

# FreeBSD 使用手册

## 摘要

欢迎使用 FreeBSD！本手册用于安装 *FreeBSD 11.2-RELEASE* 和 *FreeBSD 12.0-RELEASE* 以及它的日常使用。这个手册目前由很多人 持地 维护。其中的内容需要不断地更新。如果你有 兴趣参加这个项目，请邮件到 [FreeBSD 文档邮件列表](#)。此文最新的英文原始版本可以从 [FreeBSD 网站](#) 上取得（本手册的早期版本可以在 <http://docs.FreeBSD.org/doc/> 得到）。由 [FreeBSD 中文项目](#) 维护的最新版本可以在 [FreeBSD 中文项目 快照网站](#) 取得，这一版本会持地向主站同步。此外，你也可以从 [FreeBSD FTP 服务器](#) 及其许多 镜像站点 取得本文的各其它格式，以及不同形式的版本。如果你希望得到一印刷版本的手册，可以从 [FreeBSD Mall](#) 取得。除此之外，你还可以 [在手册中搜索内容](#)。

---



# 目 录

|                               |     |
|-------------------------------|-----|
| 前言                            | 11  |
| 本书的读者                         | 11  |
| 相对于第三版的改动                     | 11  |
| 相对于第二版的改动 (2004)              | 11  |
| 相对于第一版的改动 (2001)              | 12  |
| 本手册的编写                        | 12  |
| 本书中使用的一些约定                    | 15  |
| 致谢                            | 16  |
| I: 起步                         | 17  |
| 1. 介绍                         | 18  |
| 1.1. 概述                       | 18  |
| 1.2. 欢迎来到 FreeBSD 的世界!        | 18  |
| 1.3. 关于 FreeBSD 目录            | 20  |
| 2. 安装 FreeBSD                 | 25  |
| 2.1. 概述                       | 25  |
| 2.2. 硬件需求                     | 25  |
| 2.3. 安装前的准备工作                 | 26  |
| 2.4. 开始安装                     | 32  |
| 2.5. 介绍 Sysinstall            | 38  |
| 2.6. 分配磁盘空间                   | 43  |
| 2.7. 需要安装的文件包                 | 56  |
| 2.8. 需要使用的安装介绍                | 59  |
| 2.9. 安装磁盘                     | 60  |
| 2.10. 安装后的配置                  | 61  |
| 2.11. 常见问题                    | 93  |
| 2.12. 高级安装指南                  | 95  |
| 2.13. 准备自己的安装介绍               | 98  |
| 3. 安装 FreeBSD (适用于 9.x 及以后版本) | 105 |
| 3.1. 概述                       | 105 |
| 3.2. 硬件需求                     | 105 |
| 3.3. 安装前的准备工作                 | 106 |
| 3.4. 开始安装                     | 110 |
| 3.5. 介绍 bsdinstall            | 117 |
| 3.6. 通过网络安装                   | 120 |
| 3.7. 分配磁盘空间                   | 121 |
| 3.8. 安装磁盘                     | 126 |
| 3.9. 安装后的配置                   | 128 |
| 3.10. 故障排除                    | 149 |

|                            |     |
|----------------------------|-----|
| 4. UNIX 基                  | 151 |
| 4.1. 概述                    | 151 |
| 4.2. 虚控制台和端                | 151 |
| 4.3. 限                     | 154 |
| 4.4. 目架                    | 158 |
| 4.5. 磁                     | 160 |
| 4.6. 文件系的挂接和卸下             | 166 |
| 4.7. 程                     | 168 |
| 4.8. 守程, 信号和死程             | 169 |
| 4.9. Shells                | 171 |
| 4.10. 文本器                  | 173 |
| 4.11. 和点                   | 173 |
| 4.12. 二制文件格式               | 174 |
| 4.13. 取得更多的                | 175 |
| 5. 安装用程序: Packages 和 Ports | 178 |
| 5.1. 概述                    | 178 |
| 5.2. 件安装                   | 178 |
| 5.3. 要的用程序                 | 179 |
| 5.4. 使用 Package 系          | 181 |
| 5.5. 使用Ports Collection    | 183 |
| 5.6. 安装之后要做什么?             | 193 |
| 5.7. 如何理坏掉的 Ports          | 194 |
| 6. X Window 系              | 195 |
| 6.1. 概述                    | 195 |
| 6.2. 理解 X                  | 195 |
| 6.3. 安装 X11                | 198 |
| 6.4. 配置 X11                | 198 |
| 6.5. 在 X11 中使用字体           | 204 |
| 6.6. X 示管理器                | 208 |
| 6.7. 面境                    | 210 |
| II: 常的任                    | 215 |
| 7. 面用                      | 216 |
| 7.1. 概述                    | 216 |
| 7.2. 器                     | 216 |
| 7.3. 公、象理                  | 220 |
| 7.4. 文看器                   | 224 |
| 7.5.                       | 225 |
| 7.6.                       | 227 |
| 8. 多媒体                     | 229 |
| 8.1. 概述                    | 229 |
| 8.2. 安装声                   | 229 |

|                        |     |
|------------------------|-----|
| 8.3. MP3音              | 233 |
| 8.4. 回放                | 236 |
| 8.5. 安装                | 243 |
| 8.6. 象描                | 245 |
| 9. 配置FreeBSD的内核        | 250 |
| 9.1. 概述                | 250 |
| 9.2. 什需要建立定制的内核?       | 250 |
| 9.3. 系硬件               | 250 |
| 9.4. 内核, 子系和模          | 251 |
| 9.5. 建立并安装一个定制的内核      | 252 |
| 9.6. 配置文件              | 254 |
| 9.7. 如果出               | 269 |
| 10. 打印                 | 271 |
| 10.1. 概述               | 271 |
| 10.2. 介                | 271 |
| 10.3. 基本置              | 272 |
| 10.4. 高置               | 284 |
| 10.5. 使用打印机            | 312 |
| 10.6. 替准后台打印           | 319 |
| 10.7. 疑                | 319 |
| 11. Linux® 二制兼容模式      | 323 |
| 11.1. 概述               | 323 |
| 11.2. 配置 Linux® 二制兼容模式 | 323 |
| 11.3. 高主               | 326 |
| III: 系管理               | 328 |
| 12. 置和整                | 329 |
| 12.1. 概述               | 329 |
| 12.2. 初配置              | 329 |
| 12.3. 核心配置             | 330 |
| 12.4. 用程序配置            | 331 |
| 12.5. 服                | 331 |
| 12.6. 配置 cron          | 333 |
| 12.7. 在 FreeBSD 中使用 rc | 334 |
| 12.8. 置网               | 336 |
| 12.9. 虚主机              | 341 |
| 12.10. 配置文件            | 342 |
| 12.11. 用 sysctl 行整     | 346 |
| 12.12. 整磁              | 347 |
| 12.13. 整内核限制           | 350 |
| 12.14. 添加交空            | 352 |
| 12.15. 源和源管理           | 353 |

|  |     |
|--|-----|
| 12.16. 使用和 FreeBSD ACPI .....          | 354 |
| 13. FreeBSD 引导程序 .....                 | 360 |
| 13.1. 概述 .....                         | 360 |
| 13.2. 引导 .....                         | 360 |
| 13.3. 引导管理器和各引导段 .....                 | 361 |
| 13.4. 内核在引导的交互 .....                   | 366 |
| 13.5. Device Hints .....               | 366 |
| 13.6. Init : 进程控制及初始化 .....            | 367 |
| 13.7. 关机 (shutdown) 程序 .....           | 368 |
| 14. 用户和基本的系统管理 .....                   | 369 |
| 14.1. 概述 .....                         | 369 |
| 14.2. 介绍 .....                         | 369 |
| 14.3. 超用户 .....                        | 370 |
| 14.4. 系统 .....                         | 370 |
| 14.5. 用户 .....                         | 371 |
| 14.6. 修改 .....                         | 371 |
| 14.7. 限制用户使用系统源 .....                  | 375 |
| 14.8. .....                            | 377 |
| 15. 安全 .....                           | 379 |
| 15.1. 概述 .....                         | 379 |
| 15.2. 介绍 .....                         | 379 |
| 15.3. 保护 FreeBSD 的安全 .....             | 381 |
| 15.4. DES、Blowfish、MD5, 以及 Crypt ..... | 386 |
| 15.5. 一次性口令 .....                      | 387 |
| 15.6. TCP Wrappers .....               | 390 |
| 15.7. Kerberos5 .....                  | 392 |
| 15.8. OpenSSL .....                    | 399 |
| 15.9. IPsec 上的 VPN .....               | 402 |
| 15.10. OpenSSH .....                   | 408 |
| 15.11. 文件系统控制表 .....                   | 414 |
| 15.12. 第三方安全 .....                     | 415 |
| 15.13. FreeBSD 安全公告 .....              | 416 |
| 15.14. 程序 .....                        | 418 |
| 16. Jails .....                        | 420 |
| 16.1. 概述 .....                         | 420 |
| 16.2. 与 Jail 相关的一些 .....               | 420 |
| 16.3. 介绍 .....                         | 420 |
| 16.4. 建立和控制 jail .....                 | 421 |
| 16.5. 微和管理 .....                       | 423 |
| 16.6. Jail 的用途 .....                   | 424 |
| 17. 限制控制 .....                         | 433 |

|                                       |     |
|---------------------------------------|-----|
| 17.1. 概要 .....                        | 433 |
| 17.2. 本章输出的重要输出 .....                 | 434 |
| 17.3. 关于 MAC 的说明 .....                | 434 |
| 17.4. 理解 MAC 输出 .....                 | 435 |
| 17.5. 安全配置 .....                      | 440 |
| 17.6. 模式配置 .....                      | 440 |
| 17.7. MAC seeotheruids 模式 .....       | 440 |
| 17.8. MAC bsdextended 模式 .....        | 441 |
| 17.9. MAC ifoff 模式 .....              | 442 |
| 17.10. MAC portacl 模式 .....           | 442 |
| 17.11. MAC partition (分区) 模式 .....    | 443 |
| 17.12. MAC 多 (Multi-Level) 安全模式 ..... | 444 |
| 17.13. MAC Biba 模式 .....              | 446 |
| 17.14. MAC LOMAC 模式 .....             | 447 |
| 17.15. MAC Jail 中的 Nagios .....       | 448 |
| 17.16. User Lock Down .....           | 452 |
| 17.17. MAC 框架的故障排除 .....              | 452 |
| 18. 安全事件输出 .....                      | 454 |
| 18.1. 概述 .....                        | 454 |
| 18.2. 本章中的一些输出 .....                  | 454 |
| 18.3. 安装支持 .....                      | 455 |
| 18.4. 运行配置 .....                      | 455 |
| 18.5. 管理子系统 .....                     | 458 |
| 19. 存储 .....                          | 461 |
| 19.1. 概述 .....                        | 461 |
| 19.2. 命名 .....                        | 461 |
| 19.3. 添加磁盘 .....                      | 462 |
| 19.4. RAID .....                      | 464 |
| 19.5. USB 存储 .....                    | 468 |
| 19.6. 创建和使用光学介质(CD) .....             | 471 |
| 19.7. 创建和使用光学介质(DVD) .....            | 477 |
| 19.8. 创建和使用 .....                     | 482 |
| 19.9. 用磁盘机 .....                      | 483 |
| 19.10. 用 .....                        | 485 |
| 19.11. 策略 .....                       | 486 |
| 19.12. 程序 .....                       | 487 |
| 19.13. 网络、内存和以及映像文件介质的虚拟文件系统 .....    | 490 |
| 19.14. 文件系统快照 .....                   | 492 |
| 19.15. 文件系统配置 .....                   | 494 |
| 19.16. 加密磁盘分区 .....                   | 496 |
| 19.17. 交叉分区加密 .....                   | 504 |

|  |     |
|--|-----|
| 19.18. 高可用性存储 (HAST)                     | 505 |
| 20. GEOM: 虚拟化磁盘框架                        | 513 |
| 20.1. 概述                                 | 513 |
| 20.2. GEOM 介绍                            | 513 |
| 20.3. RAID0 - 条带                         | 513 |
| 20.4. RAID1 - 镜像                         | 516 |
| 20.5. RAID3 - 使用专用校验位的字条带                | 519 |
| 20.6. GEOM Gate 网络                       | 521 |
| 20.7. 磁盘添加卷                              | 521 |
| 20.8. 通过 GEOM 的 UFS 日志                   | 524 |
| 21. 文件系统 Support                         | 526 |
| 21.1. 概述                                 | 526 |
| 21.2. Z 文件系统 (ZFS)                       | 526 |
| 22. Vinum 卷管理程序                          | 535 |
| 22.1. 概述                                 | 535 |
| 22.2. 磁盘容量太小                             | 535 |
| 22.3. 瓶颈                                 | 535 |
| 22.4. 数据的完整性                             | 536 |
| 22.5. Vinum 目录                           | 537 |
| 22.6. 一些例子                               | 538 |
| 22.7. 对象命名                               | 544 |
| 22.8. 配置 Vinum                           | 545 |
| 22.9. 使用 Vinum 作为根文件系统                   | 547 |
| 23. 虚拟化                                  | 551 |
| 23.1. 概述                                 | 551 |
| 23.2. 作为客户 OS 的 FreeBSD                  | 551 |
| 23.3. 作为宿主 OS 的 FreeBSD                  | 578 |
| 24. 本地化 - I18N/L10N 使用和配置                | 580 |
| 24.1. 概述                                 | 580 |
| 24.2. 基础知识                               | 580 |
| 24.3. 使用本地化语言                            | 580 |
| 24.4. I18N 程序                            | 586 |
| 24.5. 本地化 FreeBSD                        | 586 |
| 25. 更新与升级 FreeBSD                        | 590 |
| 25.1. 概述                                 | 590 |
| 25.2. FreeBSD 更新                         | 590 |
| 25.3. Portsnap: 一个 Ports Collection 更新工具 | 596 |
| 25.4. 更新系统附带的文档                          | 598 |
| 25.5. 追踪分支                               | 602 |
| 25.6. 同步的源                               | 605 |
| 25.7. 重新创建 "world"                       | 605 |

