故障的许多工具都依赖于互联网控制消息协议（ICMP），该协议是专门为调试而设计的。

ICMP 协议在主机和网关之间发送和接收 控制消息，主要是为了向发送方提供有关前往目标主机途中的任何异常或困难情况的反馈。路由器使用 ICMP 协议来协商包的大小和其他传输参数，这个过程通常被称为路径 MTU 发现。

从防火墙的角度来看，某些 ICMP 控制消息容易受到已知攻击媒介的攻击。此外，让所有诊断流量无条件通过使调试更容易，但也使其他人更容易提取有关网络的信息。由于这些原因，以下规则可能不是最佳规则：

```
pass inet proto icmp from any to any
```

一种解决方案是让来自本地网络的所有 ICMP 流量通过，同时阻止来自网络外部的所有探测：

```
pass inet proto icmp from $localnet to any keep state
pass inet proto icmp from any to $ext_if keep state
```

其他选项也可用，这些选项展示了 PF 的一些灵活性。例如，可以指定 ping(8)[1931] 和 traceroute(8)[1932] 使用的消息，而不是允许所有 ICMP 消息。首先为该类型的消息定义一个宏：

```
icmp_types = "echoreq"
```

以及使用宏的规则：

[1929] https://www.freebsd.org/cgi/man.cgi?query=ftp-proxy&sektion=8&format=html
[1930] https://www.freebsd.org/cgi/man.cgi?query=ftp-proxy&sektion=8&format=html
[1931] https://www.freebsd.org/cgi/man.cgi?query=ping&sektion=8&format=html
[1932] https://www.freebsd.org/cgi/man.cgi?query=traceroute&sektion=8&format=html

```
pass inet proto icmp all icmp-type $icmp_types keep state
```

如果需要其他类型的 ICMP 数据包，可以将 `icmp_type` 扩展为这些数据包类型的列表。输入 `more /usr/src/sbin/pfctl/pfctl_parser.c` 可查看 PF 支持的 ICMP 消息类型列表。关于每种消息类型的解释，请参考 http://www.iana.org/assignments/icmp-parameters/icmp-parameters.xhtml。

由于 Unix 的 `traceroute` 默认使用 UDP，所以需要另一条规则来允许 Unix 的 `traceroute`。

```
# allow out the default range for traceroute(8):
pass out on $ext_if inet proto udp from any to any port 33433 >< 33626 keep state
```

由于 Microsoft Windows 系统上的 `TRACERT.EXE` 使用 ICMP echo 请求信息，所以只需要第一条规则就可以允许来自这些系统的网络追踪。Unix 的 `traceroute` 也可以被指示使用其他协议，如果使用 `-I`，将使用 ICMP echo 请求消息。详情请查看 traceroute(8)[1933] 手册页以了解详细信息。

33.3.2.3.1.路径 MTU 发现

互联网协议被设计为与设备无关，设备独立的一个后果是，无法始终可靠地预测给定连接的最佳数据包大小。数据包大小的主要约束是 最大传输单位（MTU），它设置了接口数据包大小的上限。键入 `ifconfig` 以查看系统网络接口的 MTU。

TCP/IP 使用称为路径 MTU 发现的过程来确定连接的正确数据包大小。此过程发送设置了 “Do not fragment” 标志集的不同大小的数据包，当达到上限时，预计 ICMP 将返回数据包 “type 3, code 4”。“type 3” 表示 “目标无法访问”，“type 4” 是 “需要分片，但设置了不分段标志” 的缩写。要允许路径 MTU 发现以支持与其他 MTU 的连接，请将类型 `destination unreachable` 添加到 `icmp_types` 宏中：

```
icmp_types = "{ echoreq, unreach }"
```

由于传递规则已使用该宏，因此无需对其进行修改即可支持新的 ICMP 类型：

```
pass inet proto icmp all icmp-type $icmp_types keep state
```

PF 允许对 ICMP 类型和代码的所有变体进行过滤。可能的类型和代码列表记录在 icmp(4)[1934] 和 icmp6(4)[1935] 中。

[1933] https://www.freebsd.org/cgi/man.cgi?query=traceroute&sektion=8&format=html
[1934] https://www.freebsd.org/cgi/man.cgi?query=icmp&sektion=4&format=html
[1935] https://www.freebsd.org/cgi/man.cgi?query=icmp6&sektion=4&format=html

33.3.2.4.使用表

某些类型的数据与给定时间的筛选和重定向相关，但它们的定义太长，无法包含在规则集文件中。PF 支持使用表，表是定义的列表，无需重新加载整个规则集即可对其进行操作，并且可以提供快速查找。表名始终括在 < > 中，如下所示：

```
table <clients> { 192.168.2.0/24, !192.168.2.5 }
```

在此示例中，网络是 192.168.2.0/24 表的一部分，但 192.168.2.5 地址除外，该地址使用 ! 运算符将其排除。还可以从每个项目位于单独行上的文件中加载表，如以下示例 **/etc/clients** 所示：

```
192.168.2.0/24
!192.168.2.5
```

要引用该文件，请按如下方式定义表：

```
table <clients> persist file "/etc/clients"
```

定义表后，可以通过规则引用它：

```
pass inet proto tcp from <clients> to any port $client_out flags S/SA keep state
```

可以使用 pfctl 实时操作表的内容。以下示例向表中添加另一个网络：

```
# pfctl -t clients -T add 192.168.1.0/16
```

注意，以这种方式做出的任何改变都会立即生效，使其成为测试的理想选择，但在断电或重启后将无法继续。要使这些变化成为永久性的，请修改规则集中的表的定义，或者编辑表所指向的文件。我们可以使用 cron(8)[1936] 工作来维护磁盘上的表的副本，该工作会定期将表的内容转储到磁盘上，使用的命令有：pfctl -t clients -T show >/etc/clients。另外，**/etc/clients** 也可以用内存中的表内容进行更新：

```
# pfctl -t clients -T replace -f /etc/clients
```

[1936] https://www.freebsd.org/cgi/man.cgi?query=cron&sektion=8&format=html

33.3.2.5.使用重载表来保护 SSH

在外部接口上运行 SSH 的用户可能在身份验证日志中看到过类似这样的内容：

```
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 41484 ssh2
```

这表明有人或某些程序试图探索用户名和密码，以让他们进入系统。

如果合法用户需要外部 SSH 访问，则更改 SSH 使用的默认端口可以提供一些保护。但是，PF 提供了一个更优雅的解决方案。Pass 规则可以包含对连接主机可以执行的操作的限制，并且可以将违规者放逐到被拒绝部分或全部访问的地址表中。甚至可以从超出限制的计算机中删除所有现有连接。

要对此进行配置，请在规则集的表部分中创建此表：

```
table <bruteforce> persist
```

然后，在规则集的早期某个位置，添加规则以阻止暴力访问，同时允许合法访问：

```
block quick from <bruteforce>
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)
```

括号中的部分定义了限制，应更改数字以满足本地要求。可以这样理解：

`max-src-conn` 是允许从一台主机同时连接的次数。

`max-src-conn-rate` 是每秒数（*5*）允许来自任何单个主机的新连接速率（*15*）。

`overload <bruteforce>` 意味着任何超过这些限制的主机都会被添加到 `bruteforce` 表中。该规则集阻止来自 `bruteforce` 表中的地址的所有流量。

最后，`flush global` 表示，当一个主机达到限制时，该主机的所有（`global`）连接将被终止（`flush`）。

> 注意
>
> 这些规则 不会阻止如 http://home.nuug.no/\~peter/hailmary2013/[1937] 中所述的速度较慢的暴力破解程序。

[1937] http://home.nuug.no/~peter/hailmary2013/

此示例规则集主要用作说明。例如，如果通常需要大量的连接，但在涉及 ssh 时希望更加严格，请在规则集的早期用如下内容补充上述规则：

```
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)
```

> 注意
>
> 可能没有必要阻止所有重载器：
>
> 值得注意的是，重载机制是一种通用技术，不仅适用于 SSH，并且完全阻止违规者的所有流量并不总是最佳的。
>
> 例如，重载规则可用于保护邮件服务或 Web 服务，重载表可用于在规则中将违规者分配到具有最小带宽分配的队列或重定向到特定网页。

随着时间的推移，表将被重载规则填充，其大小将逐渐增长，占用更多内存。有时，被阻止的 IP 地址是动态分配的 IP 地址，此后已分配给具有正当理由与本地网络中的主机通信的主机。

对于这样的情况，pfctl 提供了使表项过期的能力。例如，该命令将删除 `86400` 秒内未被引用的 `<bruteforce>` 表项。

```
# pfctl -t bruteforce -T expire 86400
```

security/expiretable[1938] 也提供了类似的功能，它删除了在指定时间段内未被访问的表条目。

安装后，可以运行 expiretable 来删除 `<bruteforce>` 表中超过指定期限的表条目。本示例删除所有早于 24 小时的条目：

```
/usr/local/sbin/expiretable -v -d -t 24h bruteforce
```

33.3.2.6.防止垃圾邮件

不要与与 spamassassin 捆绑在一起的垃圾邮件守护程序混淆，mail/spamd[1939] 可以使用 PF 进行配置，以提供针对垃圾邮件的外部防御。此垃圾邮件使用一组重定向挂接到 PF 配置中。

垃圾邮件发送者倾向于发送大量邮件，垃圾邮件主要来自少数垃圾邮件发送者友好的网络和大量被劫持的计算机，这两者都会相当快地报告到 *阻止列表*中。

当收到来自阻止列表中某个地址的 SMTP 连接时，spamd 会显示其标题，并立即切换到一种模式，在该模式下，它一次一个字节地应答 SMTP 通信。这种技术旨在将尽可能多的时间浪费在垃圾邮件发送者这一端，称为 *缓送*。使用一个字节 SMTP 答复的特定实现通常称为 *断断续续*。

[1938] https://cgit.freebsd.org/ports/tree/security/expiretable/pkg-descr

[1939] https://cgit.freebsd.org/ports/tree/mail/spamd/pkg-descr

此示例演示了使用自动更新的阻止列表设置垃圾邮件的基本过程。有关详细信息，请参阅随 mail/spamd[1940] 一起安装的手册页。

配置反垃圾邮件的过程：

1. 通过软件包或 port 安装 mail/spamd[1941]。要使用 spamd 的灰名单功能，fdescfs(5)[1942] 必须挂载在 **/dev/fd**。将以下行添加到 **/etc/fstab**：

```
fdescfs /dev/fd fdescfs rw 0 0
```

然后，挂载文件系统：

```
#  mount fdescfs
```

2. 接下来，编辑 PF 规则集以包括：

```
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
```

两个表 <spamd> 和 <spamd-white> 是必不可少的。来自 <spamd> 中所列但不在 <spamd-white> 中的地址的 SMTP 流量被重定向到在 8025 端口监听的 spamd 守护进程。

3. 下一步是在 **/usr/local/etc/spamd.conf** 中配置 spamd，并添加一些 **rc.conf** 参数。

mail/spamd[1943] 的安装包括一个示例配置文件（**/usr/local/etc/spamd.conf.sample**）和一个 **spamd.conf** 的手册页。有关此示例中所示选项之外的其他配置选项，请参阅这些选项。

配置文件中不以 # 注释符号开头的第一行包含了定义 all 列表的块，它指定了要使用的列表：

```
all:\
    :traplist:allowlist:
```

此条目添加所需的区块列表，以冒号(:)分隔。要使用白名单从黑名单中减去地址，请在该黑名单的名称之后 紧跟添加白名单的名称。例如：`:blocklist:allowlist:`

后跟指定的黑名单的定义：

```
traplist:\
    :black:\
    :msg="SPAM. Your address %A has sent spam within the last 24 hours":\
    :method=http:\
    :file=www.openbsd.org/spamd/traplist.gz
```

[1940] https://cgit.freebsd.org/ports/tree/mail/spamd/pkg-descr
[1941] https://cgit.freebsd.org/ports/tree/mail/spamd/pkg-descr
[1942] https://www.freebsd.org/cgi/man.cgi?query=fdescfs&sektion=5&format=html
[1943] https://cgit.freebsd.org/ports/tree/mail/spamd/pkg-descr

其中第一行是封杀名单的名称，第二行是指定名单类型。msg 字段包含了在 SMTP 对话中显示给被封杀的发送者的信息。method 字段指定 spamd-setup 如何获取列表数据；支持的方法有 http、ftp、挂载文件系统中的 file 和通过外部程序的 exec。最后，file 字段指定了 spamd 期望接收的文件的名称。

指定白名单的定义类似，但省略了该字段，因为不需要消息 msg：

```
allowlist:\
    :white:\
    :method=file:\
    :file=/var/mail/allowlist.txt
```

技巧

谨慎选择数据源：

使用示例 **spamd.conf** 中的所有阻止列表将阻止互联网的大块区域。管理员需要编辑该文件以创建使用适用数据源并在必要时使用自定义列表的最佳配置。

接下来，将此条目添加到 **/etc/rc.conf**。注释指定的手册页中讲解了其他标志：

```
spamd_flags="-v" # use "" and see spamd-setup(8) for flags
```

完成后，重新加载规则集，通过输入 service obspamd start 启动 spamd，并使用 spamd-setup 完成配置。最后，创建一个 cron(8)[1944] 工作，在合理的时间间隔内调用 spamd-setup 来更新这些表。

在邮件服务器前面的典型网关上，主机很快就会在几秒钟到几分钟内开始陷入困境。

PF 还支持 *灰名单*，这会暂时拒绝来自具有代码 *45n* 的未知主机的邮件。来自灰名单主机的消息在合理时间内再次尝试，将被允许通过。来自设置为在 RFC 1123 和 RFC 2821 设置的限制范围内运行的发件人的流量将立即通过。

有关灰名单技术的更多信息，请访问 greylisting.org[1945] 网站。除了简单之外，灰名单最令人惊奇的事情是它仍然有效。垃圾邮件发送者和恶意软件编写者在适应绕过这种技术方面非常缓慢。

配置灰名单的基本过程如下：

配置灰名单的过程：

1. 确保 fdescfs(5)[1946] 已按照上一过程的步骤 1 中所述进行挂载。
2. 要在灰名单模式下运行垃圾邮件，请将此行添加到 **/etc/rc.conf**：

```
spamd_grey="YES"  # use spamd greylisting if YES
```

有关其他相关参数的说明，请参阅垃圾邮件手册页。

3. 要完成灰名单设置：

[1944] https://www.freebsd.org/cgi/man.cgi?query=cron&sektion=8&format=html

[1945] http://www.greylisting.org

[1946] https://www.freebsd.org/cgi/man.cgi?query=fdescfs&sektion=5&format=html

```
# service obspamd restart
# service obspamlogd start
```

在幕后，spamdb 数据库工具和 spamlogd 白名单更新程序执行灰名单功能的基本功能。spamdb 是管理员通过 **/var/db/spamdb** 数据库的内容管理块、灰名单和允许列表的主要界面。

33.3.2.7.网络维护

本节介绍如何使用 `block-policy` 、`scrub` 和 `antispoof` 来使规则集的行为正常。

`block-policy` 选项，可以在 `options` 规则集的一部分中设置，该部分位于重定向和过滤规则之前。此选项确定 PF 向被规则阻止的主机发送哪些反馈（如果有）。该选项有两个可能的值：`drop` 没有反馈的被阻止数据包，并 `return` 返回状态代码，如 `Connection refused`。

如果未设置 `drop`，则缺省策略为 `block-policy`，要更改，请指定所需的值：

```
set block-policy return
```

在 PF 中，`scrub` 是启用网络数据包规范化的关键字。此过程将重新组合碎片数据包并丢弃具有无效标志组合的 TCP 数据包。启用 `scrub` 功能提供了一种针对基于不正确处理数据包片段的某些类型攻击的保护措施。有许多选项可用，但最简单的形式适用于大多数配置：

```
scrub in all
```

某些服务（如 NFS）需要特定的片段处理选项。有关详细信息，请参阅 https://home.nuug.no/~peter/pf/en/scrub.html。

本示例重新组合片段，清除“do not fragment”位，并将最大段大小设置为 1440 字节：

```
scrub in all fragment reassemble no-df max-mss 1440
```

`antispoof` 机制可防止来自欺骗或伪造 IP 地址的活动，主要是通过阻止出现在接口上和逻辑上不可能的方向上的数据包。

这些规则清除了来自世界其他地区的欺骗性流量以及源自本地网络的任何欺骗性数据包：

```
antispoof for $ext_if
antispoof for $int_if
```

33.3.2.8.处理不可路由的地址

即使使用正确配置的网关来处理网络地址转换，也可能必须补偿其他人的错误配置。一个常见的错误配置是让具有不可路由地址的流量流向互联网。由于来自不可路由地址的流量可以在多种 DoS 攻击技术中发挥作用，因此请考虑显式阻止来自不可路由地址的流量通过外部接口进入网络。

在此示例中，定义了一个包含不可路由地址的宏，然后将其用于阻止规则。进出这些地址的流量会悄悄地丢弃在网关的外部接口上。

```
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
```

33.3.3.启用 ALTQ

在 FreeBSD 上，ALTQ 可以与 PF 一起使用，以提供服务质量（QOS）。启用 ALTQ 后，可以在规则集中定义队列，以确定出站数据包的处理优先级。

在启用 ALTQ 之前，请参考 altq（4）[1947] 以确定系统上安装的网卡的驱动程序是否支持它。

ALTQ 不能用作可加载的内核模块。如果系统的接口支持 ALTQ，那么使用 配置 FreeBSD[1948] 内核中的说明创建一个定制内核。以下内核选项可用。需要第一个来启用 ALTQ。至少需要一个其他选项来指定排队调度程序算法：

```
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)
```

以下调度程序算法可用：

- **CBQ**

  基于类的队列（CBQ）用于将连接的带宽划分为不同的类或队列，以根据过滤规则确定流量的优先级。

- **RED**

  随机早期检测（RED）用于通过测量队列的长度并将其与队列的最小和最大阈值进行比较来避免网络拥塞。当队列超过最大值时，将随机丢弃所有新数据包。

[1947] https://www.freebsd.org/cgi/man.cgi?query=altq&sektion=4&format=html

[1948] https://docs.freebsd.org/en/books/handbook/kernelconfig/index.html#kernelconfig

- **RIO**

  在随机早期检测入库和出库（RIO）模式下，RED 保持多个平均队列长度和多个阈值，每个 QOS 级别一个阈值。

- **HFSC**

  分层公平服务曲线数据包调度程序（HFSC）在 http://www-2.cs.cmu.edu/\~hzhang/HFSC/main.html[1949] 中进行了说明。

- **PRIQ**

  优先级队列（PRIQ）始终首先传递较高队列中的流量。

有关调度算法和示例规则集的更多信息，请访问 OpenBSD 的 Web 存档[1950]。

33.4. IPFW

FreeBSD 在 **/etc/rc.firewall** 中提供了一个规则集示例，它为常见场景定义了几种防火墙类型，以帮助新手用户生成适当的规则集。IPFW 提供了一种强大的语法，专业用户可以使用该语法来制作满足给定环境安全要求的自定义规则集。

本节介绍如何启用 IPFW，概述其规则语法，并演示常见配置方案的多个规则集。

33.4.1.启用 IPFW

IPFW 作为内核可加载模块包含在基本的 FreeBSD 安装中，这意味着不需要定制内核来启用 IPFW。

对于希望将 IPFW 支持静态编译为定制内核的用户，请参阅 IPFW 内核选项[1951]。

要将系统配置为在引导时启用 IPFW，请添加 `firewall_enable="YES"` 到 **/etc/rc.conf**：

```
# sysrc firewall_enable="YES"
```

要使用 FreeBSD 提供的默认防火墙类型之一，请添加另一行来指定类型：

```
# sysrc firewall_type="open"
```

可用类型包括：

- `open`：传递所有流量。
- `client`：仅保护此计算机。
- `simple`：保护整个网络。

[1949] http://www-2.cs.cmu.edu/~hzhang/HFSC/main.html

[1950] https://web.archive.org/web/20151109213426/http://www.openbsd.org/faq/pf/queueing.html

[1951] https://docs.freebsd.org/en/books/handbook/firewalls/#firewalls-ipfw-kernelconfig

- `closed`：完全禁用除环回接口之外的 IP 流量。
- `workstation`：仅使用有状态规则保护此计算机。
- `UNKNOWN`：禁用加载防火墙规则。
- **filename**：包含防火墙规则集的文件的完整路径。

如果 `firewall_type` 设置为 `client` 或 `simple`，请修改 **/etc/rc.firewall** 中找到的默认规则以适合系统的配置。

请注意，该 `filename` 类型用于加载自定义规则集。

另一种加载自定义规则集的方法是将 `firewall_script` 变量设置为包含 IPFW 命令的可执行脚本的绝对路径。本节中使用的例子假设 `firewall_script` 被设置为 **/etc/ipfw.rules**：

```
# sysrc firewall_script="/etc/ipfw.rules"
```

要通过 syslogd(8)[1952] 启用日志记录，请加入以下行：

```
# sysrc firewall_logging="YES"
```

> 警告
> 只有带有 `log` 选项的防火墙规则才会被记录下来。默认规则不包括这个选项，必须手动添加。因此，建议对默认规则集进行编辑以获得日志。此外，如果日志被存储在一个单独的文件中，可能需要进行日志轮换。

没有 **/etc/rc.conf** 变量来设置日志记录限制。要限制每次连接尝试记录规则的次数，请在 **/etc/sysctl.conf** 中使用以下行指定次数：

```
# echo "net.inet.ip.fw.verbose_limit=5" >> /etc/sysctl.conf
```

要通过专用接口 `ipfw0` 启用日志功能，请将这一行添加到 **/etc/rc.conf**：

```
# sysrc firewall_logif="YES"
```

然后使用 tcpdump 查看正在记录的内容：

```
# tcpdump -t -n -i ipfw0
```

> 技巧
> 除非附加了 tcpdump，否则不会因日志记录而产生开销。

保存所需的编辑后，启动防火墙。要立即启用日志记录限制，还要设置上面指定的 `sysctl` 值：

[1952] https://www.freebsd.org/cgi/man.cgi?query=syslogd&sektion=8&format=html

```
# service ipfw start
# sysctl net.inet.ip.fw.verbose_limit=5
```

33.4.2. IPFW 规则的语法

当数据包进入 IPFW 防火墙时，会将其与规则集中的第一个规则进行比较，并一次推进一个规则，按顺序从上到下移动。当数据包与规则的选择参数匹配时，将执行规则的操作，并终止对规则集的搜索以查找该数据包。这被称为 “首次匹配成功”。如果数据包与任何规则都不匹配，则会被强制 IPFW 默认规则编号 65535 捕获，该规则会拒绝所有数据包并以静默方式丢弃它们。但是，如果数据包与包含 `count`、`skipto` 或 `tee` 关键字的规则匹配，则搜索将继续。有关这些关键字如何影响规则处理的详细信息，请参阅 ipfw(8)[1953]。

创建 IPFW 规则时，必须按以下顺序编写关键字。某些关键字是必需的，而其他关键字是可选的。以大写形式显示的单词表示变量，以小写形式显示的单词必须位于其后面的变量之前。`#` 该符号用于标记注释的开头，可能出现在规则的末尾或其自己的行上。空行将被忽略。

``*CMD RULE_NUMBER set SET_NUMBER ACTION log LOG_AMOUNT PROTO from SRC SRC_PORT to DST DST_PORT OPTIONS*``

本节概述了这些关键字及其选项。它不是每个可能选项的详尽列表。有关创建 IPFW 规则时可以使用的规则语法的完整说明，请参阅 ipfw(8)[1954]。

- **CMD**

  每个规则都必须以 `ipfw add` 开头。

- **RULE_NUMBER**

  每个规则都与一个从 `1` 到 `65534` 的数字相关联。该数字用于指示规则处理的顺序。多个规则可以具有相同的编号，在这种情况下，将根据添加规则的顺序应用这些规则。

- **SET_NUMBER**

  每个规则都与从 `0` 到 `31` 的设置编号相关联。可以单独禁用或启用集，从而可以快速添加或删除一组规则。如果未指定 SET_NUMBER，则该规则将被添加并设置为 `0`。

- **ACTION**

  一个规则可以与下列行动之一相关。当数据包符合规则的选择标准时，将执行指定的动作。

  `allow | accept | pass | permit`: 这些关键词是等价的，允许符合规则的数据包。

  `check-state`: 根据动态状态表检查该数据包。如果发现匹配，则执行与产生该动态规则的规则相关的操作，否则转到下一条规则。一个检查状态的规则没有选择标准。如果规则集中没有检查状态规则，则在第一个保持状态或限制规则处检查动态规则表。

  `count`: 更新所有符合该规则的数据包的计数器。搜索继续到下一条规则。

[1953] https://www.freebsd.org/cgi/man.cgi?query=ipfw&sektion=8&format=html
[1954] https://www.freebsd.org/cgi/man.cgi?query=ipfw&sektion=8&format=html

`deny | drop`：任何一个词都会默默地丢弃符合该规则的数据包。

其他操作可用。有关详细信息，请参阅 ipfw(8)[1955]。

- **LOG_AMOUNT**

  当一个数据包与带有 `log` 关键字的规则相匹配时，一条信息将被记录到 syslogd(8)[1956]，其设施名称为 SECURITY。只有当该特定规则所记录的数据包数量不超过指定的 LOG_AMOUNT 时才会发生记录。如果没有指定 LOG_AMOUNT，则限制取自 `net.inet.ip.fw.verbose_limit` 的值。一个零的值会移除日志记录的限制。一旦达到限制，可以通过使用 `ipfw resetlog` 来清除该规则的日志计数器或数据包计数器来重新启用日志记录

  注意

  更多操作日志记录是在满足所有其他数据包匹配条件之后，以及在对数据包执行最终操作之前完成的。管理员决定启用登录的规则。

- **PROTO**

  此可选值可用于指定在 **/etc/protocol** 中找到的任何协议名称或编号。

- **SRC**

  `from` 关键字后面必须有源地址或代表源地址的关键字。地址可以用 `any`、`me`（本系统接口上配置的任何地址）、`me6`（本系统接口上配置的任何 IPv6 地址）或 `table` 来表示，后面是一个包含地址列表的查询表的编号。当指定一个 IP 地址时，可以选择在其后面加上 CIDR 掩码或子网掩码。例如，`1.2.3.4/25` 或 `1.2.3.4: 255.255.255.128`。

- **SRC_PORT**

  可以使用 **/etc/services** 中的端口号或名称指定可选的源端口。

- **DST**

  `to` 关键字后必须跟目标地址或表示目标地址的关键字。SRC 部分中描述的相同关键字和地址可用于描述目标。

- **DST_PORT**

  可以使用 **/etc/services** 中的端口号或名称指定可选的目标端口。

- **OPTIONS**

  有几个关键词可以跟在源和目的地后面。顾名思义，OPTIONS 是可选的。常用的选项包括 `in` 或 `out`，它们指定数据包流动的方向，`icmp types` 后面是 ICMP 消息的类型，以及 `keep-state`。

  当保持状态规则被匹配时，防火墙将创建一个动态规则，该规则匹配使用相同协议的源和目标地址和端口之间的双向流量。

  动态规则设施很容易受到 SYN 洪水攻击的资源消耗，这种攻击会打开大量的动态规则。为了对付 IPFW 的这种类型的攻击，请使用限制。这个选项通过检查打开的动态规则，计算这个规则和 IP 地址组合出现的次数来限制同时进行的会话数量。如果这个计数大于 `limit` 所指定的值，数据包就会被丢弃。

[1955] https://www.freebsd.org/cgi/man.cgi?query=ipfw&sektion=8&format=html

[1956] https://www.freebsd.org/cgi/man.cgi?query=syslogd&sektion=8&format=html

有几十个 OPTIONS 可用。参考 ipfw(8)[1957] 以了解每个可用选项的介绍。

33.4.3.规则集示例

本节演示了如何创建名为 **/etc/ipfw.rules** 的示例有状态防火墙规则集脚本。在此示例中，所有连接规则都使用 `in` 或 `out` 来阐明方向。它们还使用 `via` 接口名称来指定数据包所经过的接口。

> 注意
> 首次创建或测试防火墙规则集时，请考虑暂时设置以下可调参数：
>
> ```
> net.inet.ip.fw.default_to_accept="1"
> ```
>
> 这会将默认策略 ipfw(8)[1958] 设置为比默认策略 `deny ip from any to any` 更宽松，使得在重新启动后立即锁定系统变得更加困难。

防火墙脚本一开始就表明它是一个 Bourne shell 脚本，并刷新了任何现有的规则。然后，它创建了 `cmd` 变量，这样就不必在每个规则的开头输入 `ipfw add`。它还定义了 `pif` 变量，代表连接到互联网的接口名称。

```
#!/bin/sh
# Flush out the list before we begin.
ipfw -q -f flush

# Set rules command prefix
cmd="ipfw -q add"
pif="dc0"     # interface name of NIC attached to Internet
```

前两个规则允许受信任的内部接口和环回接口上的所有流量：

```
# Change xl0 to 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
```

如果数据包与动态规则表中的现有条目匹配，则下一个规则允许数据包通过：

```
$cmd 00101 check-state
```

下一组规则定义了内部系统可以创建与互联网上的主机相关的有状态连接：

[1957] https://www.freebsd.org/cgi/man.cgi?query=ipfw&sektion=8&format=html
[1958] https://www.freebsd.org/cgi/man.cgi?query=ipfw&sektion=8&format=html

```
# Allow access to public DNS
# Replace x.x.x.x with the IP address of a public DNS server
# and repeat for each DNS server in /etc/resolv.conf
$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 access to ISP's DHCP server for cable/DSL configurations.
# Use the first rule and check log for IP address.
# Then, uncomment the second rule, input the IP address, and delete the 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 outbound HTTP and HTTPS connections
$cmd 00200 allow tcp from any to any 80 out via $pif setup keep-state
$cmd 00220 allow tcp from any to any 443 out via $pif setup keep-state

# Allow outbound email connections
$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 outbound ping
$cmd 00250 allow icmp from any to any out via $pif keep-state

# Allow outbound NTP
$cmd 00260 allow udp from any to any 123 out via $pif keep-state

# Allow outbound SSH
$cmd 00280 allow tcp from any to any 22 out via $pif setup keep-state

# deny and log all other outbound connections
$cmd 00299 deny log all from any to any out via $pif
```

下一组规则控制从互联网主机到内部网络的连接。它首先拒绝通常与攻击关联的数据包，然后显式允许特定类型的连接。所有源自互联网的授权服务都使用 `limit` 以防止泛滥。

```
# 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
```

(continues on next page)

(continued from previous page)

```
$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 services.
$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 fragments
$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 from ISP's DHCP server.
# Replace x.x.x.x with the same IP address used in rule 00120.
#$cmd 00360 allow udp from any to x.x.x.x 67 in via $pif keep-state

# Allow HTTP connections to internal web server
$cmd 00400 allow tcp from any to me 80 in via $pif setup limit src-addr 2

# Allow inbound SSH connections
$cmd 00410 allow tcp from any to me 22 in via $pif setup limit src-addr 2

# Reject and log all other incoming connections
$cmd 00499 deny log all from any to any in via $pif
```

最后一个规则记录与规则集中的任何规则都不匹配的所有数据包：

```
# Everything else is denied and logged
$cmd 00999 deny log all from any to any
```

33.4.4.内核内部 NAT

FreeBSD 的 IPFW 防火墙有两个 NAT 实现：用户空间实现 natd(8)[1959] 和最近的内核内部 NAT 实现。两者都与 IPFW 合作提供网络地址转换。这可用于提供互联网连接共享解决方案，以便多台内部计算机可以使用单个公共 IP 地址连接到互联网。

要做到这一点，连接到互联网的 FreeBSD 机器必须充当网关。此系统必须有两个 NIC，其中一个连接到互联网，另一个连接到内部 LAN。连接到 LAN 的每台计算机都应在专用网络空间中分配一个 IP 地址，如 RFC 1918[1960] 所定义。

需要一些额外的配置才能启用 IPFW 的内核内部 NAT 功能。要在引导时启用内核内 NAT 支持，必须在 **/etc/rc.conf** 中设置以下内容：

```
gateway_enable="YES"
firewall_enable="YES"
firewall_nat_enable="YES"
```

> 注意
>
> 当设置 `firewall_nat_enable` 但未设置 `firewall_enable` 时，它将没有效果，也不执行任何操作。这是因为内核内部的 NAT 实现仅与 IPFW 兼容。

当规则集包含有状态的规则时，NAT 规则的定位很关键，因此要使用 `skipto` 动作。`skipto` 动作需要一个规则号，这样它就知道要跳到哪条规则。下面的例子建立在上一节所示的防火墙规则集的基础上。它增加了一些额外的条目，并修改了一些现有的规则，以便为内核内 NAT 配置防火墙。它首先添加了一些额外的变量，这些变量代表要跳到的规则编号、保持状态选项和一个 TCP 端口列表，这些端口将被用来减少规则的数量。

```
#!/bin/sh
ipfw -q -f flush
cmd="ipfw -q add"
skip="skipto 1000"
pif=dc0
ks="keep-state"
good_tcpo="22,25,37,53,80,443,110"
```

对于内核内部 NAT，由于 libalias（3）的体系结构，必须禁用 TCP 分段卸载（TSO）,libalias[1961]（一个作为内核模块实现的库，以提供 IPFW 的内核内 NAT 工具）。TSO 可以使用 ifconfig(8)[1962] 在每个网络接口的基础上禁用，也可以使用 sysctl(8)[1963] 在系统范围内禁用。要禁用 TSO 系统范围，必须将其设置为 **/etc/sysctl.conf**：

[1959] https://www.freebsd.org/cgi/man.cgi?query=natd&sektion=8&format=html
[1960] https://www.ietf.org/rfc/rfc1918.txt
[1961] https://www.freebsd.org/cgi/man.cgi?query=libalias&sektion=3&format=html
[1962] https://www.freebsd.org/cgi/man.cgi?query=ifconfig&sektion=8&format=html
[1963] https://www.freebsd.org/cgi/man.cgi?query=sysctl&sektion=8&format=html

```
net.inet.tcp.tso="0"
```

一个 NAT 实例也将被配置。有可能有多个 NAT 实例，每个都有自己的配置。在这个例子中，只需要一个 NAT 实例，即 1 号 NAT 实例。配置可以有几个选项，例如：`if` 表示公共接口，`same_ports` 表示所有端口和本地端口号的映射是相同的，`unreg_only` 将导致只有未注册的（私有）地址空间被 NAT 实例处理，`reset` 将帮助保持一个正常的 NAT 实例，即使 IPFW 机器的公共 IP 地址发生变化。对于所有可以传递给单个 NAT 实例配置的可能选项，请参考 ipfw(8)[1964]。当配置一个有状态的 NAT 防火墙时，有必要允许翻译后的数据包在防火墙中被重新注入以进一步处理。这可以通过在防火墙脚本的开始处禁用 `one_pass` 行为来实现。

```
ipfw disable one_pass
ipfw -q nat 1 config if $pif same_ports unreg_only reset
```

入站 NAT 规则被插入到允许信任接口和环回接口上所有流量的两条规则之后，以及重新组合规则之后，但在检查状态规则之前。重要的是，为这个 NAT 规则选择的规则号，在这个例子中是 100，比前三个规则高，比检查状态规则低。此外，由于内核 NAT 的行为，建议在第一条 NAT 规则之前和允许在受信任接口上通信的规则之后放置一个重新组合规则。通常情况下，IP 碎片不应该发生，但是当处理 IPSEC/ESP/GRE 隧道流量时，它可能会发生，在把完整的数据包交给内核内 NAT 设施之前，有必要对碎片进行重新组合。

> 注意
>
> 用户空间 natd(8)[1965] 无需重新组装规则，因为 IPFW 操作 `divert` 的内部工作已经在传递到套接字之前负责重新组装数据包，如 ipfw(8)[1966] 中所述。
>
> 此示例中使用的 NAT 实例和规则编号与 **rc.firewall** 创建的默认 NAT 实例和规则编号不匹配。**rc.firewall** 是一个脚本，用于设置 FreeBSD 中存在的默认防火墙规则。

```
$cmd 005 allow all from any to any via xl0  # exclude LAN traffic
$cmd 010 allow all from any to any via lo0  # exclude loopback traffic
$cmd 099 reass all from any to any in       # reassemble inbound packets
$cmd 100 nat 1 ip from any to any in via $pif # NAT any inbound packets
# Allow the packet through if it has an existing entry in the dynamic rules table
$cmd 101 check-state
```

出站规则被修改为用 `$skip` 变量替换允许动作，表明规则处理将在规则 1000 处继续进行。由于 `$good_tcpo` 变量包含了七个允许的出站端口，七个 `tcp` 规则已经被规则 125 所取代。

> 注意
>
> 请记住，IPFW 的性能很大程度上取决于规则集中存在的规则数量。

[1964] https://www.freebsd.org/cgi/man.cgi?query=ipfw&sektion=8&format=html
[1965] https://www.freebsd.org/cgi/man.cgi?query=natd&sektion=8&format=html
[1966] https://www.freebsd.org/cgi/man.cgi?query=ipfw&sektion=8&format=html

```
# Authorized outbound packets
$cmd 120 $skip udp from any to x.x.x.x 53 out via $pif $ks
$cmd 121 $skip udp from any to x.x.x.x 67 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
```

入站规则保持不变，除了最后一条规则删除了 `via $pif`，以便同时捕捉入站和出站规则。NAT 规则必须在最后一条出站规则之后，必须有一个比最后一条规则更高的编号，而且规则编号必须被 `skipto` 动作所引用。在这个规则集中，规则号 1000 处理将所有数据包传递给我们配置的实例进行 NAT 处理。下一条规则允许任何经过 NAT 处理的数据包通过。

```
$cmd 999 deny log all from any to any
$cmd 1000 nat 1 ip from any to any out via $pif # skipto location for outbound␣
↪stateful rules
$cmd 1001 allow ip from any to any
```

在这个例子中，规则 `100`、`101`、`125`、`1000` 和 `1001` 控制出站和入站数据包的地址转换，以便动态状态表中的条目总是注册私人 LANIP 地址。

考虑一个内部网络浏览器，它通过 80 端口初始化一个新的出站 HTTP 会话。当第一个出站数据包进入防火墙时，它不符合规则 100，因为它是向外而不是向内。它通过了规则 101，因为这是第一个数据包，它还没有被发布到动态状态表。该数据包最终与规则 125 相匹配，因为它是在一个允许的端口上向外发送的，并且有一个来自内部局域网的源 IP 地址。在匹配这个规则时，发生了两个动作。首先，保持状态的动作为动态状态表添加了一个条目，并执行指定的动作，即跳到规则 1000。接下来，数据包经过 NAT，被发送到互联网上。这个数据包到达目的地网络服务器，在那里生成并发回一个响应数据包。这个新的数据包进入了规则集的顶部。它与规则 100 相匹配，并将其目标 IP 地址映射回原来的内部地址。然后，它被检查状态规则处理，在表中被发现是一个现有的会话，并被释放到 LAN。

在入站方面，规则集必须拒绝坏数据包，只允许授权的服务。匹配入站规则的数据包被发布到动态状态表，数据包被释放到 LAN。作为响应产生的数据包被检查状态规则识别为属于一个现有会话。然后，它被发送到规则 1000，在被释放到出站接口之前经过 NAT。

> 注意
>
> 从用户空间 natd(8)[1967] 到内核内部 NAT 的转换起初可能看起来是无缝的，但有一个小问题。使用 GENERIC 内核时，IPFW 将加载内核模块 **libalias.ko**，当 **/etc/rc.conf** 中启用 `firewall_nat_enable` 时。**libalias.ko** 内核模块只提供了基本的 NAT 功能，而用户区间实现的 natd(8)[1968] 在用户区间的库中提供了所有的 NAT 功能，而不需要任何额外的配置。所有的功能都是指以下内核模块，除了标准的 **libalias.ko** 内核模块外，在需要时还可以使用 **/etc/rc.conf** 中的 `kld_list` 指令额外加载：**alias_ftp.ko**、**alias_bbt.ko**、**skinny.ko**、**irc.ko**、**alias_pptp.ko** 和 **alias_smedia.ko**。如果使用定制内核，用户空间库的全部功能都可以在内核中使用 `options LIBALIAS` 编译。

[1967] https://www.freebsd.org/cgi/man.cgi?query=natd&sektion=8&format=html

[1968] https://www.freebsd.org/cgi/man.cgi?query=natd&sektion=8&format=html

33.4.4.1.端口重定向

NAT 的缺点通常是无法从互联网访问 LAN 客户端。LAN 上的客户端可以与世界建立传出连接，但不能接收传入连接。如果尝试在其中一台 LAN 客户端计算机上运行互联网服务，则会出现问题。解决此问题的一种简单方法是将 NAT 提供计算机上的选定互联网端口重定向到 LAN 客户端。

例如，IRC 服务器在客户端 `A` 上运行，Web 服务器在客户端 `B` 上运行。要使其正常工作，必须在端口 6667（IRC）和 80（HTTP）上接收到的连接重定向到相应的计算机。

使用内核内部 NAT 时，所有配置都在 NAT 实例配置中完成。有关内核内 NAT 实例可以使用的选项的完整列表，请参阅 ipfw(8)[1969]。IPFW 语法遵循 natd 的语法。`redirect_port` 的语法如下：

```
redirect_port proto targetIP:targetPORT[-targetPORT]
  [aliasIP:]aliasPORT[-aliasPORT]
  [remoteIP[:remotePORT[-remotePORT]]]
```

要配置上述示例设置，参数应为：

```
redirect_port tcp 192.168.0.2:6667 6667
redirect_port tcp 192.168.0.3:80 80
```

将这些参数添加到上述规则集中的 NAT 实例 1 配置后，TCP 端口将被转发到运行 IRC 和 HTTP 服务的 LAN 客户端计算机。

```
ipfw -q nat 1 config if $pif same_ports unreg_only reset \
  redirect_port tcp 192.168.0.2:6667 6667 \
  redirect_port tcp 192.168.0.3:80 80
```

单个端口上的端口范围可以用 `redirect_port` 指示。例如，*tcp 192.168.0.2：2000-3000 2000-3000* 会将端口 2000 到 3000 上收到的所有连接重定向到客户端 `A` 上的端口 2000 到 3000。

33.4.4.2.地址重定向

如果有一个以上的 IP 地址，地址重定向就很有用。每个局域网客户可以通过 ipfw(8)[1970] 分配自己的外部 IP 地址，然后用适当的外部 IP 地址重写从局域网客户发出的数据包，并将该特定 IP 地址上传入的所有流量重定向到特定的局域网客户。这也被称为静态 NAT。例如，如果有`128.1.1.1`、`128.1.1.2` 和 `128.1.1.3` 三个 IP 地址，`128.1.1.1`可以作为 ipfw(8)[1971] 机器的外部 IP 地址，而`128.1.1.2` 和 `128.1.1.3` 则被转发回 LAN 客户端 `A` 和 `B`。

`redirect_addr` 的语法如下，其中 `local IP` 是 LAN 客户端的内部 IP 地址，`public IP` 是 LAN 客户端对应的外部 IP 地址。

[1969] https://www.freebsd.org/cgi/man.cgi?query=ipfw&sektion=8&format=html
[1970] https://www.freebsd.org/cgi/man.cgi?query=ipfw&sektion=8&format=html
[1971] https://www.freebsd.org/cgi/man.cgi?query=ipfw&sektion=8&format=html

```
redirect_addr localIP publicIP
```

在示例中，参数将如下所示：

```
redirect_addr 192.168.0.2 128.1.1.2
redirect_addr 192.168.0.3 128.1.1.3
```

例如 `redirect_port`，这些参数放置在 NAT 实例配置中。使用地址重定向，不需要端口重定向，因为在特定 IP 地址上接收的所有数据都会被重定向。

ipfw(8)[1972] 计算机上的外部 IP 地址必须处于活动状态，并且与外部接口存在别名。有关详细信息，请参阅 rc.conf(5)[1973]。

33.4.4.3.用户空间 NAT

让我们从一个语句开始：用户空间 NAT 实现：natd(8)[1974]，比内核内部 NAT 具有更多的开销。对于 natd(8)[1975] 要转换数据包，数据包必须从内核复制到用户空间，然后再复制回用户空间，这带来了内核内部 NAT 所没有的额外开销。

要在引导时启用用户空间的 NAT 守护程序 natd(8)[1976]，以下是 **/etc/rc.conf** 中的最低配置。其中 `natd_interface` 设置为连接到互联网的网卡的名称。natd(8)[1977] 的 rc(8)[1978] 脚本将自动检查是否使用了动态 IP 地址，并配置自己来处理它。

```
gateway_enable="YES"
natd_enable="YES"
natd_interface="rl0"
```

通常，上述内核内部 NAT 所解释的规则集也可以与 natd(8)[1979] 一起使用。例外情况是内核内 NAT 实例 (`ipfw -q nat 1 config …`) 的配置，该配置不需要与重组规则 99 一起使用，因为其功能包含在操作中。`divert` 规则编号 100 和 1000 必须按顺序更改，如下所示。

```
$cmd 100 divert natd ip from any to any in via $pif
$cmd 1000 divert natd ip from any to any out via $pif
```

要配置端口或地址重定向，请使用与内核内 NAT 类似的语法。虽然现在，最好在配置文件中完成

[1972] https://www.freebsd.org/cgi/man.cgi?query=ipfw&sektion=8&format=html
[1973] https://www.freebsd.org/cgi/man.cgi?query=rc.conf&sektion=5&format=html
[1974] https://www.freebsd.org/cgi/man.cgi?query=natd&sektion=8&format=html
[1975] https://www.freebsd.org/cgi/man.cgi?query=natd&sektion=8&format=html
[1976] https://www.freebsd.org/cgi/man.cgi?query=natd&sektion=8&format=html
[1977] https://www.freebsd.org/cgi/man.cgi?query=natd&sektion=8&format=html
[1978] https://www.freebsd.org/cgi/man.cgi?query=rc&sektion=8&format=html
[1979] https://www.freebsd.org/cgi/man.cgi?query=natd&sektion=8&format=html

natd(8)[1980] 的配置，而不是像内核内 NAT 那样在我们的规则集脚本中指定配置。为此，必须通过 **/etc/rc.conf** 传递一个额外的标志，该参数指定配置文件的路径。

```
natd_flags="-f /etc/natd.conf"
```

> 注意
>
> 指定的文件必须包含配置选项列表，每行一个。有关配置文件和可能的变量的更多信息，请参考 natd(8)[1981]。下面是两个示例条目，每行一个：
>
> ```
> redirect_port tcp 192.168.0.2:6667 6667
> redirect_addr 192.168.0.3 128.1.1.3
> ```

33.4.5 IPFW 命令

`ipfw` 可用于在活动防火墙运行时手动添加或删除单个规则。使用此方法的问题在于，当系统重新启动时，所有更改都将丢失。建议改为将所有规则写入文件，并使用该文件在启动时加载规则，并在该文件更改时替换当前正在运行的防火墙规则。

`ipfw` 是向控制台屏幕显示正在运行的防火墙规则的有用方法。IPFW 记帐工具为每个规则动态创建一个计数器，该计数器对与规则匹配的每个数据包进行计数。在测试规则的过程中，列出规则及其计数器是确定规则是否按预期运行的一种方法。

按顺序列出所有正在运行的规则：

```
# ipfw list
```

列出所有正在运行的规则，并带有上次匹配规则的时间戳：

```
# ipfw -t list
```

下一个示例列出了匹配规则的记帐信息和数据包计数以及规则本身。第一列是规则编号，后跟匹配的数据包和字节数，后跟规则本身。

```
# ipfw -a list
```

除静态规则外，还要列出动态规则：

```
# ipfw -d list
```

要同时显示过期的动态规则，请执行以下操作：

[1980] https://www.freebsd.org/cgi/man.cgi?query=natd&sektion=8&format=html
[1981] https://www.freebsd.org/cgi/man.cgi?query=natd&sektion=8&format=html

```
# ipfw -d -e list
```

要将计数器清零：

```
# ipfw zero
```

要仅将数字为 *NUM* 的规则的计数器清零：

```
# ipfw zero NUM
```

33.4.5.1.记录防火墙消息

即使启用了日志设施，IPFW 也不会自行生成任何规则日志。由防火墙管理员决定规则集中的哪些规则将被记录，并为这些规则添加日志关键字。通常情况下，只有拒绝规则被记录。习惯上，重复“ipfw default deny everything”规则，并将 `log` 关键字作为规则集的最后一条规则。这样，就有可能看到所有不符合规则集中任何规则的数据包。

记录是一把双刃剑。如果不小心，过量的日志数据或 DoS 攻击会使磁盘上充满日志文件。日志信息不仅被写入 `syslogd`，而且还被显示在根控制台屏幕上，很快就会变得很烦人。

内核选项 `IPFIREWALL_VERBOSE_LIMIT=5` 限制了发送到 syslogd(8)[1982] 的连续信息的数量，这些信息是关于给定规则的数据包匹配。当这个选项在内核中被启用时，关于一个特定规则的连续消息的数量被限制在指定的数量。200 条相同的日志信息是没有任何好处的。当这个选项设置为 5 时，关于某个特定规则的 5 条连续信息将被记录到 syslogd，其余的相同的连续信息将被计算并发布到 syslogd，其语句如下。

```
last message repeated 45 times
```

默认情况下，所有记录的数据包消息都写入 **/var/log/security**，这在 **/etc/syslog.conf** 中定义。

33.4.5.2.构建规则脚本

大多数有经验的 IPFW 用户创建一个包含规则的文件，并以与将它们作为脚本运行兼容的方式对其进行编码。这样做的主要好处是可以批量刷新防火墙规则，而无需重新启动系统来激活它们。此方法在测试新规则时很方便，因为该过程可以根据需要执行多次。作为脚本，符号替换可用于将常用值替换为多个规则使用。

此示例脚本与 sh(1)[1983]、csh(1)[1984] 和 tcsh(1)[1985] shell 使用的语法兼容。符号替换字段以美元符号（$）为前缀。符号字段没有 $ 前缀。用于填充符号字段的值必须括在双引号（“”）中。

启动规则文件，如下所示：

[1982] https://www.freebsd.org/cgi/man.cgi?query=syslogd&sektion=8&format=html
[1983] https://www.freebsd.org/cgi/man.cgi?query=sh&sektion=1&format=html
[1984] https://www.freebsd.org/cgi/man.cgi?query=csh&sektion=1&format=html
[1985] https://www.freebsd.org/cgi/man.cgi?query=tcsh&sektion=1&format=html

```
############### 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 ############
```

规则并不重要，因为此示例的重点是如何填充符号替换字段。

如果上面的示例位于 **/etc/ipfw.rules** 中，则可以通过以下命令重新加载规则：

```
# sh /etc/ipfw.rules
```

/etc/ipfw.rules 可以位于任何位置，文件可以具有任何名称。

通过手动运行这些命令也可以完成同样的事情：

```
# 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
```

33.4.6. IPFW 内核选项

为了静态地将 IPFW 支持编译为定制内核，请参阅配置 FreeBSD 内核[1986] 中的说明。以下选项可用于定制内核配置文件：

```
options     IPFIREWALL            # enables IPFW
options     IPFIREWALL_VERBOSE       # enables logging for rules with log keyword to␣
```

(continues on next page)

[1986] https://docs.freebsd.org/en/books/handbook/kernelconfig/index.html#kernelconfig

(continued from previous page)

```
↪syslogd(8)
options    IPFIREWALL_VERBOSE_LIMIT=5   # limits number of logged packets per-entry
options    IPFIREWALL_DEFAULT_TO_ACCEPT # sets default policy to pass what is not␣
↪explicitly denied
options    IPFIREWALL_NAT       # enables basic in-kernel NAT support
options    LIBALIAS         # enables full in-kernel NAT support
options    IPFIREWALL_NAT64     # enables in-kernel NAT64 support
options    IPFIREWALL_NPTV6     # enables in-kernel IPv6 NPT support
options    IPFIREWALL_PMOD      # enables protocols modification module support
options    IPDIVERT         # enables NAT through natd(8)
```

注意

IPFW 可以作为内核模块加载：上述选项默认为模块编译，也可以在运行时使用可调参数进行设置。

33.5. IPFILTER（IPF）

IPFILTER（也称为 IPF）是一个跨平台的开源防火墙，已被移植到多个操作系统，包括 FreeBSD，NetBSD，OpenBSD 和 Solaris™。

IPFILTER 是一种内核端防火墙和 NAT 机制，可以由用户空间程序控制和监视。可以使用 ipf 设置或删除防火墙规则，可以使用 ipnat 设置或删除 NAT 规则，可以使用 ipfstat 打印 IPFILTER 内核部分的运行时统计信息，ipmon 可用于将 IPMILTER 操作记录到系统日志文件中。

IPF 最初是使用“最后一个匹配的规则成功”的规则处理逻辑编写的，并且仅使用无状态规则。自那时以来，IPF 得到了加强，以包括 `quick` 和 `keep state` 选项。

IPF 常见问题解答位于 http://www.phildev.net/ipf/index.html。IPFilter 邮件列表的可搜索存档位于 http://marc.info/?l=ipfilter。

手册的这一部分重点介绍 IPF，因为它与 FreeBSD 有关。它提供了包含 `quick` 和 `keep state` 选项的规则示例。

33.5.1.启用 IPF

IPF 作为内核可加载模块包含在 FreeBSD 基本系统中，这意味着不需要通过定制内核来启用 IPF。

对于喜欢将 IPF 支持静态编译为定制内核的用户，请参阅配置 FreeBSD 内核[1987] 中的说明。以下内核选项可用：

[1987] https://docs.freebsd.org/en/books/handbook/kernelconfig/index.html#kernelconfig

```
options IPFILTER
options IPFILTER_LOG
options IPFILTER_LOOKUP
options IPFILTER_DEFAULT_BLOCK
```

其中 `options IPFILTER` 启用对 IPFILTER 的支持，`options IPFILTER_LOG` 启用 IPF 日志，对每个有 `log` 关键字的规则使用 **ipl** 包日志伪设备，`IPFILTER_LOOKUP` 启用 IP 池，以加快 IP 查找，`options IPFILTER_DEFAULT_BLOCK` 改变默认行为，使任何不匹配防火墙 `pass` 规则的数据包被阻止。

要配置系统在启动时启用 IPF，请在 **/etc/rc.conf** 中添加以下条目。这些条目也将启用日志和 `default pass all`。要想在不编译定制内核的情况下将默认策略改为 `block all`，记得在规则集的末尾添加一个 `block all` 规则。

```
ipfilter_enable="YES"             # Start ipf firewall
ipfilter_rules="/etc/ipf.rules"   # loads rules definition text file
ipv6_ipfilter_rules="/etc/ipf6.rules" # loads rules definition text file for IPv6
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
```

如果需要 NAT 功能，还要添加以下行：

```
gateway_enable="YES"              # Enable as LAN gateway
ipnat_enable="YES"                # Start ipnat function
ipnat_rules="/etc/ipnat.rules"    # rules definition file for ipnat
```

然后，立即开启 IPF：

```
#  service ipfilter start
```

要加载防火墙规则，请使用 `ipf` 指定规则集文件的名称。以下命令可用于替换当前正在运行的防火墙规则：

```
# ipf -Fa -f /etc/ipf.rules
```

其中 `-Fa` 刷新所有内部规则表，`-f` 指定包含要加载的规则的文件。

这提供了对自定义规则集进行更改并使用规则的新副本更新正在运行的防火墙的功能，而无需重新启动系统。此方法便于测试新规则，因为该过程可以根据需要执行多次。

有关此命令可用的其他标志的详细信息，请参阅 ipf(8)[1988]。

[1988] https://www.freebsd.org/cgi/man.cgi?query=ipf&sektion=8&format=html

33.5.2. IPF 规则语法

本节描述了用于创建有状态规则的 IPF 规则语法。在创建规则时，请记住，除非规则中出现 `quick` 关键字，否则每条规则都是按顺序读取的，最后一个匹配的规则才会被应用。这意味着，即使第一个匹配数据包的规则是 `pass`，如果后面有一个匹配的规则是 `block`，那么这个数据包将被丢弃。在 **/usr/share/examples/ipfilter** 中可以找到样本规则集。

当创建规则时，`#` 字符用于标记注释的开始，可以出现在规则的末尾，以解释该规则的功能，或出现在自己的行中。任何空行都会被忽略。

在规则中使用的关键词必须按照特定的顺序书写，从左到右。有些关键词是强制性的，而有些则是可选的。有些关键词有子选项，这些子选项本身可能是关键词，也包括更多的子选项。关键字的顺序如下，大写字母显示的字代表一个变量，小写字母显示的字必须在它后面的变量之前。

``*ACTION DIRECTION OPTIONS proto PROTO_TYPE from SRC_ADDR SRC_PORT to DST_ADDR DST_PORT TCP_FLAG|ICMP_TYPE keep state STATE*``

本节介绍其中每个关键字及其选项。它不是每个可能选项的详尽列表。有关创建 IPF 规则时可以使用的规则语法的完整介绍，以及使用每个关键字的示例，请参阅 ipf(5)[1989]。

- **ACTION**

  action 关键字指示如果数据包与该规则匹配，则如何处理该数据包。每个规则都必须有一个操作。可识别以下操作：

  `block`：丢弃数据包。

  `pass`：允许数据包。

  `log`：生成日志记录。

  `count`：计算数据包和字节数，这些数据包和字节数可以指示规则的使用频率。

  `auth`：将数据包排入队列，以供另一个程序进一步处理。

  `call`：提供对 IPF 内置函数的访问，这些函数允许执行更复杂的操作。

  `decapsulate`：删除所有标头以处理数据包的内容。

- **DIRECTION**

  接下来，每个规则必须使用以下关键字之一显式声明流量方向：

  `in`：规则应用于入站数据包。

  `out`：规则应用于出站数据包。

  `all`：规则适用于任一方向。

  如果系统有多个接口，则可以将接口与方向一起指定。一个例子是 `in on fxp0`。

- **OPTIONS**

[1989] https://www.freebsd.org/cgi/man.cgi?query=ipf&sektion=5&format=html

选项是可选的。但是，如果指定了多个选项，则必须按此处显示的顺序使用它们。

`log`：执行指定的 ACTION 时，数据包标头的内容将被写入 ipl(4)[1990] 数据包日志伪设备。

`quick`：如果数据包与此规则匹配，则会发生规则指定的 ACTION，并且不会对此数据包进行任何后续规则的进一步处理。

`on`：后跟 ifconfig(8)[1991] 显示的接口名称。仅当数据包在指定方向上通过指定接口时，该规则才会匹配。当使用 `log` 关键字时，以下限定符可以按此顺序使用:

`body`：表示数据包内容的前 128 个字节将记录在标头之后。

`first`：如果 `log` 关键字与 `keep state` 选项一起使用，则建议使用此选项，以便只记录触发数据包，而不是每个匹配有状态连接的数据包。其他选项可用于指定错误返回消息。

有关更多详细信息，请参阅 ipf(5)[1992]。

- **PROTO_TYPE**

  协议类型是可选的。然而，如果规则需要指定一个 SRC_PORT 或 DST_PORT，它是必须的，因为它定义了协议的类型。当指定协议类型时，使用 `proto` 关键字，后面跟上 **/etc/protocols** 中的协议编号或名称。协议名称的例子包括 `tcp`、`udp` 或 `icmp`。如果指定了 PROTO_TYPE，但没有指定 SRC_PORT 或 DST_PORT，该协议的所有端口号都将与该规则相匹配。

  SRC_ADDR

  `from` 关键字是必需的，后跟一个表示数据包源的关键字。源可以是主机名、IP 地址后跟 CIDR 掩码、地址池或 `all` 关键字。有关示例，请参阅 ipf(5)[1993]。

  没有办法匹配 IP 地址的范围，这些 IP 地址不能使用虚线数字形式/掩码长度表示法轻松表达自己。二进制包或 ports net-mgmt/ipcalc[1994] 可用于简化 CIDR 掩码的计算。其他信息可在实用程序的网页上找到: http://jodies.de/ipcalc。

- **SRC_PORT**

  源的端口号是可选的。然而，如果使用它，它需要在规则中首先定义 PROTO_TYPE。端口号前面还必须有 proto 关键字。

  支持一些不同的比较运算符：= (等于)，! = (不等于)，< (小于)，> (大于)，⇐ (小于或等于)，和>= (大于或等于)。

  要指定端口范围，将两个端口号放在 <> (小于和大于)、>< (大于和小于) 或: (大于或等于和小于或等于)。

- **DST_ADDR**

  `to` 关键字是必需的，后跟一个表示数据包目标的关键字。与 SRC_ADDR 类似，它可以是主机名、IP 地址，后跟 CIDR 掩码、地址池或关键字 `all`。

[1990] https://www.freebsd.org/cgi/man.cgi?query=ipl&sektion=4&format=html
[1991] https://www.freebsd.org/cgi/man.cgi?query=ifconfig&sektion=8&format=html
[1992] https://www.freebsd.org/cgi/man.cgi?query=ipf&sektion=5&format=html
[1993] https://www.freebsd.org/cgi/man.cgi?query=ipf&sektion=5&format=html
[1994] https://cgit.freebsd.org/ports/tree/net-mgmt/ipcalc/pkg-descr

- **DST_PORT**

  与 SRC_PORT 类似，目标的端口号是可选的。但是，如果使用它，则需要首先在规则中定义 PROTO_TYPE。端口号前面还必须有关键字 `proto`。

- **TCP_FLAG|ICMP_TYPE**

  如果 tcp 被指定为 PROTO_TYPE，标志可以被指定为字母，每个字母代表一个可能的 TCP 标志，用于确定连接的状态。可能的值是:S（SYN）、A（ACK）、P（PSH）、F（FIN）、U（URG）、R（RST）、C（CWN）和 E（ECN）。

  如果 `icmp` 被指定为 PROTO_TYPE，可以指定要匹配的 ICMP 类型。关于允许的类型请参考 ipf(5)[1995]。

  STATE

  如果一个 `pass` 规则包含 `keep state`，IPF 将向其动态状态表添加一个条目，并允许与连接匹配的后续数据包。IPF 可以跟踪 TCP、UDP 和 ICMP 会话的状态。任何 IPF 可以确定是活动会话的一部分的数据包，即使它是不同的协议，也会被允许。

  在 IPF 中，首先根据动态状态表检查预定通过连接到公共互联网的接口出去的数据包。如果该数据包与构成活动会话的下一个预期数据包相匹配，它就会退出防火墙，并在动态状态表中更新会话会话流的状态。不属于已经激活的会话的数据包将根据出站规则集进行检查。从连接到公共互联网的接口进来的数据包首先根据动态状态表进行检查。如果数据包与构成活动会话的下一个预期数据包相匹配，它就退出防火墙，并在动态状态表中更新会话对话流的状态。不属于已经活动的会话的数据包将根据入站规则集进行检查。

  在 `keep state` 之后可以添加几个关键字。如果使用，这些关键字会设置各种控制有状态过滤的选项，比如设置连接限制或连接期限。请参考 ipf(5)[1996] 获取可用选项列表及其描述。

33.5.3.示例规则集

本节演示如何创建一个示例规则集，该规则集仅允许服务匹配 `pass` 规则并阻止所有其他规则。

FreeBSD 使用环回接口（**lo0**）和 IP 地址 `127.0.0.1` 进行内部通信。防火墙规则集必须包含允许自由移动这些内部使用的数据包的规则：

```
# no restrictions on loopback interface
pass in quick on lo0 all
pass out quick on lo0 all
```

连接到互联网的公共接口用于授权和控制所有出站和入站连接的访问。如果将一个或多个接口连接到专用网络，则这些内部接口可能需要规则来允许源自 LAN 的数据包在内部网络之间流动或流向连接到互联网的接口。规则集应分为三个主要部分：任何受信任的内部接口、通过公共接口的出站连接和通过公共接口的入站连接。

以下两个规则允许所有流量通过名为 **xl0** 的受信任 LAN 接口：

[1995] https://www.freebsd.org/cgi/man.cgi?query=ipf&sektion=5&format=html

[1996] https://www.freebsd.org/cgi/man.cgi?query=ipf&sektion=5&format=html

```
# no restrictions on inside LAN interface for private network
pass out quick on xl0 all
pass in quick on xl0 all
```

公共接口的出站和入站部分的规则应将最常匹配的规则放在不太匹配的规则之前，该部分中的最后一个规则阻止并记录该接口和方向的所有数据包。

这组规则定义名为 **dc0** 的公共接口的出站部分。这些规则保持状态，并确定内部系统授权用于互联网访问的特定服务。所有规则都使用 `quick`，并指定适当的端口号和(在适用的情况下)目的地址。

```
# interface facing Internet (outbound)
# Matches session start requests originating from or behind the
# firewall, destined for the Internet.

# Allow outbound access to public DNS servers.
# Replace x.x.x.x with address listed in /etc/resolv.conf.
# Repeat for each DNS server.
pass out quick on dc0 proto tcp from any to x.x.x.x port = 53 flags S keep state
pass out quick on dc0 proto udp from any to x.x.x.x port = 53 keep state

# Allow access to ISP's specified DHCP server for cable or DSL networks.
# Use the first rule, then check log for the IP address of DHCP server.
# Then, uncomment the second rule, replace z.z.z.z with the IP address,
# and comment out the 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 HTTP and HTTPS
pass out quick on dc0 proto tcp from any to any port = 80 flags S keep state
pass out quick on dc0 proto tcp from any to any port = 443 flags S keep state

# Allow email
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 NTP
pass out quick on dc0 proto tcp from any to any port = 37 flags S keep state

# Allow FTP
pass out quick on dc0 proto tcp from any to any port = 21 flags S keep state

# Allow SSH
pass out quick on dc0 proto tcp from any to any port = 22 flags S keep state
```

(continues on next page)

(continued from previous page)

```
# Allow ping
pass out quick on dc0 proto icmp from any to any icmp-type 8 keep state

# Block and log everything else
block out log first quick on dc0 all
```

公共接口的入站部分中的规则示例首先阻止所有不需要的数据包。这将减少最后一个规则记录的数据包数。

```
# interface facing Internet (inbound)
# 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 fragments and too short tcp packets
block in quick on dc0 all with frags
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 OS fingerprint attempts and log first occurrence
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 and ident
block in quick on dc0 proto icmp all icmp-type 8
block in quick on dc0 proto tcp from any to any port = 113

# Block incoming Netbios services
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
```

(continues on next page)

(continued from previous page)

```
block in log first quick on dc0 proto tcp/udp from any to any port = 81
```

每当带有 `log first` 选项的规则上记录消息时，请运行 `ipfstat -hio` 以评估规则匹配的次数。大量匹配可能表示系统受到攻击。

入站部分中的其余规则定义了允许从互联网启动哪些连接。最后一个规则拒绝本节中前面规则未显式允许的所有连接。

```
# Allow traffic in from ISP's DHCP server. Replace z.z.z.z with
# the same IP address used in the outbound section.
pass in quick on dc0 proto udp from z.z.z.z to any port = 68 keep state

# Allow public connections to specified internal web server
pass in quick on dc0 proto tcp from any to x.x.x.x port = 80 flags S keep state

# Block and log only first occurrence of all remaining traffic.
block in log first quick on dc0 all
```

33.5.4.配置 NAT

要启用 NAT，请将这些语句添加到 **/etc/rc.conf** 并指定包含 NAT 规则的文件的名称：

```
gateway_enable="YES"
ipnat_enable="YES"
ipnat_rules="/etc/ipnat.rules"
```

NAT 规则非常灵活，可以完成许多不同的事情，以满足商业和家庭用户的需求。此处提供的规则语法已得到简化，以演示常见用法。有关完整的规则语法说明，请参阅 ipnat(5)[1997]。

NAT 规则的基本语法如下，其中启动 `map` 规则，*IF* 应替换为外部接口的名称：

```
map IF LAN_IP_RANGE -> PUBLIC_ADDRESS
```

LAN_IP_RANGE 是内部客户端使用的 IP 地址范围。通常，它是一个专用地址范围，例如 `192.168.1.0/24`。*PUBLIC_ADDRESS* 可以是静态外部 IP 地址，也可以是表示分配给 *IF* 的 IP 地址的关键字 `0/32`。

在 IPF 中，当数据包从具有公共目标的 LAN 到达防火墙时，它首先通过防火墙规则集的出站规则。然后，数据包被传递到 NAT 规则集，该规则集从上往下读取，第一个匹配的规则成功。IPF 根据数据包的接口名称和源 IP 地址测试每个 NAT 规则。当数据包的接口名称与 NAT 规则匹配时，将检查数据包在专用 LAN 中的源 IP 地址，以查看它是否在 *LAN_IP_RANGE* 中指定的 IP 地址范围内。在匹配时，数据包的源 IP 地址

[1997] https://www.freebsd.org/cgi/man.cgi?query=ipnat&sektion=5&format=html

将使用 *PUBLIC_ADDRESS* 指定的公共 IP 地址重写。IPF 在其内部 NAT 表中发布一个条目，以便当数据包从互联网返回时，可以将其映射回其原始专用 IP 地址，然后再传递到防火墙规则进行进一步处理。

对于具有大量内部系统或多个子网的网络，将每个专用 IP 地址汇集到单个公共 IP 地址的过程将成为资源问题。有两种方法可以解决此问题。

第一种方法是分配要用作源端口的端口范围。通过添加 `portmap` 关键字，可以将 NAT 定向到只使用指定范围内的源端口:

```
map dc0 192.168.1.0/24 -> 0/32 portmap tcp/udp 20000:60000
```

另外，使用 `auto` 关键字告诉 NAT 来确定可用的端口:

```
map dc0 192.168.1.0/24 -> 0/32 portmap tcp/udp auto
```

第二种方法是使用公共地址池。当有太多的 LAN 地址无法放入单个公共地址并且有一个公共 IP 地址块可用时，这很有用。这些公共地址可用作池，NAT 从池中选择 IP 地址，因为数据包的地址在输出时被映射。

可以使用网络掩码或 CIDR 表示法指定公共 IP 地址的范围。这两个规则是等效的:

```
map dc0 192.168.1.0/24 -> 204.134.75.0/255.255.255.0
map dc0 192.168.1.0/24 -> 204.134.75.0/24
```

一个常见的做法是，将一个可公开访问的网络服务器或邮件服务器隔离到一个内部网段。来自这些服务器的流量仍然要经过 NAT，但需要进行端口重定向，将入站流量引导到正确的服务器。例如，要把一个使用内部地址 `10.0.10.25` 的 Web 服务器映射到它的公共 IP 地址 `20.20.20.5`，使用这个规则。

```
rdr dc0 20.20.20.5/32 port 80 -> 10.0.10.25 port 80
```

如果它是唯一的 Web 服务器，则此规则也可以工作，因为它将所有外部 HTTP 请求重定向到 `10.0.10.25`:

```
rdr dc0 0.0.0.0/0 port 80 -> 10.0.10.25 port 80
```

IPF 有一个内置的 FTP 代理，可以与 NAT 一起使用。它监视主动或被动 FTP 连接请求的所有出站流量，并动态创建包含 FTP 数据通道使用的端口号的临时筛选器规则。这样就无需为 FTP 连接打开大范围的高阶端口。

在此示例中，第一个规则调用来自内部 LAN 的出站 FTP 流量的代理。第二条规则将 FTP 流量从防火墙传递到互联网，第三条规则处理来自内部 LAN 的所有非 FTP 流量:

```
map dc0 10.0.10.0/29 -> 0/32 proxy port 21 ftp/tcp
map dc0 0.0.0.0/0 -> 0/32 proxy port 21 ftp/tcp
map dc0 10.0.10.0/29 -> 0/32
```

FTP `map` 规则位于 NAT 规则之前，因此当数据包与 FTP 规则匹配时，FTP 代理会创建临时筛选规则，以允许 FTP 会话数据包通过并进行 NAT。所有不是 FTP 的 LAN 数据包将不匹配 FTP 规则，但如果它们与第三个规则匹配，则将进行 NAT。

如果没有 FTP 代理，则需要以下防火墙规则。请注意，如果没有代理，则需要允许上述 `1024` 所有端口：

```
# 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
```

每当包含 NAT 规则的文件被编辑时，运行带有 `-CF` 的 `ipnat` 来删除当前的 NAT 规则并刷新动态翻译表的内容。包括 `-f` 并指定要加载的 NAT 规则集的名称。

```
# ipnat -CF -f /etc/ipnat.rules
```

要显示 NAT 统计信息：

```
# ipnat -s
```

列出 NAT 表的当前映射：

```
# ipnat -l
```

打开详细模式并显示与规则处理以及活动规则和表条目相关的信息：

```
# ipnat -v
```

33.5.5.查看 IPF 统计数据

IPF 包括 ipfstat(8)[1998]，它可用于检索和显示统计信息，这些统计信息是在数据包通过防火墙时与规则匹配时收集的。统计信息是自上次启动防火墙以来或自上次使用 `ipf -Z` 重置为零以来累积的。

默认 `ipfstat` 输出如下所示：

[1998] https://www.freebsd.org/cgi/man.cgi?query=ipfstat&sektion=8&format=html

```
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)
```

有几个选项是可用的。当提供 `-i`（入站）或 `-o`（出站）时，该命令将检索并显示当前已安装并由内核使用的适当的过滤规则列表。要想看到规则的编号，请加入 `-n`。例如，`ipfstat -on` 显示了带有规则编号的出站规则表。

```
@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
```

包括 `-h` 每个规则前面加上规则匹配次数的计数。例如，`ipfstat -oh` 显示出站内部规则表，并在每个规则前面加上其使用计数：

```
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
```

要以类似于 top(1)[1999] 的格式显示状态表，请使用 `ipfstat -t`。当防火墙受到攻击时，此选项提供了识别和查看攻击数据包的功能。可选的子标志提供了选择要实时监视的目标或源 IP、端口或协议的功能。有关详细信息，请参阅 ipfstat(8)[2000]。

[1999] https://www.freebsd.org/cgi/man.cgi?query=top&sektion=1&format=html

[2000] https://www.freebsd.org/cgi/man.cgi?query=ipfstat&sektion=8&format=html

33.5.6. IPF 日志记录

IPF 提供了 ipmon，可用于以人类可读的格式写入防火墙的日志记录信息。它要求首先使用配置 FreeBSD[2001] 内核中的说明将options IPFILTER_LOG 添加到定制内核中。

这个命令通常在守护模式下运行，以便提供一个连续的系统日志文件，从而可以审查过去事件的日志。由于 FreeBSD 有一个内置的 syslogd(8)[2002] 工具来自动旋转系统日志，默认的 **rc.conf** ipmon_flags 语句使用 -Ds

```
ipmon_flags="-Ds" # D = start as daemon
                  # s = log to syslog
                  # v = log tcp window, ack, seq
                  # n = map IP & port to names
```

日志记录提供了在事后查看信息的功能，例如丢弃了哪些数据包、它们来自哪些地址以及它们要去哪里。此信息在跟踪攻击者时很有用。

一旦在 **rc.conf** 中启用了日志记录工具并以 service ipmon start 启动，IPF 将仅记录包含 log 关键字的规则。防火墙管理员决定应记录规则集中的哪些规则，通常只记录拒绝规则。习惯上将关键字 log 包含在规则集的最后一个规则中。这样就可以查看与规则集中的任何规则都不匹配的所有数据包。

默认情况下，ipmon -Ds 模式用 local0 作日志记录工具。以下日志记录级别可用于进一步隔离记录的数据：

```
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 due to an␣
↪incomplete header
```

要将 IPF 设置为将所有数据记录到 **/var/log/ipfilter.log**，请首先创建空文件：

```
# touch /var/log/ipfilter.log
```

然后，将所有要记录的消息写入指定的文件，请将以下语句添加到 **/etc/syslog.conf**：

```
local0.* /var/log/ipfilter.log
```

要激活更改并指示 syslogd(8)[2003] 读取修改后的 **/etc/syslog.conf**，请运行 service syslogd reload。

不要忘记编辑 **/etc/newsyslog.conf** 来轮换新的日志文件。

[2001] https://docs.freebsd.org/en/books/handbook/kernelconfig/index.html#kernelconfig

[2002] https://www.freebsd.org/cgi/man.cgi?query=syslogd&sektion=8&format=html

[2003] https://www.freebsd.org/cgi/man.cgi?query=syslogd&sektion=8&format=html

`ipmon` 生成的消息由以空格分隔的数据字段组成。所有消息共有的字段包括：

1. 数据包接收日期。
2. 数据包接收时间。其格式为 HH：MM：SS.F，表示小时、分钟、秒和几分之一秒。
3. 处理数据包的接口的名称。
4. 格式为 `@0:17` 的规则的组和规则编号。
5. 操作：`p` 对于已通过、`b` 对于阻止、`S` 对于短数据包 `n` 与任何规则都不匹配以及 `L` 对于日志规则。
6. 写为三个字段的地址：源地址和端口（以逗号分隔）、→ 符号以及目标地址和端口。例如：`209.53.17.22,80 → 198.73.220.17,1722`。
7. `PR` 后跟协议名称或编号：例如，`PR tcp`。
8. `len` 后跟标头长度和数据包的总长度：例如，`len 20 40`

如果数据包是 TCP 数据包，则将有一个以连字符开头的附加字段，后跟与设置的任何标志相对应的字母。请参考 ipf(5)[2004] 获取字母及其标志的列表。

如果数据包是 ICMP 数据包，则末尾将有两个字段：第一个字段始终为 “icmp”，下一个是 ICMP 消息和子消息类型，由斜杠分隔。例如：`icmp 3/3` 对于端口无法访问的消息。

33.6. Blacklistd

Blacklistd 是一个监听套接字的守护进程，等待接收来自其他守护进程的关于连接尝试失败或成功的通知。它最广泛地用于阻止开放端口上的太多连接尝试。一个主要的例子是在互联网上运行的 SSH 从试图猜测密码并获得访问权限的机器人或脚本中获取大量请求。使用 Blacklistd，守护程序可以通知防火墙创建过滤规则，以阻止在多次尝试后从单点进行过多的连接尝试。Blacklistd 最初是在 NetBSD 上开发的，并在第 7 版中出现。FreeBSD 11 从 NetBSD 引入了 Blacklistd。

本章介绍如何设置和配置 Blacklistd，并提供如何使用 Blacklistd 的示例。读者应熟悉规则等基本防火墙概念。有关详细信息，请参阅防火墙章节。示例中使用了 PF，但 FreeBSD 上提供的其他防火墙也应该能够使用 Blacklistd。

33.6.1. Blacklistd

Blacklistd 的主要配置存储在 blacklistd.conf(5)[2005] 中。各种命令行选项也可用于更改 Blacklistd 的运行时行为。跨重新启动的持久配置应存储在 **/etc/blacklistd.conf** 中。要在系统引导期间启用守护程序，请在 **/etc/rc.conf** 中添加一行 `blacklistd_enable`，如下所示：

```
# sysrc blacklistd_enable=yes
```

[2004] https://www.freebsd.org/cgi/man.cgi?query=ipf&sektion=5&format=html

[2005] https://www.freebsd.org/cgi/man.cgi?query=blacklistd.conf&sektion=5&format=html

若要手动启动该服务，请运行以下命令：

```
# service blacklistd start
```

33.6.2.创建列入 Blacklistd 的规则集

Blacklistd 的规则在 blacklistd.conf(5)[2006] 中配置，每行一个条目。每个规则都包含一个由空格或制表符分隔的元组。规则属于`local` 或 `remote`，分别适用于运行 Blacklistd 的计算机或外部源。

33.6.2.1.本地规则

本地规则的示例 blacklistd.conf 条目如下所示：

```
[local]
ssh             stream  *       *               *       3       24h
```

该部分后面的所有规则都被视为本地规则 `[local]`（这是默认规则），并应用于本地计算机。当遇到 `[remote]` 节时，它后面的所有规则都被作为远程机器规则处理。

由制表符或空格分隔的七个字段定义了一个规则。前四个字段标识应列入 Blacklistd 的流量。后面的三个字段定义了 backlistd 的行为。通配符表示为星号（`*`），匹配此字段中的任何内容。第一个字段定义位置。在本地规则中，这些是网络端口。位置字段的语法如下：

```
[address|interface][/mask][:port]
```