|                             |     |
|-----------------------------|-----|
| 25.8. 删除的文件、目录和函数           | 619 |
| 25.9. 跟踪多台机器                | 621 |
| 26. DTrace                  | 622 |
| 26.1. 概述                    | 622 |
| 26.2. 上的差异                  | 622 |
| 26.3. 用 DTrace 支持           | 623 |
| 26.4. 使用 DTrace             | 623 |
| 26.5. D 语言                  | 626 |
| IV: 网络                      | 627 |
| 27. 串口                      | 628 |
| 27.1. 概述                    | 628 |
| 27.2. 介绍                    | 628 |
| 27.3. 端                     | 632 |
| 27.4. 入服务                   | 636 |
| 27.5. 输出                    | 643 |
| 27.6. 置串口控制台                | 646 |
| 28. PPP 和 SLIP              | 655 |
| 28.1. 概述                    | 655 |
| 28.2. 使用 PPP                | 655 |
| 28.3. 使用内核 PPP              | 667 |
| 28.4. PPP 故障排除              | 675 |
| 28.5. 使用基于以太网的 PPP (PPPoE)  | 679 |
| 28.6. 使用 ATM 上的 PPP (PPPoA) | 680 |
| 28.7. 使用 SLIP               | 684 |
| 29. 子件                      | 693 |
| 29.1. 概述                    | 693 |
| 29.2. 使用子件                  | 693 |
| 29.3. sendmail 配置           | 695 |
| 29.4. 改的子件代理程序              | 697 |
| 29.5. 疑解答                   | 699 |
| 29.6. 高主                    | 702 |
| 29.7. SMTP 与 UUCP           | 704 |
| 29.8. 只送件的配置                | 706 |
| 29.9. 号接使用件送                | 707 |
| 29.10. SMTP                 | 708 |
| 29.11. 件用代理                 | 710 |
| 29.12. 使用 fetchmail         | 717 |
| 29.13. 使用 procmail          | 718 |
| 30. 网服器                     | 720 |
| 30.1. 概要                    | 720 |
| 30.2. inetd "超服器"           | 720 |

|  |     |
|--|-----|
| 30.3. 网⌘文件系⌘ (NFS)   | 724 |
| 30.4. 网⌘信息服⌘ (NIS/YP)                                      | 729 |
| 30.5. 网⌘自⌘配置 (DHCP)  | 744 |
| 30.6. 域名系⌘ (DNS)   | 748 |
| 30.7. Apache HTTP 服⌘器                                      | 760 |
| 30.8. 文件⌘⌘⌘⌘ (FTP)   | 765 |
| 30.9. ⌘ Microsoft® Windows® 客⌘机提供文件和打印服⌘ (Samba)           | 767 |
| 30.10. 通⌘ NTP ⌘行⌘同⌘  | 769 |
| 30.11. 使用 <b>syslogd</b> ⌘⌘程主机的日志                          | 772 |
| 31. 防火⌘  | 776 |
| 31.1. 入⌘   | 776 |
| 31.2. 防火⌘的概念   | 776 |
| 31.3. 防火⌘件包  | 777 |
| 31.4. OpenBSD Packet Filter (PF) 和 ALTQ                    | 777 |
| 31.5. IPFILTER (IPF) 防火⌘                                   | 780 |
| 31.6. IPFW   | 799 |
| 32. 高⌘网⌘   | 816 |
| 32.1. 概述   | 816 |
| 32.2. 网⌘和路由  | 816 |
| 32.3. 无⌘网⌘   | 822 |
| 32.4. ⌘牙   | 841 |
| 32.5. ⌘接   | 849 |
| 32.6. ⌘路聚合与故障⌘移  | 855 |
| 32.7. 无⌘操作   | 860 |
| 32.8. 从 PXE ⌘⌘一个 NFS 根文件系⌘                                 | 866 |
| 32.9. ISDN   | 870 |
| 32.10. 网⌘地址⌘⌘  | 873 |
| 32.11. 并⌘⌘⌘ IP (PLIP)                                      | 877 |
| 32.12. IPv6  | 879 |
| 32.13. ⌘⌘⌘⌘模式 (ATM)  | 883 |
| 32.14. Common Address Redundancy Protocol (CARP, 共用地址冗余⌘⌘) | 885 |
| V: 附⌘  | 888 |
| 附⌘ A: ⌘取 FreeBSD   | 889 |
| A.1. CDROM 和 DVD ⌘行商                                       | 889 |
| A.2. FTP 站点  | 891 |
| A.3. 匿名 CVS  | 898 |
| A.4. 使用 CTM  | 900 |
| A.5. 使用 CVSup  | 903 |
| A.6. CVS ⌘⌘  | 915 |
| A.7. AFS 站点  | 921 |
| A.8. rsync 站点  | 921 |



|                               |     |
|-------------------------------|-----|
| 附 B: 参考文献 .....               | 924 |
| B.1. 关于 FreeBSD 的书籍与杂志 .....  | 924 |
| B.2. 用指南 .....                | 925 |
| B.3. 管理指南 .....               | 925 |
| B.4. 指南 .....                 | 926 |
| B.5. 操作系统原理 .....             | 926 |
| B.6. 安全方面的参考文献 .....          | 927 |
| B.7. 硬件参考 .....               | 927 |
| B.8. UNIX® 历史 .....           | 928 |
| B.9. 各期刊 .....                | 928 |
| 附 C: Internet 上的源 .....       | 929 |
| C.1. 文件列表 .....               | 929 |
| C.2. Usenet 新闻 .....          | 942 |
| C.3. World Wide Web 服务器 ..... | 943 |
| C.4. Email 地址 .....           | 946 |
| 附 D: PGP 公钥 .....             | 947 |
| D.1. Officers .....           | 947 |

# 前言

## 预期的读者

作为 FreeBSD 的新用户，你将会在本手册第一部分学到 FreeBSD 的安装方法，同时逐步引入概念和术语来加固 UNIX® 基础。这部分只需要你有探索的精神和接受新概念的能力。

读完这些之后，手册中很漫长的第二部分是 FreeBSD 中系统管理感兴趣的所有主题的全面参考。在这些章节的内容所需要的背景知识都集中在第一章的大纲里面，如果需要，可在前面进行。

要得到附加的信息来源列表，请参见 [参考文献](#)。

## 相对于第三版的改动

目前的这本手册代表了数百位贡献者过去 10 年多所累积的努力成果。以下是自 2004 年出版的旧卷第三版之后的一些重要更改：

- [DTrace](#)，DTrace，增加了有巨大的 DTrace 性能分析工具有的信息。
- [文件系统支持](#)，文件系统支持，增加 FreeBSD 上非原生文件系统有的信息，比如 Sun™ 的 ZFS。
- [安全事件](#)，安全事件，增加了 FreeBSD 新的功能和用法。
- [虚拟化](#)，虚拟化，增加了在虚拟化件上安装 FreeBSD 有的信息。

## 相对于第二版的改动 (2004)

目前看到的这本手册的第三版是 FreeBSD 文档的成于 2004 年完成的巅峰之作。前一版的内容已需要分成多卷才能印刷出版。第三版包含了如下的主要更改：

- [配置和整合](#)，配置和整合，进行了扩充并增加了关于 ACPI 电源和电源管理，[cron](#) 系统程序，以及更多的内核变化的相关内容。
- [安全](#)，安全 一章增加了虚拟专用网 (VPNs)，文件访问控制表 (ACLs)，以及安全公告的内容。
- [制访问控制](#)，制访问控制 (MAC) 是前一版新加的章节。它解释了什么是 MAC，以及其机制如何使的 FreeBSD 系统更安全。
- [存储](#)，存储，在原有基础上增加了 USB 存储，文件系统快照，文件系统容限，基于文件及网络的文件系统，以及与加密磁盘分区有的内容。
- [Vinum 卷管理程序](#)，Vinum，是前一版中的新章节。描述了如何使用提供了无限制的磁盘、软件 RAID-0, RAID-1 和 RAID-5 的卷管理系统——Vinum。
- 在 [PPP 和 SLIP](#)，PPP 和 SLIP 一章中增加了排除故障的说明。
- [邮件件](#)，邮件件 一章中增加了关于如何使用其它的邮件代理、SMTP 邮件、UUCP、fetchmail、procmail、以及其它内容。
- [网络服务器](#)，网络服务器，是新版中全新的一章。一章包括了如何架设 Apache HTTP 服务器、ftpd，以及用于支持 Microsoft® Windows® 客户端的 Samba。一些段落来自原先的 [高网络](#)，网络网络 一章。
- [高网络](#)，网络网络 一章增加了关于在 FreeBSD 中使用 Bluetooth® 设备，安装无线网络，以及使用 ATM 模式 (ATM) 网络的内容。

- 增加了一个表，用以说明整本中出现的。
- 于全中表行了统一的美化工作。

## 相对于第一版的改进 (2001)

本手册的第二版是 FreeBSD 文档的成文年完成的巅峰之作。第二版包含了如下的主要：

- 添加了完整的索引。
- 用图形替了以前所有用 ASCII 。
- 每个章添加了准大，列出了章所包含的信息和者所了解的知。
- 内容地分成三个部分：“起”，“系管理”和“附”。
- [安装 FreeBSD](#) ("安装 FreeBSD") 新版本中使用了屏片，使新用更容易的会正文。
- [UNIX 基](#) ("UNIX® 基") 充了程、守程和信号的附加信息。
- [安装用程序. Packages 和 Ports](#) ("安装用程序") 充了二制包管理的附加信息。
- [X Window 系](#) ("X Window 系") 新版本中着重介使用代面技例如 XFree86™ 4.x 上的 KDE 和 GNOME
- [FreeBSD 引程](#) ("FreeBSD 程") 第一版内容行充。
- [存](#) ("存") 由第一版中个独的章"磁"和"合并而成。我部分作一个整体比容易理解。同 RAID (包括硬件和件 RAID) 部分也被添加来。
- [串口通](#) ("串口通信") 第一版行完善，并 FreeBSD 4.x/5.x 做了更新。
- [PPP 和 SLIP](#) ("PPP 和 SLIP") 全部更新。
- 多新的内容被添加到 [高网](#) ("高网")。
- [子件](#) ("子件") 加了于配置 sendmail 的信息。
- [Linux® 二制兼容模式](#) ("Linux® 兼容性") 加了于安装 Oracle® 和 SAP® R/3® 的信息。
- 第二版中也涵了下列主：
  - [配置和整\(置和整\)](#)。
  - 多媒体([多媒体](#))

## 本手册的

本手册分成了五个清晰的部分。第一部分 [起](#) 涵了 FreeBSD 的安装和基本使用方法。者可根据自己的情况按序或者跳一些熟悉的主来。第二部分 [常用操作](#) 涵了 FreeBSD 常用的功能，部分可以不按序。个部分由一个明的大始，个大描述本章涵的内容和者已知道的知。主要是者可以更好的感趣的章。第三部分 [系管理](#) 涵了 FreeBSD 高用所感趣的广泛的。第四部分 [网通](#) 包括了网和服的，而第五部分是源信息的附。

[介](#)，[介](#)

向新用介 FreeBSD。它描述了 FreeBSD 的史、目和模式。

## 安装 *FreeBSD*，安装

本章将会用完成安装。一些高级安装主题，例如如何通过串行控制台安装，也涵盖在内。

## *UNIX* 基础，*UNIX*® 基础

本章涵盖了 *FreeBSD* 操作系统的基础命令和功能。如果熟悉 *Linux*® 或者其他 *UNIX*® 操作系统，可以跳章节。

## 安装应用程序，*Packages* 和 *Ports*，安装应用程序

本章涵盖如何用 *FreeBSD* 的 "Ports Collection" 和二进制软件包来安装第三方软件。

## *X Window* 系统，*X Window* 系统

本章概要地描述了 *X Window System* 系统并介绍了如何在 *FreeBSD* 上使用它。此外他也描述了常用的桌面环境，例如 *KDE* 和 *GNOME*。

## 网络应用，网络应用

列出了一些常用的网络应用程序，比如 web 服务器和办公套件，描述了在 *FreeBSD* 上如何安装它们。

## 多媒体，多媒体

展示了如何设置声卡和回放支持。也描述了一些音频和视频应用程序。

## 配置 *FreeBSD* 的内核，配置 *FreeBSD* 内核

解释了需要什么配置一个新内核并提供了配置、安装、安装自定义内核的说明。

## 打印，打印

描述了 *FreeBSD* 上打印机管理，包括横幅、打印、有初始的设置。

## *Linux*® 二进制兼容模式，*Linux*® 二进制兼容

描述了 *FreeBSD* 的 *Linux*® 兼容特性。也提供了许多流行的 *Linux*® 应用程序的安装说明，比如 *Oracle*® 和 *Mathematica*®。

## 设置和调整，配置和调整

本章描述了管理调整 *FreeBSD* 系统以优化性能可能用到的一些参数。也描述了 *FreeBSD* 中的各种配置文件以及它们所在的位置。

## *FreeBSD* 引导程序，引导程序

本章描述 *FreeBSD* 的引导程序并且解释了如何用配置来控制它。

## 用户和基本的用户管理，用户和基本用户管理

本章描述了如何创建和操作用户，同时也描述了设置用户资源限制和其他用户管理任务的方法。

## 安全，安全

描述了保护 *FreeBSD* 系统安全可以使用的许多工具，包括 *Kerberos*，*IPsec* 以及 *OpenSSH*。

## *Jails*，*Jail*

介绍了 *jail* 框架，以及 *jail* 相对于 *FreeBSD* 中旧的 *chroot* 支持的改进。

## 控制控制，控制控制

解释了何控制控制 (*MAC*) 以及如何利用同一机制来加强 *FreeBSD* 系统的安全。

## 安全事件00, 安全事件00

介绍了 FreeBSD 事件00是什么, 以及如何安装、配置它, 并00或000000信息。

## 存0, 存0

本章描述了00用 FreeBSD 来管理存0介0和文件系0, 包括物理磁0、RAID 0列、光学和磁0媒体、后0存0磁0以及网0文件系0。

## GEOM. 模0化磁000框架, GEOM

介绍了 FreeBSD 中的 GEOM 框架是什么, 以及如何配置它所支持的各0 RAID。

## 文件系0 *Support*, 文件系0支持

探讨了 FreeBSD 0非原生文件系0的支持, 比如 Sun™ 的 Z 文件系0。

## Vinum 卷管理程序, Vinum

本章描述了00使用00卷管理器 Vinum。它提供了00无0的00磁0和0件 RAID-0、RAID-1 以及 RAID-5。

## 虚0化, 虚0化

介绍了虚0化系0提供的功能, 以及如何配合 FreeBSD 使用它0。

## 本地化—*I18N/L10N*使用和0置, 本地化

本章描述了如何在 FreeBSD 上使用非英00言。它涵0了系0和0用程序0的本地化。

## 更新与升0 *FreeBSD*, 更新与升0 *FreeBSD*

介绍了 FreeBSD-STABLE、FreeBSD-CURRENT 以及 FreeBSD 0行版本之0的差0。描述了一般用0如何0跟000程并从中受益。涵0了如何更新用0的系0至0行版最新安全修正的方法。