地址可以指定为数字格式的 IPv4 或方括号中的 IPv6。也可以使用接口名称 `em0`。

套接字类型由第二个字段定义。TCP 套接字的类型为 `stream`，而 UDP 表示为 `dgram`。上面的示例使用 TCP，因为 SSH 使用该协议。

协议可用于列入 Blacklistd 的规则的第三个字段。可以使用以下协议：`tcp`、`udp`、`tcp6`、`udp6` 或数字。通配符（如示例中所示）通常用于匹配所有协议，除非有理由通过某个协议来区分流量。

在第四个字段中，定义了报告事件的守护程序进程的有效用户或所有者。用户名或 UID 可以在此处使用，也可以使用通配符（请参阅上面的示例规则）。

数据包筛选器规则名称由第五个字段声明，该字段启动规则的行为部分。默认情况下，blacklistd 将所有块放在 **pf.conf** 中的一个名为 `blacklistd` 的 pf 锚点下，如下所示：

```
anchor "blacklistd/*" in on $ext_if
block in
pass out
```

[2006] https://www.freebsd.org/cgi/man.cgi?query=blacklistd.conf&sektion=5&format=html

对于单独的阻止列表，可以在此字段中使用锚点名称。在其他情况下，通配符就足够了。当名称以连字符（-）开头时，表示应使用前面附加了默认规则名称的锚点。上面使用连字符的修改示例如下所示：

```
ssh             stream  *       *               -ssh       3       24h
```

使用这样的规则，任何新的阻止列表规则都将添加到名为 `blacklistd-ssh` 的锚点中。

为了阻止整个子网对单一规则的违反，可以在规则名称中使用一个 `/`。这导致名称的其余部分被解释为应用于规则中指定的地址的掩码。例如，这条规则将阻止与 `/24` 相邻的每个地址。

```
22              stream  tcp       *               */24    3       24h
```

> 注意
>
> 在这里指定适当的协议是很重要的。IPv4 和 IPv6 对 /24 的处理方式不同，这就是为什么不能在此规则的第三个字段使用 * 的原因。

此规则定义，如果该网络中的任何一台主机行为异常，则该网络上的其他所有内容也将被阻止。

第六个字段名为 `nfail`，设置将相关远程 IP 列入 Blacklistd 所需的登录失败次数。当在此位置使用通配符时，这意味着块将永远不会发生。在上面的示例规则中，定义了三个限制，这意味着在一个连接上三次尝试登录 SSH 后，IP 将被阻止。

列入 Blacklistd 的规则定义中的最后一个字段指定主机被列入 Blacklistd 的时间长度。默认单位为秒，后缀也可以分别指定为分钟 `m`、小时 `h` 和天 `d`。

这个例子的规则整体上意味着，在三次验证 SSH 后，将导致该主机的新 PF 块规则。规则的匹配是通过首先一个接一个地检查本地规则，从最具体到最不具体。当匹配发生时，将应用 `remote` 规则，并由匹配的 `remote` 规则更改 name、`nfail` 和 disable 字段。

33.6.2.2.远程规则

远程规则用于指定 Blacklistd 如何根据当前正在评估的远程主机更改其行为。远程规则中的每个字段都与本地规则中的字段相同。唯一的区别在于 Blacklistd 使用它们的方式。为了解释它，使用了以下示例规则：

```
[remote]
203.0.113.128/25 *      *       *               =/25    =       48h
```

地址字段可以是 IP 地址（v4 或 v6）、端口或两者。这允许为特定的远程地址范围设置特殊规则，如本例所示。套接字段的类型、协议和所有者的解释与本地规则中的相同。

但是，名称字段是不同的：远程规则中的等号（=）告诉 blacklistd 使用匹配的本地规则中的值。这意味着将采用防火墙规则条目并添加前缀 `/25`（的网络掩码`255.255.255.128`）。当来自该地址范围的连接被列入阻止列表时，整个子网都会受到影响。此处也可以使用 PF 锚点名称，在这种情况下，Blacklistd 会将此地址块的规则添加到该名称的锚点。指定通配符时使用默认表。

在 nfail 列中可以为一个地址定义一个自定义的失败次数。这对特定规则的例外情况很有用，也许可以允许某人不那么严格地应用规则，或者在登录尝试中更宽松一点。当在这第六个字段中使用星号时，阻断功能将被禁用。

与来自本地网络（如办公室）的尝试相比，远程规则允许对登录尝试实施更严格的限制。

33.6.3.列入 Blacklistd 的客户端配置

FreeBSD 中有一些软件包可以利用 Blacklistd 的功能。两个最突出的是 ftpd(8)[2007] 和 sshd(8)[2008]，用于阻止过多的连接尝试。要激活 SSH 守护程序中的 Blacklistd，请将以下行添加到 **/etc/ssh/sshd_config**：

```
UseBlacklist yes
```

之后重新启动 sshd 以使这些更改生效。

ftpd(8)[2009] 的 Blacklistd 使用 -B 来启用，或者在 **/etc/inetd.conf** 中，或者作为 **/etc/rc.conf** 中的一个参数，像这样:

```
ftpd_flags="-B"
```

这就是使这些程序与 Blacklistd 对话所需的全部内容。

33.6.4. Blacklistd 管理

Blacklistd 为用户提供了一个名为 blacklistctl(8)[2010] 的管理实用程序。它显示被阻止的地址和网络，这些地址和网络被 blacklistd.conf(5)[2011] 中定义的规则列入了阻止列表。要查看当前被阻止的主机列表，请像这样结合使用 dump 和 -b。

```
# blacklistctl dump -b
      address/ma:port id      nfail   last access
213.0.123.128/25:22   OK      6/3     2019/06/08 14:30:19
```

这个例子显示，在端口 22 上有 6 次允许的尝试，这些尝试来自地址范围213.0.123.128/25。列出的尝试次数多于允许的次数，因为 SSH 允许客户端在一个 TCP 连接上尝试多次登录。目前正在进行的连接不会被 blacklistd 阻止。最后的连接尝试被列在输出的 last access 列中。

要查看此主机将在 Blacklistd 上的剩余时间，请添加 -r 到上一个命令中。

[2007] https://www.freebsd.org/cgi/man.cgi?query=ftpd&sektion=8&format=html
[2008] https://www.freebsd.org/cgi/man.cgi?query=sshd&sektion=8&format=html
[2009] https://www.freebsd.org/cgi/man.cgi?query=ftpd&sektion=8&format=html
[2010] https://www.freebsd.org/cgi/man.cgi?query=blacklistctl&sektion=8&format=html
[2011] https://www.freebsd.org/cgi/man.cgi?query=blacklistd.conf&sektion=5&format=html

```
# blacklistctl dump -br
      address/ma:port id      nfail   remaining time
213.0.123.128/25:22   OK      6/3     36s
```

在此示例中，还剩下 36 秒，直到此主机不再被阻止。

33.6.5.从阻止列表中删除主机

有时，有必要在剩余时间到期之前将一个主机从封锁名单中删除。不幸的是，`blacklistd` 中没有这样的功能。不过，可以用 `pfctl` 把地址从 PF 表中删除。对于每个被封锁的端口，在 **/etc/pf.conf** 中定义的 blacklistd 锚内都有一个子锚。例如，如果有一个封锁端口 22 的子锚，它就被称为 `blacklistd/22`。在这个子锚里面有一个表，包含了被封的地址。这个表的名字是 port，后面是端口号。在这个例子中，它被称为 `port22`。有了这些信息，现在就可以用 pfctl(8)[2012] 来显示所有列出的地址，如下所示：

```
# pfctl -a blacklistd/22 -t port22 -T show
...
213.0.123.128/25
...
```

从列表中确定要取消阻止的地址后，以下命令会将其从列表中删除：

```
# pfctl -a blacklistd/22 -t port22 -T delete 213.0.123.128/25
```

该地址现已从 PF 中删除，但仍将显示在 blacklistctl 列表中，因为它不知道在 PF 中所做的任何更改。Blacklistd 数据库中的条目最终将过期，并最终从其输出中删除。如果主机再次与列入 Blacklistd 的阻止规则之一匹配，则将再次添加该条目。

[2012] https://www.freebsd.org/cgi/man.cgi?query=pfctl&sektion=8&format=html

第三十三章高级网络

34.1.概述

本章涉及了一些高级网络主题。

读完本章，你就会知道：

- 网关和路由的基础知识。
- 如何设置 USB 网络共享。
- 如何设置 IEEE® 802.11 和蓝牙 ® 设备。
- 如何让 FreeBSD 充当网桥。
- 如何设置网络 PXE 启动。
- 如何在 FreeBSD 中启用和利用共享地址冗余协议（CARP）的功能。
- 如何在 FreeBSD 上配置多个 VLAN。
- 配置蓝牙耳机。

在阅读本章之前，你应该：

- 了解 **/etc/rc** 脚本的基础知识。
- 熟悉基本的网络术语。
- 了解 FreeBSD 上的基本网络配置（FreeBSD 网络[2013]）。
- 知道如何配置和安装新的 FreeBSD 内核（配置 FreeBSD 内核[2014]）。
- 了解如何安装其他第三方软件（安装应用程序：软件包和 Port[2015]）。

[2013] https://docs.freebsd.org/en/books/handbook/network/#network

[2014] https://docs.freebsd.org/en/books/handbook/kernelconfig/index.html#kernelconfig

[2015] https://docs.freebsd.org/en/books/handbook/ports/index.html#ports

34.2.网关和路由

路由是允许一个系统查找到另一个系统的网络路径的机制。路由是一对定义的地址，表示“目标”和“网关”。路由指示在尝试到达指定目标时，通过指定的网关发送数据包。有三种类型的目标：单个主机、子网和“默认”。如果没有其他路由适用，则使用“默认路由”。还有三种类型的网关：单个主机、接口（也称为链路）和以太网硬件地址（MAC）。已知路由存储在路由表中。

本节提供路由基础知识的概述。然后，它演示了如何将 FreeBSD 系统配置为路由器，并提供了一些故障排除技巧。

34.2.1.路由基础

要查看 FreeBSD 系统的路由表，请使用 netstat(1)[2016]：

```
% netstat -r
Routing tables

Internet:
Destination      Gateway            Flags     Refs      Use     Netif Expire
default          outside-gw         UGS         37      418       em0
localhost        localhost          UH           0      181       lo0
test0            0:e0:b5:36:cf:4f   UHLW         5    63288       re0     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
```

此示例中的条目如下所示：

- **default**

  这个表中的第一个路由指定的是 `default` 路由。当本地系统需要与远程主机建立连接时，它检查路由表以确定是否存在一个已知路径。如果远程主机与表中的一个条目相匹配，系统就会检查它是否可以使用该条目中指定的接口进行连接。

  如果目的地不匹配条目，或者所有已知路径都失败，系统就使用默认路由的条目。对于局域网上的主机，默认路由中的 `Gateway` 字段被设置为与互联网直接连接的系统。读取这个条目时，要确认 `Flags` 栏中显示网关是可用的（UG）。

  一台本身作为通往外部世界的网关的机器，其默认路由将是互联网服务提供商（ISP）的网关机器。

- **localhost**

[2016] https://www.freebsd.org/cgi/man.cgi?query=netstat&sektion=1&format=html

第二条路由是 `localhost` 路由。在 `Netif` 栏中为 `localhost` 指定的接口是 **lo0**，也被称为回环设备。这表明这个目的地的所有流量都应该是内部的，而不是通过网络发送出去。

- **MAC address**

  以 `0:e0:` 开头的地址是 MAC 地址。FreeBSD 会自动识别本地以太网上的任何主机，即本例中的 `test0`，并通过以太网接口 **re0** 为该主机添加一条路由。这种类型的路由有一个超时时间，在 `Expire` 一栏中可以看到，如果主机在特定的时间内没有回应，就会使用这个路由。当这种情况发生时，到这个主机的路由将被自动删除。这些主机是使用路由信息协议（RIP）确定的，它根据最短路径确定计算到本地主机的路由。

- **subnet**

  FreeBSD 会自动为本地子网添加子网路由。在这个例子中，`10.20.30.255` 是子网 `10.20.30` 的广播地址，`example.com` 是与该子网有关的域名。指定的 `link#1` 指的是机器中的第一个以太网卡。

  本地网络主机和本地子网的路由是由一个叫 routed(8)[2017] 的守护程序自动配置。如果它未运行，则仅存在由管理员静态定义的路由。

- **host**

  `host1` 一行是指主机的以太网地址。由于它是发送主机，FreeBSD 知道要使用回环接口 **(lo0)** 而不是以太网接口。

  两行 `host2` 代表使用 ifconfig(8)[2018] 创建的别名。`lo0` 接口后面的 ⇒ 符号表示除了环回地址之外，还设置了一个别名。这样的路由只显示在支持别名的主机上，本地网络上的所有其他主机都会有一个 `Link#1` 行来显示这样的路由。

- **224**

  最后一行（目标子网 `224`）处理多播。

在列中可以看到每个路由的各种 `Flags` 属性。常见的路由表标志[2019] 总结了其中一些标志及其含义：

表 29. 经常看到的路由表标志

标志	意义
U	路由处于活动状态（up）。
H	路由目标是单个主机。
G	将此目标的任何内容发送到此网关，该网关将从那里确定将其发送到何处。
S	此路由是静态配置的。
C	基于此路由克隆新路由，供计算机连接到。这种类型的路由通常用于本地网络。
W	路由是根据局域网（克隆）路由自动配置的。
L	路由涉及对以太网（链路）硬件的引用。

[2017] https://www.freebsd.org/cgi/man.cgi?query=routed&sektion=8&format=html

[2018] https://www.freebsd.org/cgi/man.cgi?query=ifconfig&sektion=8&format=html

[2019] https://docs.freebsd.org/en/books/handbook/advanced-networking/#routeflags

在 FreeBSD 系统上，可以通过指定默认网关的 IP 地址在 **/etc/rc.conf** 中定义默认路由：

```
defaultrouter="10.20.30.1"
```

也可以使用以 `route` 手动添加路由：

```
# route add default 10.20.30.1
```

请注意，手动添加的路由将无法在重新启动后继续运行。有关手动操作网络路由表的详细信息，请参阅 route(8)[2020]。

34.2.2.使用静态路由配置路由器

FreeBSD 系统可以配置为网络的默认网关或路由器，如果它是双宿主系统。双宿主系统是驻留在至少两个不同网络上的主机。通常，每个网络都连接到一个单独的网络接口，但 IP 别名可用于将多个地址（每个地址位于不同的子网上）绑定到一个物理接口。

为了让系统在接口之间转发数据包，必须配置 FreeBSD 为路由器。互联网标准和良好的工程实践会阻止 FreeBSD 项目默认启用此功能，但是可以通过将此行添加到 **/etc/rc.conf** 来将其配置为在引导时启动：

```
gateway_enable="YES"          # Set to YES if this host will be a gateway
```

要立即启用路由，请将 sysctl(8)[2021] 变量 `net.inet.ip.forwarding` 设置为 `1`。要停止路由，请将此变量重置为 `0`。

路由器的路由表需要其他路由，以便它知道如何到达其他网络。可以使用静态路由手动添加路由，也可以使用路由协议自动学习。静态路由适用于小型网络，本节介绍如何为小型网络添加静态路由条目。

> 注意
>
> 对于大型网络，静态路由很快就会变得无法扩展。FreeBSD 附带了标准的 BSD 路由守护进程 routed(8)[2022]，它提供了路由协议 RIP 的第 1 和第 2 版以及 IRDP。可以使用软件包或 port 安装 net/quagga[2023] 以支持 BGP 和 OSPF 路由协议。

请思考以下网络：

[2020] https://www.freebsd.org/cgi/man.cgi?query=route&sektion=8&format=html
[2021] https://www.freebsd.org/cgi/man.cgi?query=sysctl&sektion=8&format=html
[2022] https://www.freebsd.org/cgi/man.cgi?query=routed&sektion=8&format=html
[2023] https://cgit.freebsd.org/ports/tree/net/quagga/pkg-descr

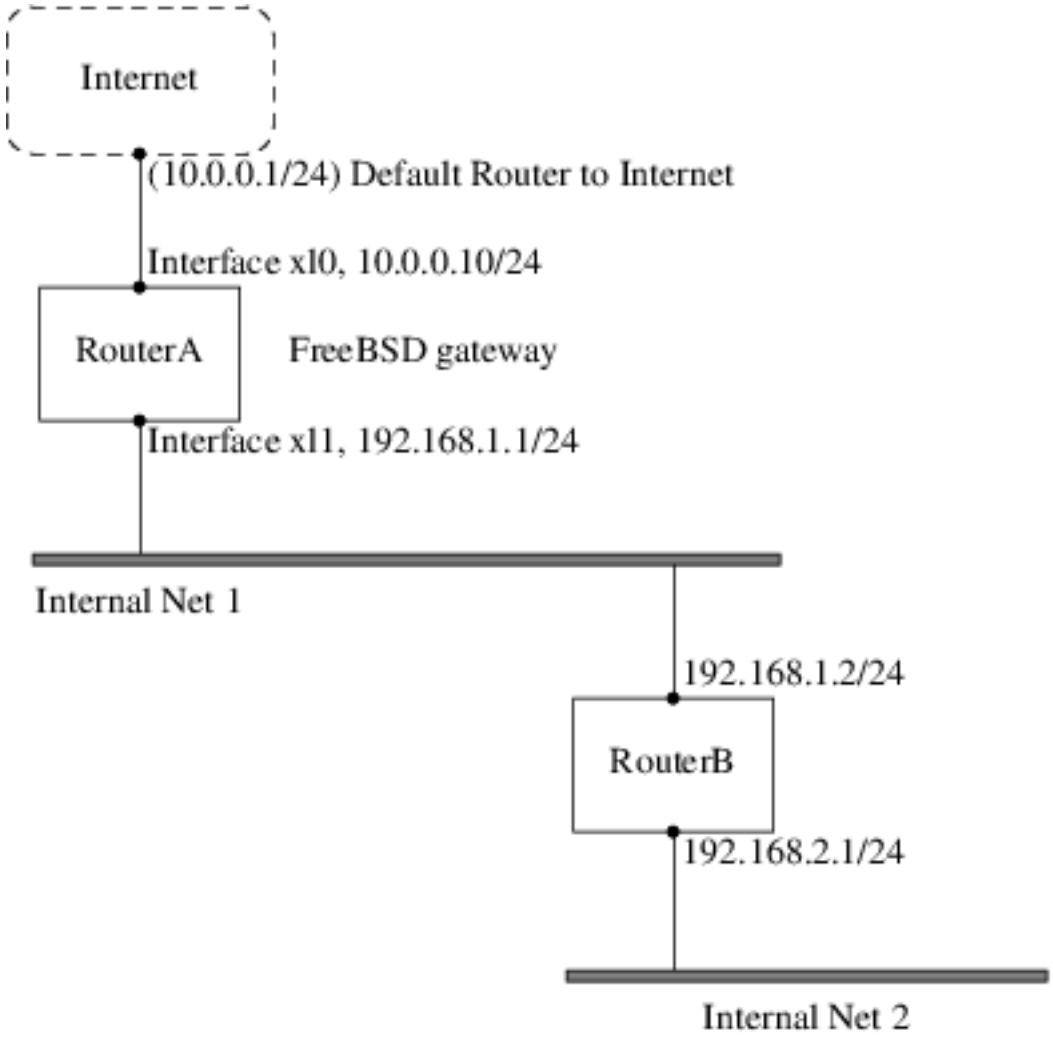


图51: 静态路由

在这种情况下，`RouterA` 是一台 FreeBSD 机器，它作为一个路由器连接到互联网的其他地方。它的默认路由设置为 `10.0.0.1`，允许它与外部世界连接。`RouterB` 已经配置为使用 `192.168.1.1` 作为默认网关。

在添加任何静态路由之前，`RouterA` 的路由表看起来是这样的：

```
% 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
```

在当前的路由表中，RouterA 没有通往 192.168.2.0/24 网络的路由。下面的命令使用 192.168.1.2 作为下一跳，将 Internal Net 2 网络添加到 RouterA 的路由表中:

```
# route add -net 192.168.2.0/24 192.168.1.2
```

现在，RouterA 可以到达 192.168.2.0/24 网络上的任何主机。然而，如果 FreeBSD 系统重新启动，路由信息将不会持续存在。如果静态路由需要持久化，请将其添加到 **/etc/rc.conf** 中。

```
# Add Internal Net 2 as a persistent static route
static_routes="internalnet2"
route_internalnet2="-net 192.168.2.0/24 192.168.1.2"
```

static_routes 配置变量是一个用空格隔开的字符串列表，每个字符串引用一个路由名称。变量 route_internalnet2 包含该路由名称的静态路由。

在 static_routes 中使用一个以上的字符串可以创建多个静态路由。下面是一个为 192.168.0.0/24 和 192.168.1.0/24 网络添加静态路由的例子。

```
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"
```

34.2.3.故障排除

当一个地址空间被分配给一个网络时，服务运营商就会配置他们的路由表，这样网络的所有流量就会被发送到网站的链接上。但是，外部站点如何知道将其数据包发送到网络的 ISP?

有一个系统跟踪所有分配的地址空间，并定义它们与互联网骨干网的连接点，或在全国和世界各地传输互联网流量的主干线。每台骨干机都有一套主表的副本，将特定网络的流量导向特定的骨干运营商，并从那里沿着服务运营商的链条向下延伸，直到到达特定网络。

服务运营商的任务是向骨干站点宣传他们是连接点，因此是一个站点的向内路径。这就是所谓的路由传播。

有时，路由传播有问题，一些站点无法连接。也许最有用的命令是 traceroute，它能试图找出路由中断的地方。它在 ping 失败时很有用。

当使用 traceroute 时，包括要连接的远程主机的地址。输出将显示尝试路径上的网关主机，最终要么到达目标主机，要么因为缺乏连接而终止。更多信息，请参考 traceroute(8)[2024]。

[2024] https://www.freebsd.org/cgi/man.cgi?query=traceroute&sektion=8&format=html

34.2.4.多播的注意事项

FreeBSD 原生支持多播应用程序和多播路由。多播应用程序不需要任何特殊配置即可在 FreeBSD 上运行。对多播路由的支持要求将以下选项编译到定制内核中：

```
options MROUTING
```

多播路由守护程序 mrouted 可以使用软件包或 port net/mrouted[2025] 进行安装。此守护程序实现 DVMRP 多播路由协议，并通过编辑 **/usr/local/etc/mrouted.conf** 进行配置，以便设置隧道和 DVMRP。安装 mrouted 还会安装 map-mbone 和 mrinfo，以及它们相关的手册页。有关配置示例，请参阅这些内容。

> 注意
>
> 在许多多播安装中，DVMRP 已在很大程度上被 PIM 协议所取代。有关详细信息，请参阅 pim(4)[2026]。

34.3.虚拟主机

FreeBSD 的一个常见用途是虚拟站点托管，其中一台服务器在网络上显示为多台服务器。这是通过将多个网络地址分配给单个接口来实现的。

给定的网络接口有一个“真实”地址，并且可以有任意数量的“别名”地址。这些别名通常是通过在 **/etc/rc.conf** 中放置别名条目来添加的，如本例所示：

```
# sysrc ifconfig_fxp0_alias0="inet xxx.xxx.xxx.xxx netmask xxx.xxx.xxx.xxx"
```

别名条目必须从 `alias0` 开始，使用连续的数字，例如 `alias0`、`alias1` 等。配置过程将在第一个缺失的数字处停止。

别名网络掩码的计算很重要。对于给定的接口，必须有一个地址正确地表示网络的掩码。该网络中的任何其他地址都必须有一个全为 1 的网络掩码，表示为 `255.255.255.255` 或 `0xffffffff`。

例如，考虑 `fxp0` 接口连接到两个网络的情况：网络掩码为 `255.255.255.0` 的 `10.1.1.0` 和网络掩码为 `255.255.255.240` 的 `202.0.75.16`。系统将配置在 `10.1.1.1` 至 `10.1.1.5` 和 `202.0.7.5.17` 至 `202.0.7.5.20` 范围内。只有给定范围中的第一个网络地址才应该具有真正的网络掩码。所有其余部分（`10.1.1.2` 至 `10.1.1.5` 和 `202.0.7.5.18` 至 `202.0.7.5.20`）必须配置 `255.255.255.255` 的网络掩码。

以下 **/etc/rc.conf** 条目为该场景配置了正确的适配器：

```
# sysrc ifconfig_fxp0="inet 10.1.1.1 netmask 255.255.255.0"
# sysrc ifconfig_fxp0_alias0="inet 10.1.1.2 netmask 255.255.255.255"
```

(continues on next page)

[2025] https://cgit.freebsd.org/ports/tree/net/mrouted/pkg-descr

[2026] https://www.freebsd.org/cgi/man.cgi?query=pim&sektion=4&format=html

(continued from previous page)

```
# sysrc ifconfig_fxp0_alias1="inet 10.1.1.3 netmask 255.255.255.255"
# sysrc ifconfig_fxp0_alias2="inet 10.1.1.4 netmask 255.255.255.255"
# sysrc ifconfig_fxp0_alias3="inet 10.1.1.5 netmask 255.255.255.255"
# sysrc ifconfig_fxp0_alias4="inet 202.0.75.17 netmask 255.255.255.240"
# sysrc ifconfig_fxp0_alias5="inet 202.0.75.18 netmask 255.255.255.255"
# sysrc ifconfig_fxp0_alias6="inet 202.0.75.19 netmask 255.255.255.255"
# sysrc ifconfig_fxp0_alias7="inet 202.0.75.20 netmask 255.255.255.255"
```

一种更简单的表达方式是使用 IP 地址范围的空格分隔列表。第一个地址将获得指定的子网掩码，其他地址将使用 255.255.255.255作为子网掩码。

```
# sysrc ifconfig_fxp0_aliases="inet 10.1.1.1-5/24 inet 202.0.75.17-20/28"
```

34.4.无线高级身份验证

FreeBSD 支持不同的无线网络连接方式。本节将介绍如何对无线网络进行高级身份验证。

要连接无线网络并进行基本身份验证，网络章中的连接无线网络并进行身份验证[2027]一节将介绍如何操作。

34.4.1.使用 EAP-TLS 的 WPA

使用 WPA 的第二种方法是使用 802.1X 后端身份验证服务器。在这种情况下，WPA 被称为 WPA Enterprise，以将其与安全性较低的 WPA Personal 区分开来。WPA 企业版中的身份验证基于可扩展身份验证协议（EAP）。

EAP 不附带加密方法。相反，EAP 嵌入在加密隧道中。有许多 EAP 身份验证方法，但 EAP-TLS、EAP-TTLS 和 EAP-PEAP 是最常见的。

具有传输层安全性（EAP-TLS）的 EAP 是一种得到良好支持的无线身份验证协议，因为它是第一个获得 Wi-Fi 联盟[2028]认证的 EAP 方法。EAP-TLS 需要三个证书才能运行：所有计算机上安装的证书颁发机构（CA）的证书、身份验证服务器的服务器证书以及每个无线客户端的一个客户端证书。在此 EAP 方法中，身份验证服务器和无线客户端都通过提供各自的证书来相互验证，然后验证这些证书是否由组织的 CA 签名。

与之前一样，配置是通过 **/etc/wpa_supplicant.conf** 完成的：

```
network={
  ssid="freebsdap" ①
  proto=RSN  ②
```

(continues on next page)

[2027] https://docs.freebsd.org/en/books/handbook/network/#wireless-authentication

[2028] http://www.wi-fi.org/

(continued from previous page)

```
  key_mgmt=WPA-EAP ③
  eap=TLS ④
  identity="loader" ⑤
  ca_cert="/etc/certs/cacert.pem" ⑥
  client_cert="/etc/certs/clientcert.pem" ⑦
  private_key="/etc/certs/clientkey.pem" ⑧
  private_key_passwd="freebsdmallclient" ⑨
}
```

① 此字段指示了网络名称（SSID）。

② 此示例使用了 RSN IEEE® 802.11i 协议，也称为 WPA2。

③ 引用要使用的密钥管理协议。在此示例中，它是使用 EAP 身份验证的 WPA。

④ 此字段指示连接的 EAP 方法。

⑤ 字段包含了 EAP 的标识字符串。

⑥ 字段指示了 CA 证书文件的路径名。需要此文件来验证服务器证书。

⑦ 行提供了客户端证书文件的路径名。此证书对于网络的每个无线客户端都是唯一的。

⑧ 字段是客户端证书私钥文件的路径名。

⑨ 字段包含了私钥的密码。

然后，将以下行添加到 **/etc/rc.conf**：

```
wlans_ath0="wlan0"
ifconfig_wlan0="WPA DHCP"
```

下一步是调出接口：

```
# 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
```

(continues on next page)

(continued from previous page)

```
    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
```

也可以使用 wpa_supplicant(8)[2029] 和 ifconfig(8)[2030] 手动调出接口。

34.4.2.使用 EAP-TTLS 的 WPA

使用 EAP-TLS，身份验证服务器和客户端都需要证书。使用 EAP-TTLS，客户端证书是可选的。此方法类似于 Web 服务器，即使访问者没有客户端证书，也会创建安全的 SSL 隧道。EAP-TTLS 使用加密的 TLS 隧道来安全传输身份验证数据。

所需的配置可以添加到 **/etc/wpa_supplicant.conf**：

```
network={
  ssid="freebsdap"
  proto=RSN
  key_mgmt=WPA-EAP
  eap=TTLS ①
  identity="test" ②
  password="test" ③
  ca_cert="/etc/certs/cacert.pem" ④
  phase2="auth=MD5" ⑤
}
```

① 此字段指定连接的 EAP 方法。

② 字段包含加密 TLS 隧道内用于 EAP 身份验证的身份字符串。

③ 字段包含用于 EAP 身份验证的密码短语。

④ 字段表示 CA 证书文件的路径名。需要此文件来验证服务器证书。

⑤ 此字段指定加密的 TLS 隧道中使用的身份验证方法。在此示例中，使用具有 MD5-Challenge 的 EAP。“内部身份验证”阶段通常称为“阶段 2”。

接下来，将以下行添加到 **/etc/rc.conf** 中：

```
wlans_ath0="wlan0"
ifconfig_wlan0="WPA DHCP"
```

下一步是调出接口：

[2029] https://www.freebsd.org/cgi/man.cgi?query=wpa_supplicant&sektion=8&format=html

[2030] https://www.freebsd.org/cgi/man.cgi?query=ifconfig&sektion=8&format=html

```
# 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
```

34.4.3.使用 EAP-PEAP 的 WPA

> 注意
> PEAPv0/EAP-MSCHAPv2 是最常见的 PEAP 方法。在本章中，术语 PEAP 用于指代该方法。

受保护的 EAP（PEAP）设计为 EAP-TTLS 的替代方案，是继 EAP-TLS 之后最常用的 EAP 标准。在具有混合操作系统的网络中，PEAP 应该是仅次于 EAP-TLS 最受支持的标准。

PEAP 类似于 EAP-TTLS，因为它使用服务器端证书通过在客户端和身份验证服务器之间创建加密的 TLS 隧道来对客户端进行身份验证，从而保护随后的身份验证信息交换。PEAP 身份验证与 EAP-TTLS 不同，因为它以明文形式广播用户名，并且仅在加密的 TLS 隧道中发送密码。EAP-TTLS 将对用户名和密码使用 TLS 隧道。

将以下行添加到 **/etc/wpa_supplicant.conf** 以配置 EAP-PEAP 相关设置：

```
network={
  ssid="freebsdap"
  proto=RSN
  key_mgmt=WPA-EAP
  eap=PEAP ①
  identity="test" ②
  password="test" ③
  ca_cert="/etc/certs/cacert.pem" ④
  phase1="peaplabel=0" ⑤
  phase2="auth=MSCHAPV2" ⑥
}
```

① 此字段指定了连接的 EAP 方法。

② 字段包含了加密 TLS 隧道中 EAP 身份验证的标识字符串。

③ 字段包含 EAP 身份验证的通行短语。

④ 字段指示了 CA 证书文件的路径名。需要此文件来验证服务器证书。

⑤ 此字段包含了身份验证的第一阶段（TLS 隧道）的参数。根据所使用的身份验证服务器，指定用于身份验证的特定标签。大多数情况下，标签将是“客户端 EAP 加密”，这是通过使用 `peaplabel=0` 设置的。更多信息可以在 wpa_supplicant.conf(5)[2031] 中找到。

⑥ 此字段指定了加密的 TLS 隧道中使用的身份验证协议。在 PEAP 的情况下，它是 `auth=MSCHAPV2`。

将以下内容添加到 **/etc/rc.conf**：

```
wlans_ath0="wlan0"
ifconfig_wlan0="WPA DHCP"
```

然后，调出界面：

```
# 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
```

[2031] https://www.freebsd.org/cgi/man.cgi?query=wpa_supplicant.conf&sektion=5&format=html

34.5.无线点对点模式

IBSS 模式，也称为点对点模式，专为点对点连接而设计。例如，要在计算机 A 和 B 之间建立 ad hoc 网络，请选择两个 IP 地址和一个 SSID。

在计算机 A 上：

```
# 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
```

adhoc 参数指示接口正在 IBSS 模式下运行。

B 现在应该能够检测到 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
```

输出中的 I 确认 A 处于 ad-hoc 模式。现在，用一个不同的 IP 地址配置 B：

```
# 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
```

A 和 B 两者现在都已准备好交换信息。

34.5.1.FreeBSD 主机接入点

FreeBSD 可以充当接入点（AP），从而无需购买硬件 AP 或运行 ad-hoc 网络。当 FreeBSD 机器充当另一个网络（如互联网）的网关时，这可能特别有用。

34.5.1.1.基本设置

在将 FreeBSD 机器配置为 AP 之前，内核必须为无线网卡配置适当的网络支持以及正在使用的安全协议。有关更多详细信息，请参阅基本设置[2032]。

> 注意
>
> Windows® 驱动程序的 NDIS 驱动程序 wrapper 当前不支持 AP 操作。只有原生的 FreeBSD 无线驱动程序支持 AP 模式。

加载无线网络支持后，检查无线设备是否支持基于主机的接入点模式（也称为主机 AP 模式）：

```
# ifconfig wlan0 create wlandev ath0
# ifconfig wlan0 list caps
drivercaps=6f85edc1<STA,FF,TURBOP,IBSS,HOSTAP,AHDEMO,TXPMGT,SHSLOT,SHPREAMBLE,MONITOR,
↪MBSS,WPA1,WPA2,BURST,WME,WDS,BGSCAN,TXFRAG>
cryptocaps=1f<WEP,TKIP,AES,AES_CCM,TKIPMIC>
```

此输出显示卡的功能。HOSTAP 确认此无线网卡可以充当 AP。还列出了各种支持的密码：WEP、TKIP 和 AES。此信息指示可以在 AP 上使用哪些安全协议。

在创建网络伪设备期间，无线设备只能进入 hostap 模式，因此必须先销毁以前创建的设备：

```
# ifconfig wlan0 destroy
```

然后在设置其他参数之前使用正确的选项重新生成：

```
# 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
```

再次使用 ifconfig(8)[2033] 查看 **wlan0** 接口的状态：

```
# 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
```

(continues on next page)

[2032] https://docs.freebsd.org/en/books/handbook/advanced-networking/#network-wireless-basic

[2033] https://www.freebsd.org/cgi/man.cgi?query=ifconfig&sektion=8&format=html

(continued from previous page)

```
        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
```

hostap参数指示接口正在基于主机的接入点模式下运行。

接口配置可以在引导时自动完成，方法是将以下行添加到 **/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"
```

34.5.1.2.基于主机的接入点，无需认证或加密

虽然不建议在没有任何身份验证或加密的情况下运行 AP，但这是检查 AP 是否正常工作的简单方法。此配置对于调试客户端问题也很重要。

配置 AP 后，从另一台无线计算机启动扫描以查找 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
```

客户端计算机找到了 AP，并且可以与之关联：

```
# 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
```

34.5.1.3.WPA2 基于主机的接入点

本节重点介绍如何使用 WPA2 安全协议设置 FreeBSD 接入点。有关 WPA 和基于 WPA 的无线客户端的配置的更多详细信息，请参阅 WPA[2034]。

hostapd(8)[2035] 守护程序用于处理启用了 WPA2 的 AP 上的客户端身份验证和密钥管理。

以下配置操作是在充当 AP 的 FreeBSD 机器上执行的。一旦 AP 正常工作，hostapd(8)[2036] 可以在启动时自动启动，在 **/etc/rc.conf** 中使用以下行：

```
hostapd_enable="YES"
```

在尝试配置 hostapd(8)[2037] 之前，首先配置基本设置中引入的基本设置[2038]。

34.5.1.3.1.WPA2-PSK

WPA2-PSK 适用于无法或不需要使用后端身份验证服务器的小型网络。

配置在 **/etc/hostapd.conf** 中完成：

```
interface=wlan0 ①
debug=1 ②
ctrl_interface=/var/run/hostapd ③
ctrl_interface_group=wheel ④
ssid=freebsdap ⑤
wpa=2   ⑥
wpa_passphrase=freebsdmall ⑦
wpa_key_mgmt=WPA-PSK ⑧
wpa_pairwise=CCMP ⑨
```

① 用于接入点的无线接口。

② 在 hostapd(8)[2039] 执行期间使用的详细级别。值为 1 表示最小级别。

③hostapd(8)[2040] 用于存储域套接字文件的目录的路径名，以便与外部程序（如hostapd_cli(8)[2041]）进行通信。此示例中使用默认值。

④ 允许访问控制接口文件的组。

⑤ 将显示在无线扫描中的无线网络名称或 SSID。

[2034] https://docs.freebsd.org/en/books/handbook/advanced-networking/#network-wireless-wpa
[2035] https://www.freebsd.org/cgi/man.cgi?query=hostapd&sektion=8&format=html
[2036] https://www.freebsd.org/cgi/man.cgi?query=hostapd&sektion=8&format=html
[2037] https://www.freebsd.org/cgi/man.cgi?query=hostapd&sektion=8&format=html
[2038] https://docs.freebsd.org/en/books/handbook/advanced-networking/#network-wireless-ap-basic
[2039] https://www.freebsd.org/cgi/man.cgi?query=hostapd&sektion=8&format=html
[2040] https://www.freebsd.org/cgi/man.cgi?query=hostapd&sektion=8&format=html
[2041] https://www.freebsd.org/cgi/man.cgi?query=hostapd_cli&sektion=8&format=html

⑥ 启用 WPA 并指定需要哪种 WPA 认证协议。值为 2 时，建议将 AP 配置为 WPA2。只有在需要过时的 WPA 时才设置为 1。

⑦ 用于 WPA 身份验证的 ASCII 密码。

⑧ 要使用的密钥管理协议。此示例设置 WPA-PSK。

⑨ 接入点接受的加密算法。在此示例中，仅接受 CCMP（AES）密码。CCMP 是 TKIP 的替代方案，在可能的情况下是强烈建议的。只有当存在无法使用 CCMP 的站点时，才应允许 TKIP。

下一步是启动 hostapd(8)[2042]：

```
# service hostapd forcestart
```

```
# ifconfig wlan0
wlan0: flags=8943<UP,BROADCAST,RUNNING,PROMISC,SIMPLEX,MULTICAST> metric 0 mtu 1500
    ether 04:f0:21:16:8e:10
    inet6 fe80::6f0:21ff:fe16:8e10%wlan0 prefixlen 64 scopeid 0x9
    nd6 options=21<PERFORMNUD,AUTO_LINKLOCAL>
    media: IEEE 802.11 Wireless Ethernet autoselect mode 11na <hostap>
    status: running
    ssid No5ignal channel 36 (5180 MHz 11a ht/40+) bssid 04:f0:21:16:8e:10
    country US ecm authmode WPA2/802.11i privacy MIXED deftxkey 2
    AES-CCM 2:128-bit AES-CCM 3:128-bit txpower 17 mcastrate 6 mgmtrate 6
    scanvalid 60 ampdulimit 64k ampdudensity 8 shortgi wme burst
    dtimperiod 1 -dfs
    groups: wlan
```

AP 运行后，客户端可以与其关联。有关更多详细信息，请参阅 WPA[2043]。可以使用 `ifconfig wlan0 list sta` 查看与 AP 关联的站。

34.6.USB 网络共享

许多手机都提供了通过 USB 共享其数据连接（通常称为 “网络共享”）的选项。此功能使用 RNDIS、CDC 或 Apple® iPhone®/iPad® 自定义协议之一。

- Android™ 设备通常使用驱动程序 urndis(4)[2044]。
- Apple® 设备使用驱动程序 ipheth(4)[2045]。
- 较旧的设备通常使用驱动程序 cdce(4)[2046]。

[2042] https://www.freebsd.org/cgi/man.cgi?query=hostapd&sektion=8&format=html
[2043] https://docs.freebsd.org/en/books/handbook/advanced-networking/#network-wireless-wpa
[2044] https://www.freebsd.org/cgi/man.cgi?query=urndis&sektion=4&format=html
[2045] https://www.freebsd.org/cgi/man.cgi?query=ipheth&sektion=4&format=html
[2046] https://www.freebsd.org/cgi/man.cgi?query=cdce&sektion=4&format=html

在连接设备之前，请将相应的驱动程序加载到内核中：

```
# kldload if_urndis
# kldload if_cdce
# kldload if_ipheth
```

连接设备后，`ue` *0* 将像普通网络设备一样可供使用。确保在设备上启用了“USB 网络共享”选项。

要使此更改永久化并在引导时将驱动程序作为模块加载，请在 **/boot/loader.conf** 中放置以下相应行：

```
if_urndis_load="YES"
if_cdce_load="YES"
if_ipheth_load="YES"
```

34.7.蓝牙

蓝牙是一种无线技术，用于创建在 2.4 GHz 无授权频段运行的个人网络，范围为 10 米。网络通常由便携式设备（如移动电话、手持设备和笔记本电脑）临时形成。与 Wi-Fi 无线技术不同，蓝牙提供更高级别的服务配置文件，例如类似 FTP 的文件服务器、文件推送、语音传输、串行线路仿真等。

本节介绍如何在 FreeBSD 系统上使用 USB 蓝牙加密狗。然后，它涉及了各种蓝牙协议和实用程序。

34.7.1.加载蓝牙支持

FreeBSD 中的蓝牙堆栈是使用 netgraph(4)[2047] 框架实现的。ng_ubt(4)[2048] 支持多种蓝牙 USB 加密狗。基于 Broadcom BCM2033 的蓝牙设备受 ubtbcmfw(4)[2049] 和 ng_ubt(4)[2050] 驱动程序的支持。3Com 蓝牙 PC 卡 3CRWB60-A 受 ng_bt3c(4)[2051] 驱动程序支持。串行和基于 UART 的蓝牙设备受 sio(4)[2052]、ng_h4(4)[2053] 和 hcseriald(8)[2054] 的支持。

在连接设备之前，请确定它使用上述驱动程序，然后加载驱动程序。例如，如果设备使用 ng_ubt(4)[2055] 驱动程序：

```
# kldload ng_ubt
```

[2047] https://www.freebsd.org/cgi/man.cgi?query=netgraph&sektion=4&format=html
[2048] https://www.freebsd.org/cgi/man.cgi?query=ng_ubt&sektion=4&format=html
[2049] https://www.freebsd.org/cgi/man.cgi?query=ubtbcmfw&sektion=4&format=html
[2050] https://www.freebsd.org/cgi/man.cgi?query=ng_ubt&sektion=4&format=html
[2051] https://www.freebsd.org/cgi/man.cgi?query=ng_bt3c&sektion=4&format=html
[2052] https://www.freebsd.org/cgi/man.cgi?query=sio&sektion=4&format=html
[2053] https://www.freebsd.org/cgi/man.cgi?query=ng_h4&sektion=4&format=html
[2054] https://www.freebsd.org/cgi/man.cgi?query=hcseriald&sektion=8&format=html
[2055] https://www.freebsd.org/cgi/man.cgi?query=ng_ubt&sektion=4&format=html

如果在系统启动期间将蓝牙设备连接到系统，则可以通过将驱动程序添加到 **/boot/loader.conf**，将系统配置为在引导时加载模块：

```
ng_ubt_load="YES"
```

加载驱动程序后，插入 USB 蓝牙适配器。如果驱动程序加载成功，则类似于以下内容的输出应显示在控制台和 **/var/log/messages** 中：

```
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
```

要启动和停止蓝牙堆栈，请使用其启动脚本。最好在拔下设备之前停止堆栈。启动蓝牙堆栈可能需要启动 hcsecd(8)[2056]。启动堆栈时，输出应类似于以下内容：

```
# 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
```

34.7.2.查找其他蓝牙设备

主机控制器接口（HCI）为访问蓝牙基带功能提供了一种统一的方法。在 FreeBSD 中，为每个蓝牙设备创建一个 netgraph HCI 节点。有关更多详细信息，请参阅 ng_hci(4)[2057]。

最常见的任务之一是发现射频邻近范围内的蓝牙设备。此操作称为 *轮询*。轮询和其他与 HCI 相关的操作均使用 hccontrol(8)[2058] 完成。以下示例显示了如何找出哪些蓝牙设备在范围内。设备列表应在几秒钟内显示。请注意，远程设备只有在设置为 *可发现* 模式时才会回答查询。

[2056] https://www.freebsd.org/cgi/man.cgi?query=hcsecd&sektion=8&format=html
[2057] https://www.freebsd.org/cgi/man.cgi?query=ng_hci&sektion=4&format=html
[2058] https://www.freebsd.org/cgi/man.cgi?query=hccontrol&sektion=8&format=html

```
% 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]
```

BD_ADDR 是蓝牙设备的唯一地址，类似于网卡的 MAC 地址。该地址用于与设备的进一步通信，可以为 BD_ADDR 分配一个人类可读的名称。关于已知蓝牙主机的信息包含在 **/etc/bluetooth/hosts** 中。下面的示例演示如何获取分配给远程设备的人类可读名称：

```
% hccontrol -n ubt0hci remote_name_request 00:80:37:29:19:a4
BD_ADDR: 00:80:37:29:19:a4
Name: Pav's T39
```

如果在远程蓝牙设备上执行查询，则会发现计算机为 your.host.name (ubt0)。分配给本地设备的名称可以随时更改。

可以在 **/etc/bluetooth/hosts** 中为远程设备分配别名。有关 **/etc/bluetooth/hosts** 文件的更多信息，可以在 bluetooth.hosts(5)[2059] 中找到。

蓝牙系统在两个蓝牙设备之间提供点对点连接，或在多个蓝牙设备之间共享点对多点连接。下面的示例演示如何创建与远程设备的连接：

```
% hccontrol -n ubt0hci create_connection BT_ADDR
```

create_connection 接受 BT_ADDR 以及 /etc/bluetooth/hosts 中的主机别名

下面的示例演示如何获取本地设备的活动基带连接列表：

```
% 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
```

当需要终止基带连接时，连接柄很有用，尽管通常不需要手动执行此操作。堆栈将自动终止非活动基带连接。

[2059] https://www.freebsd.org/cgi/man.cgi?query=bluetooth.hosts&sektion=5&format=html

```
# hccontrol -n ubt0hci disconnect 41
Connection handle: 41
Reason: Connection terminated by local host [0x16]
```

键入 `hccontrol help` 以获得可用 HCI 命令的完整清单。大多数 HCI 命令不需要超级用户权限。

34.7.3.设备配对

默认情况下，蓝牙通信是不需要认证的，任何设备都可以与任何其他设备通信。蓝牙设备，如手机，可以选择要求认证以提供特定服务。蓝牙认证通常是通过 PIN 码完成的，这是一个长度不超过 16 个字符的 ASCII 字符串。用户需要在两个设备上输入相同的 *PIN 码*。一旦用户输入了 PIN 码，两台设备将生成一个 *链接密钥*。之后，链接密钥可以存储在设备中或持久性存储中。下一次，两台设备将使用先前生成的链接密钥。这个过程被称为 *配对*。注意，如果任何一个设备丢失了链路密钥，必须重新配对。

hcsecd(8)[2060] 守护程序负责处理蓝牙认证请求。默认的配置文件是 **/etc/bluetooth/hcsecd.conf**。下面是一个手机的例子部分，PIN 码设置为 `1234`：

```
device {
        bdaddr  00:80:37:29:19:a4;
        name    "Pav's T39";
        key     nokey;
        pin     "1234";
      }
```

对 PIN 码的唯一限制是长度。一些设备，如蓝牙耳机，可能有一个固定的 PIN 码。`-d` 开关强制 hcsecd(8)[2061] 停留在前台，所以很容易看到正在发生的事情。将远程设备设置为接收配对，并启动与远程设备的蓝牙连接。远程设备应该表示接受配对并要求输入 PIN 码。输入 **hcsecd.conf** 中列出的相同 PIN 码。现在计算机和远程设备已经配对。另外，也可以在远程设备上启动配对。

以下行可以添加到 **/etc/rc.conf** 中，以将 hcsecd(8)[2062] 配置为在系统启动时自动启动：

```
hcsecd_enable="YES"
```

以下是 hcsecd(8)[2063] 守护程序输出的示例：

```
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',
```

(continues on next page)

[2060] https://www.freebsd.org/cgi/man.cgi?query=hcsecd&sektion=8&format=html
[2061] https://www.freebsd.org/cgi/man.cgi?query=hcsecd&sektion=8&format=html
[2062] https://www.freebsd.org/cgi/man.cgi?query=hcsecd&sektion=8&format=html
[2063] https://www.freebsd.org/cgi/man.cgi?query=hcsecd&sektion=8&format=html

(continued from previous page)

```
↪ link key doesn't exist
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 exists
hcsecd[16484]: Sending PIN_Code_Reply to 'ubt0hci' for remote bdaddr 0:80:37:29:19:a4
```

34.7.4.使用 PPP 配置文件进行网络访问

拨号网络（DUN）配置文件可用于将移动电话配置为无线调制解调器，以便连接到拨号互联网访问服务器。它还可用于配置计算机以接收来自移动电话的数据呼叫。

具有 PPP 配置文件的网络访问可用于为单个蓝牙设备或多个蓝牙设备提供 LAN 访问。它还可以通过串行电缆仿真使用 PPP 网络提供 PC 到 PC 连接。

在 FreeBSD 中，这些配置文件是用 ppp(8)[2064] 和 rfcomm_pppd(8)[2065] 包装器实现的，它把蓝牙连接转换成 PPP 可以使用的东西。在使用配置文件之前，必须在 **/etc/ppp/ppp.conf** 中创建新的 PPP 标签。有关示例，请参考 rfcomm_pppd(8)[2066]。

在本例中，rfcomm_pppd(8)[2067] 用于在 DUNRFCOMM 通道上以 `00:80:37:29:19:a4` 的 `BD_ADDR` 打开到远端设备的连接:

```
# rfcomm_pppd -a 00:80:37:29:19:a4 -c -C dun -l rfcomm-dialup
```

实际的通道号将使用 SDP 协议从远程设备获取。可以手动指定 RFCOMM 信道，在这种情况下，rfcomm_pppd(8)[2068] 不会执行 SDP 查询。使用 sdpcontrol(8)[2069] 找出远程设备上的 RFCOMM 信道。

为了通过 PPPLAN 服务提供网络访问，sdpd(8)[2070] 必须正在运行，并且必须在 **/etc/ppp/ppp.conf** 中为 LAN 客户端创建一个新条目。有关示例，请参考 rfcomm_pppd(8)[2071]。最后，在有效的 RFCOMM 信道号上启动 RFCOMMPPP 服务器。RFCOMMPPP 服务器将自动向本地 SDP 守护程序注册蓝牙 LAN 服务。下面的示例显示了如何启动 RFCOMMPPP 服务器。

```
# rfcomm_pppd -s -C 7 -l rfcomm-server
```

[2064] https://www.freebsd.org/cgi/man.cgi?query=ppp&sektion=8&format=html
[2065] https://www.freebsd.org/cgi/man.cgi?query=rfcomm_pppd&sektion=8&format=html
[2066] https://www.freebsd.org/cgi/man.cgi?query=rfcomm_pppd&sektion=8&format=html
[2067] https://www.freebsd.org/cgi/man.cgi?query=rfcomm_pppd&sektion=8&format=html
[2068] https://www.freebsd.org/cgi/man.cgi?query=rfcomm_pppd&sektion=8&format=html
[2069] https://www.freebsd.org/cgi/man.cgi?query=sdpcontrol&sektion=8&format=html
[2070] https://www.freebsd.org/cgi/man.cgi?query=sdpd&sektion=8&format=html
[2071] https://www.freebsd.org/cgi/man.cgi?query=rfcomm_pppd&sektion=8&format=html

34.7.5.蓝牙协议

本节概述了各种蓝牙协议、其功能和相关实用程序。

34.7.5.1.逻辑链路控制和适配协议（L2CAP）

逻辑链路控制和适配协议（L2CAP）为上层协议提供面向连接和无连接的数据服务。L2CAP 允许更高级别的协议和应用程序发送和接收长度高达 64 KB 的 L2CAP 数据包。

L2CAP 基于 *信道* 的概念。信道是基带连接之上的逻辑连接，其中每个信道都以多对一的方式绑定到单个协议。多个信道可以绑定到同一协议，但一个信道不能绑定到多个协议。在信道上接收的每个 L2CAP 数据包都定向到相应的更高级别的协议。多个信道可以共享同一基带连接。

在 FreeBSD 中，为每个蓝牙设备创建一个 netgraph L2CAP 节点。此节点通常连接到下游蓝牙 HCI 节点和上游蓝牙插座节点。L2CAP 节点的默认名称是 `devicel2cap`。有关更多详细信息，请参阅 ng_l2cap(4)[2072]。

一个有用的命令是 l2ping(8)[2073]，它可以用来 ping 其他设备。一些蓝牙实现可能不会返回所有发送给它们的数据，所以下面例子中的 `0 bytes` 是正常的。

```
# 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
```

l2control(8)[2074] 实用程序用于在 L2CAP 节点上执行各种操作。此示例演示如何获取本地设备的逻辑连接（信道）列表和基带连接列表：

```
% 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
```

另一个诊断工具是 btsockstat(1)[2075]。它类似于 netstat(1)[2076]，但适用于与蓝牙网络相关的数据结构。下面的示例显示了与上面的 l2control(8)[2077] 相同的逻辑连接。

[2072] https://www.freebsd.org/cgi/man.cgi?query=ng_l2cap&sektion=4&format=html
[2073] https://www.freebsd.org/cgi/man.cgi?query=l2ping&sektion=8&format=html
[2074] https://www.freebsd.org/cgi/man.cgi?query=l2control&sektion=8&format=html
[2075] https://www.freebsd.org/cgi/man.cgi?query=btsockstat&sektion=1&format=html
[2076] https://www.freebsd.org/cgi/man.cgi?query=netstat&sektion=1&format=html
[2077] https://www.freebsd.org/cgi/man.cgi?query=l2control&sektion=8&format=html

```
% 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
```

34.7.5.2.射频通讯（RFCOMM）

RFCOMM 协议通过 L2CAP 协议提供串行端口的仿真。RFCOMM 是一种简单的传输协议，具有用于模拟 RS-232（EIATIA-232-E）串行端口的 9 个电路的附加规定。它支持两个蓝牙设备之间的多达 60 个同步连接（RFCOMM 信道）。

出于 RFCOMM 的目的，完整的通信路径涉及在通信端点上运行的两个应用程序，它们之间有一个通信段。RFCOMM 旨在涵盖使用它们所在的设备的串行端口的应用。通信段是从一个设备到另一个设备的直接连接蓝牙链路。

RFCOMM 仅涉及直接连接情况下的设备之间的连接，或网络情况下的设备与调制解调器之间的连接。RFCOMM 可以支持其他配置，例如一端通过蓝牙无线技术进行通信并在另一端提供有线接口的模块。

在 FreeBSD 中，RFCOMM 是在蓝牙套接字层实现的。

34.7.5.3.服务发现协议（SDP）

服务发现协议（SDP）为客户端应用程序提供了发现服务器应用程序提供的服务的存在以及这些服务的属性的方法。服务的属性包括所提供服务的类型或类别，以及利用服务所需的机制或协议信息。

SDP 涉及 SDP 服务器和 SDP 客户端之间的通信。服务器维护一个服务记录列表，这些记录描述与服务器关联的服务的特征。每个服务记录都包含有关单个服务的信息。客户端可以通过发出 SDP 请求从 SDP 服务器维护的服务记录中检索信息。如果客户端或与客户端关联的应用程序决定使用某项服务，则它必须打开与服务提供程序的单独连接才能使用该服务。SDP 提供了一种用于发现服务及其属性的机制，但它不提供用于利用这些服务的机制。

通常，SDP 客户端会根据服务的某些所需特征搜索服务。但是，有时需要发现 SDP 服务器的服务记录描述了哪些类型的服务，而无需有关服务的任何先前信息。查找任何提供的服务的过程称为 浏览。

蓝牙 SDP 服务器 sdpd(8)[2078] 和命令行客户端 sdpcontrol(8)[2079] 都包含在标准的 FreeBSD 安装中。下面的示例演示如何执行 SDP 浏览查询。

[2078] https://www.freebsd.org/cgi/man.cgi?query=sdpd&sektion=8&format=html

[2079] https://www.freebsd.org/cgi/man.cgi?query=sdpcontrol&sektion=8&format=html

```
% 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
```

请注意，每个服务都有一个属性列表，例如 RFCOMM 信道。根据服务的不同，用户可能需要记下某些属性。某些蓝牙实现不支持服务浏览，可能会返回空列表。在这种情况下，可以搜索特定服务。下面的示例显示了如何搜索 OBEX 对象推送（OPUSH）服务：

```
% sdpcontrol -a 00:01:03:fc:6e:ec search OPUSH
```

在 FreeBSD 上向蓝牙客户端提供服务是通过 sdpd(8)[2080] 服务器完成的。以下行可以添加到 **/etc/rc.conf** 中：

```
sdpd_enable="YES"
```

然后 sdpd(8)[2081] 守护程序可以启动：

```
# service sdpd start
```

要向远程客户端提供蓝牙服务的本地服务器应用程序将向本地 SDP 守护程序注册该服务。此类应用程序的一个例子是 rfcomm_pppd(8)[2082]。启动后，它将向本地 SDP 守护程序注册蓝牙 LAN 服务。

[2080] https://www.freebsd.org/cgi/man.cgi?query=sdpd&sektion=8&format=html

[2081] https://www.freebsd.org/cgi/man.cgi?query=sdpd&sektion=8&format=html

[2082] https://www.freebsd.org/cgi/man.cgi?query=rfcomm_pppd&sektion=8&format=html

通过本地控制信道发出 SDP 浏览查询，可以获得在本地 SDP 服务器注册的服务列表：

```
# sdpcontrol -l browse
```

34.7.5.4.OBEX 对象推送（OPUSH）

对象交换（OBEX）是一种广泛使用的协议，用于移动设备之间的简单文件传输。它的主要用途是红外通信，用于笔记本电脑或 PDA 之间的通用文件传输，以及用于在手机和其他具有个人信息管理器（PIM）应用程序的设备之间发送名片或日历条目。

OBEX 服务器和客户端由 obexapp 实现，可以使用软件包或 port 来安装 comms/obexapp[2083]。

OBE X 客户端被用来从 OBEX 服务器推送和/或拉取对象。例如是一张名片或一个约会的对象。OBEX 客户端可以通过 SDP 从远程设备获得 RFCOMM 频道号码。这可以通过指定服务名称而不是 RFCOMM 信道号码来实现。支持的服务名称是：IrMC、FTRN 和 OPUSH。也可以把 RFCOMM 信道指定为一个数字。下面是一个 OBEX 会话的示例，其中从移动电话中提取设备信息对象，并将一个新对象(名片)推入电话的目录。

```
% 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)
```

为了提供 OPUSH 服务，sdpd(8)[2084] 必须正在运行，并且必须创建一个根文件夹，其中将存储所有传入的对象。根文件夹的默认路径是 **/var/sline/obex**。最后，在有效的 RFCOMM 信道号上启动 OBEX 服务器。OBEX 服务器将自动向本地 SDP 守护程序注册 OPUSH 服务。下面的示例显示了如何启动 OBEX 服务器。

```
# obexapp -s -C 10
```

34.7.5.5.串行端口配置文件（SPP）

串行端口配置文件（SPP）允许蓝牙设备执行串行电缆仿真。此配置文件允许传统应用程序通过虚拟串行端口抽象使用蓝牙作为电缆替代品。

在 FreeBSD 中，rfcomm_sppd(1)[2085] 实现了 SPP，伪 tty 被用作虚拟串行端口抽象。以下示例显示了如何连接到远程设备的串行端口服务。RFCOMM 信道不必指定，因为 rfcomm_sppd(1)[2086] 可以通过 SDP 从远程设备获取它。若要重写此值，请在命令行上指定 RFCOMM 信道。

[2083] https://cgit.freebsd.org/ports/tree/comms/obexapp/pkg-descr
[2084] https://www.freebsd.org/cgi/man.cgi?query=sdpd&sektion=8&format=html
[2085] https://www.freebsd.org/cgi/man.cgi?query=rfcomm_sppd&sektion=1&format=html
[2086] https://www.freebsd.org/cgi/man.cgi?query=rfcomm_sppd&sektion=1&format=html

```
# rfcomm_sppd -a 00:07:E0:00:0B:CA -t
rfcomm_sppd[94692]: Starting on /dev/pts/6...
/dev/pts/6
```

连接后，伪 tty 可用作串行端口：

```
# cu -l /dev/pts/6
```

伪 tty 打印在 stdout 上，可以通过包装脚本读取：

```
PTS=`rfcomm_sppd -a 00:07:E0:00:0B:CA -t`
cu -l $PTS
```

34.7.6.故障排除

默认情况下，当 FreeBSD 接受新连接时，它会尝试执行角色切换并成为主节点。某些不支持角色切换的旧版蓝牙设备将无法连接。由于角色切换是在建立新连接时执行的，因此无法询问远程设备是否支持角色切换。但是，有一个 HCI 选项用于在本地端禁用角色切换：

```
# hccontrol -n ubt0hci write_node_role_switch 0
```

要显示蓝牙数据包，请使用第三方软件包 hcidump，该软件包可以使用软件包或 port 来安装 comms/hcidump[2087]。此实用程序类似于 tcpdump(1)[2088]，可用于在终端上显示蓝牙数据包的内容，并将蓝牙数据包转储到文件中。

34.8.桥接

有时将网络（如以太网段）划分为网段很有用，而无需创建 IP 子网并使用路由器将这些网段连接在一起。以这种方式将两个网络连接在一起的设备称为“网桥”。

网桥的工作原理是学习其每个网络接口上设备的 MAC 地址。仅当源和目标 MAC 地址位于不同的网络上时，它才会在网络之间转发流量。在许多方面，网桥就像一个端口很少的以太网交换机。具有多个网络接口的 FreeBSD 系统可以被配置为充当网桥。

桥接在以下情况下可能很有用：

- **连接网络**

  网桥的基本操作是连接两个或多个网段。使用基于主机的网桥而不是网络设备的原因有很多，例如布线约束或防火墙。网桥还可以将以 hostap 模式运行的无线接口连接到有线网络，并充当接入点。

[2087] https://cgit.freebsd.org/ports/tree/comms/hcidump/pkg-descr

[2088] https://www.freebsd.org/cgi/man.cgi?query=tcpdump&sektion=1&format=html

- **过滤/流量整形防火墙**

  当需要防火墙功能时，可以使用网桥，而无需路由或网络地址转换（NAT）。例如，通过 DSL 或 ISDN 连接到 ISP 的小公司。网络上有来自 ISP 的 13 个公共 IP 地址和 10 台计算机。在这种情况下，由于子网划分问题，使用基于路由器的防火墙很困难。可以配置基于网桥的防火墙，而不会出现任何 IP 寻址问题。

- **网络分路器**

  网桥可以连接两个网段，以便在网桥接口上使用 bpf(4)[2089] 和 tcpdump(1)[2090] 检查在网桥之间传递的所有以太网帧，或者通过发送所有帧的副本输出一个称为 span 端口的附加接口。

- **2 层 VPN**

  通过将网络桥接到 EtherIP 隧道或基于 tap(4)[2091] 的解决方案（如 OpenVPN），可以跨 IP 链路连接两个以太网。

- **2 层冗余**

  网络可以通过多个链路连接在一起，并使用生成树协议（STP）来阻止冗余路径。

本节说明如何使用 if_bridge(4)[2092] 将 FreeBSD 系统配置为网桥。netgraph 桥接驱动程序也是可用的，在 ng_bridge(4)[2093] 中描述。

> 注意
>
> 数据包过滤可以与任何挂接到 pfil(9)[2094] 框架中的防火墙软件包一起使用。该桥可以用作具有 altq(4)[2095] 或 dummynet（4）[2096] 的流量调整器。

34.8.1.启用网桥

在 FreeBSD 中，if_bridge(4)[2097] 是一个内核模块，在创建桥接接口时由 ifconfig(8)[2098] 自动加载。也可以通过在定制的内核配置文件中加入 `device if_bridge` 来将桥接支持编译到定制的内核中。

网桥是使用接口克隆创建的。要创建网桥接口：

```
# ifconfig bridge create
bridge0
# ifconfig bridge0
bridge0: flags=8802<BROADCAST,SIMPLEX,MULTICAST> metric 0 mtu 1500
```

(continues on next page)

[2089] https://www.freebsd.org/cgi/man.cgi?query=bpf&sektion=4&format=html
[2090] https://www.freebsd.org/cgi/man.cgi?query=tcpdump&sektion=1&format=html
[2091] https://www.freebsd.org/cgi/man.cgi?query=tap&sektion=4&format=html
[2092] https://www.freebsd.org/cgi/man.cgi?query=if_bridge&sektion=4&format=html
[2093] https://www.freebsd.org/cgi/man.cgi?query=ng_bridge&sektion=4&format=html
[2094] https://www.freebsd.org/cgi/man.cgi?query=pfil&sektion=9&format=html
[2095] https://www.freebsd.org/cgi/man.cgi?query=altq&sektion=4&format=html
[2096] https://www.freebsd.org/cgi/man.cgi?query=dummynet&sektion=4&format=html
[2097] https://www.freebsd.org/cgi/man.cgi?query=if_bridge&sektion=4&format=html
[2098] https://www.freebsd.org/cgi/man.cgi?query=ifconfig&sektion=8&format=html

(continued from previous page)

```
        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
```

创建网桥接口时，会自动分配一个随机生成的以太网地址。`maxaddr` 和 `timeout` 参数控制了网桥在其转发表中保留多少个 MAC 地址，以及每个条目在最后一次被看到后有多少秒会被删除。其他参数控制 STP 的运行方式。

接下来，指定要添加为网桥成员的网络接口。要使网桥转发数据包，所有成员接口和网桥都需要处于打开状态：

```
# ifconfig bridge0 addm fxp0 addm fxp1 up
# ifconfig fxp0 up
# ifconfig fxp1 up
```

网桥现在可以在 **fxp0** 和 **fxp1** 之间转发以太网帧。将以下行添加到 **/etc/rc.conf** 中，以便在启动时创建网桥：

```
cloned_interfaces="bridge0"
ifconfig_bridge0="addm fxp0 addm fxp1 up"
ifconfig_fxp0="up"
ifconfig_fxp1="up"
```

如果网桥主机需要 IP 地址，请在网桥接口上设置该地址，而不是在成员接口上设置该地址。该地址可以静态设置，也可以通过 DHCP 设置。此示例设置静态 IP 地址：

```
# ifconfig bridge0 inet 192.168.0.1/24
```

也可以将 IPv6 地址分配给网桥接口。要使更改永久化，请将寻址信息添加到 **/etc/rc.conf**。

> 注意
>
> 启用数据包筛选后，桥接数据包将通过网桥接口上原始接口上的筛选器入站，并在相应的接口上通过出站。任何一个阶段都可以禁用。当数据包流的方向很重要时，最好在成员接口上设置防火墙，而不是网桥本身。
>
> 该网桥具有多个可配置的设置，用于传递非 IP 和 IP 数据包，以及使用 ipfw(8)[2099] 的第 2 层防火墙。有关详细信息，请参阅 if_bridge(4)[2100]

[2099] https://www.freebsd.org/cgi/man.cgi?query=ipfw&sektion=8&format=html

[2100] https://www.freebsd.org/cgi/man.cgi?query=if_bridge&sektion=4&format=html

34.8.2.启用生成树

要使以太网网络正常工作，两个设备之间只能存在一个活动路径。STP 协议检测环路并将冗余链路置于阻塞状态。如果其中一个活动链路发生故障，STP 将计算不同的树，并启用其中一个被阻止的路径以恢复与网络中所有点的连接。

快速生成树协议（RSTP 或 802.1w）提供与传统 STP 的向后兼容性。RSTP 提供更快的收敛速度，并与相邻交换机交换信息，从而快速过渡到转发模式，而无需创建环路。FreeBSD 支持 RSTP 和 STP 作为操作模式，RSTP 是默认模式。

STP 可以使用 ifconfig(8)[2101] 在成员接口上启用。对于以 **fxp0** 和 **fxp1** 作为当前接口的网桥，请使用以下命令启用 STP：

```
# 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
```

该网桥的生成树 ID 为 `00:01:02:4b:d4:50`，优先级为 `32768`。由于根 ID 是相同的，它表明这是该树的根桥。

网络上的另一个网桥也启用了 STP：

```
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
```

行 `root id 00:01:02:4b:d4:50 priority 32768 ifcost 400000 port 4` 表示根桥为

[2101] https://www.freebsd.org/cgi/man.cgi?query=ifconfig&sektion=8&format=html

00:01:02:4b:d4:50，从该桥到该桥的路径开销为400000。到根桥的路径是通过 port 4，也就是 **fxp0**。

34.8.3.桥接接口参数

有几个 ifconfig 参数是桥接接口所特有的。本节总结了这些参数的一些常见用途。完整的可用参数列表在 ifconfig(8)[2102] 中介绍。

- **private**

  一个专用接口不转发任何流量到任何其他也被指定为专用接口的端口。流量被无条件地阻止，因此没有以太网帧会被转发，包括 ARP 数据包。如果需要有选择地阻断流量，应使用防火墙来代替。

- **span**

  span 端口传输网桥接收到的每个以太网帧的副本。网桥上配置的 span 端口数不受限制，但如果将接口指定为跨度端口，则不能将其用作常规网桥端口。这对于在连接到网桥的跨端口之一的另一台主机上被动侦听桥接网络最有用。例如，要将所有帧的副本发送到名为 **fxp4** 的接口，请执行以下操作：

  ```
  # ifconfig bridge0 span fxp4
  ```

- **sticky**

  如果一个网桥成员接口被标记为粘性，动态学习的地址条目会被视为转发缓存中的静态条目。粘性条目不会从缓存中过期或被替换，即使该地址在不同的接口上被看到。这就提供了静态地址条目的好处，而不需要预先填充转发表。在网桥的一个特定网段上学习的客户不能漫游到另一个网段。

  使用粘性地址的一个例子是将网桥与 VLAN 结合起来，以便在不浪费 IP 地址空间的情况下隔离客户网络。假设 CustomerA 在 vlan100 上，CustomerB 在 vlan101 上，网桥的地址为 192.168.0.1：

  ```
  # ifconfig bridge0 addm vlan100 sticky vlan100 addm vlan101 sticky vlan101
  # ifconfig bridge0 inet 192.168.0.1/24
  ```

在这个例子中，两个客户都把 192.168.0.1 看作他们的默认网关。由于网桥缓存是粘性的，一个主机不能欺骗另一个客户的 MAC 地址，以拦截他们的通信。

VLAN 之间的任何通信都可以用防火墙来阻断，或者如本例中所见，用私有接口来阻断：

```
# ifconfig bridge0 private vlan100 private vlan101
```

客户之间是完全隔离的，并且可以分配完整的 /24 地址范围而不需要划分子网。

一个接口后面的唯一源 MAC 地址的数量可以被限制。一旦达到限制，具有未知源地址的数据包将被丢弃，直到现有的主机缓存条目过期或被删除。

下面的例子将 vlan100 上 CustomerA 的最大以太网设备数设置为 10：

[2102] https://www.freebsd.org/cgi/man.cgi?query=ifconfig&sektion=8&format=html

```
# ifconfig bridge0 ifmaxaddr vlan100 10
```

网桥接口还支持监视模式，在该模式下，数据包在 bpf(4)[2103] 处理后被丢弃，并且不会进一步处理或转发。这可用于将两个或多个接口的输入多路复用到单个 bpf(4)[2104] 流中。这对于重建通过两个单独的接口传输 RX/TX 信号的网络分路器的流量非常有用。例如，要将来自四个网络接口的输入作为一个流读取：

```
# ifconfig bridge0 addm fxp0 addm fxp1 addm fxp2 addm fxp3 monitor up
# tcpdump -i bridge0
```

34.8.4.SNMP 监控

桥接接口和 STP 参数可以通过 bsnmpd(1)[2105] 进行监控，它包含在 FreeBSD 基础系统中。导出的网桥 MIB 符合 IETF 标准，因此任何 SNMP 客户端或监控包都可用于检索数据。

要在网桥上启用监视，请在 **/etc/snmpd.config** 中取消注释此行，方法是删除开头的 # 号：

```
begemotSnmpdModulePath."bridge" = "/usr/lib/snmp_bridge.so"
```

可能需要在此文件中修改其他配置设置，例如社区名称和访问列表。有关更多信息，请参见 bsnmpd(1)[2106] 和 snmp_bridge(3)[2107]。这些编辑被保存后，将此行添加到 **/etc/rc.conf**：

```
bsnmpd_enable="YES"
```

然后，启动 bsnmpd（1）[2108]：

```
# service bsnmpd start
```

以下示例使用 Net-SNMP 软件（net-mgmt/net-snmp[2109]）从客户端系统查询网桥。也可以使用 net-mgmt/bsnmptools[2110] 端口。从运行 Net-SNMP 的 SNMP 客户端中，将以下行添加到 **$HOME/.snmp/snmp.conf** 中，以便导入网桥 MIB 定义：

```
mibdirs +/usr/share/snmp/mibs
mibs +BRIDGE-MIB:RSTP-MIB:BEGEMOT-MIB:BEGEMOT-BRIDGE-MIB
```

要使用 IETF BRIDGE-MIB（RFC4188）监视单个网桥：

[2103] https://www.freebsd.org/cgi/man.cgi?query=bpf&sektion=4&format=html
[2104] https://www.freebsd.org/cgi/man.cgi?query=bpf&sektion=4&format=html
[2105] https://www.freebsd.org/cgi/man.cgi?query=bsnmpd&sektion=1&format=html
[2106] https://www.freebsd.org/cgi/man.cgi?query=bsnmpd&sektion=1&format=html
[2107] https://www.freebsd.org/cgi/man.cgi?query=snmp_bridge&sektion=3&format=html
[2108] https://www.freebsd.org/cgi/man.cgi?query=bsnmpd&sektion=1&format=html
[2109] https://cgit.freebsd.org/ports/tree/net-mgmt/net-snmp/pkg-descr
[2110] https://cgit.freebsd.org/ports/tree/net-mgmt/bsnmptools/pkg-descr

```
% 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)
```

`dot1dStpTopChanges.0` 值为 2，表示 STP 网桥拓扑已更改两次。拓扑更改意味着网络中的一个或多个链路已更改或发生故障，并且已计算出新树。`dot1dStpTimeSinceTopologyChange.0` 值将显示发生这种情况的时间。

要监视多个网桥接口，可以使用专用 BEGEMOT-BRIDGE-MIB：

```
% 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) 0:19:29.27 centi-seconds
BEGEMOT-BRIDGE-MIB::begemotBridgeStpTimeSinceTopologyChange."bridge2" = Timeticks:␣
↪(82773) 0:13:47.73 centi-seconds
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
```

要改变通过 mib-2.dot1dBridge 子树监控的网桥接口：

```
% snmpset -v 2c -c private bridge1.example.com
BEGEMOT-BRIDGE-MIB::begemotBridgeDefaultBridgeIf.0 s bridge2
```

34.9.链路聚合与故障转移

FreeBSD 提供了 lagg(4)[2111] 接口，它可以用来将多个网络接口聚合到一个虚拟接口中，以便提供故障转移和链路聚合。故障转移允许流量继续流动，只要至少有一个聚合网络接口具有已建立的链接。链路聚合在支持 LACP 的交换机上效果最佳，因为此协议在响应单个链路的故障时双向分配流量。

滞后接口支持的聚合协议确定哪些端口用于传出流量以及特定端口是否接受传入流量。lagg(4)[2112] 支持以下协议：

- **failover**

  此模式仅通过主端口发送和接收流量。如果主端口不可用，则使用下一个活动端口。添加到虚拟接口的第一个接口是主端口，随后添加的所有接口都用作故障转移设备。如果故障转移到非主端口，则原始端口在再次变为可用后将成为主端口。

- **loadbalance**

  这提供了静态设置，并且不会与对等帧或交换帧协商聚合以监视链路。如果交换机支持 LACP，则应该用该参数。

- **lacp**

  IEEE® 802.3ad 链路聚合控制协议（LACP）将一组可聚合链路与对等方协商到一个或多个链路聚合组（LAG）中。每个 LAG 由相同速度的端口组成，设置为全双工操作，并且流量在 LAG 中具有最大总速度的端口之间均衡。通常，只有一个 LAG 包含所有端口。如果物理连接发生更改，LACP 将快速收敛到新配置。

  LACP 根据散列协议标头信息均衡活动端口之间的传出流量，并接受来自任何活动端口的传入流量。哈希包括以太网源地址和目标地址，以及 VLAN 标记（如果可用）以及 IPv4 或 IPv6 源地址和目标地址。

- **roundrobin**

  此模式使用轮循机制调度程序在所有活动端口之间分配传出流量，并接受来自任何活动端口的传入流量。由于此模式违反了以太网帧排序，因此应谨慎使用。

- **broadcast**

  此模式将传出流量发送到滞后接口上配置的所有端口，并在任何端口上接收帧。

[2111] https://www.freebsd.org/cgi/man.cgi?query=lagg&sektion=4&format=html
[2112] https://www.freebsd.org/cgi/man.cgi?query=lagg&sektion=4&format=html

34.9.1.配置示例

本节演示如何配置思科 ® 交换机和 FreeBSD 系统以实现 LACP 负载均衡。然后，它演示如何在故障转移模式下配置两个以太网接口，以及如何在以太网和无线接口之间配置故障转移模式。

例 **1.**使用 **Cisco®** 交换机进行 **LACP** 聚合

此示例将 FreeBSD 机器上的两个 fxp(4)[2113] 以太网接口连接到 Cisco® 交换机上的前两个以太网端口，作为单个负载均衡和容错链路。可以添加更多接口以提高吞吐量和容错能力。替换示例中显示的 Cisco® 端口、以太网设备、通道组编号和 IP 地址的名称，以匹配本地配置。

帧排序在以太网链路上是强制性的，两个站之间的任何流量始终流经同一物理链路，从而将最大速度限制为一个接口的速度。传输算法尝试使用尽可能多的信息来区分不同的流量，并在可用接口之间均衡流量。

在 Cisco® 交换机上，将 *FastEthernet0/1* 和 *FastEthernet0/2* 接口添加到通道 *group 1*：

```
interface FastEthernet0/1
channel-group 1 mode active
channel-protocol lacp
!
interface FastEthernet0/2
channel-group 1 mode active
channel-protocol lacp
```

在 FreeBSD 系统上，使用物理接口 *fxp0* 和 *fxp1* 创建 lagg(4)[2114] 接口，并使这些接口的 IP 地址为 *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
```

接下来，验证虚拟接口的状态：

```
# 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 10.0.0.3 netmask 0xffffff00 broadcast 10.0.0.255
       media: Ethernet autoselect
       status: active
       laggproto lacp
       laggport: fxp1 flags=1c<ACTIVE,COLLECTING,DISTRIBUTING>
       laggport: fxp0 flags=1c<ACTIVE,COLLECTING,DISTRIBUTING>
```