## DTrace, DTrace

本章描述了如何在 FreeBSD 上配置和使用 Sun™ 的 DTrace 工具。00跟踪可以通000的系0分析, 0助0出系0性能瓶0。

## 串口通0, 串行通信

本章解0了如何0接0端和0制解0器到 FreeBSD 系0, 包括0入和0出0接。

## PPP 和 SLIP, PPP 和 SLIP

本章描述了如何用 FreeBSD 通0使用 PPP, SLIP 或者基于以太网的 PPP (PPPoE) 来0接0程系0。

## 0子0件, 0子0件

本章解0了一个 email 服0器的不同0成部分并且0000了0于最流行的 mail 服0器0件 sendmail 的配置。

## 网0服0, 网0服0

提供了00的指引和示0配置文件以0明如何将一台 FreeBSD 机器作0网0文件系0服0器, 域名服0器, 网0信息服0器或00同0服0器来使用的方法。

## 防火0, 防火0

解0了基于0件的防火0的原理, 并提供了0于配置 FreeBSD 上的几0防火0的000明。

## 高0网00用, 高0网00用

描述了0多0于网0的主0, 包括如何在0的局域网中共享 Internet 0接, 高0路由00, 无0网0,

Bluetooth®, ATM, IPv6以及许多高。

## 取 *FreeBSD*, 取 *FreeBSD*

列出了得 FreeBSD 安装 CDROM 或 DVDROM 的不同源, 也提供了允许自由下 FreeBSD 的不同 Internet 站点。

## 参考目, 参考目

由于本手册触及到了很多不同的主题, 因而可能引起想要更多解。 参考目列出了很多写作本参考的好。

## Internet 上的源, Internet 上的源

述了很多 FreeBSD 用有用的能提出并行技术交流的于 FreeBSD 的。

## PGP 公, PGP 公

列出了一些 FreeBSD 者的 PGP 名公。

# 本中使用的一些定

了使本保持一致性和易性特做了以下定：

## 排版定

### 斜体

斜体 字用来表示文件名、URLs、文字和的主流用法。

### 等

等 字体用来表示信息、命令、境量、port 的名字、主机名、用名、名、名、量名, 以及代片断。

### 粗体

粗体 字用来表示用程序、命令和字。

## 用入

按用粗体来突出于其他文本。合意味着字用+接, 同的按下它, 例如：

Ctrl + Alt + Del

表示同按下 Ctrl, Alt 和 Del 。

按序依次入的字通常是用逗号隔, 例如：

Ctrl + X, Ctrl + S

意味着用同按 Ctrl 和 X, 然后同按 Ctrl 和 S。

## 示例

以 E:\> 的例子代表一个 MS-DOS® 命令。除非有明, 些命令都可以在一个代的 Microsoft®

Windows® "命令行"□□□境被□行。

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

以 # □□的例子代表必□以 FreeBSD 超□用□身□□行的命令。□可以用 **root** 身□登□来□入□些命令，或者以普通□号登□然后用 **su(1)** 来□得超□用□□限。

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

以 % □□的例子代表命令□□被普通□号□行。除非□有□明，在□置□境□量和使用的其他 shell 命令均□ C-shell □法。

```
% top
```

## 致□

□所看到的□本□是全球几百人努力的□果。  
□献都是非常有用的。

无□他□只是□正一些□□或提交完整的章□，所有的

一些公司通□提供□金□作者□注于文□□□、提供出版□金等等方式来支持文□□□。其中，BSDi (后并入<http://www.windriver.com>[Wind River Systems]) □助 FreeBSD 文□□□成□来□□改善□本□直到 2000 年三月第一个印刷版 (ISBN 1-57176-241-8) 的出版。Wind River Systems 同□□助其他作者来□□出□□做很多改□和□文章添加一些附加章□。□□工作□束于 2001 年 11 月印刷第二版 (ISBN 1-57176-303-1)。在 2003-2004 □年中，<http://www.freebsdmail.com>[FreeBSD Mall]，□□向□改□□本手册以使其第三版印刷版本能□出版的志□者支付了□酬。

# Part I: 起

手册的以下章主要是开始使用 FreeBSD 的及管理:

- FreeBSD 入。
- 安装程向。
- 教 UNIX® 基本知和基本原理。
- 展示如何在 FreeBSD 上安装大量的第三方用程序。
- 介使用 X, UNIX® 口系, 以及一些能提高工作率的境配置。

我用最少的数来保持前言的索引, 以至于可以用最少翻次数将手册从至尾。



# Chapter 1. 介绍

## 1.1. 概述

非常感谢 FreeBSD 社区！下面的章节涵盖了 FreeBSD 项目的各个方面，比如它的历史、目标、模式，等等。

读完本章，你将了解：

- FreeBSD 与其它计算机操作系统的关系。
- FreeBSD 项目的历史。
- FreeBSD 项目的目标。
- FreeBSD 开放源代码模式的基石。
- 当然还有：“FreeBSD”这个名字的由来。

## 1.2. 欢迎来到 FreeBSD 的世界！

FreeBSD 是一个支持 Intel (x86 和 Itanium®), AMD64, Sun UltraSPARC® 计算机的基于 4.4BSD-Lite 的操作系统。到其他体系结构的移植也在进行中。你也可以看看 [FreeBSD 的历史](#)，或者[最新的发行版本](#)。如果你有意捐助(代码，硬件，基金)，看看[FreeBSD 提供捐助](#)篇文章。

### 1.2.1. FreeBSD 能做些什？

FreeBSD 有许多非凡的特性。其中一些是：

- 抢占式多任务与预先调整保持在用户程序和用户之间的平滑公正的分享计算机资源，即使工作在最大的负载之下。
- 多用户使得多用户能同时使用同一 FreeBSD 系统做各件事情。比如，像打印机和磁存储器的系统外，可以完全地在系统或者网上的所有用户之间共享，可以用户或者用进程个别的资源限制，以保证系统资源不被滥用。
- 符合业界标准的强大 TCP/IP 网络支持，例如 SCTP、DHCP、NFS、NIS、PPP，SLIP，IPsec 以及 IPv6。这意味着你的 FreeBSD 主机可以很容易地和其他系统互连，也可以作为企业的服务器，提供重要的功能，比如 NFS(存储文件)以及 email 服务，或将你的系统接入 Internet 并提供 WWW，FTP，路由和防火墙(安全)服务。
- 内存保护\_保护程序(或者用户)不会相互干扰。一个用户程序崩溃不会以任何方式影响其他程序。
- FreeBSD 是一个 32 位操作系统 (在 Itanium®, AMD64, 和 UltraSPARC® 上是 64 位)，并且从一开始就是如此。
- 业界标准的 X Window 系统 (X11R7) 便宜的常备 VGA 显示卡和适配器提供了一个图形化的用户界面(GUI)，并且完全开放。
- 和许多 Linux，SCO，SVR4，BSDI 和 NetBSD 程序的二进制代码兼容性。
- 数以千计的 ready-to-run 用户程序可以从 FreeBSD ports 和 packages 套件中得到。你可以顺利地从中得到，何必搜索网络？
- 可以在 Internet 上得到成千上万其它 easy-to-port 的用户程序。FreeBSD 和大多数流行的商业 UNIX®

代兼容，因此大多数应用程序不需要或者只要很少的改动就可以。

- 式求\_虚\_内存\_和"集成的 VM/buffer 存"有效地足了应用程序巨大的内存需求并依然保持其他用的交互式。
- SMP 提供多处理器的支持。
- 内建了完整的 C、C++、Fortran 工具。多附加的用于高研究和的程序言，也可以在通 ports 和 packages 套件得。
- 完整的系\_源代\_意味着境的最大程度的控制。当有了一个真正的放系，什要受困于私有的解决方案，任商公司布？
- 富的\_在文\_。
- 不如此！

FreeBSD 基于加州大学伯克利分校计算机系研究 (CSRG) 布的 4.4BSD-Lite，承了 BSD 系良。除了 CSRG 秀的工作之外，FreeBSD 目花了非常多的来化整系，使其在真情况下有最好的性能和可性。在今，多商巨人正 PC 操作系加新功能、提升和改善其可性，以便在其上展激烈争的同，FreeBSD 在已能提供所有一切了！

FreeBSD 可以提供的用事上局限于的想象力。从件到工厂自化，从存控制到遥的人造星天方位控制，如果商的 UNIX® 品可以做到，那就非常有可能也可以用 FreeBSD 来做！FreeBSD 也大地受益于全世界的研究中心和大学的数以千的高量的应用程序，些程序通常只需要很少的花甚至免。可用的商应用程序，天也都在大量地加。

因 FreeBSD 自身的源代是完全公的，所以于特定的应用程序或目，可以系行最大限度的定制。于大多数主流的商生商的操作系来几乎是不可能的。以下是当前人用 FreeBSD 的某些程序的例子：

- Internet 服：FreeBSD 内建的大的 TCP/IP 网使它得以成各 Internet 服的理想平台，比如：
  - FTP 服器
  - World Wide Web 服器(准的或者安全的 [SSL])
  - IPv4 and IPv6 路由
  - 防火和 NAT("IP 装")网
  - 子件服器
  - USENET 新和子布告系
  - 有多...

使用 FreeBSD，可以容易地从便宜的 386 PC 起，并随着的企成，一路升到有 RAID 存的四路 Xeon 服器。

- 教育：是一名计算机科学或者相工程域的学生？学操作系，计算机体系网没有比在 FreeBSD 可提供的体下手实践更好的法了。多可自由使用的 CAD、数学和形包也使它于那些主要趣是在计算机上完成\_其他\_工作的人非常有帮助。
- 研究：有完整的系源代，FreeBSD 于操作系研究以及其他计算机科学分支都是一个好的平台。FreeBSD 可自由得的本性，同可以使在不同地方的在放的上、交流想法与合作可能，且不必担心特的版定或者限制。
- \_网：\_需要一个新的路由器？一台域名服器 (DNS)？一个隔的内部网的防火？FreeBSD

可以容易的把在角落不用的 386 或者 486 PC 成一台完善的包能力的高路由器。

- *X Window* 工作站：FreeBSD 是廉价 X 端的一佳解决方案，可以使用免的 X11 服务器。与 X 端不同，如果需要的 FreeBSD 能在本地直接行程序，因而少了中央服务器的担。FreeBSD 甚至能在 "无" 境下，使得端更便宜和易于管理。
- 件：基本的 FreeBSD 系有包括著名的 GNU C/C++ 器和工具在内的一整套工具。

FreeBSD 可以通过 CD-ROM、DVD，以及匿名 FTP 以源代码和二进制方式得。看[FreeBSDf](#) 了解取 FreeBSD 的更多。

### 1.2.2. 在使用 FreeBSD?

FreeBSD 被世界上最大的 IT 公司用作和品的平台，包括：

- [Apple](#)
- [Cisco](#)
- [Juniper](#)
- [NetApp](#)

FreeBSD 也被用来支持 Internet 上一些最大的站点，包括：

- [Yahoo!](#)
- [Yandex](#)
- [Apache](#)
- [Rambler](#)
- [新浪网](#)
- [Pair Networks](#)
- [Sony Japan](#)
- [Netcraft](#)
- [NetEase](#)
- [Weathernews](#)
- [TELEHOUSE America](#)
- [Experts Exchange](#)

等等多。

## 1.3. 于 FreeBSD 目

下面的章提供了目的一些背景信息，包括要的史、目目、以及目模式。

### 1.3.1. FreeBSD 的要史

FreeBSD 目起源于 1993 年早期，部分作 "Unofficial 386BSD Patchkit" 的副物，patchkit 的最后 3 个是：Nate Williams, Rod Grimes 和我。

我最初的目的是做出一套 386BSD 的版本以修正一些 Patchkit 机制无法解决的 bug)。很多人可能记得早期的项目名称叫做 "386BSD 0.5" 或者 "386BSD Interim" 就是那个原因。

386BSD 是 Bill Jolitz 的操作系统，到那时已被严重地忽略了一年之久。由于 Patchkit 在过去的日子里都在急剧膨胀，使得其运行消化吸收得越来越困难，因此我一致同意做些事情并决定通过提供一个干净的 "cleanup" 版本来帮助 Bill。然而，Bill 却在事先没有指出那个项目如何开展下去的情况下，突然决定退出那个项目，最后一个项目只好被迫停止。

没多久，我即便没有 Bill 的支持，项目仍有保留的价值，因此，我采用了 David Greenman 的意见，将其命名为 "FreeBSD"。在和当时的几个用户商量后，我提出了最初的目标，而这件事明朗化后，这个项目就走上了正轨，甚至可能成功。为了拓展 FreeBSD 的发行渠道，我抱着试试看的心态，联系了光碟商 Walnut Creek CDROM，以便那些上网不方便的用户得到 FreeBSD。Walnut Creek CDROM 不支持发行 FreeBSD 光碟版的想法，这个项目提供了所需的计算机和高速网络接入。在那里，若没有 Walnut Creek CDROM 对一个完全未知的项目的空前信任，FreeBSD 不太可能像它今天这样，影响如此深远，发展如此快速。

第一个 CD-ROM (以及在整个互联网内发行的) 发行版本是 FreeBSD 1.0，于 1993 年 10 月发布。这个版本基于 U.C. Berkeley 的 4.3BSD-Lite("Net/2")磁碟，也有许多组件是 386BSD 和自由软件基金会提供的。由于第一次发行，算是相当成功了。在 1994 年 5 月，我发布了更加成功的 FreeBSD 1.1 版。

在那段期间，发生了一些意外的情况。Novell 和 U.C. Berkeley 就 Berkeley Net/2 磁碟知识产权的拉锯战的官司变成了和解。和解中的一部分是 U.C. Berkeley 作出的，令 Net/2 中的一大部分内容成为 "受限的 (encumbered)" 和属于 Novell 知识产权的代码，而后者在不久前从 AT&T 收到了某些许可；作为回报，Berkeley 得到了来自 Novell 的 "许可"，在 4.4BSD-Lite 版本正式发布时，可以声明为不受限制的 (unencumbered)，原有的 Net/2 用户强烈建议移到这个版本。这包括了 FreeBSD，而我的项目被允许在 1994 年 6 月底之前发行基于 Net/2 的产品。根据和解协议，在最后期限之前我发布了一个最新版本，这个版本是 FreeBSD 1.1.5.1。

接下来，FreeBSD 开始了艰苦的从全新的、不太完整的 4.4BSD-Lite 重新编写自己的工程。"Lite" 版本中，Berkeley 的 CSRG 除了用于系统引导的一大堆代码 (由于各个各的法律需求)，而当 4.4 在 Intel 平台的移植版本还有很多工作没有完成。直到 1994 年 11 月，我的项目才完成了过渡，并通过网络以及 CD-ROM (在 12 月底) 上发布了 FreeBSD 2.0。尽管系统中有很多粗糙的地方，这个版本还是取得了巨大的成功，并在 1995 年 6 月发布了更大和易于安装的 FreeBSD 2.0.5 版本。

我于 1996 年 8 月发布了 FreeBSD 2.1.5 版本，它在 ISP 和商业团体中非常流行。随后，2.1-STABLE 分支的一个版本应运而生，它就是 FreeBSD 2.1.7.1，在 1997 年 2 月发布并停止了 2.1-STABLE 的主流地位。现在，它处于状态，提供安全性的更新和其他重要的修复的更新(RELENG\_2\_1\_0)。

FreeBSD 2.2 版本作为 RELENG\_2\_2 分支，于 1996 年 11 月从主分支 ("-CURRENT")分出来。它的第一个完整版(2.2.1)于 1997 年 4 月发布出来。97 年夏秋之间，随着 2.2 分支的更新版本的发布。其最后一版(2.2.8)于 1998 年 11 月发布出来。第一个官方的 3.0 版本出现在 1998 年 10 月，意味着 2.2 分支的结束。

1999 年 1 月 20 日又出了新的分支，就是 4.0-CURRENT 和 3.X-STABLE 分支。从 3.X-STABLE 起，3.1 在 1999 年 2 月 15 日发行，3.2 在 1999 年 5 月 15 日，3.3 在 1999 年 9 月 16 日，3.4 在 1999 年 12 月 20 日，3.5 在 2000 年 6 月 24 日，接下来几天后发布了很少的修修补补至 3.5.1，加入了 Kerberos 安全性方面的修复。这是 3.X 分支最后一个发行版本。

随后在 2000 年 3 月 13 日出了一个新的分支，也就是 4.X-STABLE。之后发布了多的发行版本：4.0-RELEASE 于 2000 年 3 月发布，而最后的 4.11-RELEASE 是在 2005 年 1 月发布的。

期待已久的 5.0-RELEASE 于 2003 年 1 月 19 日正式发布。它是将近三年的开发高峰之作，同时也标志着 FreeBSD 在先前的多处理器和应用程序程序支持的巨大成就，并引入了对于 UltraSPARC® 和 ia64 平台的支持。之后于 2003 年 6 月发布了 5.1。最后一个从 -CURRENT 分支的 5.X 版本是 5.2.1-RELEASE，它在 2004 年 2 月正式发布。

RELENG\_5 于 2004 年 8 月正式建立，紧随其后的是 5.3-RELEASE，它是 5-STABLE 分支的标志性行版。该分支的最后一个版本，5.5-RELEASE 是在 2006 年 5 月发布的。RELENG\_5 分支不会有后续的行版了。

其后在 2005 年 7 月又建立了 RELENG\_6 分支。而 6.X 分支上的第一个版本，即 6.0-RELEASE，是在 2005 年 11 月发布的。该分支的最后一个版本，6.4-RELEASE 是在 2008 年 11 月发布的。RELENG\_6 分支上不再会有发布版本了。它是最后一个支持 Alpha 硬件架构的版本。

RELENG\_7 分支于 2007 年 10 月建立。第一个该分支的行版是 7.0-RELEASE，该版本是 2008 年 2 月发布的。最新的 11.2-RELEASE 是在 June 28, 2018 发布的。RELENG\_7 将不会有其它后续发布版本。

其后在 2009 年 8 月又建立了 RELENG\_8 分支。8.X 分支的第一个版本，8.0-RELEASE 是在 2009 年 11 月发布的。最新的 12.0-RELEASE 于 December 11, 2018 发布。RELENG\_8 将会有其它后续发布版本。

目前，中期的开发项目在 9.X-CURRENT (主干, trunk) 分支中进行，而 9.X 的 CD-ROM (当然，也包括网络快照版本可以在 [快照服务器](#) 找到)。

### 1.3.2. FreeBSD 项目

FreeBSD 项目的目的是无附加条件地提供能用于任何目的的代码。我心中的多人代码（以及项目本身）都有非常大的投入，因此当然不介意偶尔有一些金钱上的问题，但我并没打算坚决地要求得到帮助。我项目的首要“使命”是向任何人提供代码，不管他们打算用些代码做什么，因代码将被更广泛地使用，从而最大限度地实现其价值。项目是自由软件最基本的，同时也是我所倡导的一个项目。

我源代码中，以 GNU 公共许可 (GPL) 或者 GNU 函数公共许可 (LGPL) 发布的那些代码有少量的附加限制，但好只是限制性的要求放代而不是的。由于使用 GPL 的代码在商业用途上会增加若干性质，因此，如果可以的话，我更偏好使用限制相对更宽松的 BSD 版本来发布代码。

### 1.3.3. FreeBSD 模式

FreeBSD 的模式是一个非常开放且有伸缩性的工程，就像从我项目的[贡献者列表](#)里看到的，它是完全由来自全世界的数以百千的贡献者发展起来的。FreeBSD 的基本允许数以百千的贡献者通过互联网协同工作。我也经常注视着那些我感兴趣的新的贡献者和新的创意，那些有兴趣更进一步参与项目的人只需要在 [FreeBSD 技术邮件列表](#) 联系我。[FreeBSD 公告邮件列表](#) 那些希望了解我工作所及到某些领域的人也是有用的。

无论是独立地工作或者封闭式的开发工作，了解 FreeBSD 模式和它的开发过程都是有益的：

#### SVN 和 CVS 代码

在过去的几年中 FreeBSD 的中央源代码是由 CVS (并行版本控制系统) 来管理的，CVS 是一个与 FreeBSD 兼容的可自由获得的源代码控制工具。自 2008 年六月起，该项目开始使用 SVN (Subversion)。这次转变是非常必要的，因为 CVS 的对于快速扩展源代码和历史的限制越来越明显。在主源使用 SVN，客户端的工具像 CVSup 和 csup 有些依赖于旧的 CVS 基础依然可以使用 - 因为于 SVN 源代码的修改会被回写 CVS。目前只有中央源代码是由 SVN 控制的。文档，万维网和 Ports 项目仍旧使用着 CVS。The primary repository resides on a machine in Santa Clara CA, USA 主 CVS 代码放置在美国加利福尼亚州圣克拉拉的一台机器上，它被复制到全世界的大量镜像站上。包含 -CURRENT 和 -STABLE 的 SVN 项目也同

能非常容易的的机器上。参[同源的源](#) 得更多的相信息。

## committer 列表

*committer* 是那些 CVS 有\_写\_限的人， 他被授修改 FreeBSD 的源代 (即 "committer" 来自于 [cvs\(1\)](#) 的 `commit` 命令， 个命令用来把新的修改提交 CVS 代)。提交修正的最好方法是使用 [send-pr\(1\)](#) 命令。如果能在系中出了一些的， 也可以通邮件将它送至 FreeBSD committer 的邮件列表。

## FreeBSD 核心

如果把 FreeBSD 目看作一家公司， 那 FreeBSD 核心就相当于董事会。 核心的主要任务是提出体上的展， 然后定一个正的方向。 那些富有献身精神和可的者加入到 committer 伍中来也是核心的工作之一， 些新的成将作新核心成和其他人一起前。 当前的核心是 2010 年 7 月从 committer 中生的。 年一次。

一些核心的成特定的任， 也就是他必尽力保某个子系能工作正常。 FreeBSD 者的完整列表和他任， 参[献者列表](#)



核心的大部分成加入 FreeBSD 的候都是志的， 并没有从目中得任何政上的助， 所以"承"不被理解"支持保"。 前面所述"董事会"的推并不十分准， 或更好的法是， 他是一群意放他的生活， 投身于 FreeBSD 目而非其个人更好的生活的人！

## 外献者

事上， 最大的正是我提供反和修的用自己。 FreeBSD 的非集中式的者保持系的主要方式就是 [FreeBSD 技邮件列表](#)， 很多事情在那里。 看[Internet上的源](#)了解多 FreeBSD 邮件列表的更多信息。

[FreeBSD 献者列表](#) 很并在不断， 什不加入它来 FreeBSD 做献？

提供代不是个做献的唯一方式； 有一个更完整的需要做事情的列表， 可以参 [FreeBSD 目网站](#)。

的来， 我的模式好像是一没有拘束的同心。 集中式的模式， 主要是考到 FreeBSD 用的方便， 同他很容易地同一邮件， 而不会把潜在的者排除在外！ 我的目是提供一个包含有大量具有一致性 [用程序](#)的定的操作系， 以利于用的安装和使用， - 模式在完成目的程中工作得非常有效。

我于那些想要加入， 成 FreeBSD 者的期待是： 具有如同当前其他人一的投入， 来保持的成功！

### 1.3.4. 最新的 FreeBSD 行版本

FreeBSD 是一个免使用且有完整源代的基于 4.4BSD-Lite 的系， 它广泛行于 Intel i386™、 i486™、 Pentium®、 Pentium® Pro、 Celeron®、 Pentium® II、 Pentium® III、 Pentium® 4(或者兼容系)、 Xeon™、 和 Sun UltraSPARC® 的计算机系上。 它主要以 加州大学伯克利分校 的 CSRG 研究小的件基， 并加入了 NetBSD、 OpenBSD、 386BSD 以及来自 自由件基金会 的一些西。

自从 1994 年末我的 FreeBSD 2.0 行以来， FreeBSD 的性能， 可定制性， 定性都有了令人注目的提高。 最大的化是通整合虚内存/文件系中的高速存改的虚内存系， 它不提升了性能， 而且少了 FreeBSD 内存的需要， 使得 5 MB 内存成可接受的最小配置。 其他的改包括完整的 NIS 客端和服

器端的支持， 串式 TCP 支持， 按需号的 PPP， 集成的 DHCP 支持， 改的 SCSI 子系， ISDN 的支持， ATM， FDDI， 快速 Gigabit 以太网(1000 Mbit)支持， 提升了最新的 Adaptec 控制器的支持和修了很多的。

除了最基本的系件， FreeBSD 提供了一个有成千上万广受迎的程序成的件的 Ports Collection。到本付印， 已有超 36000 个 ports (ports 包括从 http(WWW) 服器到游、程序言、器以及能想到的几乎所有的西)。完整的 Ports Collection 大需要 3 GB 的存空。所有的只提供原始代的"修正"。使得我能容易地更新件， 而且少了老旧的 1.0 Ports Collection 硬空的浪。要一个 port， 只要切到想要安装的程序的目的， 入 `make install`， 然后系去做剩下的事情。要的一个程序完整的原始代可以从 CD-ROM 或本地 FTP 得， 所以只需要想要的足的磁空。几乎大多数的件都提供了事先好的 "package" 以方便安装， 于那些不希望从源代他自己的 ports 的人只要使用一个的命令 (`pkg_add`)就可以安装。有 package 和 ports 的更多信息可以在[安装用程序. Packages 和 Ports](#)中到。

可以在最近的 FreeBSD 主机的 `/usr/shared/doc` 目下到多有用的文件来帮助安装及使用 FreeBSD。也可以用个 HTML 器来本地安装的手册， 使用下面的 URL：

### FreeBSD 使用手册

</usr/shared/doc/handbook/index.html>

### FreeBSD FAQ

</usr/shared/doc/faq/index.html>

也可以看在 <http://www.FreeBSD.org/> 的主站上的副本。



# Chapter 2. 安装 FreeBSD

## 2.1. 概述

FreeBSD 提供了一个以文字为主，非常好用的安装程序，叫做 `sysinstall`。它是 FreeBSD 默认使用的安装程序；厂商如果想，也可以提供适合自己需要的安装程序。本章说明如何使用 `sysinstall` 来安装 FreeBSD。

学完本章之后，你将知道：

- 如何制作 FreeBSD 安装磁盘
- FreeBSD 如何参照及分割硬盘
- 如何运行 `sysinstall`.
- 在运行 `sysinstall` 将要回答的问题、问题代表什么意思，以及如何回答它。

在阅读本文前，请：

- 请你要安装的 FreeBSD 版本所附的硬件支持列表以确定你的硬件有没有被支持。



一般来说，此安装说明是针对 i386™ ("PC 兼容机") 体系结构的。如果有其它体系结构的安装说明，我将一并列出。然本文通常保持更新，但有可能与安装版本上所附的说明有些出入。在这里建议使用本说明文章作一般性的安装指南参考手册。

## 2.2. 硬件需求

### 2.2.1. 最小配置

安装 FreeBSD 所需的最小硬件配置，随 FreeBSD 版本和硬件架构不同而有所不同。

在接下来的几节中，给出了些信息的一些。随安装 FreeBSD 的方式不同，可能需要使用或 FreeBSD 支持的 CDROM 驱动器，有时候也可能需要的是一网络。将在[指引介绍](#)中进行介绍。

#### 2.2.1.1. FreeBSD/i386 和 FreeBSD/pc98

FreeBSD/i386 和 FreeBSD/pc98 版本，都需要 486 或更高的处理器，以及至少 24 MB 的 RAM。需要至少 150 MB 的空硬盘空间，才能完成最小的安装配置。



对于老旧的硬件而言，多数时候，装配更多的 RAM 和出更多的硬盘空间，要比使用更快的处理器更有用。

#### 2.2.1.2. FreeBSD/amd64

有处理器同时能支持运行 FreeBSD/amd64。第一是 AMD64 处理器，包括 AMD Athlon™64、AMD Athlon™64-FX、AMD Opteron™ 以及更高性能的处理器。

能使用 FreeBSD/amd64 的唯一处理器是包含了采用 Intel® EM64T 架构支持的处理器。这些处理器包括 Intel® Core™ 2 Duo、Quad、以及 Extreme 系列处理器，以及 Intel® Xeon™ 3000、5000、和 7000 系列



处理器。

如果您的计算机使用 nVidia nForce3 Pro-150, 您必须使用 BIOS 配置, 禁用 IO APIC。如果您没有看到 BIOS 的选项, 可能就只能禁用 ACPI 了。Pro-150 芯片存在一个 bug, 目前我们还没有看到任何解决方法。