[2113] https://www.freebsd.org/cgi/man.cgi?query=fxp&sektion=4&format=html

[2114] https://www.freebsd.org/cgi/man.cgi?query=lagg&sektion=4&format=html

标记为 ACTIVE 的端口是已经与远程交换机协商好的 LAG 的一部分。流量将通过这些活动端口传输和接收。在上述命令中添加 -v，以查看 LAG 标识符。

要查看 Cisco® 交换机上的端口状态：

```
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
```

有关更多详细信息，请键入 show lacp neighbor detail。

要在重新引导时保留此配置，请将以下条目添加到 FreeBSD 系统上的 **/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"
```

例 **2.**故障转移模式

如果主接口上的链路丢失，则故障转移模式可用于切换到辅助接口。要配置故障转移，请确保底层物理接口已启动，然后创建 lagg(4)[2115] 接口。在此示例中，*fxp0* 是主接口，*fxp1* 是辅助接口，并为虚拟接口分配了 IP 地址 *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
```

虚拟接口应如下所示：

```
# 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
```

(continues on next page)

[2115] https://www.freebsd.org/cgi/man.cgi?%3Equery=lagg&sektion=4&format=html

(continued from previous page)

```
        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>
```

流量将在 *fxp0* 上传输和接收。如果链接在 *fxp0* 上丢失，*fxp1* 将成为活动链接。如果在主接口上恢复了链接，它将再次成为活动链接。

要在重新启动后保留此配置，请将以下条目添加到 **/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"
```

例 **3.**以太网和无线接口之间的故障转移模式

对于便携式计算机用户，通常需要将无线设备配置为辅助设备，仅在以太网连接不可用时使用。使用 lagg(4)[2116] 时，可以配置一个故障转移，由于性能和安全原因，它更喜欢以太网连接，同时保持通过无线连接传输数据的能力。

这是通过将以太网接口的 MAC 地址替换为无线接口的 MAC 地址来实现的。

从理论上讲，以太网或无线 MAC 地址都可以更改以匹配另一个。但是，一些流行的无线接口不支持覆盖 MAC 地址。因此，我们建议为此覆盖以太网 MAC 地址。

如果无线接口的驱动程序没有在 `GENERIC` 或定制内核中加载，而计算机运行的是 FreeBSD 12.1，可以在 `/boot/loader.conf` 中加载相应的 `.ko`，在该文件中加入 `driver_load="YES"` 并重启。另一个更好的方法是在 `/etc/rc.conf` 中加载驱动程序，将其添加到该文件的 `kld_list`,详见（rc.conf(5)[2117]）中，然后重新启动。这是有必要的，因为否则在设置 lagg(4)[2118] 接口的时候，驱动程序还没有被加载。

在此示例中，以太网接口 *re0* 是主接口，无线接口 *wlan0* 是故障转移。*wlan0* 接口是从 *ath0* 物理无线接口创建的，以太网接口将使用无线接口的 MAC 地址进行配置。首先，启动无线接口（将 *FR* 替换为你自己的 2 个字母的国家/地区代码），但不要设置 IP 地址。更换 *wlan0* 以匹配系统的无线接口名称：

```
# ifconfig wlan0 create wlandev ath0 country FR ssid my_router up
```

现在，你可以确定无线接口的 MAC 地址：

```
# ifconfig wlan0
wlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
```

(continues on next page)

[2116] https://www.freebsd.org/cgi/man.cgi?query=lagg&sektion=4&format=html

[2117] https://www.freebsd.org/cgi/man.cgi?query=rc.conf&sektion=5&format=html

[2118] https://www.freebsd.org/cgi/man.cgi?query=lagg&sektion=4&format=html

(continued from previous page)

```
  ether b8:ee:65:5b:32:59
  groups: wlan
  ssid Bbox-A3BD2403 channel 6 (2437 MHz 11g ht/20) bssid 00:37:b7:56:4b:60
  regdomain ETSI country FR indoor ecm authmode WPA2/802.11i privacy ON
  deftxkey UNDEF AES-CCM 2:128-bit txpower 30 bmiss 7 scanvalid 60
  protmode CTS ampdulimit 64k ampdudensity 8 shortgi -stbctx stbcrx
  -ldpc wme burst roaming MANUAL
  media: IEEE 802.11 Wireless Ethernet MCS mode 11ng
  status: associated
  nd6 options=29<PERFORMNUD,IFDISABLED,AUTO_LINKLOCAL>
```

ether 行将包含指定接口的 MAC 地址。现在，更改以太网接口的 MAC 地址以匹配：

```
# ifconfig re0 ether b8:ee:65:5b:32:59
```

确保 *re0* 接口已启动，然后创建以 *re0* 作为主接口的 lagg(4)[2119] 接口，并故障转移到 *wlan0*：

```
# ifconfig re0 up
# ifconfig lagg0 create
# ifconfig lagg0 up laggproto failover laggport re0 laggport wlan0
```

虚拟接口应如下所示：

```
# ifconfig lagg0
lagg0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
       options=8<VLAN_MTU>
       ether b8:ee:65:5b:32:59
       laggproto failover lagghash l2,l3,l4
       laggport: re0 flags=5<MASTER,ACTIVE>
       laggport: wlan0 flags=0<>
       groups: lagg
      media: Ethernet autoselect
       status: active
```

然后，启动 DHCP 客户端以获取 IP 地址：

```
# dhclient lagg0
```

要在重新启动后保留此配置，请将以下条目添加到 **/etc/rc.conf**：

```
ifconfig_re0="ether b8:ee:65:5b:32:59"
wlans_ath0="wlan0"
ifconfig_wlan0="WPA"
```

(continues on next page)

[2119] https://www.freebsd.org/cgi/man.cgi?query=lagg&sektion=4&format=html

(continued from previous page)

```
create_args_wlan0="country FR"
cloned_interfaces="lagg0"
ifconfig_lagg0="up laggproto failover laggport re0 laggport wlan0 DHCP"
```

34.10.使用 PXE 进行无盘操作

英特尔预启动执行环境（Intel® Preboot eXecution Environment，PXE）允许通过网络引导操作系统。例如，可以从网络引导 FreeBSD 系统，并在没有本地磁盘的情况下运行，即使用从 NFS 服务器挂载的文件系统。PXE 支持通常在 BIOS 中可用。要在机器启动时使用 PXE，请在 BIOS 设置中选择 `Boot from network` 该选项，或在系统初始化期间键入功能键。

为了提供操作系统通过网络引导所需的文件，PXE 安装程序还需要正确配置 DHCP、TFTP 和 NFS 服务器，其中：

- 初始参数（如 IP 地址、可执行启动文件名和位置、服务器名称和根路径）是从 DHCP 服务器获取的。
- 操作系统加载程序文件是使用 TFTP 引导的。
- 使用 NFS 加载文件系统。

当计算机 PXE 启动时，它会通过 DHCP 接收有关从何处获取初始启动加载程序文件的信息。主计算机收到此信息后，通过 TFTP 下载引导加载程序，然后执行引导加载程序。在 FreeBSD 中，引导加载程序文件是 **/boot/pxeboot**。在 **/boot/pxeboot** 执行之后，FreeBSD 内核被加载，其余的 FreeBSD 引导序列继续进行，如 FreeBSD 引导过程[2120]中所述。

本节概述如何在 FreeBSD 系统上配置这些服务，以便其他系统可以 PXE 引导到 FreeBSD。有关详细信息，请参阅 diskless(8)[2121]。

> 当心
>
> 如前所述，提供这些服务的系统是不安全的。它应该位于网络的受保护区域中，并且不受其他主机的信任。

34.10.1.设置 PXE 环境

本节中显示的步骤配置内置 NFS 和 TFTP 服务器。下一节演示如何安装和配置 DHCP 服务器。在此示例中，将包含 PXE 用户使用的文件的目录是 **/b/tftpboot/FreeBSD/install**。重要的是，此目录存在，并且在 **/etc/inetd.conf** 和 **/usr/local/etc/dhcpd.conf** 中都设置了相同的目录名称。

> 注意

[2120] https://docs.freebsd.org/en/books/handbook/boot/index.html#boot

[2121] https://www.freebsd.org/cgi/man.cgi?query=diskless&sektion=8&format=html

下面的命令示例假定使用了 sh(1)[2122] shell。csh(1)[2123] 和 tcsh(1)[2124] 用户需要启动 sh(1)[2125] shell 或使命令适应 csh(1)[2126] 语法。

1. 创建将包含要挂载 NFS 的 FreeBSD 安装的根目录：

```
# export NFSROOTDIR=/b/tftpboot/FreeBSD/install
# mkdir -p ${NFSROOTDIR}
```

2. 通过将此行添加到 **/etc/rc.conf** 来启用 NFS 服务器：

```
nfs_server_enable="YES"
```

3. 通过 NFS 导出无盘根目录，方法是将以下内容添加到 **/etc/exports**：

```
/b -ro -alldirs -maproot=root
```

4. 启动 NFS 服务器：

```
# service nfsd start
```

5. 通过向 **/etc/rc.conf** 添加以下行来启用 inetd(8)[2127]：

```
inetd_enable="YES"
```

6. 取消注释 **/etc/inetd.conf** 中的以下行，方法是确保它不以符号 # 开头：

```
tftp dgram udp wait root /usr/libexec/tftpd tftpd -l -s /b/tftpboot
```

注意

某些版本的 PXE 需要 TCP 版本的 TFTP。在这种情况下，请取消对第二行 `tftp` 的注释，该行包含 `stream tcp`。

7. 启动 inetd(8)[2128]：

```
# service inetd start
```

8. 将基本系统安装到 **${NFSROOTDIR}**，可以通过解压官方压缩文件或构建 FreeBSD 内核和用户空间来实现（更详细的说明请参考 “从源代码更新 FreeBSD”[2129]，但不要忘记在运行 `make installkernel` 和 `make installworld` 命令时加上 **DESTDIR=${NFSROOTDIR}**。
9. 测试 TFTP 服务器工作，可以下载引导加载程序，将通过 PXE 获取：

[2122] https://www.freebsd.org/cgi/man.cgi?query=sh&sektion=1&format=html
[2123] https://www.freebsd.org/cgi/man.cgi?query=csh&sektion=1&format=html
[2124] https://www.freebsd.org/cgi/man.cgi?query=tcsh&sektion=1&format=html
[2125] https://www.freebsd.org/cgi/man.cgi?query=sh&sektion=1&format=html
[2126] https://www.freebsd.org/cgi/man.cgi?query=csh&sektion=1&format=html
[2127] https://www.freebsd.org/cgi/man.cgi?query=inetd&sektion=8&format=html
[2128] https://www.freebsd.org/cgi/man.cgi?query=inetd&sektion=8&format=html
[2129] https://docs.freebsd.org/en/books/handbook/cutting-edge/index.html/#makeworld

```
# tftp localhost
tftp> get FreeBSD/install/boot/pxeboot
Received 264951 bytes in 0.1 seconds
```

10. 编辑 **${NFSROOTDIR}/etc/fstab** 并创建一个条目以通过 NFS 挂载根文件系统：

```
# Device                                          Mountpoint    FSType  ␣
↪Options  Dump Pass
myhost.example.com:/b/tftpboot/FreeBSD/install       /          nfs      ro  ␣
↪      0    0
```

将 *myhost.example.com* 替换为 NFS 服务器的主机名或 IP 地址。在此示例中，根文件系统以只读方式挂载，以防止 NFS 客户端删除根文件系统的内容。

11. 在 PXE 环境中为正在 PXE 引导的客户端计算机设置 root 密码：

```
# chroot ${NFSROOTDIR}
# passwd
```

12. 如果需要，通过编辑 **${NFSROOTDIR}/etc/ssh/sshd_config** 并启用 `PermitRootLogin` 来为正在 PXE 启动的客户机启用 ssh(1)[2130] root 登录。这个选项在 sshd_config(5)[2131] 中有记载。
13. 在 **${NFSROOTDIR}** 中执行 PXE 环境的任何其他所需自定义。这些自定义可能包括安装软件包或使用 vipw（8）[2132] 编辑密码文件之类的事情。

当从 NFS root 卷启动时，**/etc/rc** 会检测到 NFS 启动并运行 `/etc/rc.initdiskless`。在这种情况下，**/etc** 和 **/var** 需要成为有内存支持的文件系统，以便这些目录可以写入，但 NFS 根目录是只读的：

```
# 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
```

当系统启动时，**/etc** 和 **/var** 的内存文件系统将被创建和挂载，**cpio.gz** 文件的内容将被复制到其中。默认情况下，这些文件系统的最大容量为 5 兆字节。如果你的档案不合适，通常在安装了二进制软件包后，**/var** 会出现这种情况，可以在 **:math:`{NFSROOTDIR}/conf/base/etc/md_size** 和 **`{NFSROOTDIR}/conf/base/var/md_size** 文件中分别为 **/etc** 和 **/var** 文件系统申请一个更大的容量（例如，5 兆字节是 10240 个扇区）。

[2130] https://www.freebsd.org/cgi/man.cgi?query=ssh&sektion=1&format=html
[2131] https://www.freebsd.org/cgi/man.cgi?query=sshd_config&sektion=5&format=html
[2132] https://www.freebsd.org/cgi/man.cgi?query=vipw&sektion=8&format=html

34.10.2.配置 DHCP 服务器

DHCP 服务器不需要与 TFTP 和 NFS 服务器位于同一台计算机，但需要在网络中可访问它。

DHCP 不是 FreeBSD 基本系统的一部分，但可以使用 net/isc-dhcp44-server[2133] port 或软件包进行安装。

安装后，编辑配置文件 **/usr/local/etc/dhcpd.conf**。如本例所示，配置 TFTP 服务器、文件名和根路径设置：

```
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-servers 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/" ;

}
```

`next-server` 指令用于指定 TFTP 服务器的 IP 地址。

文件名指令定义了 **/boot/pxeboot** 的路径。使用了一个相对的文件名，意味着在路径中并不包括 **/b/tftpboot**。

root-path 选项定义了到 NFS 根文件系统的路径。

编辑完毕后，在 **/etc/rc.conf** 中添加以下一行，在启动时启用 DHCP。

```
dhcpd_enable="YES"
```

然后启动 DHCP 服务：

```
# service isc-dhcpd start
```

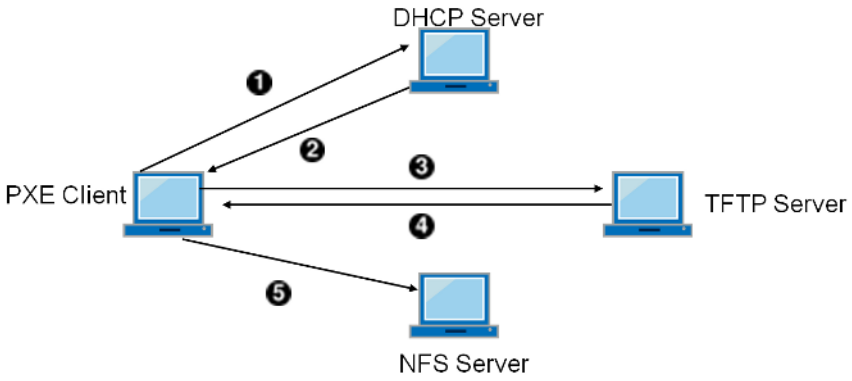
[2133] https://cgit.freebsd.org/ports/tree/net/isc-dhcp44-server/pkg-descr

34.10.3.调试 PXE 问题

一旦所有服务都配置并启动，PXE 客户端应该能够通过网络自动加载 FreeBSD。如果特定客户端无法连接，当该客户端计算机启动时，进入 BIOS 配置菜单并确认它已设置为从网络启动。

本节介绍一些故障排除提示，以便在没有客户端能够 PXE 启动时隔离配置问题的根源。

1. 使用软件包或 port 来安装 net/wireshark[2134] 调试 PXE 启动过程中涉及的网络流量，如下图所示。



   1. 客户端广播 DHCP 发现消息。
   2. DHCP 服务器使用 IP 地址、下一个服务器、文件名和根路径值进行响应。
   3. 客户端向下一个服务器发送 TFTP 请求，要求检索文件名。
   4. TFTP 服务器响应并将文件名发送到客户端。
   5. 客户端执行文件名 pxeboot（8），然后加载内核。当内核执行时，根路径指定的根文件系统将通过 NFS 挂载。

2. 在 TFTP 服务器上，读取 **/var/log/xferlog** 以确保从正确的位置检索 **pxeboot**。要测试此示例配置，请执行以下操作：

```
# tftp 192.168.0.1
tftp> get FreeBSD/install/boot/pxeboot
Received 264951 bytes in 0.1 seconds
```

   tftpd(8)[2135] 和 tftp(1)[2136] 中的 BUGS 部分记录了 TFTP 的一些限制。

3.确保根文件系统可以通过 NFS 挂载。要测试此示例配置，请执行以下操作：

```
# mount -t nfs 192.168.0.1:/b/tftpboot/FreeBSD/install /mnt
```

[2134] https://cgit.freebsd.org/ports/tree/net/wireshark/pkg-descr
[2135] https://www.freebsd.org/cgi/man.cgi?query=tftpd&sektion=8&format=html
[2136] https://www.freebsd.org/cgi/man.cgi?query=tftp&sektion=1&format=html

34.11.共用地址冗余协议（CARP）

共用地址冗余协议（CARP）允许多个主机共享相同的 IP 地址和虚拟主机 ID（VHID），以便为一个或多个服务提供 高可用性。这意味着一个或多个主机可能会发生故障，而其他主机将透明地接管，以便用户不会看到服务故障。

除了共享 IP 地址之外，每个主机都有自己的 IP 地址用于管理和配置。共享 IP 地址的所有计算机都具有相同的 VHID。每个虚拟 IP 地址的 VHID 在网络接口的广播域中必须是唯一的。

使用 CARP 的高可用性在 FreeBSD 中内置，尽管它的配置步骤因 FreeBSD 版本的不同而略有不同。本节为 FreeBSD 10 之前和之后的版本提供相同的示例配置。

这个例子配置了三个主机的故障转移支持，它们都有独特的 IP 地址，但提供相同的网络内容。它有两个不同的主机，名为 `hosta.example.org` 和 `hostb.example.org`，有一个共享备份，名为 `hostc.example.org`。

这些计算机使用轮循机制 DNS 配置进行负载平衡。主计算机和备份计算机的配置相同，但其主机名和管理 IP 地址除外。这些服务器必须具有相同的配置并运行相同的服务。发生故障转移时，仅当备份服务器有权访问相同的内容时，才能正确应答对共享 IP 地址上的服务的请求。备份计算机有两个附加的 CARP 接口，每个接口对应于主内容服务器的 IP 地址。发生故障时，备份服务器将拾取故障主计算机的 IP 地址。

34.11.1.在 FreeBSD 10 及更高版本上使用 CARP

通过在 **/boot/loader.conf** 中添加 **carp.ko** 内核模块的条目来启用对 CARP 的引导时支持：

```
carp_load="YES"
```

要立即加载模块而不重新启动：

```
# kldload carp
```

对于喜欢使用定制内核的用户，在定制内核配置文件中包含以下行，并按照配置 FreeBSD 内核[2137]中所述编译内核：

```
device  carp
```

主机名、管理 IP 地址和子网掩码、共享 IP 地址和 VHID 都是通过向 **/etc/rc.conf** 添加条目来设置的。此示例适用于 `hosta.example.org`：

```
hostname="hosta.example.org"
ifconfig_em0="inet 192.168.1.3 netmask 255.255.255.0"
ifconfig_em0_alias0="inet vhid 1 pass testpass alias 192.168.1.50/32"
```

[2137] https://docs.freebsd.org/en/books/handbook/kernelconfig/index.html#kernelconfig

下一组条目适用于 hostb.example.org。由于它表示第二个主服务器，因此它使用不同的共享 IP 地址和 VHID。但是，然而，与 pass 一起指定的密码必须是相同的，因为 CARP 只会监听和接受来自具有正确密码的机器的广告。

```
hostname="hostb.example.org"
ifconfig_em0="inet 192.168.1.4 netmask 255.255.255.0"
ifconfig_em0_alias0="inet vhid 2 pass testpass alias 192.168.1.51/32"
```

第三台机器，hostc.example.org，被配置为处理来自任一主站的故障切换。这台机器配置了两个 CARPVHID，一个用于处理每个主控主机的虚拟 IP 地址。设置 CARP 通告 skew (advskew) 是为了确保备份主机的通告时间晚于主主机，因为当有多个备份服务器时，advskew 控制优先级的顺序。

```
hostname="hostc.example.org"
ifconfig_em0="inet 192.168.1.5 netmask 255.255.255.0"
ifconfig_em0_alias0="inet vhid 1 advskew 100 pass testpass alias 192.168.1.50/32"
ifconfig_em0_alias1="inet vhid 2 advskew 100 pass testpass alias 192.168.1.51/32"
```

配置了两个 CARPVHID 意味着如果任何一个主服务器变得不可用，hostc.example.org 将注意到。如果主服务器不能在备份服务器之前发布广告，备份服务器将获得共享的 IP 地址，直到主服务器再次变得可用。

> 注意
>
> 如果原来的主服务器再次可用，hostc.example.org 将不会自动释放虚拟 IP 地址给它。要做到这一点，必须启用抢占功能。该功能默认是禁用的，它通过 sysctl(8)[2138] 变量 net.inet.carp.preempt 来控制的。管理员可以强制备份服务器将 IP 地址返回给主服务器：
>
> ```
> # ifconfig em0 vhid 1 state backup
> ```

配置完成后，重新启动网络或重新启动每个系统。现已启用高可用性。

CARP 功能可以通过 carp(4)[2139]）手册页中记录的多个 sysctl(8)[2140]）变量进行控制。其他操作可以通过使用 devd(8)[2141] 从 CARP 事件触发。

[2138] https://www.freebsd.org/cgi/man.cgi?query=sysctl&sektion=8&format=html
[2139] https://www.freebsd.org/cgi/man.cgi?query=carp&sektion=4&format=html
[2140] https://www.freebsd.org/cgi/man.cgi?query=sysctl&sektion=8&format=html
[2141] https://www.freebsd.org/cgi/man.cgi?query=devd&sektion=8&format=html

34.11.2.在 FreeBSD 9 及更早版本上使用 CARP

这些版本的 FreeBSD 的配置与上一节中提及的配置类似，不同之处在于必须首先在配置中创建和引用 CARP 设备。

通过在 **/boot/loader.conf** 中加载 **if_carp.ko** 内核模块来启用对 CARP 的引导时支持：

```
if_carp_load="YES"
```

要立即加载模块而不重新启动：

```
# kldload carp
```

对于喜欢使用定制内核的用户，在定制内核配置文件中包含以下行，并按照配置 FreeBSD 内核[2142] 中所述编译内核：

```
device  carp
```

接下来，在每个主机上，创建一个 CARP 设备：

```
# ifconfig carp0 create
```

通过在 **/etc/rc.conf** 中添加必要的行来设置主机名、管理 IP 地址、共享 IP 地址和 VHID。由于使用的是虚拟 CARP 设备而不是别名，实际的子网掩码是 `/24` 而不是 `/32`。下面是 `hosta.example.org` 的条目。

```
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"
```

在`hostb.example.org`上

```
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"
```

第三台计算机 `hostc.example.org` 配置为处理来自任一主主机的故障转移：

```
hostname="hostc.example.org"
ifconfig_fxp0="inet 192.168.1.5 netmask 255.255.255.0"
cloned_interfaces="carp0 carp1"
```

(continues on next page)

[2142] https://docs.freebsd.org/en/books/handbook/kernelconfig/index.html#kernelconfig

(continued from previous page)

```
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"
```

注意

在 FreeBSD **GENERIC** 内核中，抢占被禁用。如果在定制内核中启用了抢占功能，`hostc.example.org` 可能不会将 IP 地址放回给原始内容服务器。管理员可以用命令强制备份服务器将 IP 地址返回给主服务器。

```
# ifconfig carp0 down && ifconfig carp0 up
```

这应该在与正确的主机相对应的 **carp** 接口上进行。

配置完成后，重新启动网络或重新启动每个系统。现已启用高可用性。

34.12.VLANs

VLAN 是一种将网络虚拟划分为许多不同子网（也称为分段）的方法。每个网段都有自己的广播域，并与其他 VLAN 隔离。

在 FreeBSD 上，网卡驱动程序必须支持 VLAN。要查看哪些驱动程序支持 vlan，请参阅 vlan(4)[2143] 手册页。

配置 VLAN 时，必须知道几条信息。首先，哪个网络接口？其次，什么是 VLAN 标记?

要在运行时配置 VLAN，网卡为 `em0`，VLAN 标签为 `5`，命令如下。

```
# ifconfig em0.5 create vlan 5 vlandev em0 inet 192.168.20.20/24
```

注意

了解接口名称如何包含 NIC 驱动程序名称和 VLAN 标记（以句点分隔）？这是一种最佳做法，用于在计算机上存在许多 VLAN 时轻松维护 VLAN 配置。

要在引导时配置 VLAN，必须更新 **/etc/rc.conf**。要复制上述配置，需要添加以下内容：

```
vlans_em0="5"
ifconfig_em0_5="inet 192.168.20.20/24"
```

可以添加额外的 VLAN，只需在 `vlans_em0` 字段中添加标签，并在该 VLAN 标签的接口上增加一行配置网络的内容。

为一个接口指定一个符号名称是很有用的，这样当相关的硬件被改变时，只需要更新一些配置变量。例如，安全摄像机需要在 `em0` 的 VLAN 1 上运行。以后，如果 em0 卡被替换成使用 ixgb(4)[2144] 驱动的卡，所有

[2143] https://www.freebsd.org/cgi/man.cgi?query=vlan&sektion=4&format=html

[2144] https://www.freebsd.org/cgi/man.cgi?query=ixgb&sektion=4&format=html

对 em0.1 的引用就不必改为 ixgb0.1 了。

要在网卡 em0 上配置 VLAN 5，指定接口名称为 cameras，并为该接口分配一个 24 位前缀的 IP 地址 ``*192.168.20.20*``，请使用此命令：

```
# ifconfig em0.5 create vlan 5 vlandev em0 name cameras inet 192.168.20.20/24
```

对于名为 video 的接口，请使用以下命令：

```
# ifconfig video.5 create vlan 5 vlandev video name cameras inet 192.168.20.20/24
```

要在引导时应用更改，请将以下行添加到 **/etc/rc.conf**：

```
vlans_video="cameras"
create_args_cameras="vlan 5"
ifconfig_cameras="inet 192.168.20.20/24"
```

第五部分：附录

附录 A. 获取 FreeBSD

A.1.镜像站

FreeBSD 项目的官方镜像站是由许多机器组成的，这些机器由项目集群管理员操作，并由 GeoDNS 引导用户到最近的可用镜像。目前的位置是澳大利亚、巴西、德国、日本（两个区域）、马来西亚、南非、台湾、英国、美国（加利福尼亚、新泽西和华盛顿）。

官方镜像站服务:

域名	协议	详情
docs.FreeBSD.org	https[2145]	FreeBSD 文档门户。
download.FreeBSD.org	https[2146] ftp[2147]	与 ftp.FreeBSD.org 的内容相同，ftp 是一个传统的名字；建议使用 download.FreeBSD.org。
git.FreeBSD.org	git over `https` and `ssh`	更多细节见使用 git[2148] 章节
pkg.FreeBSD.org	pkg(8)[2149] over `http` and `https`	pkg(8)[2150] 程序所使用的 FreeBSD 官方软件包库。
vuxml.FreeBSD.org / **www.VuXML.org**	https[2151]	FreeBSD 项目 VuXML 网页。pkg audit 会从这个服务中获取漏洞列表。
www.FreeBSD.org	https[2152]	FreeBSD 网站

所有的官方镜像都支持 IPv4 和 IPv6。

[2145] https://docs.freebsd.org/

[2146] https://download.freebsd.org/

[2147] ftp://download.freebsd.org/pub/FreeBSD/

[2148] https://docs.freebsd.org/en/books/handbook/mirrors/#git

[2149] https://www.freebsd.org/cgi/man.cgi?query=pkg&sektion=8&format=html

[2150] https://www.freebsd.org/cgi/man.cgi?query=pkg&sektion=8&format=html

[2151] https://www.vuxml.org/

[2152] https://www.freebsd.org/

http://ftp-archive.FreeBSD.org 不在 GeoDNS 基础设施内，只在一个地方（美国）托管。

该项目正在寻找新的站点；那些愿意赞助的人，请联系群组管理员团队以获得更多信息。

由社区和其他公司维护的镜像站列表：

国家/地区	域名	协议
Australia[2153]	ftp.au.FreeBSD.org	http[2154] http_v6[2155] rsync rsync_v6
	ftp3.au.FreeBSD.org	http[2156] ftp[2157] rsync
Austria[2158]	ftp.at.FreeBSD.org	http[2159] http_v6[2160] ftp[2161] ftp_v6[2162] rsync rsync_v6
Brazil[2163]	ftp2.br.FreeBSD.org	http[2164] rsync rsync_v6
	ftp3.br.FreeBSD.org	http[2165] ftp[2166] rsync
Bulgaria[2167]	ftp.bg.FreeBSD.org	ftp[2168] ftp_v6[2169] rsync rsync_v6
Czech Republic[2170]	ftp.cz.FreeBSD.org	http[2171] http_v6[2172] rsync rsync_v6
Denmark[2173]	ftp.dk.FreeBSD.org	http[2174] http_v6[2175] ftp[2176] ftp_v6[2177] rsync rsync_v6
Finland[2178]	ftp.fi.FreeBSD.org	ftp[2179]
France[2180]	ftp.fr.FreeBSD.org	http[2181] http_v6[2182] ftp[2183] ftp_v6[2184] rsync rsync_v6
	ftp3.fr.FreeBSD.org	ftp[2185]
	ftp6.fr.FreeBSD.org	http[2186] ftp[2187] rsync

[2153] hostmaster@au.FreeBSD.org
[2154] http://ftp.au.freebsd.org/pub/FreeBSD
[2155] http://ftp.au.freebsd.org/pub/FreeBSD
[2156] http://ftp3.au.freebsd.org/pub/FreeBSD
[2157] ftp://ftp3.au.freebsd.org/pub/FreeBSD
[2158] hostmaster@at.FreeBSD.org
[2159] http://ftp.at.freebsd.org/pub/FreeBSD/
[2160] http://ftp.at.freebsd.org/pub/FreeBSD/
[2161] ftp://ftp.at.freebsd.org/pub/FreeBSD/
[2162] ftp://ftp.at.freebsd.org/pub/FreeBSD/
[2163] hostmaster@br.FreeBSD.org
[2164] http://ftp2.br.freebsd.org/FreeBSD
[2165] http://ftp3.br.freebsd.org/pub/FreeBSD
[2166] ftp://ftp3.br.freebsd.org/pub/FreeBSD
[2167] mirror@telepoint.bg
[2168] ftp://ftp.bg.freebsd.org/pub/FreeBSD
[2169] ftp://ftp.bg.freebsd.org/pub/FreeBSD
[2170] hostmaster@cz.FreeBSD.org
[2171] http://ftp.cz.freebsd.org/pub/FreeBSD
[2172] http://ftp.cz.freebsd.org/pub/FreeBSD
[2173] staff@dotsrc.org
[2174] http://ftp.dk.freebsd.org/FreeBSD/
[2175] http://ftp.dk.freebsd.org/FreeBSD/
[2176] ftp://ftp.dk.freebsd.org/FreeBSD/
[2177] ftp://ftp.dk.freebsd.org/FreeBSD/
[2178] hostmaster@fi.FreeBSD.org
[2179] ftp://ftp.fi.freebsd.org/pub/FreeBSD
[2180] hostmaster@fr.FreeBSD.org
[2181] http://ftp.fr.freebsd.org/pub/FreeBSD
[2182] http://ftp.fr.freebsd.org/pub/FreeBSD
[2183] ftp://ftp.fr.freebsd.org/pub/FreeBSD
[2184] ftp://ftp.fr.freebsd.org/pub/FreeBSD
[2185] ftp://ftp3.fr.freebsd.org/pub/FreeBSD
[2186] http://ftp6.fr.freebsd.org/pub/FreeBSD
[2187] ftp://ftp6.fr.freebsd.org/pub/FreeBSD

国家/地区	域名	协议
Germany[2188]	ftp.de.FreeBSD.org	ftp[2189] ftp_v6[2190] rsync rsync_v6
	ftp1.de.FreeBSD.org	http[2191] http_v6[2192] ftp[2193] ftp_v6[2194] rsync rsync_v6
	ftp2.de.FreeBSD.org	http[2195] http_v6[2196] ftp[2197] ftp_v6[2198] rsync rsync_v6
	ftp5.de.FreeBSD.org	ftp[2199] ftp_v6[2200]
	ftp7.de.FreeBSD.org	http[2201] http_v6[2202] ftp[2203] ftp_v6[2204]
Greece[2205]	ftp.gr.FreeBSD.org	http[2206] http_v6[2207] ftp[2208] ftp_v6[2209]
	ftp2.gr.FreeBSD.org	http[2210] http_v6[2211] ftp[2212] ftp_v6[2213] rsync
Japan[2214]	ftp.jp.FreeBSD.org	http[2215] http_v6[2216] ftp[2217] ftp_v6[2218] rsync rsync_v6
	ftp2.jp.FreeBSD.org	ftp[2219] rsync rsync_v6
	ftp3.jp.FreeBSD.org	http[2220] rsync
	ftp4.jp.FreeBSD.org	ftp[2221]
	ftp6.jp.FreeBSD.org	http[2222] http_v6[2223] ftp[2224] ftp_v6[2225] rsync rsync_v6

[2188] de-bsd-hubs@de.FreeBSD.org
[2189] ftp://ftp.de.freebsd.org/pub/FreeBSD
[2190] ftp://ftp.de.freebsd.org/pub/FreeBSD
[2191] http://ftp1.de.freebsd.org/pub/FreeBSD
[2192] http://ftp1.de.freebsd.org/pub/FreeBSD
[2193] ftp://ftp1.de.freebsd.org/pub/FreeBSD
[2194] ftp://ftp1.de.freebsd.org/pub/FreeBSD
[2195] http://ftp2.de.freebsd.org/pub/FreeBSD
[2196] http://ftp2.de.freebsd.org/pub/FreeBSD
[2197] ftp://ftp2.de.freebsd.org/pub/FreeBSD
[2198] ftp://ftp2.de.freebsd.org/pub/FreeBSD
[2199] ftp://ftp5.de.freebsd.org/pub/FreeBSD
[2200] ftp://ftp5.de.freebsd.org/pub/FreeBSD
[2201] http://ftp7.de.freebsd.org/pub/FreeBSD
[2202] http://ftp7.de.freebsd.org/pub/FreeBSD
[2203] ftp://ftp7.de.freebsd.org/pub/FreeBSD
[2204] ftp://ftp7.de.freebsd.org/pub/FreeBSD
[2205] hostmaster@gr.FreeBSD.org
[2206] http://ftp.gr.freebsd.org/pub/FreeBSD
[2207] http://ftp.gr.freebsd.org/pub/FreeBSD
[2208] ftp://ftp.gr.freebsd.org/pub/FreeBSD
[2209] ftp://ftp.gr.freebsd.org/pub/FreeBSD
[2210] http://ftp2.gr.freebsd.org/pub/FreeBSD
[2211] http://ftp2.gr.freebsd.org/pub/FreeBSD
[2212] ftp://ftp2.gr.freebsd.org/pub/FreeBSD
[2213] ftp://ftp2.gr.freebsd.org/pub/FreeBSD
[2214] hostmaster@jp.FreeBSD.org
[2215] http://ftp.jp.freebsd.org/pub/FreeBSD
[2216] http://ftp.jp.freebsd.org/pub/FreeBSD
[2217] ftp://ftp.jp.freebsd.org/pub/FreeBSD
[2218] ftp://ftp.jp.freebsd.org/pub/FreeBSD
[2219] ftp://ftp2.jp.freebsd.org/pub/FreeBSD
[2220] http://ftp3.jp.freebsd.org/pub/FreeBSD
[2221] ftp://ftp4.jp.freebsd.org/pub/FreeBSD
[2222] http://ftp6.jp.freebsd.org/pub/FreeBSD
[2223] http://ftp6.jp.freebsd.org/pub/FreeBSD
[2224] ftp://ftp6.jp.freebsd.org/pub/FreeBSD
[2225] ftp://ftp6.jp.freebsd.org/pub/FreeBSD

国家/地区	域名	协议
Korea[2226]	ftp.kr.FreeBSD.org	http[2227] https[2228] ftp[2229] rsync
	ftp2.kr.FreeBSD.org	rsync
Latvia[2230]	ftp.lv.FreeBSD.org	http[2231] ftp[2232]
Netherlands[2233]	ftp.nl.FreeBSD.org	http[2234] http_v6[2235] ftp[2236] ftp_v6[2237] rsync rsync_v6
	ftp2.nl.FreeBSD.org	http[2238] ftp[2239] rsync
New Zealand[2240]	ftp.nz.FreeBSD.org	http[2241] ftp[2242]
Norway[2243]	ftp.no.FreeBSD.org	ftp[2244] ftp_v6[2245] rsync rsync_v6
Poland[2246]	ftp.pl.FreeBSD.org	http[2247] http_v6[2248] ftp[2249] rsync rsync_v6
Russia[2250]	ftp.ru.FreeBSD.org	http[2251] http_v6[2252] ftp[2253] ftp_v6[2254] rsync rsync_v6
	ftp2.ru.FreeBSD.org	https[2255] ftp[2256] rsync
Slovenia[2257]	ftp.si.FreeBSD.org	http[2258] http_v6[2259] ftp[2260] ftp_v6[2261]

[2226] hostmaster@kr.FreeBSD.org
[2227] http://ftp.kr.freebsd.org/pub/FreeBSD
[2228] https://ftp.kr.freebsd.org/pub/FreeBSD
[2229] ftp://ftp.kr.freebsd.org/pub/FreeBSD
[2230] hostmaster@lv.FreeBSD.org
[2231] http://ftp.lv.freebsd.org/pub/Freebsd
[2232] ftp://ftp.lv.freebsd.org/pub/freebsd
[2233] hostmaster@nl.FreeBSD.org
[2234] http://ftp.nl.freebsd.org/pub/FreeBSD
[2235] http://ftp.nl.freebsd.org/pub/FreeBSD
[2236] ftp://ftp.nl.freebsd.org/pub/FreeBSD
[2237] ftp://ftp.nl.freebsd.org/pub/FreeBSD
[2238] http://ftp2.nl.freebsd.org/pub/FreeBSD
[2239] ftp://ftp2.nl.freebsd.org/pub/FreeBSD
[2240] hostmaster@nz.FreeBSD.org
[2241] http://ftp.nz.freebsd.org/pub/FreeBSD
[2242] ftp://ftp.nz.freebsd.org/pub/FreeBSD
[2243] hostmaster@no.FreeBSD.org
[2244] ftp://ftp.no.freebsd.org/pub/FreeBSD
[2245] ftp://ftp.no.freebsd.org/pub/FreeBSD
[2246] hostmaster@pl.FreeBSD.org
[2247] http://ftp.pl.freebsd.org/pub/FreeBSD
[2248] http://ftp.pl.freebsd.org/pub/FreeBSD
[2249] ftp://ftp.pl.freebsd.org/pub/FreeBSD
[2250] hostmaster@ru.FreeBSD.org
[2251] http://ftp.ru.freebsd.org/pub/FreeBSD
[2252] http://ftp.ru.freebsd.org/pub/FreeBSD
[2253] ftp://ftp.ru.freebsd.org/pub/FreeBSD
[2254] ftp://ftp.ru.freebsd.org/pub/FreeBSD
[2255] https://ftp2.ru.freebsd.org/pub/FreeBSD
[2256] ftp://ftp2.ru.freebsd.org/pub/FreeBSD
[2257] hostmaster@si.FreeBSD.org
[2258] http://ftp.si.freebsd.org/pub/FreeBSD
[2259] http://ftp.si.freebsd.org/pub/FreeBSD
[2260] ftp://ftp.si.freebsd.org/pub/FreeBSD
[2261] ftp://ftp.si.freebsd.org/pub/FreeBSD

国家/地区	域名	协议
South Africa[2262]	ftp.za.FreeBSD.org	https[2263] https_v6[2264] rsync rsync_v6
	ftp2.za.FreeBSD.org	http[2265] http_v6[2266] ftp_v6[2267]
	ftp4.za.FreeBSD.org	http[2268] ftp[2269] rsync
Sweden[2270]	ftp.se.FreeBSD.org	http[2271] http_v6[2272] ftp[2273] ftp_v6[2274] rsync rsync_v6
Taiwan[2275]	ftp4.tw.FreeBSD.org	https[2276] ftp[2277] rsync
	ftp5.tw.FreeBSD.org	http[2278] ftp[2279]
Ukraine[2280]	ftp.ua.FreeBSD.org	http[2281] ftp[2282] ftp_v6[2283] rsync rsync_v6
United Kingdom[2284]	ftp.uk.FreeBSD.org	http[2285] http_v6[2286] ftp[2287] ftp_v6[2288] rsync rsync_v6
	ftp2.uk.FreeBSD.org	http[2289] http_v6[2290] https[2291] https_v6[2292] ftp[2293] ftp_v6[2294]
United States of America[2295]	ftp11.FreeBSD.org	http[2296] http_v6[2297] ftp[2298] ftp_v6[2299] rsync rsync_v6
	ftp14.FreeBSD.org	ftp[2300] rsync (Former official tier 1)
	ftp5.FreeBSD.org	http[2301] http_v6[2302] ftp[2303] ftp_v6[2304]

目前社区镜像站支持的协议列表最后一次更新是在 2022-01-31，但并做不保证。

[2262] hostmaster@za.FreeBSD.org
[2263] https://ftp.za.freebsd.org/pub/FreeBSD
[2264] https://ftp.za.freebsd.org/pub/FreeBSD
[2265] http://ftp2.za.freebsd.org/pub/FreeBSD
[2266] http://ftp2.za.freebsd.org/pub/FreeBSD
[2267] ftp://ftp2.za.freebsd.org/pub/FreeBSD
[2268] http://ftp4.za.freebsd.org/pub/FreeBSD
[2269] ftp://ftp4.za.freebsd.org/pub/FreeBSD
[2270] hostmaster@se.FreeBSD.org
[2271] http://ftp.se.freebsd.org/pub/FreeBSD
[2272] http://ftp.se.freebsd.org/pub/FreeBSD
[2273] ftp://ftp.se.freebsd.org/pub/FreeBSD
[2274] ftp://ftp.se.freebsd.org/pub/FreeBSD
[2275] hostmaster@se.FreeBSD.org
[2276] https://ftp4.tw.freebsd.org/pub/FreeBSD
[2277] ftp://ftp4.tw.freebsd.org/pub/FreeBSD
[2278] http://ftp5.tw.freebsd.org/pub/FreeBSD
[2279] ftp://ftp5.tw.freebsd.org/pub/FreeBSD
[2280] hostmaster@ua.FreeBSD.org
[2281] http://ftp.ua.freebsd.org/pub/FreeBSD
[2282] ftp://ftp.ua.freebsd.org/pub/FreeBSD
[2283] ftp://ftp.ua.freebsd.org/pub/FreeBSD
[2284] hostmaster@uk.FreeBSD.org
[2285] http://ftp.uk.freebsd.org/pub/FreeBSD
[2286] http://ftp.uk.freebsd.org/pub/FreeBSD
[2287] ftp://ftp.uk.freebsd.org/pub/FreeBSD
[2288] ftp://ftp.uk.freebsd.org/pub/FreeBSD
[2289] http://ftp2.uk.freebsd.org/pub/FreeBSD
[2290] http://ftp2.uk.freebsd.org/pub/FreeBSD
[2291] https://ftp2.uk.freebsd.org/pub/FreeBSD
[2292] https://ftp2.uk.freebsd.org/pub/FreeBSD
[2293] ftp://ftp2.uk.freebsd.org/pub/FreeBSD
[2294] ftp://ftp2.uk.freebsd.org/pub/FreeBSD
[2295] hostmaster@us.FreeBSD.org
[2296] http://ftp11.freebsd.org/pub/FreeBSD
[2297] http://ftp11.freebsd.org/pub/FreeBSD
[2298] ftp://ftp11.freebsd.org/pub/FreeBSD
[2299] ftp://ftp11.freebsd.org/pub/FreeBSD
[2300] ftp://ftp14.freebsd.org/pub/FreeBSD
[2301] http://ftp5.freebsd.org/pub/FreeBSD
[2302] http://ftp5.freebsd.org/pub/FreeBSD
[2303] ftp://ftp5.freebsd.org/pub/FreeBSD
[2304] ftp://ftp5.freebsd.org/pub/FreeBSD

A.2.使用 Git

A.2.1.简介

从 2020 年 12 月起，FreeBSD 使用 git 作为主要的版本控制系统来存储所有 FreeBSD 的基本源代码和文档。从 2021 年 4 月起，FreeBSD 使用 git 作为唯一的版本控制系统来存储所有的 FreeBSD ports。

> 注意
>
> Git 通常是一个开发者工具。用户可能更喜欢使用 `freebsd-update`（“更新 FreeBSD”[2305]）来更新 FreeBSD 基本系统，以及 `git`（“使用 ports”[2306]）来更新 FreeBSD ports。

这一节演示了如何在 FreeBSD 系统上安装 Git 并使用它来创建 FreeBSD 源代码仓库的本地拷贝。

A.2.2.安装

Git 可以从 ports 中安装，也可以作为一个软件包安装。