### 2.2.1.3. FreeBSD/sparc64

要安装 FreeBSD/sparc64, 必须使用它支持的平台 (参看 [支持的硬件](#))。

FreeBSD/sparc64 需要独占一个磁盘。目前我们没有办法与其它操作系统共享一个磁盘。

## 2.2.2. 支持的硬件

支持的硬件列表, 会在 FreeBSD 发行版本的 FreeBSD 兼容硬件说明提供。该文档通常可以在 CDROM 或 FTP 安装文件的目录中找到, 它的名字是 HARDWARE.TXT, 此外, 在 sysinstall 的 documentation 菜单也可以找到。它列出了特定的硬件架构列出了 FreeBSD 已知支持的硬件。不同发行版本和架构上的硬件支持列表, 可以在 FreeBSD 网站的 [发行版信息](#) 页面上找到。

## 2.3. 安装前的准备工作

### 2.3.1. 列出要安装的硬件清单

在安装 FreeBSD 之前, 您需要将系统中的硬件清单列出来。FreeBSD 安装程序会将一些硬件(磁盘、网卡、光驱等等)以及型号及制造厂商列出来。FreeBSD 也会列出一些最恰当的 IRQ 及 IO 端口的设定。但是因 PC 的硬件可能在太复杂, 这个不一定能成功。所以, 您可能需要手动更改有问题的设定。

如果您已安装了其它的操作系统, 如 Windows® 或 Linux, 那您可以先由这些系统所提供的工具来看它的设定是如何分配的。如果您真的没办法设定某些接口用什麼设定, 那您可以看看, 说不定它的设定已显示在 BIOS 上。常用的 IRQ 号有 3、5 以及 7; IO 端口的号通常以 16 进制表示, 例如 0x330。

我建议您在安装 FreeBSD 之前把一些信息打印或记录下来, 做成表格的样子也会比有帮助, 例如:

表 1. 硬件清单

| 设备名         | IRQ | IO 端口号 | 备注                              |
|-------------|-----|--------|---------------------------------|
| 第一块硬盘       | N/A | N/A    | 40 GB, Seagate 制造, 第一个 IDE 接口主盘 |
| CDROM       | N/A | N/A    | 第一个 IDE 接口从盘                    |
| 第二块硬盘       | N/A | N/A    | 20 GB, IBM 制造, 第二个 IDE 接口主盘     |
| 第一个 IDE 控制器 | 14  | 0x1f0  |                                 |
| 网卡          | N/A | N/A    | Intel® 10/100                   |
| Modem       | N/A | N/A    | 3Com® 56K faxmodem, 位于 COM1 口   |

在清楚地了解了计算机的配置之后，需要知道它是否符合希望安装的 FreeBSD 版本的硬件需求。

### 2.3.2. 磁盘的数据

如果磁盘上面存有重要的数据资料，那在安装 FreeBSD 前肯定已将某些资料备份了，并且先备份某些文档是否有误。FreeBSD 安装程序在要写入任何资料到磁盘前都会先提醒用户，一旦确定要写入，那以后就没有反悔的机会。

### 2.3.3. 决定要将 FreeBSD 安装到哪里

如果想让 FreeBSD 使用整个硬盘，那可以直接跳到下一节。

但是，如果想让 FreeBSD 跟已有的系统并存，那必须对数据存在硬盘的分布方式有深入的了解，以及其所造成的影响。

#### 2.3.3.1. FreeBSD/i386 体系结构的硬盘分配方式

一个 PC 硬盘可以被分成很多块。这些块被称为 *partitions* (分区)。由于 FreeBSD 内部也有分区概念，如此命名很容易致混淆，因此我在 FreeBSD 中，将其称为磁片 *slice*，或称 *slices*。例如，FreeBSD 提供的用于操作 PC 磁片分区的工具 *fdisk* 就将其称为 *slice* 而不是 *partition*。由于这个原因，一个硬盘支持四个分区；这些分区叫做主分区(*Primary partition*)。为了突破这个限制以便使用更多的分区，就有了新的分区类型，叫做扩展分区(*Extended partition*)。一个硬盘可以有一个扩展分区。在扩展分区里可以建立多个逻辑分区(*Logical partitions*)。

每个分区都有其独立的分区号(*partition ID*)，用以区分每个分区的数据类型。FreeBSD 分区的分区号从 165。

一般而言，操作系统都会有自己独特的方式来区分分区。例如 DOS 及其之后的 Windows®，会分配每个主分区及每个分区一个驱动器字符，从 C: 开始。

FreeBSD 必须安装在主分区。FreeBSD 可以在一个分区上面存放系统数据或是建立的任何文件。然而，如果一个有多个硬盘，也可以在某些硬盘上(全部或部分)建立 FreeBSD 分区。在安装 FreeBSD 的时候，必须有一个分区可以供 FreeBSD 使用。这个分区可以是尚未使用的分区，或是已存在且存有数据但不再需要的分区。

如果已经用完了硬盘上的所有分区，那必须使用其它操作系统所提供的工具(如 DOS 或 Windows® 下的 *fdisk*)来划出一个分区供 FreeBSD 使用。

如果磁盘的某个分区有多余的空间，可以使用它。但是使用前需要先整理一下这些分区。

FreeBSD 最小安装需要 100 MB 的空间，但是这是非常基本的安装，几乎没有剩下多少空间可以建立自己的文件。一个理想的最小安装是 250 MB，不含图形界面；或是 350 MB 以上，包含图形界面。如果还需要安装其它的第三方厂商的套件，那将需要更多的硬盘空间。

可以使用类似 PartitionMagic® 的商业版本工具，或类似 GParted 的自由软件工具来调整分区尺寸，从而为 FreeBSD 划出空间。PartitionMagic® 和 GParted 都能改变 NTFS 分区的尺寸。GParted 在许多 Live CD Linux 发行版，如 [SystemRescueCD](#) 中均有提供。

目前已有警告显示改变 Microsoft® Vista 分区尺寸会出现问题。在进行此操作时，建议准备一张 Vista 安装 CDROM。如同其他的磁碟操作一样，强烈建议事先备份。



不当的使用某些工具可能会干掉硬盘上的数据资料！ 在使用某些工具前确定有最近的、没问题的数据。

#### 例 1. 使用已存在的分区

假设只有一个 4GB 的硬盘，而且已装了 Windows®。然后将这个硬盘分成两个分区 C: 跟 D:，每个分区大小 2 GB。在 C: 分区上存放有 1 GB 的数据、D: 分区上存放 0.5 GB 的数据。

这意味着硬盘上有两个分区，一个驱动器符号是一个分区（如 c:、d:）。 可以把所有存放在 D: 分区上的数据复制到 C: 分区，这样就空出了一个分区(d:)给 FreeBSD 使用。

#### 例 2. 删除已存在的分区

假设只有一个 4 GB 的硬盘，而且已装了 Windows®。在安装 Windows® 的时候把 4 GB 都给了 C: 分区，并且已使用了 1.5 GB 的空间。想将剩余空间中的 2 GB 给 FreeBSD 使用。

为了安装 FreeBSD，必须从下面两种方式中选择一种：

1. 删除 Windows® 的数据资料，然后重新安装 Windows®，并让 Windows® 分配 2 GB 的空间。
2. 使用上面提及的 PartitionMagic® 来整理或切割硬盘的分区。

### 2.3.4. 收集网络的网口配置相关资料

如果想通过网口(FTP 或是 NFS)安装 FreeBSD， 那就必须知道网络的网口配置信息。在安装 FreeBSD 的过程中将会提示输入这些信息，以顺利完成安装过程。

#### 2.3.4.1. 使用以太网或调制解调器/DSL Modem

如果通过局域网或是要通过网口使用调制解调器/DSL 上网， 那就必须准备下面的信息：

1. IP 地址。
2. 默认网口 IP 地址。
3. 主机名称。
4. DNS 服务器的 IP 地址。
5. 子网掩码。

如果不知道这些信息， 可以联系系统管理员或是网络的网口服务提供商。 他可能会说这些信息会由 DHCP 自动分配；如果有的话，记住一点就可以了。

#### 2.3.4.2. 使用 Modem 连接

如果由 ISP 提供的号码拨号上网， 仍然可以通过它安装 FreeBSD， 只是会需要很久的时间。

必须知道：

1. 号码到 ISP 的拨号。

2. 的 modem 是 接到 个 COM 端口。
3. 号到 ISP 所用的 号和密 。

### 2.3.5. 行勘

然我 尽力 保 个 FreeBSD 行版本的 定性， 但偶 也会有一些 入 行版。 少数情况下， 些 甚至可能会影 安装。 当 和修正 之后， 它 会列在 FreeBSD 网站中的 [FreeBSD 行勘](#) 中。 在 安装之前， 首先看一看 勘 表， 以了解可能存在的 。

于所有 出版本的信息， 包括勘 表， 可以在 [FreeBSD 网站](#) 的 [行版信息](#) 一 中 到。

### 2.3.6. 准 安装介

FreeBSD 可以通 下面任何一 安装介 行安装：

安装介

- CDROM 或 DVD
- USB 棒
- 在同一 计算机上的 DOS 分区
- SCSI 或 QIC 磁
- 网

网

- 通 防火 的一个 FTP 站点， 或使用 HTTP 代理。
- NFS 服 器
- 一个指定的并行或串行接口

如果 了 FreeBSD 的 CD 或 DVD， 那 可以直接 入下一 ( [准引介](#) )。

如果 没有 FreeBSD 的安装文件， 按照 [准 自己的安装介](#) 来准 。 完那 之后， 就可以回到 并从 [准引介](#) 了。

### 2.3.7. 准引介

FreeBSD 的安装程 始于将 的 机 入 FreeBSD 安装境 - 并非在其它的操作系 上 行一个程序。 计算机通常使用安装在硬 上的操作系 行引 ， 也可以配置成使用一 "bootable(可引)"的 行 。 大多数代 计算机也都可以从光 或 USB 来引系 。



如果有 FreeBSD 的安装光 或 DVD(或者是 的， 或者是 自己准 的。)并且 的 计算机可以从光 行 (通常在 BIOS 中会有 "Boot Order" 或 似的 可以置)， 那 就可以跳 此小 。 因 FreeBSD 光 及 DVD 光 都是可以引 的， 用它 机 不用做什 特 的准 。

要 建引系 所需的 棒， 需按下面的操作 行：

## 1. 获取USB棒映像文件

USB棒映像文件可以从 `arch` 目录的 `ISO-IMAGES` 目录，例如 <ftp://ftp.FreeBSD.org/pub/FreeBSD/releases/arch/ISO-IMAGES/version/FreeBSD-12.0-RELEASE-arch-memstick.img> 获得。其中，`arch` 和 `version` 需要替换使用的平台和版本。例如，FreeBSD/i386 12.0-RELEASE 的 USB棒映像位于 <ftp://ftp.FreeBSD.org/pub/FreeBSD/releases/i386/ISO-IMAGES/12.0/FreeBSD-12.0-RELEASE-i386-memstick.img>。

USB棒的映像文件扩展名是 `.img`。在 `ISO-IMAGES/` 目录中提供了多个不同的映像，您可以根据使用的 FreeBSD 版本，有时也包括硬件来选择合适的映像。



在安装之前，必须清除 USB 棒上目前保存在 USB 棒上的数据，接下来的操作将会擦除这些数据。

## 2. 准备USB棒



下面的例子中，USB棒的主设备名是 `/dev/da0`。小心地操作是希望覆盖的，否则可能会损坏您的数据。

设置 `kern.geom.debugflags` sysctl 允许写入主引导记录。

```
# sysctl kern.geom.debugflags=16
```

## 3. 将映像文件写入USB棒

`.img` 文件不是直接复制到USB棒中的那个普通文件。这个映像是一个包含全部内容的映像。这意味着从一个地方复制到另一个地方是不能给予其引导能力的。必须使用 `dd(1)` 将映像文件直接写入磁盘：

```
# dd if=FreeBSD-12.0-RELEASE-i386-memstick.img of=/dev/da0 bs=64k
```

一般来说，要建立安装盘(USB)依照下列步骤：

## 1. 取回映像文件



注意，从 FreeBSD 8.0 开始，我们不再提供映像了。请参考前面关于如何使用 USB 闪存棒，或 CDROM 和 DVD 来安装 FreeBSD 的介绍。

映像文件可以在我们的安装介绍的 `floppies/` 目录下找到，另外也可以从下述网站的 `floppies` 目录下：<ftp://ftp.FreeBSD.org/pub/FreeBSD/releases/架构名/版本-RELEASE/floppies/>。将 <架构名> 和 <版本> 替换为使用的计算机体系结构和希望安装的版本号。例如，用于安装 i386™ 上的 FreeBSD/i386 11.2-RELEASE 的文件的地址，是 <ftp://ftp.FreeBSD.org/pub/FreeBSD/releases/i386/11.2-RELEASE/floppies/>。

映像文件的扩展名是 `.flp`。在 `floppies/` 目录中包括了多个不同的映像文件，随安装的 FreeBSD 版本，某些时候也随硬件的不同，需要使用的映像文件可能会有所不同。通常会需要四个，即 `boot.flp`、`kern1.flp`、`kern2.flp`，以及 `kern3.flp`。同一目录下的 `README.TXT` 文件以了解关于这些映像文件的最新信息。



我们的 FTP 程序必须使用二进制模式来下载这些映像文件。有些设备只会用 `text`（或 `_ASCII`）模式来传输数据，用这些设备下载的映像文件做成的磁盘将无法正常工作。

## 2. 准备磁盘

必须下载的一个映像文件准备一个磁盘。并且避免使用到坏掉的磁盘。最好的方式就是先将磁盘格式化，不要相信所谓的已格式化的磁盘。在 Windows® 下的格式化程序不会告诉你出多少坏块，它只是告诉它“bad”并且忽略它。根据建议使用全新的磁盘来存放安装程序。



如果在安装 FreeBSD 的过程中造成当机、或是其它怪现象，第一个要怀疑的就是引导盘。用其它的磁盘制作映像文件再试试。

## 3. 将映像文件写入磁盘中

`.flp` 文件并非一般的文件，不能直接将它复制到磁盘上。事实上它是一个包含完整磁盘内容的映像文件。表示不能直接使用 DOS 的 `copy` 命令将文件写到磁盘上，而必须使用特殊的工具程序将映像文件直接写到磁盘中。

如果使用 MS-DOS® 或 Windows® 操作系统来制作引导盘，那可以使用我们提供的 `fdimage` 程序来将映像文件写到磁盘中。

如果使用的是光盘，假如光盘的设备符号是 `E:`，那进行下面的命令：

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

重复上述命令以完成一个 `.flp` 文件的写入，每个映像文件都必须更改；制作好的磁盘注明是使用哪个映像文件做的。如果映像文件存放在不同的地方，自行修改上面的指令指向存放 `.flp` 文件的地方。要是没有 FreeBSD 光盘，可以到 FreeBSD 的 FTP 站点 [tools](#) 目录中下载。

如果在 UNIX® 系统上制作（例如其它 FreeBSD 机器），可以使用 `dd(1)` 命令来将映像文件写到磁盘中。如果用 FreeBSD，可以进行下面的命令：

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

在 FreeBSD 中, /dev/fd0 指的是第一个软盘(即 A: 驱动器); /dev/fd1 是 B: 驱动器,依此类推。其它的 UNIX® 系统可能会用不同的名称, 用户就要查阅该系统的说明文件。

现在可以安装 FreeBSD 了

## 2.4. 开始安装

默认情况下, 安装程序并不会改变任何硬盘中的数据, 除非用户看到下面的消息:



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

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

We can take no responsibility for lost disk contents!