```
# pkg install git
```

A.2.3.运行 Git

要获取一个干净的源码副本到本地目录，请使用 `git clone`。这个目录下的文件被称为 *工作树*。

Git 使用链接来指定一个仓库。有三个不同的仓库，`src` 是指 FreeBSD 系统的源代码，`doc` 是指文档，而 `ports` 则是指 FreeBSD ports。这三个仓库都可以通过两种不同的协议到达：HTTPS 和 SSH。例如，链接 https://git.FreeBSD.org/src.git 指定了 src 仓库的主分支，并使用了 https 协议。

表 1. FreeBSD Git 仓库的链接表

项目	Git 链接
只读 src 仓库，使用 HTTPS	https://git.FreeBSD.org/src.git[2307]
只读 src 仓库，使用 anon-ssh	ssh://anongit@git.FreeBSD.org/src.git
只读 doc 仓库，使用 HTTPS	https://git.FreeBSD.org/doc.git[2308]
只读 doc 仓库，使用 anon-ssh	ssh://anongit@git.FreeBSD.org/doc.git
只读 ports 仓库，使用 HTTPS	https://git.FreeBSD.org/ports.git[2309]
只读 ports 仓库，使用 anon-ssh	ssh://anongit@git.FreeBSD.org/ports.git

[2305] https://docs.freebsd.org/en/books/handbook/cutting-edge/index.html#updating-upgrading-freebsdupdate

[2306] https://docs.freebsd.org/en/books/handbook/ports/index.html#ports-using

[2307] https://git.freebsd.org/src.git

[2308] https://git.freebsd.org/doc.git

[2309] https://git.freebsd.org/ports.git

由项目成员维护的外部镜像也可以使用，请参考外部镜像[2310]部分。

要克隆一份 FreeBSD 系统源代码的副本。

```
# git clone -o freebsd https://git.FreeBSD.org/src.git /usr/src
```

`-o freebsd` 选项了指定了源；根据 FreeBSD 文档的惯例，源被假定为 `freebsd`。因为初始检出必须下载远程仓库的完整分支，所以可能需要一些时间。请耐心等待。

最开始的工作树包含主分支的源代码，对应的是 CURRENT。要切换到 13-STABLE：

```
# cd /usr/src
# git checkout stable/13
```

可以用 `git pull` 更新工作树。要更新上面的例子中创建的 **/usr/src**，请使用：

```
# cd /usr/src
# git pull --rebase
```

更新比检出要快得多，只传输有变化的文件。

A.2.4.基于网络的资源库浏览器

FreeBSD 项目使用 cgit 作为基于网络的版本仓库浏览器：https://cgit.FreeBSD.org/。

A.2.5. 对于开发者

关于对存储库的写入权限的信息，请参见 提交者指南[2311]。

A.2.6. 外部镜像

这些镜像并不托管在 FreeBSD.org 中，但仍由项目成员维护。我们欢迎用户和开发人员拉取或浏览这些镜像上的仓库。目前正在接受对 `doc` 和 `src` GitHub 仓库的拉取请求；除此之外，项目与这些镜像的工作流程仍在讨论中。

Codeberg

- doc: https://codeberg.org/FreeBSD/freebsd-doc
- ports: https://codeberg.org/FreeBSD/freebsd-ports
- src: https://codeberg.org/FreeBSD/freebsd-src

[2310] https://docs.freebsd.org/en/books/handbook/mirrors/#external-mirrors

[2311] https://docs.freebsd.org/en/articles/committers-guide/#git-mini-primer

GitHub

- doc: https://github.com/freebsd/freebsd-doc
- ports: https://github.com/freebsd/freebsd-ports
- src: https://github.com/freebsd/freebsd-src

GitLab

- doc: https://gitlab.com/FreeBSD/freebsd-doc
- ports: https://gitlab.com/FreeBSD/freebsd-ports
- src: https://gitlab.com/FreeBSD/freebsd-src

A.2.7. 邮件列表

在 FreeBSD 项目中，关于 git 的一般使用和问题的主要邮件列表是 freebsd-git[2312]。更多细节，包括提交信息列表，请参见邮件列表[2313]章节。

A.2.8. SSH 主机密钥

- gitrepo.FreeBSD.org 主机密钥指纹：
  - ECDSA 密钥指纹是 `SHA256:seWO5D27ySURcx4bknTNKlC1mgai0whP443PAKEvvZA`
  - ED25519 密钥指纹是 `SHA256:lNR6i4BEOaaUhmDHBA1WJsO7H3KtvjE2r5q4sOxtIWo`
  - RSA 密钥指纹是 `SHA256:f453CUEFXEJAXlKeEHV+ajJfeEfx9MdKQUD7lIscnQI`
- git.FreeBSD.org 主机密钥指纹：
  - ECDSA 密钥指纹是 `SHA256:/UlirUAsGiitupxmtsn7f9b7zCWd0vCs4Yo/tpVWP9w`
  - ED25519 密钥指纹是 `SHA256:y1ljKrKMD3lDObRUG3xJ9gXwEIuqnh306tSyFd1tuZE`
  - RSA 密钥指纹是 `SHA256:jBe6FQGoH4HjvrIVM23dcnLZk9kmpdezR/CvQzm7rJM`

这些也作为 SSHFP 记录公布在 DNS 中。

[2312] https://lists.freebsd.org/subscription/freebsd-git

[2313] https://docs.freebsd.org/en/books/handbook/handbook/eresources/index.html#eresources-mail

A.3.使用 Subversion

A.3.1.简介

从 2020 年 12 月起，FreeBSD 使用 git 作为主要的版本控制系统来存储所有 FreeBSD 的源代码和文档。在 stable/11, stable/12 和相关的 releng 分支上的 git repo 的改动会被导出到 subversion 仓库中。这种输出将持续到这些分支的生命周期。从 2012 年 7 月到 2021 年 3 月，FreeBSD 使用 Subversion 作为唯一的版本控制系统来存储所有 FreeBSD 的 ports。从 2021 年 4 月起，FreeBSD 使用 git 作为存储所有 FreeBSD ports 的唯一版本控制系统。

> 注意
>
> Subversion 通常是一个开发者工具。用户可能更喜欢使用 freebsd-update（“更新 FreeBSD”[2314]）来更新 FreeBSD 基本系统，以及 git（“使用 ports”[2315]）来更新 FreeBSD ports。在 2021 年 3 月之后，subversion 的使用只针对传统的分支（`stable/11` 和 `stable/12`）。

这一节演示了如何在 FreeBSD 系统上安装 Subversion 并使用它来创建 FreeBSD 仓库的本地副本。还包括了关于使用 Subversion 的其他信息。

A.3.2.Svnlite

一个轻量级的 Subversion 版本已经作为 `svnlite` 安装在 FreeBSD 上。只有在需要 Python 或 Perl API，或者需要较高版本的 Subversion 时，才需要移植或软件包版本的 Subversion。

与正常的 Subversion 使用的唯一区别是，命令的名字是 `svnlite`。

A.3.3.安装

如果 svnlite 不可用或者需要完整版的 Subversion，那么必须手动安装它。

Subversion 可以从 ports 中安装:

```
# cd /usr/ports/devel/subversion
# make install clean
```

也可以通过软件包来安装 Subversion：

```
# pkg install subversion
```

[2314] https://docs.freebsd.org/en/books/handbook/cutting-edge/index.html#updating-upgrading-freebsdupdate

[2315] https://docs.freebsd.org/en/books/handbook/ports/index.html#ports-using

A.3.4. 运行 Subversion

要获取一个干净的源代码拷贝到本地目录中，请使用 svn。这个目录中的文件被称为本地工作副本。

> 警告
> 在第一次使用 checkout 之前，移动或删除现有的目标目录。在现有的非 svn 目录上进行检出会导致现有文件和从版本库带入的文件之间的冲突。

Subversion 使用 URL 来指定一个版本库，其形式为 *protocol://hostname/path*。路径的第一个组成部分是要访问的 FreeBSD 代码库。有三个不同的版本库，base 是 FreeBSD 基本系统的源代码，ports 是 ports，doc 是文档。例如，URL https://svn.FreeBSD.org/base/head/ 使用 https 协议指定了 src 代码库的主分支。

从一个给定的版本库中检出，可以用这样的命令进行：

```
# svn checkout https://svn.FreeBSD.org/repository/branch lwcdir
```

其中：

- repository 是项目仓库中的一个：base、ports 或 doc。
- branch 取决于所使用的版本库。ports 和 doc 主要在 head 分支中更新，而 base 则在 head 下维护 head 的 -CURRENT 版本，在 stable/11（11.x）和 stable/12（12.x）下维护各自的最新版本 -STABLE 分支。
- lwcdir 是指定分支的内容应被放置的目标目录。对于 ports 来说，这通常是 **/usr/ports**，对于 base 来说是 **/usr/src**，而对于 doc 来说是 **/usr/doc**。

这个例子使用 HTTPS 协议从 FreeBSD 版本库中检出源代码树，将本地工作拷贝放在 **/usr/src** 中。如果 **/usr/src** 已经存在，但不是由 svn 创建的，记得在签出前重命名或删除它。

```
# svn checkout https://svn.FreeBSD.org/base/head /usr/src
```

因为初始检出必须下载远程版本库的完整分支，所以可能需要一些时间。请耐心等待。

在初始检出后，可以通过运行本地工作副本来更新：

```
# svn update lwcdir
```

要更新上面例子中创建的 **/usr/src**，请使用：

```
# svn update /usr/src
```

update 比 checkout 要快得多，只传输有变化的文件：

另一种在签出后更新本地工作拷贝的方法是由 **/usr/ports**、**/usr/src** 和 **/usr/doc** 目录中的 **Makefile** 提供的。设置 SVN_UPDATE 并使用 update 目标。例如，要更新 **/usr/src**：

```
# cd /usr/src
# make update SVN_UPDATE=yes
```

A.3.5.Subversion 镜像站

FreeBSD 的 Subversion 存储库是:

```
svn.FreeBSD.org
```

这是一个可以公开访问的镜像网络，使用 GeoDNS 来选择合适的后端服务器。要通过浏览器查看 FreeBSD Subversion 仓库，请使用 https://svnweb.FreeBSD.org/[2316]。

HTTPS 是首选协议，但需要安装 **security/ca_root_nss** 软件包，以便自动验证证书。

A.3.6.了解更多信息

关于使用 Subversion 的其他信息，请参见 `Subversion`，标题为使用 Subversion 进行版本控制[2317]，或 Subversion 文档[2318]。

A.4. CD 和 DVD 套装

FreeBSD 的 CD 和 DVD 套装可以从以下几个在线零售商那里买到:

- FreeBSD Mall, Inc.

  1164 Claremont Dr

  Brentwood, CA

  94513

  USA

  手机：+1 925 240-6652

  传真：+1 925 674-0821

  Email: info@freebsdmall.com

  WWW: https://www.freebsdmall.com

- Getlinux

[2316] https://svnweb.freebsd.org/

[2317] http://svnbook.red-bean.com/

[2318] http://subversion.apache.org/docs/

WWW: https://www.getlinux.fr/

- Dr. Hinner EDV

  Schäftlarnstr. 10 // 4. Stock

  D-81371 München

  Germany

  手机：+49 171 417 544 6

  Email: infow@hinner.de

  WWW: http://www.hinner.de/linux/freebsd.html

附录 B. 书目

虽然手册页为 FreeBSD 操作系统的每个部分都提供了确切的参考，但它们很少说明如何将这些部分联合起来，使整个操作系统顺利运行。在这方面，UNIX® 系统管理方面的好书或用户手册是无可替代的。

B.1. FreeBSD 参考书目

- **Absolute FreeBSD: The Complete Guide To FreeBSD**, Third Edition, published by No Starch Press[2319], 2018. ISBN: 978-1593278922
- **FreeBSD Mastery: Storage Essentials**, published by Tilted Windmill Press[2320], 2014. ISBN: 978-1642350098
- **FreeBSD Mastery: Specialty Filesystems**,published by Tilted Windmill Press[2321], 2015. ISBN: 978-1642350111
- **FreeBSD Mastery: ZFS**,published by Tilted Windmill Press[2322], 2015. ISBN: 978-1642350005
- **FreeBSD Mastery: Advanced ZFS**,published by Tilted Windmill Press[2323], 2016. ISBN: 978-0692688687
- **FreeBSD Mastery: Jails**,published by Tilted Windmill Press[2324], 2019. ISBN: 978-1642350241
- **FreeBSD Device Drivers: A Guide for the Intrepid**, published by No Starch Press[2325], 2012. ISBN: 978-1593272043
- **The Design And Implementation Of The Freebsd Operating System**, Second Edition, published by Pearson Education, Inc[2326], 2014. ISBN: 978-0321968975

[2319] https://nostarch.com/absfreebsd3

[2320] https://www.tiltedwindmillpress.com/product/freebsd-mastery-storage-essentials/

[2321] https://www.tiltedwindmillpress.com/product/freebsd-mastery-storage-essentials/

[2322] https://www.tiltedwindmillpress.com/product/freebsd-mastery-storage-essentials/

[2323] https://www.tiltedwindmillpress.com/product/freebsd-mastery-storage-essentials/

[2324] https://www.tiltedwindmillpress.com/product/freebsd-mastery-storage-essentials/

[2325] https://nostarch.com/absfreebsd3

[2326] https://www.pearson.com/store/p/design-and-implementation-of-the-freebsd-operating-system-the/P200000000463/9780321968975

- **UNIX and Linux System Administration Handbook**, Fifth Edition, published by Pearson Education, Inc[2327], 2017. ISBN: 978-0134277554
- **Designing BSD Rootkits**, published by No Starch Press[2328], 2007. ISBN: 978-1593271428
- **FreeBSD Jails using VNETs**, published in gumroad[2329].

B.2. 安全性参考文献

- **The Book of PF: A No-Nonsense Guide to the OpenBSD Firewall**, Third Edition, published by No Starch Press[2330], 2014. ISBN: 978-1593275891
- **SSH Mastery: OpenSSH, PuTTY, Tunnels, and Keys**, Second Edition, 2018. ISBN: 978-1642350029

B.3. UNIX® 历史

- 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[2331]
- 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[2332].
- Don Libes, Sandy Ressler *Life with UNIX* - special edition. Prentice-Hall, Inc., 1989. ISBN 0-13-536657-7
- *The BSD family tree.* https://cgit.freebsd.org/src/tree/share/misc/bsd-family-tree or /usr/share/misc/bsd-family-tree[2333] on a FreeBSD machine.
- *Networked Computer Science Technical Reports Library.*
- *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.
- Kernighan, Brian *Unix: A History and a Memoir.* Kindle Direct Publishing, 2020. ISBN 978-169597855-3

[2327] https://www.pearson.com/store/p/design-and-implementation-of-the-freebsd-operating-system-the/P200000000463/9780321968975
[2328] https://nostarch.com/absfreebsd3
[2329] https://rderik.gumroad.com/l/uwOLZ
[2330] https://nostarch.com/pf3
[2331] http://www.catb.org/~esr/jargon/html/index.html
[2332] http://www.simson.net/ref/ugh.pdf
[2333] https://cgit.freebsd.org/src/tree/usr/share/misc/bsd-family-tree

B.4. 期刊和杂志

- Admin Magazin[2334] (in German), published by Medialinx AG. ISSN: 2190-1066
- BSD Now - Video Podcast[2335], published by Jupiter Broadcasting LLC
- FreeBSD Journal[2336], published by S&W Publishing, sponsored by The FreeBSD Foundation. ISBN: 978-0-615-88479-0

[2334] https://www.admin-magazin.de/

[2335] https://www.bsdnow.tv/

[2336] https://freebsdfoundation.org/our-work/journal/

附录 C. 网络资源

FreeBSD 的发展太快，印刷媒体无法让人们及时了解信息。为了让人们了解 FreeBSD 的发展：最好的办法是用电子媒体代替印刷媒体。

FreeBSD 用户社区提供了大量的技术支持，其中论坛、聊天和电子邮件是最受欢迎和最有效的交流方式。

最重要的联系点概述如下。维基社区[2337]可能提供更多最新信息。

如果有重复的或尚未在下面列出的资源，请告知 FreeBSD 文档项目邮件列表。

C.1. 网站

- FreeBSD 论坛[2338]为 FreeBSD 问题和技术讨论提供了一个基于网络的讨论区。
- FreeBSD Wiki[2339] 提供了各种尚未进入手册的信息。
- 文档门户网站[2340]提供的内容远不止《FreeBSD 手册》，还有四十多本书籍和文章。
- FreeBSD 杂志[2341]是由 FreeBSD 基金会[2342]发行的免费的、经过专业编辑的、双月的技术杂志。
- BSDConferences YouTube 频道[2343]提供了一个来自世界各地 BSD 会议的高质量视频。这是观看主要开发者就 FreeBSD 的新工作进行演讲的一个好方法。
- FreeBSD 状态报告[2344]每三个月发布一次，跟踪 FreeBSD 的开发进度。
- 在 r/freebsd 有一个以 FreeBSD 为中心的 Reddit 小组[2345]。

[2337] https://wiki.freebsd.org/Community
[2338] https://forums.freebsd.org/
[2339] https://wiki.freebsd.org/
[2340] https://docs.freebsd.org/
[2341] https://freebsdfoundation.org/our-work/journal/browser-based-edition/
[2342] https://freebsdfoundation.org/
[2343] http://www.youtube.com/bsdconferences
[2344] https://www.freebsd.org/status/
[2345] https://www.reddit.com/r/freebsd/

- 超级用户[2346]和服务器故障[2347]，系统管理员的 Stack Exchange 服务。
- FreeBSD Discord 服务器[2348]，是一个通信和社区建设服务，FreeBSD 社区成员可以在这里进行社交，获得支持或支持他人，学习，贡献，合作，并保持与所有 FreeBSD 相关的最新信息。
- IRC 频道[2349]，广泛实施的、技术成熟的、开放标准的文本聊天。

C.2. 邮件列表

邮件列表是解决问题或向集中的 FreeBSD 受众展开技术讨论的最直接的方式。有许多关于不同的 FreeBSD 主题的列表。将问题发送到最合适的邮件列表中，必然会得到更快、更准确的回应。

技术列表主题应保持技术性。

所有 FreeBSD 的用户和开发者都应该订阅 FreeBSD 公告邮件列表[2350]。

> 注意
>
> 要测试向 FreeBSD 列表发送邮件的能力，请向 FreeBSD 测试邮件列表[2351]发送测试信息。请不要向任何其他列表发送测试信息。

当对向哪个列表发布问题有疑问时，请看如何从 FreeBSD-questions 邮件列表中获得最佳效果[2352]。

在向任何列表发帖之前，请一定要：

- 通过阅读邮件列表常见问题（FAQ）文件[2353]，了解如何最好地使用邮件列表，例如如何帮助避免频繁重复的讨论
- 搜索归档，以确定是否有人已经发布了您打算发布的内容

归档搜索界面包括：

- https://lists.freebsd.org/search (FreeBSD, experimental)
- https://www.freebsd.org/search/ (DuckDuckGo)
- https://freebsd.markmail.org/ (MarkMail)

请注意，这也意味着发送到 FreeBSD 邮件列表的信息将被永久存档。如果需要保护隐私，请考虑使用一次性的辅助电子邮件地址，并只发布公开信息。

FreeBSD 提供的归档文件：

- 不会特别标识超链接

[2346] https://superuser.com/questions/tagged/freebsd
[2347] https://serverfault.com/questions/tagged/freebsd
[2348] https://wiki.freebsd.org/Discord
[2349] https://wiki.freebsd.org/IRC/Channels
[2350] https://lists.freebsd.org/subscription/freebsd-announce
[2351] https://lists.freebsd.org/subscription/freebsd-test
[2352] https://docs.freebsd.org/en/articles/freebsd-questions/
[2353] https://docs.freebsd.org/en/articles/mailing-list-faq/

- 不显示内联图像
- 不显示 HTML 信息的 HTML 内容

可在此处[2354]查阅 FreeBSD 公共邮件列表。

C.2.1. 如何订阅或取消订阅

要订阅一个列表，请在https://lists.freebsd.org[2355]，点击列表名称。显示的页面应包含该列表的所有必要的订阅说明。

要实际发布到一个特定的列表，请发送邮件到 listname@FreeBSD.org。然后，它将被重新分配给世界各地的邮件列表成员。

C.2.2. 列表基本规则

所有的 FreeBSD 邮件列表都有一些基本的规则，使用它们的人必须遵守。如果不遵守这些规定，将导致 FreeBSD Postmaster postmaster@FreeBSD.org 发出两次书面警告，之后，如果是第三次犯错，发帖者将被从所有 FreeBSD 邮件列表中删除，并被过滤掉不再发帖。我们很遗憾这样的规则和措施是必要的，但今天的互联网是一个相当严酷的环境，似乎很多人都没有意识到它的一些机制是多么的脆弱。

列表的规则:

- 任何帖子的主题都应该遵守它所发布的列表的基本章程。如果该列表是关于技术问题的，帖子应该包含技术讨论。持续的无关紧要的闲聊或谩骂只会减损邮件列表对每个人的价值，是不能被容忍的。对于没有特定主题的自由讨论，FreeBSD 聊天邮件列表[2356]是免费提供的，应该使用它。
- 不应该在 2 个以上的邮件列表上发帖，而且只有在明确和明显需要在两个列表上发帖的情况下才可以。对于大多数列表，已经有大量的订阅者重叠，除了最深奥的混合（例如`-stable & -scsi`），真的没有理由同时发布到一个以上的列表。如果收到的邮件在 抄送栏里有多个邮件列表，请在回复前修剪 抄送栏。回复的人仍然要对交叉张贴负责，无论发起人是谁。
- 个人攻击和亵渎（在争论的背景下）是不允许的，这包括用户和开发者。严重违反网络礼仪的行为，例如在没有或不会得到许可的情况下摘录或转贴私人邮件，是不被允许的，但没有特别强制执行。然而，也有很少的情况下，这样的内容会符合列表的章程，因此它可能会被警告（或禁止），仅此而已。
- 严禁宣传非 FreeBSD 相关的产品或服务，如果明显是以垃圾邮件的方式进行宣传，将被立即禁止。

[2354] https://lists.freebsd.org/

[2355] https://lists.freebsd.org/

[2356] https://lists.freebsd.org/subscription/freebsd-chat

C.2.3. 邮件列表过滤

FreeBSD 邮件列表通过多种方式进行过滤，以避免垃圾邮件、病毒和其它不需要的邮件的传播。本节所描述的过滤操作并不包括用于保护邮件列表的所有操作。

邮件列表只允许某些类型的附件。所有 MIME 内容类型不在以下列表中的附件都将在邮件列表分发前被删除。

- 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

> 注意
> 有些邮件列表可能允许使用其他 MIME 内容类型的附件，但上述列表应适用于大多数邮件列表。

如果多部分邮件包含 text/plain 和 text/html 两部分：

- 收件人将同时收到两部分
- lists.freebsd.org 将显示 text/plain，并提供查看原始文本（源代码，其中包括原始 HTML）的选项。

如果 text/plain 没有与 text/html 同时出现：

- 将 HTML 转换为纯文本。

C.3. Usenet 新闻组

除了两个专门的 FreeBSD 新闻组之外，还有许多其他讨论 FreeBSD 或与 FreeBSD 用户相关的新闻组。

C.3.1. BSD 专用新闻组

- comp.unix.bsd.freebsd.announce
- comp.unix.bsd.freebsd.misc
- de.comp.os.unix.bsd (德语)
- fr.comp.os.bsd (法语)

C.3.2. 其他可能会感兴趣的 UNIX® 新闻组

- comp.unix
- comp.unix.questions
- comp.unix.admin
- comp.unix.programmer
- comp.unix.shell
- comp.unix.misc
- comp.unix.bsd

C.3.3. X 窗口系统

- comp.windows.x

附录 D. OpenPGP 密钥

D.1. 官方成员

FreeBSD.org 官方成员的 OpenPGP 密钥在此列出。这些密钥可以用来验证签名或向某位官方成员发送加密的电子邮件。FreeBSD OpenPGP 密钥的完整列表可以在 PGP 密钥[2357]文章中找到。完整的密钥链可以在 pgpkeyring.txt[2358] 下载。

D.1.1. 安全长官小组 security-officer@FreeBSD.org

```
pub   rsa4096/D39792F49EA7E5C2 2017-08-16 [SC] [expires: 2023-01-02]
      Key fingerprint = FC0E 878A E5AF E788 028D  6355 D397 92F4 9EA7 E5C2
uid                           FreeBSD Security Officer <security-officer@FreeBSD.org>
sub   rsa4096/6DD0A349F26ADEFD 2017-08-16 [E] [expires: 2023-01-02]
```

```
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBFmT2+ABEACrTVJ7Z/MuDeyKFqoTFnm5FrGG55k66RLeKivzQzq/tT/6RKO9
K8DaEvSIqD9b0/xgK02KgLSdp0Bucq8HLDFYUk3McFa6Z3YwjobNCWkxc72ipvVl
uAOGN4H6fuoYOpeg4cLK1H9pktUIrzONTCixaZzc/Bu6X+aX4ywGeCfsuu8g5v03
fLCPBLLgf3Bm5wsyZ6ZaGmsmILrWzd+d/rbr35Mcc5BekdgywUI4R191qo1bdrw9
mEJP1V7Ik3jpExOsNnuhMTvm5OQMeCTfUvVEOtBU15QtbT+1LXF5FIOgML0LwS5v
RHZN+5w/xvzSnEULpj24UuMKLDs/u9rj8U/zET8QaE+oG7m/mr4jJWZEmdX8HKdO
WrpnVj6UAppk72qdBIEfLsOW2xB/NOjJppbCQH3+sw7DRYA2UnKE9Mptj/KKiE4
cs4c8Cupo2WSu93lEZDC5rCrULpT2lFeEXnRYlC/5oIgY5w9sFide9VI4CzHkkWX
Z2NPW/i1w3mFhoXjvnNLGOYMfAMKPxsRC2/Bn3bY0IhKvuIZ4rAeu7FTmKDDqFKQ
YEcrUOW74ZVng17AB29xzjWr4zNJVvp/CybFiUb8JoKkwtVWRqAVZIEgenAjU40d
```

(continues on next page)

[2357] https://docs.freebsd.org/en/articles/pgpkeys/

[2358] https://docs.freebsd.org/pgpkeys/pgpkeys.txt

(continued from previous page)

```
G5+W4e+ccL0mfTQfEBbXRjnL2BL2tnaoBR42cTfbZGRucPHz7MrlKBEeZQARAQAB
tDdGcmVlQlNEIFNlY3VyaXR5IE9mZmljZXIgPHNlY3VyaXR5LW9mZmljZXJARnJl
ZUJTRC5vcmc+iQJUBBMBCgA+FiEE/A6HiuWv54gCjWNV05eS9J6n5cIFAlmT2+AC
GwMFCQoek4AFCwkIBwMFFQoJCAsFFgIDAQACHgECF4AACgkQ05eS9J6n5cKd9A/9
Fz3uGjNy28D0ALT1d/JJGzdQ2R3YwspHk9KHBr1LePkog9wf1WRalwCeNtPmA+g5
cn24psuzOeh1tRElImTZ2eE2ENPZ9XzK/J0ok0nK42MvmIwmMCyz+CaWv9GXW+FK
0oXnFmHi4YaQUVN3p+45TGkD9T+O5biVww7P47n/NnWsTfhLx0bzC7LyjPKXINai
/LgPgtlcOgY65/YhW/qhADCkoU7qMp9is41jMjTu1WB3OBPJkUkNpHfu6r15y8FN
Wqsk7K4W6Obr/WQ6VKGGXgh/a5mTcaEoFGMO16uHijAY4nXeb2HGZlBKxgmPH9Ur
aT4A9Pz/n+rIRMrK+rs+msFPemQHHNBYxy+x99uBpRBNyT2Su6GouZIxu5J16aIM
V0ZyOy/dy7m/uJ4sMhJPqKkd8a+MoQs/2L1M1y1EAzsO/QZqIrKrCluaftNN9k/B
qU0XClSDqB6sRMF7HFzYqb+f+M6cwSL/3Cp1Yx4rZ/onEE/MdWp64+3R87dETTXd
5tWXQw04qOhfPri5cBTI7r3t/qMO1iNXCGSG5RJbGkas6N6t6Mj83L4ItjI8doLf
aSIWZjj1XP3/me2hFJ6h2G5y5A+khO4ZwhC0ATFSq1fYbVGHw5AtfthIgNn8FoWu
+Sb8h7/RqTr7F6LgWagAoAh0GtVj02SVABZjcNZz/AKJAjcEEAEKACEWIQQc9/9v
rfXKn74bjLLtZ+zWXc9q5wUCWZPcTAMFAngACgkQ7Wfs1l3PauflkRAAgYcaBX0Y
ic4btxKoP/eOVpgUciOPPKEhDCiloQDyf4XQnZFDoMfjgcHpbLTBZ6kiAz2UzDGr
fJ4yUqrD+xfixUfCd5YpwzsaSpCGzDzSxOBcP/SpuAFhe40awSOIf5MruQar9Mlf
33JyslDLULXXeewAq2pcGk0/WrrOragI6Cs2vPGy9XP96VvLxyhjrWjlKmnO+//w
UF8oIO5hhKoqbtoxxlcqJgsWVyHch0mnPzvr6GWwoPhFXocnh1oPdbLjX1AwmGm9
ltEYMge4QxONIXlXJR0TvuDuJOaLNvTOC3OI8L97fdBcZS7eNJrG5FAYR5Ft3ISf
KJowIsSLGDt/cYApqpyP2pv7FpCvnwHgXHYar7/q4zhngCFRxQ2DPUx1cIJQ3Bgh
HZolKyK1X7XE5ZVDfZ3s3gcHSVKS89pipgHHZNr4sSmOanA8rXHcyHS4o2zSi1ie
r4iBwnOk6cCd6UNzEIiq0y/XhP/sc7xeL0mn3wDuV7jDBP9sp65sexL1qtIAfnzL
pLQevm0z41ifrUH5nNeL6RdbXpaoXc8M4PJJeQKJDu04KzLcQpZdUdCJsbS6QO9w
srWR8enQXPEhz2CO4L77bM9TgYO29222jTqEPcbXcmxF/klxO1rpssTTHUnHHi1Z
LUGYCbZPjt+laTJ2YPHTjUtN1Jw85vSKCEuJATMEEAEKAB0WIQS7KNQLNg7uk2rt
FW/l97zLo73d+AUCWjSYRwAKCRDl97zLo73d+JKyB/9N5Ytao12nD5QzMLvceGh5
otCLN99TUryYiDVDLoNkBivq3jHQA/hOX2rwEueFq0+LF8/2DnglJuUICNtCxIzL
WXXf/Hr5iWBUQ0JxYNPQzzjdMSXGE0WMwYVpAbCGxHpIsetKLdHUCwneYhaywe3I
KzmRJSDJGV1IJB0sAfoFtgybZXHgIR61jQjtnNmmyYXliYCd0wmIhXQDFN91tzzG
+EZdJ3Fao9JsMC+x55jO6EOLVySZgRF5E8vCeKUWemQciKFC7EhKcljILPYAA21u
NmHCAgRHKWU9JMdFK0w9lQuN2HQaNfkahjarTNM/Q6LwxY0dLG0vVYifE085WFAf
uQINBFmT2+ABEACxi39m5nQZexzY3c9sg/w5mUYCD89ZNSkj427gduQMYYGn7YW6
jSPfVJ/V3+PDK824c0a0XasyDapQFY1CPTZYrReRPoyjb8tJjsSVGXXCTFpJZlFU
br6kS9mgcx58Sypke2PMVk73+W1N1Yco+nahfTECRuM2/T2zHHr0AdKuBPF28U+H
TxyLatKoIgQwHDs4E/f4ZTbAoHvu3PixAl7XHVXCgz0cHaLhRljXizbZDXngOdGm
lqdFlAIpL6/l8E3m1Er0m3IfFo6qSzWRHg/KaBGIL4YKetJ6ACjlkCe5qbatDpmk
gWlg3Ux4RBVjyCK834Xh7eZpEcNf2iwpm28glWh7XMHGUplTHkU3PWQ4vGfNxXB8
HBOd9r02/cHL6MiHwhCAfIzZGVtqR0i9Ira57TMdXTpJWNXUcgsCMsi/Bg2a+hsn
aiYLrZc18uNL5nqOqsqKG3c1TcmeN7nbxVgnrNST4AjteulkhmB9p8tNOXA3u979
OO0T5LPwdqIpobdZ0lfw4URnAGw4Wd4Sm9PtRw0RvuAk2M2e5KXNyxPWAuMVkoRR
a7wG6h/R8pki54Gexyc+JkfB4ZcOrzHNLurw6DhxroyfRs8WEgX0wNIGmJvCXSBG
```

(continues on next page)

(continued from previous page)

```
54jb5w9qudYwzIg4YPfvuX8sfeY8MTNhal3rF0tvVloGj3l709wlaWlBYwARAQAB
iQI8BBgBCgAmFiEE/A6HiuWv54gCjWNV05eS9J6n5cIFAlmT2+ACGwwFCQoek4AA
CgkQ05eS9J6n5cKhWw/+PT0R4r2gPAxI8ESEe380BYOmneNAH24MFOgWXqWCj4zX
Uz992BVnW2aL5nH4O5d822LGeCrYUC7SCpQvlifdHZHjobgtizLTwuu40bc3gSOz
cxWlx2jKfx3Ezn6QQz2mhhK6fZ1AO0ObiQxQq25ldURep95L78E/C8XkCe11YlUR
ng3wQKeHM7awZWRw/QBC92haHuVtU3cx7At+zQL7jTBKSZqd34zzs0uoXIhk2h94
O07MMDZ8z8MeU337vdL+RKYtD2bljLwpf7/kqg1D/q44RJ4ZpZcha9G0GvtLaQg2
+MAPlLg1vOWZ8wOTLaQHm+uzYRpkqxkIV8OuVd4UikCd8t3VNjNG5rG/YRNIAX0A
UEzs6oMF5YOFE8LmykesbUHAbC07Vcb0AsT5u3XKixDiIpPdnYSwGlkvoOVVLdeh
q/aXLK9V8BpViG5+a8xP2fdF1eMqdnrKAsiO4GEiq193PN/FA049VeIs3fd0izAa
x7+ag1MGtoF5Pij5iTVJm6phH5SUd1P3FY3OmclxWj/MbL4ba/G/6FWcy5NXxdw9
L1bRqaM2KEHJ67aF6NZz7UMldwExAWzFbUon1LUpKysAukxVf0EnntydBeVOQ+JO
HdqEpirrVLMpxPttUB2xxbo947nMj7/Bnme2gvb0vxaC9xSGVxrpW9cg5iCwSdc=
=8rds
-----END PGP PUBLIC KEY BLOCK-----
```

D.1.2. 安全小组秘书 secteam-secretary@FreeBSD.org