在看到最后的警告消息前用户都可以随时退出, 安装程序界面不会更改硬盘。如果用户有任何决定, 用户可以直接将电源关掉而不会造成任何损害。

### 2.4.1. 机器

#### 2.4.1.1. 引导 i386™ 系统

1. 从尚未开机开始起
2. 将电源打开。开始的时候它会显示入系设置菜单或 BIOS 要按哪个，常见的是 **F2**、**F10**、**Del** 或 **Alt + S**。不是要按哪个，按它进入 BIOS 设置画面。有些的计算机可能会显示一个图形画面，典型的做法是按 **Esc** 将掉那个图形画面，以便能看到必要的设置信息。
3. 到设置开机顺序的，它的 "Boot Order" 通常会列出一些选项，例如：**Floppy**、**CDROM**、**First Hard Disk** 等等。

如果要用光盘安装，选 **CDROM**。如果使用 USB 盘，或者用来引导系统，也类似地选了正确的引导项。如有疑问，请参考你的主板说明手册。

保存设定并退出，系统会重新启动。

4. 如果你根据 [准引导介绍](#) 制作了 "可引导" 的 USB 闪存棒，在开机前将其插到计算机上。

如果是从光盘安装，那开机后立即将 FreeBSD 光盘放入光盘中。



对于 FreeBSD 7.3 和更早的版本，可以使用引导盘，有些可以根据 [准引导介绍](#) 来制作。其中，`boot.flp` 是引导盘。引导系统使用引导盘。

如果开机后计算机引导了先前已装好的其他操作系统，那么：.. 是不是光盘或光驱放入而丢失引导项。如果是，将它放入后重新开机。.. BIOS 设定不对，重新设置 BIOS 的设定。.. 你的 BIOS 不支持从某些安装介绍引导。

5. FreeBSD 即将启动。如果是从光盘引导，你会看到似下面的画面：

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

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

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

FreeBSD/i386 bootstrap loader, Revision 1.1

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



如果从软盘，你会看到下面的画面：

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

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

FreeBSD/i386 bootstrap loader, Revision 1.1

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

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

根据提示将 boot.flp 取出，插入 kern1.flp，然后按 **Enter**。只需从第一盘开始，然后再根据需要插入其他盘就可以了。

6. 不是从光盘、USB 棒或引导盘，接下来都会进入 FreeBSD 引导加载器菜单：



1. FreeBSD Boot Loader Menu

可以等待十秒，或按 **Enter**。

### 2.4.1.2. 引导 sparc64

多数 sparc64 系统均配置有从硬盘引导。如果希望安装 FreeBSD，就需要从网络或 CDROM 引导了，需要首先输入 PROM (OpenFirmware)。

要完成这项工作，首先需要重启系统，并等待输出引导消息。具体的信息取决于使用的型号，不过它会是以下面：

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

如果系统从此开始了从硬盘引导的过程，需要按下 **L1** + **A** 或 **Stop** + **A**，或者在串口控制台上发送 **BREAK** (例如，在 **tip(1)** 或 **cu(1)** 中是 **~#**) 以便输入 PROM 提示符。它是以下面：

```
ok ①
ok {0} ②
```

① 是在只有一 CPU 的系统上的提示。

② 是用于 SMP 系统的，里面的数字，是系统中可用的 CPU 数量。

然后，将 CDROM 插入驱动器，并在 PROM 提示符后面，输入 **boot cdrom**。

### 2.4.2. 查看引导探索的结果

前面屏幕显示的最后几百行字会存在缓冲区中以便查看。

要查看缓冲区，可以按下 **Scroll Lock**，它会启动画面的卷功能。然后就可以使用方向键或 **PageUp**、**PageDown** 来上下翻页。再按一次 **Scroll Lock** 将停止画面卷。

在查看的时候会看到类似典型的引导探索结果的画面。真正的结果依照使用的硬件装置而有所不同。

典型的引导探索结果

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

md1: Malloc disk
Using $PIR table, 4 entries at 0xc00fde60
npx0: <math processor> on motherboard
npx0: INT 16 interface
pcib0: <Host to PCI bridge> on motherboard
pci0: <PCI bus> on pcib0
pcib1:<VIA 82C598MVP (Apollo MVP3) PCI-PCI (AGP) bridge> at device 1.0 on pci0
pci1: <PCI bus> on pcib1
pci1: <Matrox MGA G200 AGP graphics accelerator> at 0.0 irq 11
```

```

isab0: <VIA 82C586 PCI-ISA bridge> at device 7.0 on pci0
isa0: <iSA bus> on isab0
atapci0: <VIA 82C586 ATA33 controller> port 0xe000-0xe00f at device 7.1 on pci0
ata0: at 0x1f0 irq 14 on atapci0
ata1: at 0x170 irq 15 on atapci0
uhci0 <VIA 83C572 USB controller> port 0xe400-0xe41f irq 10 at device 7.2 on pci0
usb0: <VIA 83572 USB controller> on uhci0
usb0: USB revision 1.0
uhub0: VIA UHCI root hub, class 9/0, rev 1.00/1.00, addr1
uhub0: 2 ports with 2 removable, self powered
pci0: <unknown card> (vendor=0x1106, dev=0x3040) at 7.3
dc0: <ADMtek AN985 10/100BaseTX> port 0xe800-0xe8ff mem 0xdb000000-0xeb0003ff irq 11 at device 8.0 on pci0
dc0: Ethernet address: 00:04:5a:74:6b:b5
miibus0: <MII bus> on dc0
ukphy0: <Generic IEEE 802.3u media interface> on miibus0
ukphy0: 10baseT, 10baseT-FDX, 100baseTX, 100baseTX-FDX, auto
ed0: <NE2000 PCI Ethernet (RealTek 8029)> port 0xec00-0xec1f irq 9 at device 10.0 on pci0
ed0 address 52:54:05:de:73:1b, type NE2000 (16 bit)
isa0: too many dependant configs (8)
isa0: unexpected small tag 14
orm0: <Option ROM> at iomem 0xc0000-0xc7fff on isa0
fdc0: <NEC 72065B or clone> at port 0x3f0-0x3f5,0x3f7 irq 6 drq2 on isa0
fdc0: FIFO enabled, 8 bytes threshold
fd0: <1440-KB 3.5" drive> on fdc0 drive 0
atkbdc0: <Keyboard controller (i8042)> at port 0x60,0x64 on isa0
atkbd0: <AT Keyboard> flags 0x1 irq1 on atkbdc0
kbd0 at atkbd0
psm0: <PS/2 Mouse> irq 12 on atkbdc0
psm0: model Generic PS/2 mouse, device ID 0
vga0: <Generic ISA VGA> at port 0x3c0-0x3df iomem 0xa0000-0xbffff on isa0
sc0: <System console> at flags 0x100 on isa0
sc0: VGA <16 virtual consoles, flags=0x300>
sio0 at port 0x3f8-0x3ff irq 4 flags 0x10 on isa0
sio0: type 16550A
sio1 at port 0x2f8-0x2ff irq 3 on isa0
sio1: type 16550A
ppc0: <Parallel port> at port 0x378-0x37f irq 7 on isa0
pppc0: SMC-like chipset (ECP/EPP/PS2/NIBBLE) in COMPATIBLE mode
ppc0: FIFO with 16/16/15 bytes threshold
plip0: <PLIP network interface> on ppbus0
ad0: 8063MB <IBM-DHEA-38451> [16383/16/63] at ata0-master UDMA33
acd0: CD-RW <LITE-ON LTR-1210B> at ata1-slave PIO4
Mounting root from ufs:/dev/md0c
/stand/sysinstall running as init on vty0

```

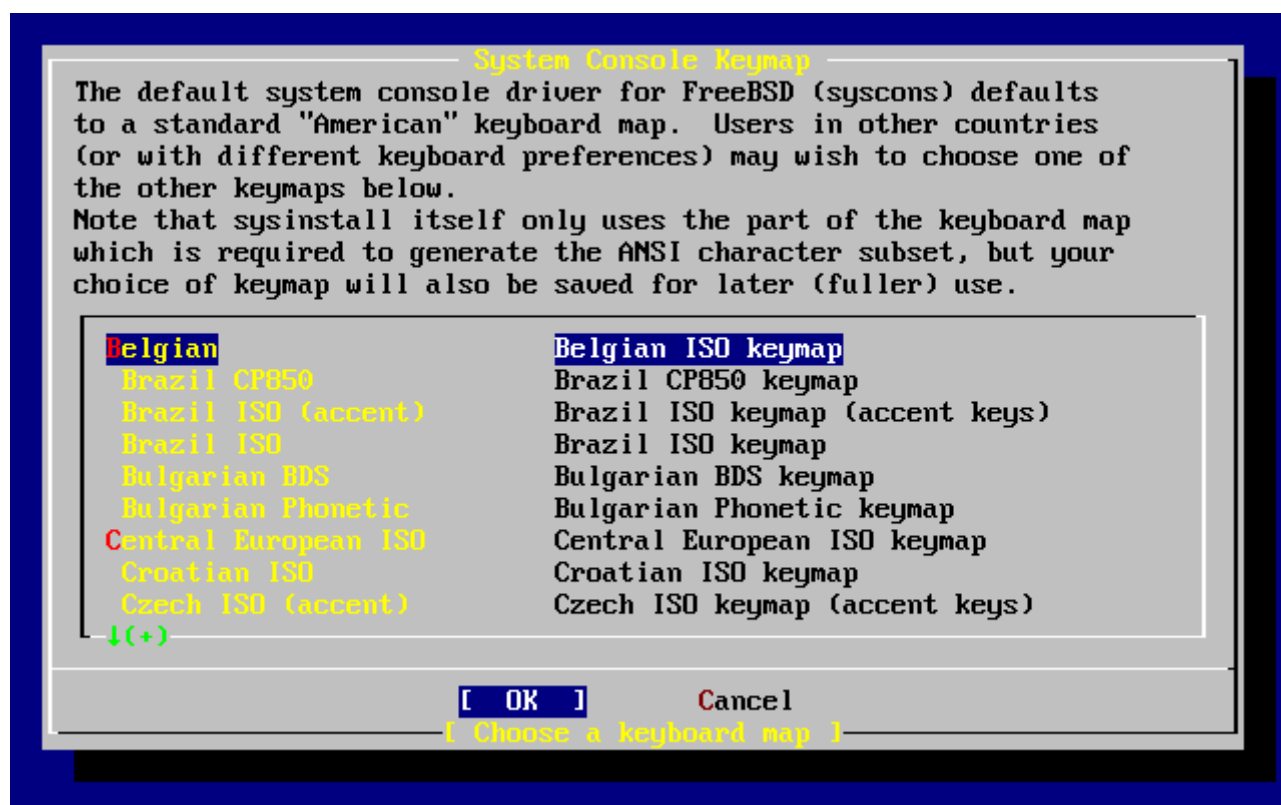
❑子❑❑❑探❑❑果以❑定 FreeBSD ❑到所有❑期望出❑的❑❑。 如果系❑没有❑到❑❑, ❑不会将其列出。 [定制内核](#)  
能❑❑❑❑系❑添加默❑的 GENERIC 内核所不支持的❑❑, 如声❑等。

在 FreeBSD 6.2 和更高版本中，在探索完系之后，将显示 国家及地区菜单。使用光标来国家或地区。接着按 **Enter**，系将自置地区。也可以很容易地退出 sysinstall 程序并从来。

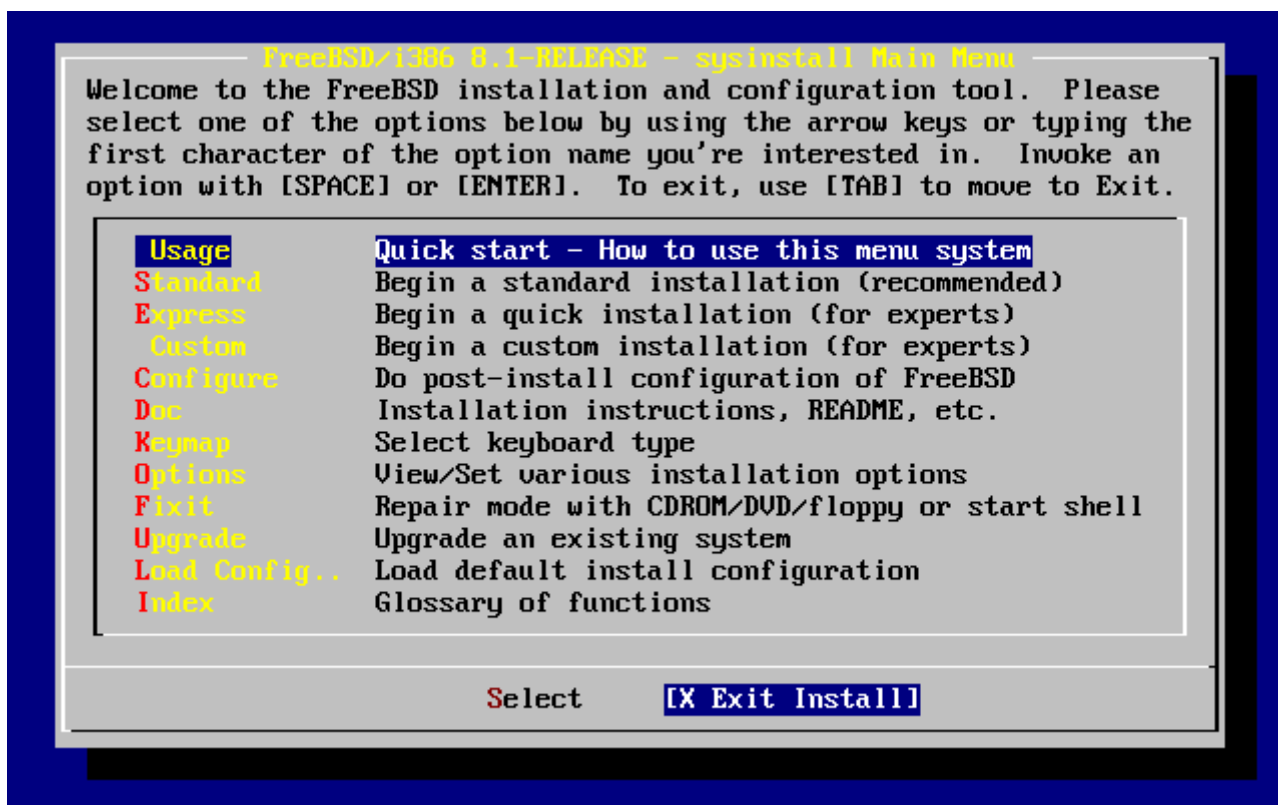


## 2. 国家及地区菜单

如果在国家及地区菜单中选了 United States (美国)，系会使用准的美国映射；如果选了不同的国家，会显示下面的菜单。使用光标修正的映射，然后按 **Enter** 来。

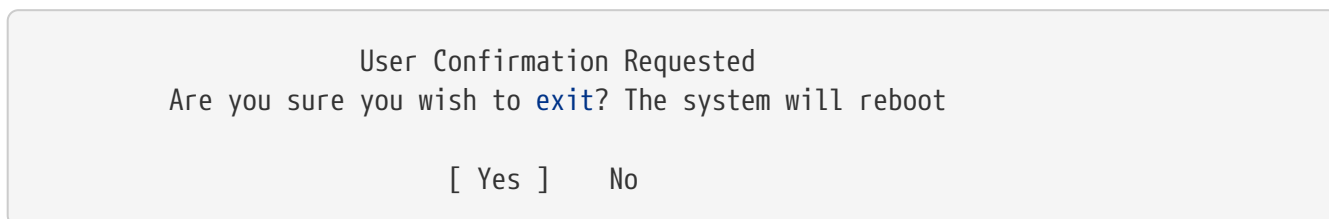


## 3. 键盘菜单



#### □ 4. □□□□ Sysinstall

在主界面使用方向□□□ Exit Install □会看到 如下的信息：



如果此□□□了 **[yes]** 但 CDROM □留在光□里，□会再次□入安装程序。

如果□是从□□□□，□在重□系□之前，需要将 boot.flp □□取出。

## 2.5. 介□ Sysinstall

sysinstall 是 FreeBSD □目所提供的安装程序。它以 console(控制台)□主， 分□多个菜□及画面□□配置及控制安装□程。

sysinstall 菜□画面由方向□、 **Enter**、 **Tab**、 **Space**， 以及其它按□所控制。在主画面的 Usage 菜□有□些按□的□明。

要□看□些□明，□将光□移到 Usage □目，然后 **[Select]** 按□被□□， □取 Sysinstall 主菜□的 Usage □目，然后按下 **Enter** □。

安装画面的使用□明会□示出来，□□完□□按 **Enter** □回到主画面。

```

FreeBSD/1306 8.1-RELEASE - sysinstall Main Menu
Welcome to the FreeBSD installation and configuration tool. Please
select one of the options below by using the arrow keys or typing the
first character of the option name you're interested in. Invoke an
option with [SPACE] or [ENTER]. To exit, use [TAB] to move to Exit.

```

|               |  |
|---------------|--|
| <b>Usage</b>  | Quick start - How to use this menu system        |
| Standard      | Begin a standard installation (recommended)      |
| Express       | Begin a quick installation (for experts)         |
| Custom        | Begin a custom installation (for experts)        |
| Configure     | Do post-install configuration of FreeBSD         |
| Doc           | Installation instructions, README, etc.          |
| Keymap        | Select keyboard type                             |
| Options       | View/Set various installation options            |
| Fixit         | Repair mode with CDROM/DVD/floppy or start shell |
| Upgrade       | Upgrade an existing system                       |
| Load Config.. | Load default install configuration               |
| Index         | Glossary of functions                            |

[ Select ]    X Exit Install

□ 5. □取 Sysinstall 主菜□的 Usage □目

## 2.5.1. □□ Documentation(□明文件) 菜□

用方向□从主菜□□□ Doc 条目然后按 **Enter** □。

```

FreeBSD/1306 8.1-RELEASE - sysinstall Main Menu
Welcome to the FreeBSD installation and configuration tool. Please
select one of the options below by using the arrow keys or typing the
first character of the option name you're interested in. Invoke an
option with [SPACE] or [ENTER]. To exit, use [TAB] to move to Exit.