```
pub   4096R/3CB2EAFCC3D6C666 2013-09-24 [expires: 2018-01-01]
      Key fingerprint = FA97 AA04 4DF9 0969 D5EF  4ADA 3CB2 EAFC C3D6 C666
uid                          FreeBSD Security Team Secretary <secteam-
↪secretary@FreeBSD.org>
sub   4096R/509B26612335EB65 2013-09-24 [expires: 2018-01-01]
```

```
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBFJBjIIBEADadvvpXSkdnBOGV2xcsFwBBcSwAdryWuLk6v2VxjwsPcY6Lwqz
NAZr2Ox1BaSgX7106Psa6v9si8nxoOtMc5BCM/ps/fmedFU48YtqOTGF+utxvACg
Ou6SKintEMUa1eoPcww1jzDZ3mxx49bQaNAJLjVxeiAZoYHe9loTe1fxsprCONnx
Era1hrI+YA2KjMWDORcwa0sSXRCI3V+b4PUnbMUOQa3fFVUriM4QjjUBU6hW0Ub0
GDPcZq45nd7PoPPtb3/EauaYfk/zdx8Xt0OmuKTi9/vMkvB09AEUyShbyzoebaKH
dKtXlzyAPCZoH9dihFM67rhUg4umckFLc8vc5P2tNblwYrnhgL8ymUaOIjZB/fOi
Z2OZLVCiDeHNjjK3VZ6jLAiPyiYTG1Hrk9E8NaZDeUgIb9X/K06JXVBQIKNSGfX5
LLp/j2wr+Kbg3QtEBkcStlUGBOzfcbhKpE2nySnuIyspfDb/6JbhD/qYqMJerX0T
d5ekkJ1tXtM6aX2iTXgZ8cqv+5gyouEF5akrkLi1ySgZetQfjm+zhy/1x/NjGd0u
35QbUye7sTbfSimwzCXKIIpy06zIO4iNA0P/vgG4v7ydjMvXsW8FRULSecDT19Gq
xOZGfSPVrSRSAhgNxHzwUivxJbr05NNdwhJSbx9m57naXouLfvVPAMeJYwARAQAB
tD9GcmVlQlNEIFNlY3VyaXR5IFRlYW0gU2VjcmV0YXJ5IDxzZWN0ZWFtLXNlY3Jl
dGFyeUBGcmVlQlNELm9yZz6JAj0EEwEKACcFAlJBjIICGwMFCQgH7b8FCwkIBwMF
FQoJCAsFFgIDAQACHgECF4AACgkQPLLq/MPWxmYt8Q/+IfFhPIbqglh4rwFzgR58
8YonMZcq+5Op3qiUBh6tE6yRz6VEqBqTahyCQGIk4xGzrHSIOIj2e6gEk5a4zYtf
```

(continues on next page)

(continued from previous page)

```
0jNJprk3pxu2Og05USJmd8lPSbyBF20FVm5W0dhWMKHagL5dGS8zInlwRYxr6mMi
UuJjj+2Hm3PoUNGAwL1SH2BVOeAeudtzu80vAlbRlujYVmjIDn/dWVjqnWgEBNHT
SD+WpA3yW4mBJyxWil0sAJQbTlt5EM/XPORVZ2tvETxJIrXea/Sda9mFwvJ02pJn
gHi6TGyOYydmbu0ob9Ma9AvUrRlxv8V9eN7eZUtvNa6n+IT8WEJj2+snJlO4SpHL
D3Z+l7zwfYeM8FOdzGZdVFgxeyBU7t3AnPjYfHmoneqgLcCO0nJDKq/98ohz5T9i
FbNR/vtLaEiYFBeX3C9Ee96pP6BU26BXhw+dRSnFeyIhD+4g+/AZ0XJ1CPF19D+5
z0ojanJkh7lZn4JL+V6+mF1eOExiGrydIiiSXDA/p5FhavMMu8Om4S0sn5iaQ2aX
wRUv2SUKhbHDqhIILLeQKlB3X26obx1Vg0nRhy47qNQn/xc9oSWLAQSVOgsShQeC
6DSzrKIBdKB3V8uWOmuM7lWAoCP53bDRW+XIOu9wfpSaXN2VTyqzU7zpTq5BHX1a
+XRw8KNHZGnCSAOCofZWnKyJAhwEEAEKAAYFAlJBjYgACgkQ7Wfs1l3PaudFcQ//
UiM7EXsIHLwHxez32TzA/0uNMPWFHQN4Ezzg4PKB6Cc4amva5qbgbhoeCPuP+XPI
2ELfRviAHbmyZ/zIgqplDC4nmyisMoKlpK0Yo1w4qbix9EVVZr2ztL8F43qN3Xe/
NUSMTBgt/Jio7l5lYyhuVS3JQCfDlYGbq6NPk0xfYoYOMOZASoPhEquCxM5D4D0Z
3J3CBeAjyVzdF37HUw9rVQe2IRlxGn1YAyMb5EpR2Ij612GFad8c/5ikzDh5q6JD
tB9ApdvLkr0czTBucDljChSpFJ7ENPjAgZuH9N5Dmx2rRUj2mdBmi7HKqxAN9Kdm
+pg/6vZ3vM18rBlXmw1poQdc3srAL+6MHmIfHHrq49oksLyHwyeL8T6BO4d4nTZU
xObP7PLAeWrdrd1Sb3EWlZJ9HB/m2UL9w9Om1c6cb6X2DoCzQAStVypAE6SQCMBK
pxkWRj90L41BS62snja+BlZTELuuLTHULRkWqS3fFkUxlDSMUn96QksWlwZLcxCv
hKxJXOX+pHAiUuMIImaPQ0TBDBWWf5d8zOQlNPsyhSGFR5Skwzlg+m9ErQ+jy7Uz
UmNCNztlYgRKeckXuvr73seoKoNXHrn7vWQ6qB1IRURj2bfphsqlmYuITmcBhfFS
Dw0fdYXSDXrmG9wad98g49g4HwCJhPAl0j55f93gHLGIRgQQEQoABgUCUkGO5gAK
CRAV1ogEymzfsol4AKCI7rOnptuoXgwYx2Z9HkUKuugSRwCgkyW9pxa5EovDijEF
j1jG/cdxTOaJAhwEEAEKAAYFAlJBkdUACgkQkshDRW2mpm6aLxAAzpWNHMZVFt7e
wQnCJnf/FMLTjduGTEhVFnVCkEtI+YKarveE6pclqKJfSRFDxruZ6PHGG2CDfMig
J6mdDdmXCkN//TbIlRGowVgsxpIRg4jQVh4S3D0Nz50h+Zb7CHbjp6WAPVoWZz7b
Myp+pN7qx/miJJwEiw22Eet4Hjj1QymKwjWyY146V928BV/wDBS/xiwfg3xIVPZr
RqtiOGN/AGpMGeGQKKplkeITY7AXiAd+mL4H/eNf8b+o0Ce2Z9oSxSsGPF3DzMTL
kIX7sWD3rjy3Xe2BM20stIDrJS2a1fbnIwFvqszS3Z3sF5bLc6W0iyPJdtbQ0pt6
nekRl9nboAdUs0R+n/6QNYBkj4AcSh3jpZKe82NwnD/6WyzHWtC0SDRTVkcQWXPW
EaWLmv8VqfzdBiw6aLcxlmXQSAr0cUA6zo6/bMQZosKwiCfGl3tR4Pbwgvbyjoii
pF+ZXfz7rWWUqZ2C79hy3YTytwIlVMOnp3MyOV+9ubOsFhLuRDxAksIMaRTsO7ii
5J4z1d+jzWMW4g1B50CoQ8W+FyAfVp/8qGwzvGN7wxN8P1iR+DZjtpCt7J+Xb9Pt
L+lRKSO/aOgOfDksyt2fEKY4yEWdzq9A3VkRo1HCdUQY6SJ/qt7IyQHumxvL90F6
vbB3edrR/fVGeJsz4vE10hzy7kI1QT65Ag0EUkGMggEQAMTsvyKEdUsgEehymKz9
MRn9wiwfHEX5CLmpJAvnX9MITgcsTX8MKiPyrTBnyY/QzA0rh+yyhzkY/y55yxMP
INdpL5xgJCS1SHyJK85HOdN77uKDCkwHfphlWYGlBPuaXyxkiWYXJTVUggSjuO4b
jeKwDqFl/4Xc0XeZNgWVjqHtKF91wwgdXXgAzUL1/nwN3IglxiIR31y10GQdOQEG
4T3ufx6gv73+qbFc0RzgZUQiJykQ3tZK1+Gw6aDirgjQYOc90o2Je0RJHjdObyZQ
aQc4PTZ2DC7CElFEt2EHJCXLyP/taeLq+IdpKe6sLPckwakqtbqwunWVoPTbgkxo
Q1eCMzgrkRu23B2TJaY9zbZAFP3cpL65vQAVJVQISqJvDL8K5hvAWJ3vi92qfBcz
jqydAcbhjkzJUI9t44v63cIXTI0+QyqTQhqkvEJhHZkbb8MYoimebDVxFVtQ3I1p
EynOYPfn4IMvaItLFbkgZpR/zjHYau5snErR9NC4AOIfNFpxM+fFFJQ7W88JP3cG
JLl9dcRGERq28PDU/CTDH9rlk1kZ0xzpRDkJijKDnFIxT2ajijVOZx7l2jPL1njx
```

(continues on next page)

(continued from previous page)

```
s4xa1jK0/39kh6XnrCgK49WQsJM5IflVR2JAi8BLi2q/e0NQG2pgn0QL695Sqbbp
NbrrJGRcRJD9sUkQTpMsLlQTABEBAAGJAiUEGAEKAA8FAlJBjIICGwwFCQgH7b8A
CgkQPLLq/MPWxmZAew//et/LToMVR3q6/qP/pf9ob/QwQ3MgejkC0DY3Md7JBRl/
6GWfySYnO0Vm5IoJofcv1hbhc/y3OeZTvK4s+BOQsNokYe34mCxZG4dypNaepkQi
x0mLujeU/n4Y0p0LTLjhGLVdKina2dM9HmllgYr4KumT58g6eGjxs2oZD6z5ty0L
viU5tx3lz3o0c3I9soH2RN2zNHVjXNW0EvWJwFLxFeLJbk/Y3UY1/kXCtcyMzLua
S5L5012eUOEvaZr5iYDKjy+wOxY4SUCNYf0GPmSej8CBbwHOF2XCwXytSzm6hNb3
5TRgCGbOSFTIy9MxfV5lpddQcdzijmuFSl8LySkL2yuJxjlI7uKNDN+NlfODIPMg
rdH0hBSyKci6Uz7Nz/Up3qdE+aISq68k+Hk1fiKJG1UcBRJidheds29FCzj3hoyZ
VDmf6OL60hL0YI1/4GjIkJyetlPzjMp8J7K3GweOUkfHcFihYZlbiMe7z+oIWEc7
0fNScrAGF/+JN3L6mjXKB6Pv+ER5ztzpfuhBJ/j7AV5BaNMmDXAVO4aTphWl7Dje
iecENuGTpkK8Ugv5cMJc4QJaWDkj/9sACc0EFgigPo68KjegvKg5R8jUPwb8E7T6
lIjBtlclVhaUrE2uLx/yTz2Apbm+GAmD8M0dQ7IYsOFlZNBW9zjgLLCtWDW+p1A=
=5gJ7
-----END PGP PUBLIC KEY BLOCK-----
```

D.1.3. 核心小组秘书 core-secretary@FreeBSD.org

```
pub   rsa4096/BE3DF7A86914D607 2022-07-29 [SC] [expires: 2024-07-28]
      Key fingerprint = E0C0 73CF 01A2 A902 800C  3680 BE3D F7A8 6914 D607
uid                           FreeBSD Core Team Secretary <core-secretary@freebsd.
↪org>
sub   rsa4096/7882C7A2CA320B52 2022-07-29 [E] [expires: 2024-07-28]
      Key fingerprint = 7828 1422 F522 802B 00AE  0410 7882 C7A2 CA32 0B52
```

```
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBGLkULYBEACsS9RbAv8gIyZWtIWgBeK6+ircHRW0LsetvHqQYlY6gfRWDLN+
467o0dHwAz4c1jyq7Or+1gy2Kr2VpcPZr1kjNTx5NbvoybQJMIMs77o9LS3Q2pA0
3Dpi8LSaM77rCmIXFmKEspbuPyjjTjKUtpOmzvwMDq8ke7ZHrqkJOessKQVGUSJc
o+4hcN6S4gGRgzRHLOuPtDqIfxFuHjr+4ZEGeispJHZqtl+HwBOEdoG966hKo/Ae
eyIRMB1vioqf8GHuiVHZ3YJbbcLN/4oMMtN/aIgucfspKo2O95zUETkimGTBlEl0
RVFXF5kq3gq2p7+7Sl3OQ3Kw8LQ9ueinWNJ5/3X06UowsRbOxtx7Jtp4DFwhpHj/
LTdEUSjiVzeZKiqDOqvgjq2ZN4hLHEZYan3mv55AUnwzwzMD04P42mNHetCJnuNP
ZGGL/wmABc8X4tx/fGddECKBzCM8hHBG7WQkDUnMTpODhBXCcC+rm6Vz7FS1zv67
CftMneqWnDDZtf/XclnHI6iOJZPcY/ljV+QxGs+oLvn2mlR6xzHu9osYfGuozA6x
PIxrSgB7PWFSuSvqtN7fSwAiXOAI5zFpZ4HP8wjFt7SWfMaovc/FR8rzYaZSZYAk
0l+FmsMbBGvkHNdPyk/4CWqj3dg4HCFAeqUWVLo3qKHdTOQaTqavyYYBCwARAQAB
tDhGcmVlQlNEIENvcmUgVGVhbSBTZWNyZXRhcnkgPGNvcmUtc2VjcmV0YXJ5QGZy
ZWVic2Qub3JnPokCVAQTAQoAPhYhBODAc88BoqkCgAw2gL4996hpFNYHBQJi5FC2
AhsDBQkDwmcABQsJCAcDBRUKCQgLBRYDAgEAAh4FAheAAAoJEL4996hpFNYHPRwP
```

(continues on next page)

(continued from previous page)

```
/R5lWY8RcsAxAwXqzCRww1N2D6aRVUK+wJfxlMYRFjT9o0phmlSQxhORJATbjLm3
prLkpFBsMOScDmG7kWttPSHoQFKBiA7IznVeT4hka9c3Id21EL54GEjjDyp6AFev
G5kNCu8vS6SxmyUD9U2Q7PiEiEq7907tfydJfJ5BEzS5Az6KTOITaZ716qxQVQFR
6i6ChMCBABT059QngTiRp0sY2TPzTepHxEyrE//8M0mgyBsaRWPQP712sVujjx82
/3fXMxKwnJTGRhy0qR+DbIeSK/OiU4OJISG1XG/IkAQEqPwG6UilXy60l5PectJD
FAGRgky/Jmh9QTL4lhFLmpEyQhjVOZANZ1lqfD6lEjsf/Adr3stcKLNMLT8xewKo
LrFzSWMXh35HsSM/ZJng9CiJlAckGNBwuYp+5z49vBWaUXj9/KzK+0uKO6OspDIG
sF7M33BOqOP48ssIZWdJihwMA8qSX+ZFq+yMn8YdPmszXcEe4H4U1curdGWMm7IF
DwUl9cMEfYiPhvuO5taJBDerEbynyMI6oYbFnfl3Bb4rknlatiCzKXrzR1OuRtho
RkAPVSnEru0FTgCHToRdj81qyAwa6VMsEtVvqhNtsWBvr8W9Bdj1zZUV0hCJOzZN
UfAmlRXkud8lK4UyxBHUrNSy4ufGRjMNOhuUmBZVwrTSiQIzBBABCgAdFiEES2Tp
4L3ps+zAa1xm2MjIO0nybxcFAmLkVMoACgkQ2MjIO0nybxdeBBAAhCqcnsVVUpL6
w9CQV71kkSoT0649GkbWeG+ob1XgXvjxCSRb7mxSx7KCiUkLtnzVUOe+qp15pbAm
o9zlHZ/yQQxA3pUX5npIIiWXSWknA3DGNKE52RJ1tbpIVhFjrseXa4gsrxrUVtCF
dHsCFl/G6Zr1h+OutL6DHxQH0ZI2OmxRQFxE0Bwjyx3HdLmtKCEZVYUuhMoCya9m
5Uu98nQOMH7jDMbj8za7JJiwZwjKNZfiNck8Ekq0DEXr3CKSueobHr3JqSLt9VKV
WzaGKDQ/sVQz4L7GZfR1yMAJQlWulsTvxGG4SWxnFsus9GNydQIEmkTfDLmBXkl4
PNNsTirNir3ld7KEIY07jffqK88uV3nZ+Cc2VUdM4GRddH4LdJnKyY+O8AXrEAse
9aUmWGmesbOSAV0ATLGfFM96eFuGMdgwxl25W3RxKlLABAdLRd5smW5RCDOSyrqZ
0F3mpqCGMG84Us93/G5hdYsULuJxlufMur4SO5kRp6h/+tuDR1TAzkAoHut16ODD
fnto8OhmLk1W3yfddh8i5e5jBU1NZkmHSYOKhs0iBcSBR0t594E3xNCUQYNZswlZ
rVNCL7o1HEuln0p+qrcOVWqQ9ovb4mE5qupRydr4+moxmJy6y5bQp3QaIxentHnY
1aq8A8o2OV8sFNbH70p7+fB+W1HeviS5Ag0EYuRQtgEQAMT6Pik/xPzARb/UjjVS
Wvo7AQjtLC3yKNg1yAey3T0gp6YKYs5vlMJckS48LDZagiqgcDucly52nk2sMWqu
+yllgBLrwm+SY4g7hqrDgXrOspZUyLysKB5fyF60qOGcjfmZgAFPh8MN4Zym/tD1
3dThrSsBJJ9jrX8OCBLlV5sXbbpx6jwtLe4wzeJOfMctW+U3U6zmJw8ZOYU7cG/M
5xSh1s9W1iju4DXo63gOtnyYad27BexHu19e/nAwxQwLaofDX9R9Y0pORFHI1SuO
IrQIHXhwgZHVRBNulPtM2zVVN1jWC5X9YLu4rc/F01BO0B4GQosYtlmcK7Rm7obx
Dm+o0pPw3xyFnl1vOXWsmRDUP6qATk3YKuHdYRe33SxFPm7iWEB+rVL41dqquMRO
L7HIt9ho9MWTac2a9jHIX17xK9Q/P/zy6ZLjwtcyirPezct4GWvIPJ+mdrmI39nn
Y/TFcBZ3G0BtwasFnuFjHbkjcBlqvtHc1Zg/hISaEDTbSDr0TMLpxG2OpT5xqMkq
PO8S04IpMtKWd1aRliv9vE65UHAGHVe8EOVmGT7Tk9cCqFxrxpwE8n+KU5JGUGTc
BnpuCedT3txnZZc90d/+yMotJPpLUmOdndj782Y4B5y5JXpTefvxN2lBOorVN1xu
zH43SkAA7lvuN0x4QpmzWJoLABEBAAGJAjwEGAEKACYWIQTgwHPPAaKpAoAMNoC+
PfeoaRTWBwUCYuRQtgIbDAUJA8JnAAAKCRC+PfeoaRTWB/RdD/94yvoMl/VtJqTR
Lc7Qu41y/SdczDAfCGPts49uu56xRqCfcLZOLr4PNXh49x0UWFroTcpcFlcsS++D
EOoR2DoyxB1+KKRhceBo1CmQ8Y7RQ0LpQzYPkqBmzVEZK/I5dKf+RX83E7sa5L28
UpCrsDSsp7dVrxmwdiiOsJBD1vA5vIGgtgTGewMNmPO4wbhmjFTIxPnSND6wRYyW
SBgKkEz5lnA6zHOMYiXabKI/oY23xq4YRu2UOVoZMUkXoqR09McwrqMI4+XJBSwC
c2G6FI4uEO6YIDGXlUZhCGDGwE+HVcP6/jshyi1HOtUpcemle//YSvyrp6N/8XkZ
RX/dvUNIBl+ykIE/wb75PWI7QTNLWkJmCU/ft9m1KEHAwccyxHJxXWurnBbMgMan
VfLYH4uJ0eH/O65zTIzRdZcMO4kY1vVllAKXY9Httxpdua0n+4rHplxL4ZfRL7Y4
5h7/Xiz5xfGcYxPd5/ezfYzcvfDr4danX8fBe7U9F2VaO/QOhcgCLV8TevR1Ku/0
```

(continues on next page)

(continued from previous page)

```
GUOfPAH6rhZaLqqz92Y2hOX1QQ6MabB92DUFZh+5SUxCzqI6cAiH60Rbf9ZI579s
L32GpxZ6BPISnsy69SNAVBiczw8EthEY1KhdN9QOuHppqcGsOlIz8cKVojqzILwj
GT6wtOZXl/ri7I8x3Fr89V3sUvmg1w==
=2Il1
-----END PGP PUBLIC KEY BLOCK-----
```

D.1.4. Ports 管理小组秘书 portmgr-secretary@FreeBSD.org

```
pub   rsa2048/D8294EC3BBC4D7D5 2012-07-24 [SC]
      Key fingerprint = FB37 45C8 6F15 E8ED AC81  32FC D829 4EC3 BBC4 D7D5
uid                           FreeBSD Ports Management Team Secretary <portmgr-
↪secretary@FreeBSD.org>
sub   rsa2048/5CC117965F65CFE7 2012-07-24 [E]
sub   rsa4096/CA20328577064EB7 2013-10-05 [S]
sub   rsa4096/8B114B3613867E00 2013-10-05 [E]
```

```
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQENBFAOzqYBCACYd+KGv0/DduIRpSEKWZG2yfDILStzWfdaQMD+8zdWihB0x7dd
JDBUpV0o0Ixzt9mvu5CHybx+9lOHeFRhZshFXc+bIJOPyi+JrSs100o7Lo6jg6+c
Si2vME0ixG4x9YjCi8DisXIGJ1kZiDXhmVWwCvL+vLInpeXrtJnK8yFkmszCOr4Y
Q3GXuvdU0BF2tL/Wo/eCbSf+3U9syopVS2L2wKcP76bbYU0ioO35Y503rJEK6R5G
TchwYvYjSXuhv4ec7N1/j3thrMC9GNpoqjVninTynOk2kn+YZuMpO3c6b/pfoNcq
MxoizGlTu8VT4OO/SF1y52OkKjpAsENbFaNTABEBAAG0R0ZyZWVCU0QgUG9ydHMg
TWFuYWdlbWVudCBUZWFtIFNlY3JldGFyeSA8cG9ydG1nci1zZWNyZXRhcnlARnJl
ZUJTRC5vcmc+iQE4BBMBAgAiBQJQDs6mAhsDBgsJCAcDAgYVCAIJCgsEFgIDAQIe
AQIXgAAKCRDYKU7Du8TX1QW2B/0coHe8utbTfGKpeM4BY9IyC+PFgkE58Hq50o8d
shoB9gfommcUaK9PNwJPxTEJNlwiKPZy+VoKs/+dO8gahovchbRdSyP1ejn3CFy+
H8pol0hDDU4n7Ldc50q54GLuZijdcJZqlgOloZqWOYtXFklKPZjdUvYN8KHAntgf
u361rwM4DZ40HngYY9fdGc4SbXurGA5m+vLAURLzPv+QRQqHfaI1DZF6gzMgY49x
qS1JBF4kPoicpgvs3o6CuX8MD9ewGFSAMM3EdzV6ZdC8pnpXC8+8Q+p6FjNqmtjk
GpW39Zq/p8SJVg1RortCH6qWLe7dW7TaFYov7gF1V/DYwDN5iEYEEBECAAYFAlN2
WksACgkQtzkaJjSHbFtuMwCg0MXdQTcGMMOma7LC3L5b4MEoZ+wAn0WyUHpHwHnn
pn2oYDlfAbwTloWIiQEiBBABAgAMBQJQGiE/BQMAEnUAAAoJEJcQuJvKV618AhwH
/ie5fi+wL0aapUHHs454xUv8xtdPfKpA35U4R2ZaVZ6wTCWWl3by+i81YIiGFVmQ
uQkkms1vRFhY6fVYzOxQR3VhWTTFexgLlLdI88eoBlzlsIvv7/4bNT9dUcg2TeYS
Ah3TzZsmVbqXIg8XvrCBD/WhG1cXGonHKszP6RcyFSDDD+bQogONqujM94dIcuoZ
7VCqbUvFKJ+rI3uXA1XFZSgFI9cDtnKYQqpGJcDEH2VeSfX/4XFuTg64g93AcCvE
qHANgquSWBfJGAy9fqV51LrRp7wB1GLfO0Vr0HSpn148wXDBSdi5T1AdfiBb8ENd
1fUfI/QfxHS5klkGE/akQNqJAhwEEwEKAAYFAldVzKcACgkQrbv4YQo3ibdgsg//
SS3vgpNtwiZHkUhfKAvuEhTlHUgJno0s55JT8JhApX3wAWHf5HJDtU+rpvthVHF7
```

(continues on next page)

(continued from previous page)

```
iWUBNZyilDb2fKQVsLJtUQx7Mq6Cov1wddGtU3cf675VPPMrW3i31Ai86Aq3+qZD
EFqFOv96kunpVdFsDpQ7n5Pl4w9bI2tsLrLIRvMvtVggOSYz962XQ/TVXc6OQ4JK
LU0VH7mDdSySDeM5Q4B4+xVT+8Fu/kdHGRaDojbCyQgNBTfEoslPfFMrI2sC4iGP
XUyc+l1QNA4AdE2Vjt5log4yE3iq3RnKnzzYiYZr1CgZJ9O2KUzzX5KE0/6f6uQo
WJrtLmxnsktRCyo5pJSfg88p32S87512FzYKOXJHrqhKlDcTNCet7/JZn4Q0UThb
3qh1nadFzMV1ZkeWzLp0pXYORGbKC8kGzfknCD5w0wBDxqwrX9LfOEUMTUHhE+v1
gW2X83kngBZqQRQnYnLGejI8LcI6+PqNeZ9UDE2XKNTWk6n/L18M5fcDOGmLYNTp
BeM/B3R7sudAhwnreJk+XiYMGVHxb13WzKR16axCNGPTji+GjIpwtXoBd3V7f9J+
e4ughPhNpN1CTzbzK7adB2gxJymfHH5AVBAc5FGIucMKp7R1diUYrKqEZx3byU+y
JpRzHu1mibInwSvO+D506GioPD+itU6Yfhci8VC2mI2JAhwEEgEIAAYFAlQYNs8A
CgkQ6rA8WL/cR49ZJRAA12CXRwt87Hce8A7CbLbVncpHGaDnf1VClHTdAQFO9py1
dLS5+LhYxEwOjIR2Vcc8CA03iwfJKshKXH2Z6FW6iSFCc9bCVzHTrcuwsGc5WZQM
JlGiHxYmsyWNad/fCW8mH0q1SjTqcPcApg2+1j/XQa4rcL42i87vB0LIs7TtEv+Y
WvxA301N6qHAVE42wrRkfS6gJjxOF/PpYhdhH2TBKCXd6yVrGL9sv5OqUoAvzKUH
sJ8q+n/iUP+/NVKGQej6cQsX7uA+sZTsnREQVIPhXwwFmEfil/gxjmpUZVMmks/D
b8ZfP29CJvwEIcfWn6shGcSA6YOds7fIybd8Mdt2D/BEBlIMBfKSK498BUgUht2v
3LJIgTVCIQoSEfYUK1T1GoTf48ewxgBsZPymvarERmEx30q/Y401EPobX3hQUQ0d
J5GSjKOflpP3AHASlWz5QacL580934fsC3BdxKwW8eY6fgaBXnvA+gE3lL0MTHIF
Wwz1JIDLMtxLSjFTtdI4SKF6FVfkoPRmfMzBMtbTkSp9M7HdmYFwtrESTpubbQkn
WEOHXWzNQPY4OrYkceIREtehg8sL5C/D9jPfqsiH40MYxlN93/6/M6fQ8jU9VZa7
cIRcbTqDHZq7G2Dgk42uF4NeIStYUWnNJBge24+sPe6fLjK0E9z5UjLaMq0cqO25
AQ0EUA7OpgEIAMzcSWKX+T66lfKXUV5J9+hXuOauu9WLW2dlq0+w9AlWfGfgGbMZ
z/7YmNUqzNjHqQngNsQQIShrFgVqNWkZOZn2TpLCrIl7Itl3gVYXWZQpKzjCgU1Y
mp+yNZR+fWZzSDF9DZdLZY9y9SI3lljb33o+N1EVfNUvRJksKY2lm6Gra9gPtaCZ
LuZu5zw/SSDnJBNU8wbMoce8yFDfcOjCq+4BcPAws8qcFCsmICmE6FZvEJZE/y9w
dH7kB1oU+kThpOKPEgu9YmubXNU81MxhGdQEDvG2525iSwQyr3lhwqGngJEcyDc6
A/cAJ/c+KZ4pc3ufULSgtXDqlJLE7wUg4esAEQEAAYkBHwQYAQIACQUCUA7OpgIb
DAAKCRDYKU7Du8TX1Th6B/4uVcoIFXcKaj1/GAZjeZl0M4mGTUKIjjfCITrMaV+F
tp1wo+ZujAAlepyQ1VesMlB/P1bw0AvZx2qwntaGjbB1yc9B2pmLeYBydBtW2LgX
LiafxRIy2uPFxSO510IyR5/K5cGTIyvz1o9ZcNlmYnpL4rs0xAx/HN2ErRiUmvoG
HvBMqt1yVQHtlUDoSJqFoBg0d7Vn4K8mvyHRBW/45ox0huV1H2yNLzcKpX/N9gak
kCjxv2YseDJgSYLwM+Ee9Tx/FgZm5jRYw3pcVcn8/NPojIBG/Wi/dTH3gTQtigda
RZoPdZkE7Iqu2o8FS1SSvDaZRwJHS5pAQjvDIDI8xoZRuQINBFJPlYgBEAC5Ao/e
SZr5n4MrTsZ0dcPMZ0AblMbWMX/gVsKlNCHzjAcQJlw0X233vQXNww2coJo62dEQ
g3FSeDpgzUXP776E7LfO9LKOPXTNYdoumllIYQ+B8nDY5MdMACMmfY14nU9y8gIQ
k0n/xo7aXDkQr5gLF3CR84YWyeXjNkgZF5jXZ+DHgv4Qm31H8zeuH5rBkeLSiiJT
z2sskGkL/3Tj96HxIeElkXPCrYxAOriMh87VjFhCUCjf+50nlbsA6VSS/ZeMlBgd
E6pStKf53nKEJDqJZx1F+aCuxJop/U4z2AhqmABFrjNQW1l4mCHYr5RJt2VPr6qu
gm0sP0ig2dmbiuFMwxkJ1u1snLG16mdfGnsIdZnX/ufmAcXl2s0Sb3xWqQf1r7Va
ovOCzloqK9SATDINzwsgYgo4R8F76FeVejN5bYa51YMogk63i170yzYCCkEBVL8F
tibHLsGtx4QYJIZCQUgj5qVgCGfXgNUboLxUPPd14bfL8vs991qsYFM4Q/GaJygg
RGWtvWY+0niGF3GZlZsQm0beD2JIBaKulQvvd6rV7jmjrwvtG/apJjY+RKr6M0be
kjb9LA3vOOE9C2nEekEvT6v2DIwLPGqN4MNIsDgJVb6YmX6BG+qbq39E89If1EUR
```

(continues on next page)

(continued from previous page)

```
1GZ4BAbBZoXImcbgoPYPxmg/KP/j19J8zlh+DwARAQABiQM+BBgBAgAJBQJST5WI
AhsCAikJENgpTsO7xNfVwV0gBBkBAgAGBQJST5WIAAoJEMogMoV3Bk63xfYP/Aui
Ybn8sZ2DtA5j1EQhVdHpt1o13MUJIUGtN7v7FPDhTcqdlDgfI8SN1wLBnuEZ3JTx
du9yuRuTn1ACdoY1hRqzKlxjS5bLibwyQlJnR5yOnLMNUbzp9psjM1Ek6sT0Mp5L
57boIaosl2xkGqMUCVVo67oaIHCJFWLzk+JzBlwspPUPrqKCzcPMac7GQcPN851o
Y483prvwWzn412396nSvTVjJAX3GkfbA4UQPmGOG3KO6EylxX4d4DDdaak7gDPHW
xnNP20uefMjULhWifnF95W0i18+aImyYupLnci/RPwpFbzBVky3EEFgj0uaytHXL
aHVtcfCZ8FZ2hvNLaxSMZkPsJOEteegnWME7/9S21hinFc/ToNXvcCC9i7FWiw/T
iEZ8eBvHS6mSc+KZhhBEuwl+q6ybkqV8TToOD7YYj7NM/6vXXqISiZFII+rZZ9Dg
oCPNtNxkki6txUwBXq9lUvptVzJANpqpNVr+rWXsmDJ2AGQgc3NehdszIOXVSpld
deGx5kU19G+UEDQeyVNwt+JHU2oIy6+Hc/+QYV4KH84z/R4FUCgqeyb03VE+zdMv
5PpkTJpyvL51nWzmR+PCDhCw4TJFOUCIih6CWl7MO8FaOvC7UkCLXyKAP5uhDkRm
cVFgov/BvpASujRBMUdXZQFvjPn0YczYKr6qC63H0eMIAJEBqtzNv4RnVP2dY+gg
yhLwNf4kkaZwZ31TPClEeutUKxHLZiKNWMElO8efoB+ZXw1I1NnCJQ/h4H8RESsJ
foW9imPTsi6tgFIYX/KwLxtnVPXYw6e01iLxd8HsS5YruAuhFnMJZxHnAqZD1GL+
/zym9IdMQMd9lJm+nc5H3duJolmUyPxoO0dUMpdQwsATPZID0YBIZ1d7pprA9yj1
3+VMMxz7JYalkztadojC+HsZ15ZJy+5iSo4+Hd4MouS8VVUs7dsYuTMgpBYYhxZE
nyLYoXEaR3c8bAVvu2FsNRC5XCjHjpGUHMgZt1aRP2Q+A0Cus8Bn5OMvFkuwOUiR
Iya5Ag0EUk+VvwEQAMG0+JaBKCu2WPH87fIcoXQYEm6VxVgAesnshg2KfbnBuMko
5X2SAHsPxDsyEf+vKSohsyBNPpMSo5b1Q3CaKln664KJqJ3RitqMit+ajrGDpWnu
rbGZu9yJybhkgvn2dVb9BwSJ/CWaSVxVe8+BaLWhuYmig3WQo8WsFRFg5e89Mw0i
yoDy7SuLkrBFbECIT6OWC0qk+5gp5XgqnTbt0uHLp7V2mZpkmZ3796p5IjzTTUmo
GazFCt+PX1Qnb5yeISg2zABIG8WVcBHgeBxYnI8ADVEEztaknPofG8JKzf0YGk9G
c952pPgyTjqykVeVVlnpseDstqE9PEqjEW4Yqf73iC761fqyh/pW/bsadL2Fp0b1
qHTpRJWtDXgc38+kr/GSNg/sSydutJiAeH+AVGAgYkg7vFCkw332G2vudCPUYFff
rpidlk3KfzwOf+fTPXukdhd3XDNEHsvIVQ3H/qz2NrYV8pkvOSB3D2rg4746R6cd
cb4aBLtol9MicAQ1F/8VVsB/5Xouw09HHdXQdaSoj8zb1kqD6es4anAerVLxOvQi
0c6LZm9jogJBTkXhFZHAEy/mxvTwTdM1rUlybgki+VgG/rMkcwRsCwO/okDmNMlw
mfhLZegRvK7MkJAUcKUxuY18wylE5doZihhZlO9sXHD3jcYW6tbRjBnw1VZpABEB
AAGJAR8EGAECAAkFAlJPlb8CGwwACgkQ2ClOw7vE19XBDgf9FsF98hfzAvLqSNGA
G44bfihuHC4FOqqNcnWYEMh6QH/HLbc6v7gg7uMCxINAg5q3k16IjUfVtFC4ZJ9S
xZGUCOmdrkh/8KkFCaxOD/TqOtU1wpSM74waBLn5Et24N8LgCokOHA9V8ck4VBt6
muJ2xlbyouHlmLrm6cyaH+0A++S7nR5i53mOJrQjQGYJoBFgNlLZ9BljdUDX6MtT
Rni7dJM2tFRY5upeinjHNvFC2Wc7YNCm7hZkCK3jfSR0Gya66l/Hr+e0qhLWp6D8
l+CI8VSRRJk5+MVUpUV2NrhWWbZQdUrshD2Jq3Z426QzfWVIFTpP3PtPubvhtyQU
WLuj9Q==
=e71t
-----END PGP PUBLIC KEY BLOCK-----
```

D.1.5. doceng-secretary@FreeBSD.org

```
pub   rsa2048/E1C03580AEB45E58 2019-10-31 [SC] [expires: 2022-10-30]
      Key fingerprint = F24D 7B32 B864 625E 5541  A0E4 E1C0 3580 AEB4 5E58
uid                           FreeBSD Doceng Team Secretary <doceng-
↪secretary@freebsd.org>
sub   rsa2048/9EA8D713509472FC 2019-10-31 [E] [expires: 2022-10-30]
```

```
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQENBF27FFcBCADeoSsIgyQUY8vREwkTikwFFlNg31MVy5s/Nq1cNK1PRfRMnprS
yfB62KqbYuz16bmQKaA9zHN4FGfiTvR6tl66LVHm1s/5HPiLv8sP14GsruLro9zN
v72dO7a9i68bMw+jarPOnu9dGiDFEI0dACOkdCGEYKEUapQeNpmWRrQ46BeXyFwF
JcNx76bJJUkwk6fWC0W63D762e6lCEX6ndoaPjjLBnFvtx13heNGUc8RukBwe2mA
U5pSGHj47J05bdWiRSwZaXa8PcW+20zTWaP755w7zWe4h60GANY7OsT9nuOqsioJ
QonxTrJuZweKRV8fNQ1EfDws3HZr7/7iXvO3ABEBAAG0PEZyZWVCU0QgRG9jZW5n
IFRlYW0gU2VjcmV0YXJ5IDxkb2Nlbmctc2VjcmV0YXJ5QGZyZWVic2Qub3JnPokB
VAQTAQoAPhYhBPJNezK4ZGJeVUGg5OHANYCutF5YBQJduxRXAhsDBQkFo5qABQsJ
CAcDBRUKCQgLBRYDAgEAAh4BAheAAAoJEOHANYCutF5YB2IIALw+EPYmOz9qlqIn
oTFmk/5MrcdzC5iLEfxubbF6TopDWsWPiOh5mAuvfEmROSGf6ctvdYe9UtQV3VNY
KeeyskeFrIBOFo2KG/dFqKPAWef6IfhbW3HWDWo5uOBg01jHzQ/pB1n6SMKiXfsM
idL9wN+UQKxF3Y7S/bVrZTV0isRUolO9+8kQeSYT/NMojVM0H2fWrTP/TaNEW4fY
JBDAl5hsktzdl8sdbNqdC0GiX3xb4GvgVzGGQELagsxjfuXk6PfOyn6Wx2d+yRcI
FrKojmhihBp5VGFQkntBIXQkaW0xhW+WBGxwXdaAl0drQlZ3W+edgdOl705x73kf
Uw3Fh2a5AQ0EXbsUVwEIANEPAsltM4vFj2pi5xEuHEcZIrIX/ZJhoaBtZkqvkB+H
4pu3/eQHK5hg0Dw12ugffPMz8mi57iGNI9TXd8ZYMJxAdvEZSDHCKZTX9G+FcxWa
/AzKNiG25uSISzz7rMB/lV1gofCdGtpHFRFTiNxFcoacugTdlYDiscgJZMJSg/hC
GXBdEKXR5WRAgAGandcL8llCToOt1lZEOkd5vJM861w6evgDhAZ2HGhRuG8/NDxG
r4UtlnYGUCFof/Q4oPNbDJzmZXF+8OQyTNcEpVD3leEOWG1Uv5XWS2XKVHcHZZ++
ISo/B5Q6Oi3SJFCVV9f+g09YF+PgfP/mVMBgif2fT20AEQEAAYkBPAQYAQoAJhYh
BPJNezK4ZGJeVUGg5OHANYCutF5YBQJduxRXAhsMBQkFo5qAAAoJEOHANYCutF5Y
kecIAMTh2VHQqjXHTszQMsy3NjiTVVITI3z+pzY0u2EYmLytXQ2pZMzLHMcklmub
5po0X4EvL6bZiJcLMI2mSrOs0Gp8P3hyMI40IkqoLMp7VA2LFlPgIJ7K5W4oVwf8
khY6lw7qg2l69APm/MM3xAyiL4p6MU8tpvWg5AncZ6lxyy27rxVflzEtCrKQuG/a
oVaOlMjH3uxvOK6IIxlhvWD0nKs/e2h2HIAZ+ILE6ytS5ZEg2GXuigoQZdEnv71L
xyvE9JANwGZLkDxnS5pgN2ikfkQYlFpJEkrNTQleCOHIIIp8vgJngEaP51xOIbQM
CiG/y3cmKQ/ZfH7BBvlZVtZKQsI=
=MQKT
-----END PGP PUBLIC KEY BLOCK-----
```

术语表

这个术语表包含了在 FreeBSD 社区和文档中使用的术语和首字母缩写。

A

ACL

见访问控制列表[2359]。

ACPI

见高级配置和电源接口[2360]。

AMD

见自动挂载程序[2361]。

AML

见 ACPI 机器语言[2362]。

API

见应用编程接口[2363]。

APIC

见高级可编程中断控制器[2364]。

[2359] https://docs.freebsd.org/en/books/handbook/glossary/#acl-glossary
[2360] https://docs.freebsd.org/en/books/handbook/glossary/#acpi-glossary
[2361] https://docs.freebsd.org/en/books/handbook/glossary/#amd-glossary
[2362] https://docs.freebsd.org/en/books/handbook/glossary/#aml-glossary
[2363] https://docs.freebsd.org/en/books/handbook/glossary/#api-glossary
[2364] https://docs.freebsd.org/en/books/handbook/glossary/#apic-glossary

APM

见高级电源管理[2365]。

APOP

见经过认证的邮局协议[2366]。

ASL

见 ACPI 源语言[2367]。

ATA

见高级技术附加[2368]。

ATM

见异步传输模式[2369]。

ACPI Machine Language，ACPI 机器语言

伪代码，由符合 ACPI 标准的操作系统内的虚拟机解释，在底层硬件和呈现给操作系统的文件接口之间提供一个层次。

ACPI Source Language，ACPI 源语言

AML 采用的编程语言。

Access Control List，访问控制列表

附加到一个对象的权限列表，通常是一个文件或一个网络设备。

Advanced Configuration and Power Interface，高级配置和电源接口

一种规范，它提供了硬件呈现给操作系统的接口的抽象，因此，操作系统应该不需要了解底层硬件就可以充分利用它。ACPI 发展并取代了以前由 APM、PNPBIOS 和其他技术提供的功能，并提供了控制功耗、机器暂停、设备启用和禁用等设施。

Application Programming Interface,应用编程接口

一套程序、协议和工具，规定了一个或多个程序部分的典型互动；它们如何、何时和为何一起工作，以及它们共享或操作什么数据。

Advanced Power Management，高级电源管理

一个 API，使操作系统能够与 BIOS 一起工作，以实现电源管理。在大多数应用中，APM 已经被更为通用和强大的 ACPI 规范所取代。

Advanced Programmable Interrupt Controller，高级可编程中断控制器

[2365] https://docs.freebsd.org/en/books/handbook/glossary/#apm-glossary
[2366] https://docs.freebsd.org/en/books/handbook/glossary/#apop-glossary
[2367] https://docs.freebsd.org/en/books/handbook/glossary/#asl-glossary
[2368] https://docs.freebsd.org/en/books/handbook/glossary/#ata-glossary
[2369] https://docs.freebsd.org/en/books/handbook/glossary/#atm-glossary

Advanced Technology Attachment，高级技术附加

Asynchronous Transfer Mode，异步传输模式

Authenticated Post Office Protocol，经过认证的邮局协议

Automatic Mount Daemon，自动挂载程序

一个当文件系统中的文件或目录被访问时自动挂载该文件系统的守护程序。

B

BAR

见基址寄存器[2370]。

BIND

见伯克利互联网名称域[2371]。

BIOS

见基本输入/输出系统[2372]。

BSD

见伯克利软件套件[2373]。

Base Address Register，基址寄存器

决定 PCI 设备将响应哪个地址范围的寄存器。

Basic Input/Output System，基本输入/输出系统

BIOS 的定义在一定程度上取决于上下文。有些人把它称为具有基本程序集的 ROM 芯片，以提供软件和硬件之间的接口。另一些人把它称为芯片中包含的帮助启动系统的一套程序。有些人可能还把它称为用于配置启动过程的屏幕。BIOS 是 PC 特有的，但其他系统也有类似的东西。

Berkeley Internet Name Domain，伯克利互联网名称域

一个 DNS 协议的实现。

Berkeley Software Distribution，伯克利软件套件

这是加州大学伯克利分校[2374]的计算机系统研究小组（CSRG）为他们对 AT&T 的 32V UNIX® 的改进和修改所起的名字。FreeBSD 是 CSRG 工作的后裔。

Bikeshed Building，自行车棚效应

[2370] https://docs.freebsd.org/en/books/handbook/glossary/#bar-glossary
[2371] https://docs.freebsd.org/en/books/handbook/glossary/#bind-glossary
[2372] https://docs.freebsd.org/en/books/handbook/glossary/#bios-glossary
[2373] https://docs.freebsd.org/en/books/handbook/glossary/#bsd-glossary
[2374] http://www.berkeley.edu/

一种现象，即许多人对一个不复杂的话题发表意见，而对一个复杂的话题却很少或没有讨论。关于该术语的起源，请参见 FAQ[2375]。

C

CD

见载波检测[2376]。

CHAP

见挑战握手认证协议[2377]。

CLIP

见 ATM 上的经典 IP[2378]。

COFF

见通用对象文件格式[2379]。

CPU

见中央处理单元[2380]。

CTS

见清除发送[2381]。

Carrier Detect，载波检测

一个 RS232C 信号，表示已经检测到载波。

Central Processing Unit，中央处理单元

也被称为处理器。这是计算机的大脑，所有计算都在这里进行。有许多不同的架构，有不同的指令集。其中比较知名的是英特尔 -x86 及其衍生产品、Arm 和 PowerPC。

Challenge Handshake Authentication Protocol，挑战握手认证协议

一种基于客户端和服务器之间共享的秘钥来验证用户身份的方法。

Classical IP over ATM，ATM 上的经典 IP

Clear To Send，，清除发送

一个 RS232C 信号，给予远程系统发送数据的许可。

[2375] https://docs.freebsd.org/en/books/faq/#bikeshed-painting
[2376] https://docs.freebsd.org/en/books/handbook/glossary/#cd-glossary
[2377] https://docs.freebsd.org/en/books/handbook/glossary/#chap-glossary
[2378] https://docs.freebsd.org/en/books/handbook/glossary/#clip-glossary
[2379] https://docs.freebsd.org/en/books/handbook/glossary/#coff-glossary
[2380] https://docs.freebsd.org/en/books/handbook/glossary/#cpu-glossary
[2381] https://docs.freebsd.org/en/books/handbook/glossary/#cts-glossary

另见请求发送[2382]。

Common Object File Format，通用对象文件格式

****D****

DAC

见自由裁量的访问控制[2383]。

DDB

见调试器[2384]。

DES

见数据加密标准[2385]。

DHCP

见动态主机配置协议[2386]。

DNS

见域名系统[2387]。

DSDT

见区分系统描述表[2388]。

DSR

见数据集就绪[2389]。

DTR

见数据终端就绪[2390]。

DVMRP

见距离矢量组播路由选择协议[2391]。

Discretionary Access Control，自由裁量的访问控制

Data Encryption Standard，数据加密标准

[2382] https://docs.freebsd.org/en/books/handbook/glossary/#rts-glossary
[2383] https://docs.freebsd.org/en/books/handbook/glossary/#dac-glossary
[2384] https://docs.freebsd.org/en/books/handbook/glossary/#ddb-glossary
[2385] https://docs.freebsd.org/en/books/handbook/glossary/#des-glossary
[2386] https://docs.freebsd.org/en/books/handbook/glossary/#dhcp-glossary
[2387] https://docs.freebsd.org/en/books/handbook/glossary/#dns-glossary
[2388] https://docs.freebsd.org/en/books/handbook/glossary/#dsdt-glossary
[2389] https://docs.freebsd.org/en/books/handbook/glossary/#dtr-glossary
[2390] https://docs.freebsd.org/en/books/handbook/glossary/#dtr-glossary
[2391] https://docs.freebsd.org/en/books/handbook/glossary/#dvmrp-glossary

一种加密信息的方法，传统上被用作 UNIX® 密码和 crypt(3) 函数的加密方法。

Data Set Ready，数据集就绪

从调制解调器发送至计算机或终端的 RS232C 信号，表示准备好发送和接收数据。

另见数据终端就绪[2392]。

Data Terminal Ready，数据终端就绪

从计算机或终端发送至调制解调器的 RS232C 信号，表示准备好发送和接收数据。

Debugger，调试器

一种用于检查系统状态的交互式内核设施，通常在系统崩溃后使用，以确定围绕故障的事件。

Differentiated System Description Table，区分系统描述表

ACPI 表，提供关于基本系统的基本配置信息。

Distance-Vector Multicast Routing Protocol，距离矢量组播路由选择协议

Domain Name System，域名系统

将人类可读的主机名（即 mail.example.net）转换为互联网地址的系统，反之亦然。

Dynamic Host Configuration Protocol，动态主机配置协议

当一台计算机（主机）向服务器请求 IP 地址时，该协议会动态地分配给它。该地址分配被称为“租赁”。

E

ECOFF

见扩展的 COFF[2393]。

ELF

见可执行和链接格式[2394]。

ESP

见封装安全有效载荷[2395]。

Encapsulated Security Payload，封装安全有效载荷

Executable and Linking Format，可执行和链接格式

Extended COFF，扩展的 COFF

F

[2392] https://docs.freebsd.org/en/books/handbook/glossary/#dtr-glossary

[2393] https://docs.freebsd.org/en/books/handbook/glossary/#ecoff-glossary

[2394] https://docs.freebsd.org/en/books/handbook/glossary/#elf-glossary

[2395] https://docs.freebsd.org/en/books/handbook/glossary/#esp-glossary

FADT

见固定 ACPI 描述表[2396]。

FAT

见文件分配表[2397]。

FAT16

见文件分配表（16 位）[2398]。

FTP

见文件传输协议[2399]。

File Allocation Table，文件分配表

File Allocation Table (16-bit)，文件分配表（16 位）

File Transfer Protocol，文件传输协议

在 TCP 基础上实现的高级协议系列的一个成员，可用于在 TCP/IP 网络上传输文件。

Fixed ACPI Description Table，固定 ACPI 描述表

G

GUI

见图形用户界面[2400]。

Giant

一个保护大量内核资源的互斥机制（sleep `mutex`）的名称。尽管在一台机器可能只有几十个进程，一块网卡，当然也只有一个处理器的时代，简单的锁机制已经足够了，但在现在，它是一个不可接受的性能瓶颈。FreeBSD 的开发者们正在积极努力用保护单个资源的锁来取代它，这将使单处理器和多处理器的机器都能有更大程度的并行性。

Graphical User Interface，图形用户界面

一个用户和计算机图形互动的系统。

H

HTML

见超文本标记语言[2401]。

[2396] https://docs.freebsd.org/en/books/handbook/glossary/#fadt-glossary
[2397] https://docs.freebsd.org/en/books/handbook/glossary/#fat-glossary
[2398] https://docs.freebsd.org/en/books/handbook/glossary/#fat16-glossary
[2399] https://docs.freebsd.org/en/books/handbook/glossary/#ftp-glossary
[2400] https://docs.freebsd.org/en/books/handbook/glossary/#gui-glossary
[2401] https://docs.freebsd.org/en/books/handbook/glossary/#html-glossary

HUP

请看 HangUp[2402]。

HangUp

HyperText Markup Language，超文本标记语言

用来创建网页的标记语言。

I

I/O

见输入/输出[2403]。

IASL

见英特尔的 ASL 编译器[2404]。

IMAP

见互联网信息访问协议[2405]。

IP

见互联网协议[2406]。

IPFW

见 IP 防火墙[2407]。

IPP

见互联网打印协议[2408]。

IPv4

见 第 4 版 IP[2409]。

IPv6

见 第 6 版 IP[2410]。

ISP

见互联网服务提供商[2411]。

[2402] https://docs.freebsd.org/en/books/handbook/glossary/#hup-glossary
[2403] https://docs.freebsd.org/en/books/handbook/glossary/#io-glossary
[2404] https://docs.freebsd.org/en/books/handbook/glossary/#iasl-glossary
[2405] https://docs.freebsd.org/en/books/handbook/glossary/#imap-glossary
[2406] https://docs.freebsd.org/en/books/handbook/glossary/#ip-glossary
[2407] https://docs.freebsd.org/en/books/handbook/glossary/#ipfw-glossary
[2408] https://docs.freebsd.org/en/books/handbook/glossary/#ipp-glossary
[2409] https://docs.freebsd.org/en/books/handbook/glossary/#ipv4-glossary
[2410] https://docs.freebsd.org/en/books/handbook/glossary/#ipv6-glossary
[2411] https://docs.freebsd.org/en/books/handbook/glossary/#isp-glossary

IP Firewall，IP 防火墙

IP Version 4，第 4 版 IP

第 4 版 IP，使用 32 位进行寻址。这个版本仍然是使用最广泛的，但它正慢慢被 IPv6 取代。

另见第 6 版 IP[2412]。

IP Version 6，第 6 版 IP

新的 IP 协议。发明的原因是 IPv4 的地址空间正在耗尽。使用 128 位进行寻址。

Input/Output，输入/输出

Intel’s ASL compiler，英特尔的 ASL 编译器

英特尔用于将 ASL 转换为 AML 的编译器。

Internet Message Access Protocol，互联网信息访问协议

一种用于访问邮件服务器上的电子邮件信息的协议，其特点是信息通常保存在服务器上，而不是下载到邮件阅读器客户端。

另见邮政协议版本 3。

Internet Printing Protocol，互联网打印协议

Internet Protocol，互联网协议

数据包传输协议，是互联网上的基本协议。最初在美国国防部开发，是 TCP/IP 协议栈的一个极其重要的部分。没有互联网协议，互联网就不会成为今天的样子。更多信息，见 RFC 791[2413]。

Internet Service Provider，互联网服务提供商

一家提供互联网接入的公司。

K

KAME

KAME 是日语“乌龟”的意思，在计算机界被用来指 KAME 项目[2414]，他们致力于 IPv6 的实施。

KDC

见密钥分发中心[2415]。

KLD

参见内核 ld(1)[2416]。

KSE

[2412] https://docs.freebsd.org/en/books/handbook/glossary/#ipv6-glossary
[2413] ftp://ftp.rfc-editor.org/in-notes/rfc791.txt
[2414] http://www.kame.net/
[2415] https://docs.freebsd.org/en/books/handbook/glossary/#kdc-glossary
[2416] https://docs.freebsd.org/en/books/handbook/glossary/#kld-glossary

见内核调度器实体[2417]。

KVA

见内核虚拟地址[2418]。

Kbps

见千比特每秒[2419]。

Kernel ld(1)[2420]

一种在不重启系统的情况下动态加载功能到 FreeBSD 内核的方法。

Kernel Scheduler Entities，内核调度器实体

一个由内核支持的线程系统。更多细节请见项目主页[2421]。

Kernel Virtual Address，内核虚拟地址

Key Distribution Center，密钥分配中心

Kilo Bits Per Second，千比特每秒

用于测量带宽（在指定的时间内有多少数据可以通过一个给定的点）。可用 Mega、Giga、Tera 等代替前缀 Kilo。

L

LAN

见局域网[2422]。

LOR

见Lock Order Reversal[2423]。

LPD

见行式打印机守护程序[2424]。

Line Printer Daemon，行式打印机守护程序

Local Area Network，局域网

在局部地区使用的网络，如办公室、家庭等。

Lock Order Reversal

[2417] https://docs.freebsd.org/en/books/handbook/glossary/#kse-glossary
[2418] https://docs.freebsd.org/en/books/handbook/glossary/#kva-glossary
[2419] https://docs.freebsd.org/en/books/handbook/glossary/#kbps-glossary
[2420] https://www.freebsd.org/cgi/man.cgi?query=ld&sektion=1&format=html
[2421] http://www.freebsd.org/kse
[2422] https://docs.freebsd.org/en/books/handbook/glossary/#lan-glossary
[2423] https://docs.freebsd.org/en/books/handbook/glossary/#lor-glossary
[2424] https://docs.freebsd.org/en/books/handbook/glossary/#lpd-glossary

FreeBSD 内核采用了一系列资源锁来判定抢占资源的行为。位于 FreeBSD-CURRENT 内核的“运行程序锁诊断系统”，又称 witness(4)[2425]（Release 中已移除）用来检测因加锁错误可能发生的死锁。（witness(4)[2426] 实际上略微保守，有出现误报的可能）。准确的报告意味着“如果你运气再差点，这里已经出现死锁了”。

真正重要的 LOR 往往很快就会被修复，所以在发布到邮件列表之前，请检查 https://lists.FreeBSD.org/subscription/freebsd-current 和 LORs Seen[2427] 页面。

M

MAC

见强制访问控制[2428]。

MADT

请参阅多 APIC 描述表[2429]。

MFC

请参阅从 Current 合并[2430]。

MFH

见从 Head 合并[2431]。

MFS

见从 Stable 合并[2432]。

MFV

参见从 Vendor 处合并[2433]。

MIT

见麻省理工学院[2434]。

MLS

见多级安全[2435]。

MOTD

[2425] https://www.freebsd.org/cgi/man.cgi?query=witness&sektion=4&format=html
[2426] https://www.freebsd.org/cgi/man.cgi?query=witness&sektion=4&format=html
[2427] http://sources.zabbadoz.net/freebsd/lor.html
[2428] https://docs.freebsd.org/en/books/handbook/glossary/#mac-glossary
[2429] https://docs.freebsd.org/en/books/handbook/glossary/#madt-glossary
[2430] https://docs.freebsd.org/en/books/handbook/glossary/#mfc-glossary
[2431] https://docs.freebsd.org/en/books/handbook/glossary/#mfh-glossary
[2432] https://docs.freebsd.org/en/books/handbook/glossary/#mfs-glossary
[2433] https://docs.freebsd.org/en/books/handbook/glossary/#mfv-glossary
[2434] https://docs.freebsd.org/en/books/handbook/glossary/#mit-glossary
[2435] https://docs.freebsd.org/en/books/handbook/glossary/#mls-glossary

见今日消息[2436]。

MTA

见邮件传输代理[2437]。

MUA

见邮件用户代理[2438]。

Mail Transfer Agent，邮件传输代理

一个用于传输电子邮件的应用程序。传统上，MTA 是 BSD 基本系统的一部分。今天，Sendmail 被包含在基本系统中，但还有许多其他的 MTA，如 postfix、qmail 和 Exim。

Mail User Agent，邮件用户代理

用户用于显示和书写电子邮件的应用程序。

Mandatory Access Control，强制访问控制

Massachusetts Institute of Technology，麻省理工学院

Merge From Current，从 Current 合并

将功能或补丁从 -CURRENT 分支合并到另一个分支，通常是 -STABLE。

Merge From Head，从 Head 合并

将版本库 HEAD 中的功能或补丁合并到一个较早的分支。

Merge From Stable，从 Stable 合并

在正常的 FreeBSD 开发过程中，一个修改在被合并到 -STABLE 之前会被提交到 -CURRENT 分支进行测试。在极少数情况下，一个改动会先进入 -STABLE，然后被合并到 -CURRENT。

当一个补丁从 -STABLE 合并到安全分支时也会用到这个术语。

另请参见从 Current 合并[2439]。

Merge From Vendor，从 Vendor 处合并

Message Of The Day，今日消息

一条信息，通常在登录时显示，通常用于向系统的用户分发信息。

Multi-Level Security，多级安全

Multiple APIC Description Table，多 APIC 描述表

N

[2436] https://docs.freebsd.org/en/books/handbook/glossary/#motd-glossary

[2437] https://docs.freebsd.org/en/books/handbook/glossary/#mta-glossary

[2438] https://docs.freebsd.org/en/books/handbook/glossary/#mua-glossary

[2439] https://docs.freebsd.org/en/books/handbook/glossary/#mfc-glossary

NAT

见网络地址转换[2440]。

NDISulator

见 Evil 项目[2441]。

NFS

见网络文件系统[2442]。

NTFS

见新技术文件系统[2443]。

NTP

见网络时间协议[2444]。

Network Address Translation，网络地址转换

一种技术，IP 数据包在通过网关时被改写，使网关后面的许多机器能够有效地共享一个 IP 地址。

Network File System，网络文件系统

New Technology File System，新技术文件系统

一种由微软开发的文件系统，可用于其“新技术”操作系统，如 Windows® 2000、Windows NT® 和 Windows® XP。

Network Time Protocol，网络时间协议

通过网络同步时钟的一种手段。

O

OBE

见被事件所取代[2445]。

ODMR

见按需邮件中继[2446]。

OS

见操作系统[2447]。

[2440] https://docs.freebsd.org/en/books/handbook/glossary/#nat-glossary
[2441] https://docs.freebsd.org/en/books/handbook/glossary/#projectevil-glossary
[2442] https://docs.freebsd.org/en/books/handbook/glossary/#nfs-glossary
[2443] https://docs.freebsd.org/en/books/handbook/glossary/#ntfs-glossary
[2444] https://docs.freebsd.org/en/books/handbook/glossary/#ntp-glossary
[2445] https://docs.freebsd.org/en/books/handbook/glossary/#obe-glossary
[2446] https://docs.freebsd.org/en/books/handbook/glossary/#odmr-glossary
[2447] https://docs.freebsd.org/en/books/handbook/glossary/#os-glossary

On-Demand Mail Relay，按需邮件中继

Operating System，操作系统

一组程序、库和工具，提供对计算机硬件资源的访问。今天，操作系统的范围从支持一次只运行一个程序、只访问一个设备的简单设计到完全的多用户、多任务和多进程系统，这些系统可以同时为成千上万的用户服务，每个用户都运行几十个不同的应用程序。

Overtaken By Events，被事件所取代

表示建议的修改（例如问题报告或功能请求），由于后来 FreeBSD 的变化、网络标准的变化、受影响的硬件已经过时等原因，已经不再相关或适用。

P

PAE

见物理地址扩展[2448]。

PAM

见可插入式认证模块[2449]。

PAP

见密码认证协议[2450]。

PC

见个人电脑[2451]。

PCNSFD

见个人计算机网络文件系统守护程序[2452]。

PDF

见便携式文档格式[2453]。

PID

见进程 ID[2454]。

POLA

见最小惊奇原则[2455]。

POP

[2448] https://docs.freebsd.org/en/books/handbook/glossary/#pae-glossary
[2449] https://docs.freebsd.org/en/books/handbook/glossary/#pam-glossary
[2450] https://docs.freebsd.org/en/books/handbook/glossary/#pap-glossary
[2451] https://docs.freebsd.org/en/books/handbook/glossary/#pc-glossary
[2452] https://docs.freebsd.org/en/books/handbook/glossary/#pcnfsd-glossary
[2453] https://docs.freebsd.org/en/books/handbook/glossary/#pdf-glossary
[2454] https://docs.freebsd.org/en/books/handbook/glossary/#pid-glossary
[2455] https://docs.freebsd.org/en/books/handbook/glossary/#pola-glossary

见邮件协议[2456]。

POP3

见邮件协议版本 3[2457]。

PPD

见 PostScript 打印机描述[2458]。

PPP

见点对点协议[2459]。

PPPoA

见 ATM 上的 PPP[2460]。

PPPoE

见以太网上的 PPP[2461]。

PPP over ATM，ATM 上的 PPP

PPP over Ethernet，以太网上的 PPP

PR

见问题报告[2462]。

PXE

见预启动执行环境[2463]。

Password Authentication Protocol，密码认证协议

Personal Computer，个人电脑

Personal Computer Network File System Daemon，个人计算机网络文件系统守护程序

Physical Address Extensions，物理地址扩展

一种使系统能够访问高达 64GB 的 RAM 的方法，这些系统只具有 32 位宽的地址空间（因此在没有 PAE 的情况下会被限制在 4GB）。

Pluggable Authentication Modules，可插入式认证模块

Point-to-Point Protocol，点对点协议

[2456] https://docs.freebsd.org/en/books/handbook/glossary/#pop-glossary
[2457] https://docs.freebsd.org/en/books/handbook/glossary/#pop3-glossary
[2458] https://docs.freebsd.org/en/books/handbook/glossary/#ppd-glossary
[2459] https://docs.freebsd.org/en/books/handbook/glossary/#ppp-glossary
[2460] https://docs.freebsd.org/en/books/handbook/glossary/#pppoa-glossary
[2461] https://docs.freebsd.org/en/books/handbook/glossary/#pppoe-glossary
[2462] https://docs.freebsd.org/en/books/handbook/glossary/#pr-glossary
[2463] https://docs.freebsd.org/en/books/handbook/glossary/#pxe-glossary

Pointy Hat

一件神话般的头饰，很像一顶傻瓜帽，它被授予任何破坏构建、使修订号倒退或在源代码库中制造任何其他破坏的 FreeBSD 提交者。任何有价值的提交者都会很快积累起一个庞大的收藏。其用法是（几乎总是如此）幽默的。

Portable Document Format，便携式文档格式

Post Office Protocol，邮件协议

另见邮件协议版本 3[2464]。

Post Office Protocol Version 3，邮件协议版本 3

一种用于访问邮件服务器上的电子邮件信息的协议，其特点是信息通常从服务器下载到客户端，而不是留在服务器上。

另见互联网信息访问协议。

PostScript Printer Description，PostScript 打印机描述

Preboot eXecution Environment，预启动执行环境

Principle Of Least Astonishment,最小惊奇原则

随着 FreeBSD 的发展，用户可以看到的变化应该尽可能地保持不令人惊讶。例如，在 **/etc/defaults/rc.conf** 中任意地重新安排系统启动变量就违反了 POLA。开发人员在考虑对用户可见的系统进行修改时应考虑 POLA。

Problem Report

对 FreeBSD 源代码或文档中发现的某种问题的描述。参见撰写 FreeBSD 问题报告。

Process ID

一个数字，对系统中的一个特定进程来说是唯一的，它可以识别它并允许对它采取行动。

Evil 项目

NDISulator 的工作标题，由 Bill Paul 撰写，他的名字是指首先需要有这样的东西是多么的可怕 (从哲学的角度来看)。NDISulator 是一个特殊的兼容模块，允许 Microsoft Windows™ NDIS miniport 网络驱动程序在 FreeBSD/i386 上使用。这通常是使用驱动是闭源的卡的唯一方法。参见**src/sys/compat/ndis/subr_ndis.c**。

R

RA

见路由器通告[2465]。

RAID

[2464] https://docs.freebsd.org/en/books/handbook/glossary/#imap-glossary

[2465] https://docs.freebsd.org/en/books/handbook/glossary/#ra-glossary

见独立冗余磁盘阵列[2466]。

RAM

见随机存取存储器[2467]。

RD

见接收的数据[2468]。

RFC

见征求意见稿[2469]。

RISC

见精简指令集计算机[2470]。

RPC

见远程过程调用[2471]。

RS232C

见推荐 232C 标准[2472]。

RTS

请看请求发送[2473]。

Random Access Memory

Revision Control System

修订控制系统（*RCS*）是对普通文件实施“修订控制”的最古老的软件套件之一。它允许对每个文件进行存储、检索、归档、记录、识别和合并多个修订版。RCS 由许多小工具组成，它们一起工作。它缺少一些在更现代的修订控制系统中发现的功能，如 Git，但它的安装、配置和使用于一小部分文件非常简单。

另见 Subversion[2474]。

Received Data

接收数据的 RS232C 引脚或导线。

另见传输的数据[2475]。

Recommended Standard 232C，232C 标准

[2466] https://docs.freebsd.org/en/books/handbook/glossary/#raid-glossary
[2467] https://docs.freebsd.org/en/books/handbook/glossary/#ram-glossary
[2468] https://docs.freebsd.org/en/books/handbook/glossary/#rd-glossary
[2469] https://docs.freebsd.org/en/books/handbook/glossary/#rfc-glossary
[2470] https://docs.freebsd.org/en/books/handbook/glossary/#risc-glossary
[2471] https://docs.freebsd.org/en/books/handbook/glossary/#rpc-glossary
[2472] https://docs.freebsd.org/en/books/handbook/glossary/#rs232c-glossary
[2473] https://docs.freebsd.org/en/books/handbook/glossary/#rts-glossary
[2474] https://docs.freebsd.org/en/books/handbook/glossary/#svn-glossary
[2475] https://docs.freebsd.org/en/books/handbook/glossary/#td-glossary

串行设备之间的通信标准。

Reduced Instruction Set Computer，精简指令集计算机

一种处理器的设计方法，硬件可以执行的操作被简化，但尽可能做到通用。这可以导致更低的功耗，更少的晶体管，在某些情况下，更好的性能和增加代码密度。RISC 处理器的例子包括 Alpha、SPARC®、ARM® 和 PowerPC®。

Redundant Array of Inexpensive Disks，独立冗余磁盘阵列

Remote Procedure Call，远程过程调用

Request For Comments，征求意见稿

一套定义互联网标准、协议等的文件。见 www.rfc-editor.org。

当有人提出修改建议并希望得到反馈时，也可作为一个一般术语使用。

Request To Send，请求发送

一个 RS232C 信号，要求远程系统开始传输数据。

另见清除发送[2476]。

Router Advertisement，路由器通告

S

SCI

见系统控制中断[2477]。

SCSI

见小型计算机系统接口[2478]。

SG

见信号地线[2479]。

SMB

见服务器消息区块[2480]。

SMP

见对称多处理器[2481]。

SMTP

[2476] https://docs.freebsd.org/en/books/handbook/glossary/#cts-glossary
[2477] https://docs.freebsd.org/en/books/handbook/glossary/#sci-glossary
[2478] https://docs.freebsd.org/en/books/handbook/glossary/#scsi-glossary
[2479] https://docs.freebsd.org/en/books/handbook/glossary/#sg-glossary
[2480] https://docs.freebsd.org/en/books/handbook/glossary/#smb-glossary
[2481] https://docs.freebsd.org/en/books/handbook/glossary/#smp-glossary

见简单邮件传输协议[2482]。

SMTP AUTH

见 SMTP 认证[2483]。

SSH

见安全 shell[2484]。

STR

见挂起到内存[2485]。

SVN

见 Subversion[2486]。

SMTP Authentication，SMTP 认证

Server Message Block，服务器消息区块

Signal Ground，信号地线

RS232 引脚或导线，是信号的接地参考。

Simple Mail Transfer Protocol，简单邮件传输协议

Secure Shell，安全 shell

Small Computer System Interface，小型计算机系统接口

Subversion

Subversion 是 FreeBSD 项目目前使用的一个版本控制系统。

Suspend To RAM，挂起到内存

Symmetric MultiProcessor，对称多处理器

System Control Interrupt，系统控制中断

T

TCP

见传输控制协议[2487]。

TCP/IP

[2482] https://docs.freebsd.org/en/books/handbook/glossary/#smtp-glossary
[2483] https://docs.freebsd.org/en/books/handbook/glossary/#smtpauth-glossary
[2484] https://docs.freebsd.org/en/books/handbook/glossary/#ssh-glossary
[2485] https://docs.freebsd.org/en/books/handbook/glossary/#str-glossary
[2486] https://docs.freebsd.org/en/books/handbook/glossary/#svn-glossary
[2487] https://docs.freebsd.org/en/books/handbook/glossary/#tcp-glossary

见传输控制协议/互联网协议[2488]。

TD

见传输的数据[2489]。

TFTP

见简单的 FTP[2490]。

TGT

见凭据授予[2491]。

TSC

见时间戳计数器[2492]。

Ticket-Granting Ticket，凭据授予

Time Stamp Counter，时间戳计数器

现代 Pentium® 处理器内部的剖析计数器，计算核心频率的时钟刻度。

Transmission Control Protocol，传输控制协议

一种位于（例如）IP 协议之上的协议，保证数据包以可靠、有序的方式交付。

Transmission Control Protocol/Internet Protocol，传输控制协议/互联网协议

在 IP 协议上运行的 TCP 协议的组合术语。互联网的大部分内容都在 TCP/IP 上运行。

Transmitted Data，传输的数据

一个 RS232C 针脚或电线，数据通过它传输。

另见接收的数据[2493]。

Trivial FTP，简单的 FTP

U

UDP

见用户数据报协议[2494]。

UFS1

见 Unix 文件系统版本 1[2495]。

[2488] https://docs.freebsd.org/en/books/handbook/glossary/#tcpip-glossary
[2489] https://docs.freebsd.org/en/books/handbook/glossary/#td-glossary
[2490] https://docs.freebsd.org/en/books/handbook/glossary/#tftp-glossary
[2491] https://docs.freebsd.org/en/books/handbook/glossary/#tgt-glossary
[2492] https://docs.freebsd.org/en/books/handbook/glossary/#tsc-glossary
[2493] https://docs.freebsd.org/en/books/handbook/glossary/#rd-glossary
[2494] https://docs.freebsd.org/en/books/handbook/glossary/#udp-glossary
[2495] https://docs.freebsd.org/en/books/handbook/glossary/#ufs1-glossary

UFS2

见 Unix 文件系统第 2 版[2496]。

UID

见用户 ID[2497]。

URL

见统一资源定位符[2498]。

USB

见通用串行总线[2499]。

Uniform Resource Locator，统一资源定位符

一种定位资源的方法，如互联网上的文件和识别该资源的手段。

Unix File System Version 1，Unix 文件系统第 1 版

最初的 UNIX® 文件系统，有时称为伯克利快速文件系统。

Unix File System Version 2，Unix 文件系统第 2 版

UFS1 的扩展，在 FreeBSD 5-CURRENT 中引入。UFS2 增加了 64 位块指针（打破了 1T 的障碍），支持扩展文件存储和其他功能。

Universal Serial Bus，通用串行总线

一种硬件标准，用于将各种计算机外围设备连接到一个通用接口。

User ID，用户 ID

分配给计算机的每个用户的唯一号码，通过它可以识别分配给该用户的资源和权限。

User Datagram Protocol,用户数据报协议

一种简单、不可靠的数据报协议，用于在 TCP/IP 网络上交换数据。UDP 不像 TCP 那样提供错误检查和纠正。

V

VPN

见虚拟专用网络[2500]。

Virtual Private Network,虚拟专用网络

一种使用公共电信（如互联网）的方法，以提供对局部网络（如公司局域网）的远程访问。

[2496] https://docs.freebsd.org/en/books/handbook/glossary/#ufs2-glossary
[2497] https://docs.freebsd.org/en/books/handbook/glossary/#uid-glossary
[2498] https://docs.freebsd.org/en/books/handbook/glossary/#url-glossary
[2499] https://docs.freebsd.org/en/books/handbook/glossary/#usb-glossary
[2500] https://docs.freebsd.org/en/books/handbook/glossary/#vpn-glossary

术语翻译对照表

英语	翻译	说明
copyleft	版权/著作权	根据【著作权法】所述：本法所称的著作权即版权。
Mandatory Access Control	强制访问控制	
masquerade	冒充	但是 IP masquerading 为 IP 伪装
remote groups	异地团体?	不知道怎么翻译
layer 4-7	第4-7 层	
load balancing	负载均衡	
URL	网址	
Installing Applications: Packages and Ports	安装应用程序：软件包和 Ports	
glossary	术语表	
inquiry	轮询	
connection handle	??? 链接柄	不知道怎么翻译
channels	信道	
USB dongle	USB 适配器	需要结合语境判断是具体是什么
Internet	互联网	无需加“公开”，互联网在理论上本就是开放的
Ticket	凭据	
Wrapper	Wrapper	专有名词，维持英文
Ports Collection	Ports	`Collection` 似乎看起来无用。Ports Collection 实际上就是 Ports，但是单个的软件则为 port
raw device	裸设备	原始设备?

continues on next page

表 5 – continued from previous page

英语	翻译	说明
media	光盘/设备	不翻译为不明确的词语“介质”
PAE（Physical Address Extension）	物理地址扩展	
embedded boards	嵌入式板卡	
system	设备/系统	
MBR	主引导记录	
GPT	全局唯一标识分区表	
UEFI	统一可扩展固件接口	
image	镜像	
bug	bug	专有名词，维持英文
slices	slices	
note	注意	应该缩进**加粗**使用
warning	警告	应该缩进**加粗**使用
important	重要	应该缩进**加粗**使用
caution	当心	应该缩进**加粗**使用
USB stick	U 盘/优盘	
space	空格键	空格键
Enter	回车键	回车键
Backspace	退格键	退格键
describe	说明	
tip	技巧	应该缩进**加粗**使用
keymap	键盘布局	
fully-qualified	完全限定	
root-on-ZFS	使用 ZFS 作为 root 分区/ root 分区使用 ZFS	
advanced users	专业用户	
print spooler	打印后台处理程序	台湾是说法“列印排存器”，列印即打印，排存器不知道是什么东西
partition schemes	分区表	
stripe	条带	
drive	结合语境可能是“磁盘”	
Distribution Files	安装文件	
Hardening Security	强化安全	
scroll-back buffer	回滚缓冲区	
Network Interfaces	网络接口/网卡	
Mirror	镜像站	
build	构建/编译	
Disk Organization	磁盘结构	3.6. Disk Organization

continues on next page

表 5 – continued from previous page

英语	翻译	说明
Video Modes	分辨率	
key bindings	组合键	
sticky directories	粘滞位	
tarball	源码包/压缩文件	?
Quarterly Branche	季度分支	
Latest Branche	最新分支	
ports tree	树无意义，不翻译	
ports skeleton	ports 框架	
Procedure	过程	
heckout	检出	github 命令参考此处[2501]
log rotation	日志轮替	
fork	复刻	github 命令参考此处[2502]
Tuning	优化	# 15.4. 中为“调整”
Kernel Limits	内核参数	内核限制实际上就是内核参数[2503]
boot	引导	不是“启动”
Process Accounting	进程审计	“进程记账”看起来不相干。“审计”更符合原意
hash	哈希:raw-latex:散列	有时代指“加密方式”
Login Classes	登录分级	
Export Regulations	出口管制	
principal	主体	Kerberos
realm	领域	Kerberos
hardened	加固	如，APK 加固，JAVA 加固
flag	标签	ACL
programmatic method	计划任务	14.10. 监测第三方安全问题
security advisory	安全公告	
per-user calls	每用户调用数	
core files	核心转储	等于 core dump
utility	软件/工具	实际上就是软件
jail	jail	一般应该小写
inode	节点	
loopback	回环	
monitoring	监控	
issue command	执行命令	不是“发出”
facility	设施/机制/工具	
Audit Review	审计复核	
reduction	精选	用于安全审计
disk	磁盘	

continues on next page

表 5 – continued from previous page

英语	翻译	说明
off-site	离线	
Memory Disks	内存盘	用于 18 章，不是闪存设备
memory-backed disks	内存盘	用于 18 章，不是闪存设备
provider	provider	专有名词, A provider is “a disk-like thing which appears in /dev”
world	世界	指除了内核以外的基本系统的所有部分
sound sink	音频接收器	第 7.2 节出现
bootstrap	自举	
custom kernel	定制内核	
stateful firewall	状态防火墙	

[2501] https://linux.cn/article-12245-1.html

[2502] https://linux.cn/article-12245-1.html

[2503] https://eloquence.marxmeier.com/sdb/html/linux_limits.html

后记

本书是数百位“FreeBSD 文档项目”贡献者的共同成果。原文是用 AsciiDoc 编写的。