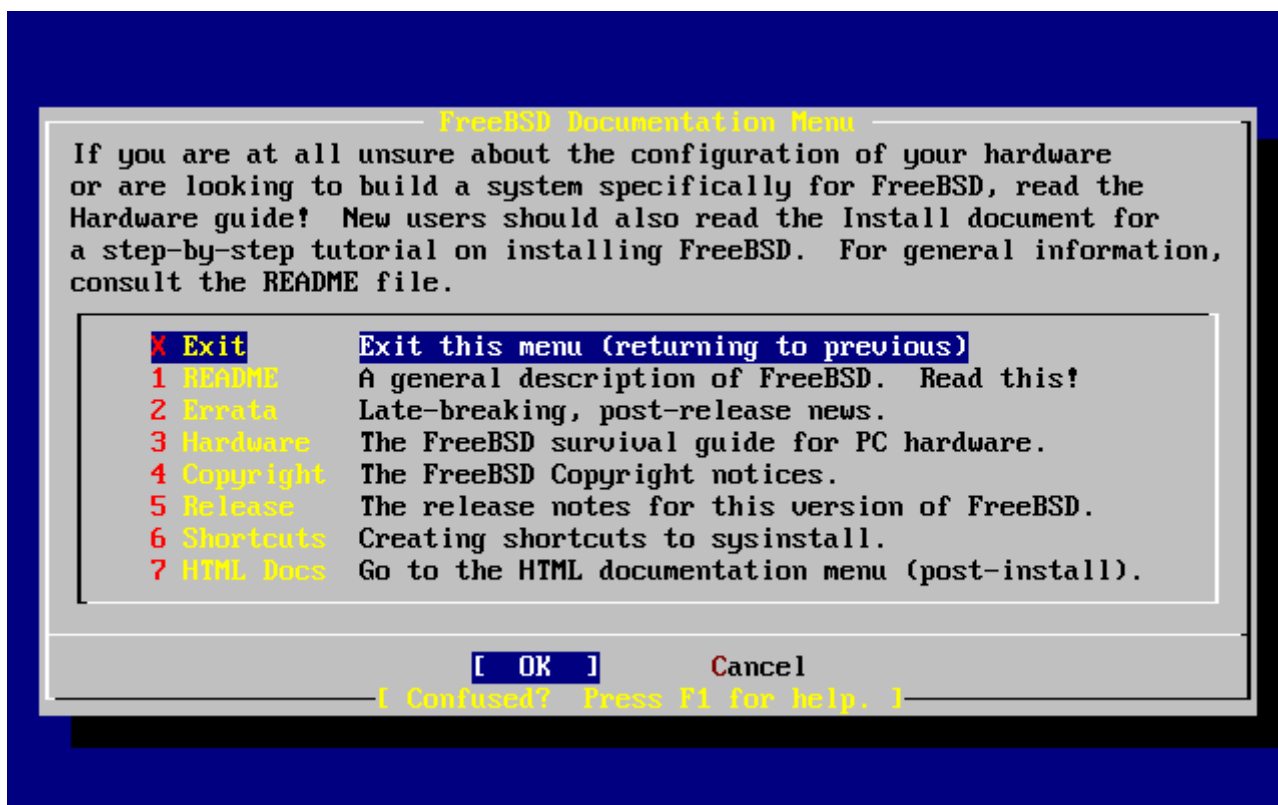
```

|               |  |
|---------------|--|
| Usage         | Quick start - How to use this menu system        |
| Standard      | Begin a standard installation (recommended)      |
| Express       | Begin a quick installation (for experts)         |
| Custom        | Begin a custom installation (for experts)        |
| Configure     | Do post-install configuration of FreeBSD         |
| <b>Doc</b>    | <b>Installation instructions, README, etc.</b>   |
| Keymap        | Select keyboard type                             |
| Options       | View/Set various installation options            |
| Fixit         | Repair mode with CDROM/DVD/floppy or start shell |
| Upgrade       | Upgrade an existing system                       |
| Load Config.. | Load default install configuration               |
| Index         | Glossary of functions                            |

[ Select ]    X Exit Install

□ 6. □□□明文件菜□

□将会□入□明文件菜□。



□ 7. Sysinstall □明文件菜□

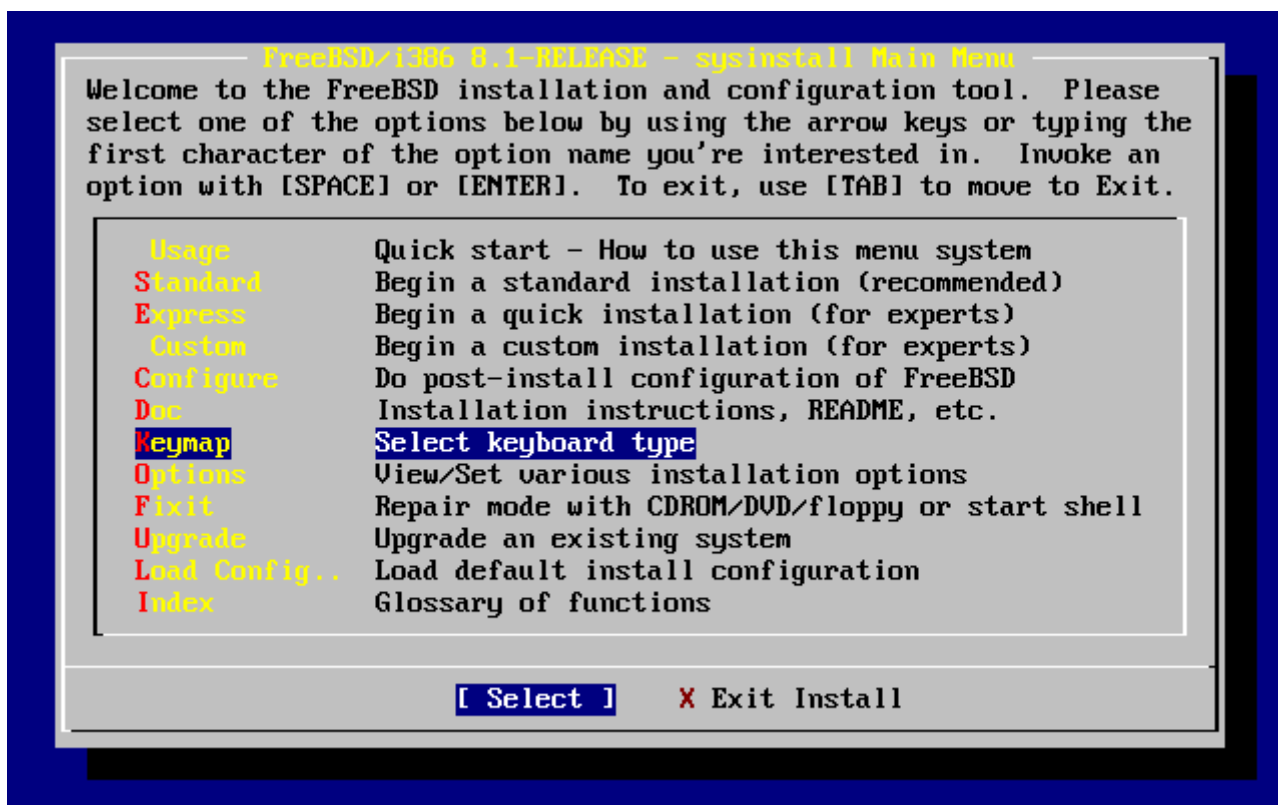
□□□些□明文件很重要。

要□□一篇文章，□用方向□□取要□□的文章然后按  □。□□中再按一下  就会回到□明文件画面。

若要回到主菜□，用方向□□□ Exit 然后按下  □。

## 2.5.2. □□□□□□(Keymap)菜□

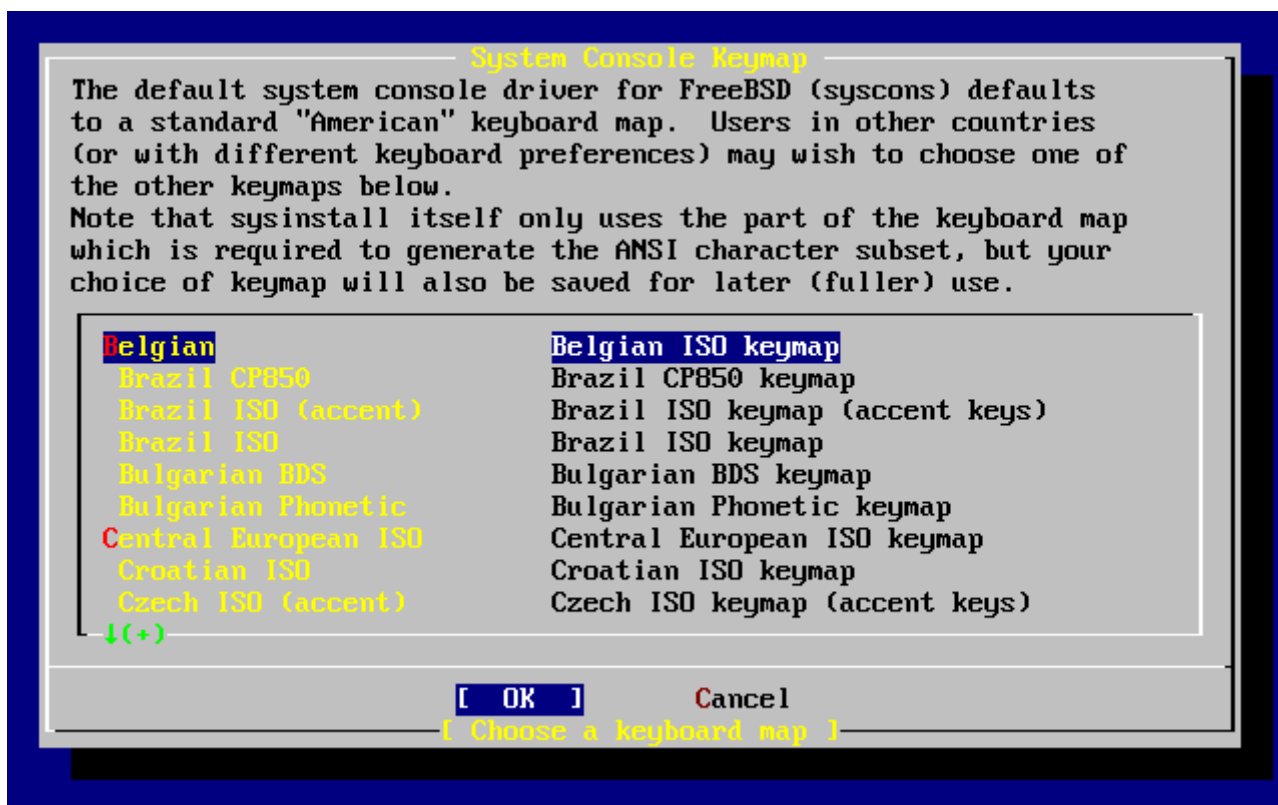
如果要改□□□按□的□□方式，□在主菜□□取 Keymap 然后按  □。一般情况下不改□此□，除非□使用了非□准□□或非美国□□。



## □ 8. Sysinstall 主菜□

□可以使用上下□移□到□想使用的□□□□方式， 然后按下  □以□取它；再按  □可以取消□取。当□完成后， □□□ [OK] 然后按  □。

□一屏幕只□示出部分列表。□□ [Cancel] 按  □将使用默□的□□□□， 并返回到主菜□



## □ 9. Sysinstall □□□□菜□



### 2.5.3. 安装配置画面

Options 然后按 `Enter` 。

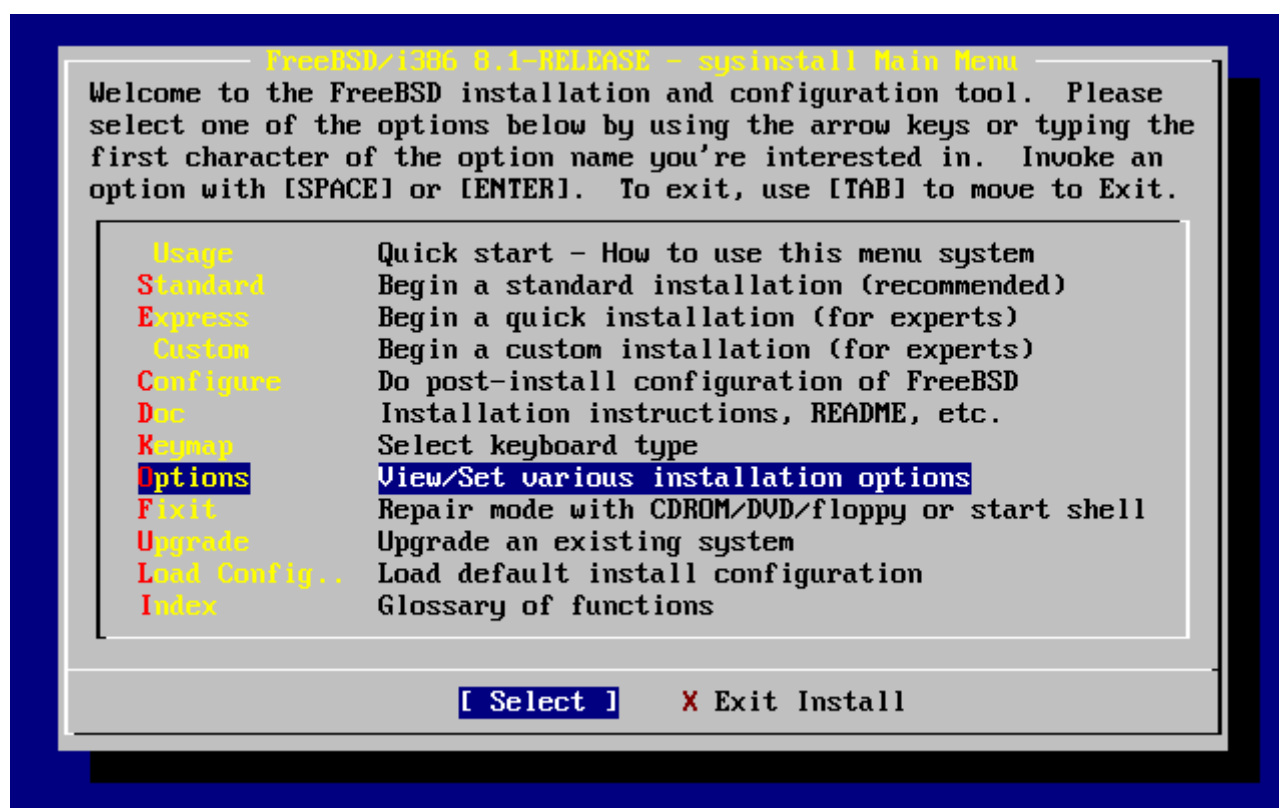


图 10. Sysinstall 主菜单

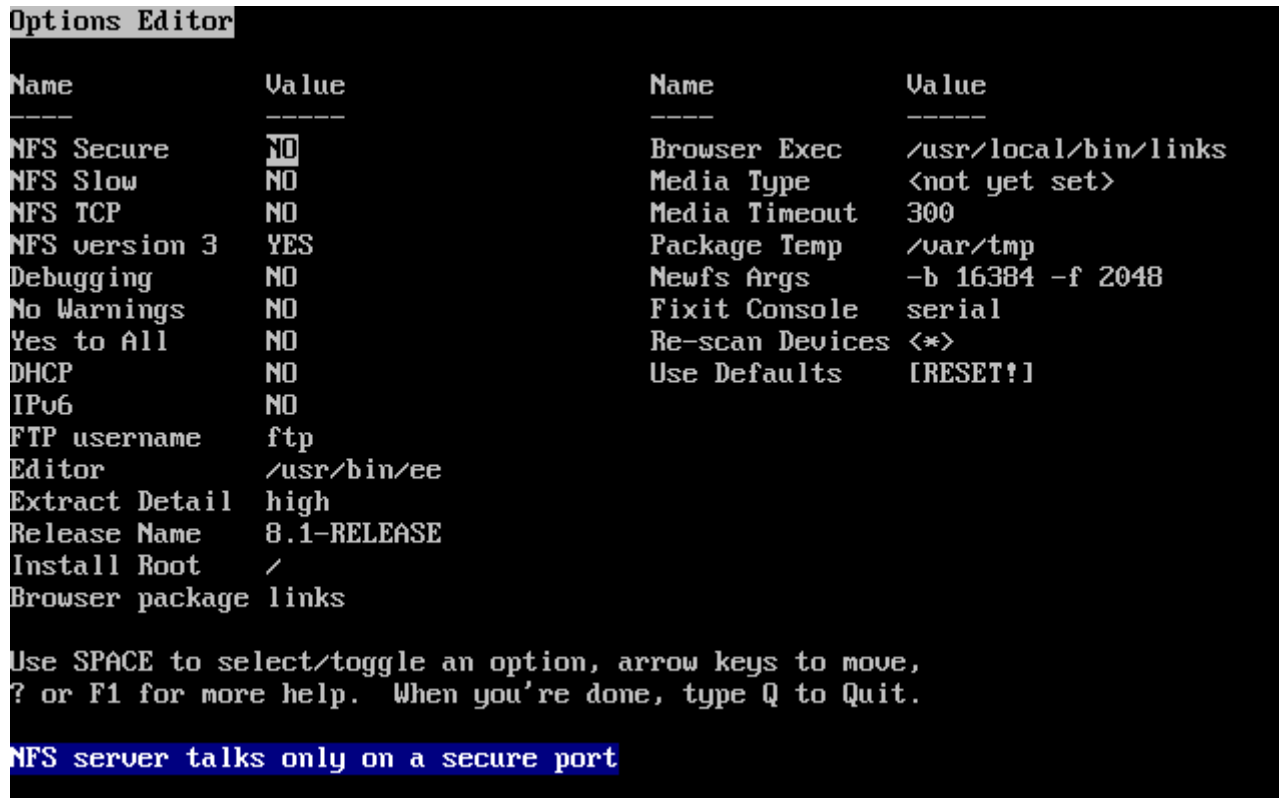


图 11. Sysinstall 配置

通常可以用于大部分的使用者，并不需要改它。版本名称要根据安装的版本行化。

目前每个项目的描述会在屏幕下方以黑底白字显示。 注意其中有一个项目是 Use Defaults(使用默认) 你可以由此将所有的设定还原。

可以按下 **F1** 来查看各项目的说明。

按 **Q** 可以回到主画面。

## 2.5.4. 开始行标准安装

Standard(标准) 安装用于那些 UNIX® 或 FreeBSD 的初级使用者。用方向键 Standard 然后按 **Enter** 可开始入标准安装。

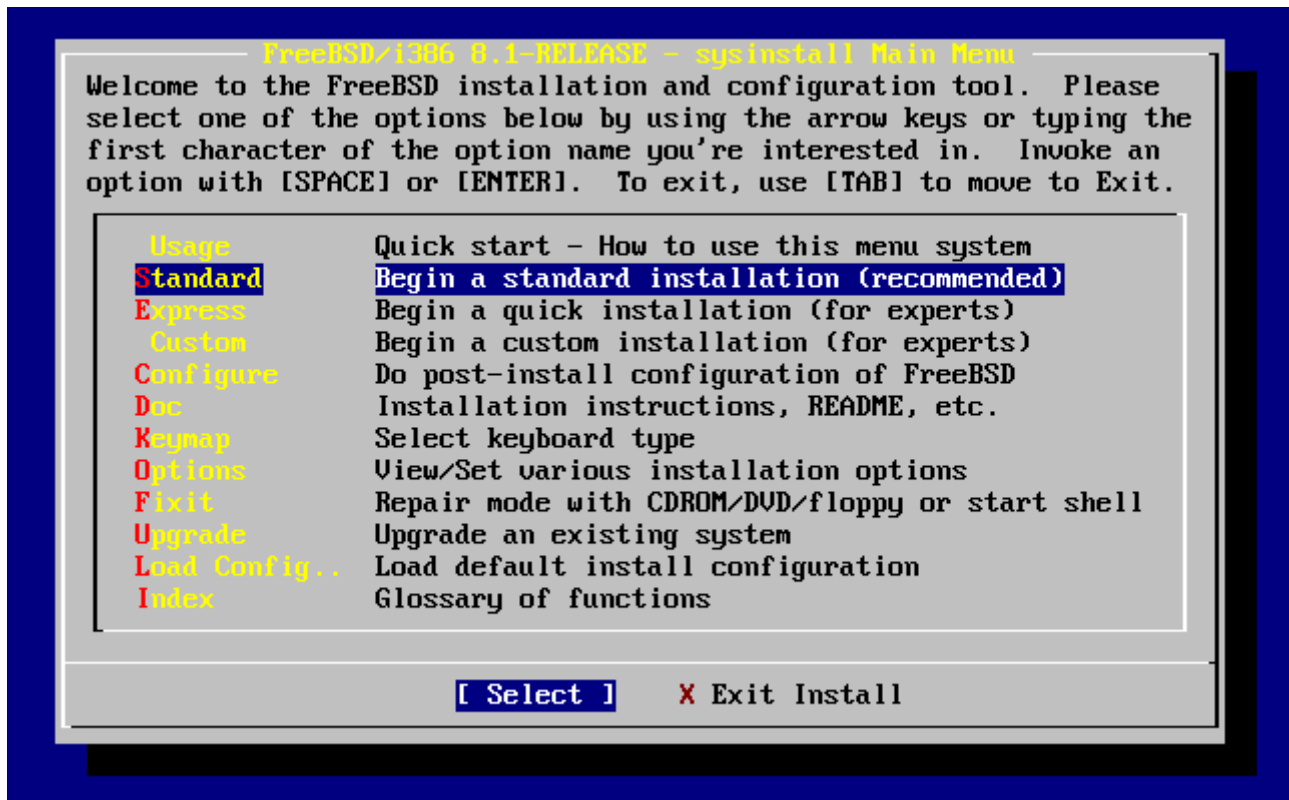


图 12. 开始行标准安装

## 2.6. 分配磁盘空间

它的第一工作就是要分配 FreeBSD 用的硬盘空间以便 sysinstall 先做好一些准备。为了完成它的工作，你必须先了解 FreeBSD 如何得到磁盘信息做一个了解。

### 2.6.1. BIOS 磁盘编号

当你在系统上安装配置 FreeBSD 之前，有一个重要的事情一定要注意，尤其是当有多个硬盘的时候。

在 PC 架构，当它像 MS-DOS® 或 Microsoft® Windows® 跟 BIOS 相关的操作系统的时候，BIOS 有能力改变正常的磁盘顺序，然后这些操作系统会跟着 BIOS 做改变。它使用者不一定非要有所谓的 "primary master" 硬盘。许多人最简单而便宜的方式就是再去买一模一样的硬盘，然后定期将数据从第一个硬盘复制到第二个硬盘，使用 Ghost 或 XCOPY。所以，当第一个硬盘死了，或者是被病毒破坏，或者有坏道，他可以调整 BIOS 中的盘机顺序而直接用第二硬盘。就像交换硬盘的数据，但是无需打机箱。

比如昂，配有 SCSI 控制器的系统通常可以延伸 BIOS 的功能来 SCSI 盘 (可七个) 到类似改变顺序的功能。

对于使用这种方式的使用者可能会感到困惑，因为在 FreeBSD 中并非如此。FreeBSD 不会参考 BIOS，而且也不知道所谓的“BIOS 磁头”是怎么回事。会让人感觉很疑惑，明明就是一块硬盘而且数据也完全从一块硬盘来的，结果却没法像以前那样用。

当使用 FreeBSD 以前，会将 BIOS 中的硬盘开机顺序回正常的顺序，并且以后不要再改。如果一定要交硬盘顺序，那用硬件的方式，打开机箱并调整。

Bill 替 Fred 把旧的 Wintel 的机器装上了 FreeBSD。他装了一台 SCSI 硬盘，ID 是 0，然后把 FreeBSD 装在上面。

Fred 开始使用他新的 FreeBSD 系统；但是过了几天，他旧的 SCSI 硬盘生了很多小问题。之后，他就跟 Bill 起这件事。

又过了几天，Bill 决定是时候解决了，所以他从后面房里的硬盘“收藏”中出了一个一模一样的硬盘，并且硬盘表面标示硬盘没有坏。因此，Bill 将它的 ID 改成 4，然后安装到 Fred 的机器，并且将数据从磁头 0 复制到磁头 4。在新硬盘装好了，而且看起来好像一切正常；所以，Bill 开始在可以开始用它了。Bill 于是到 SCSI BIOS 中设定 SCSI ID 4 开机，用磁头 4 重新开机后，一切都很顺利。

用了几天后，Bill 跟 Fred 决定要来玩点新的：将 FreeBSD 升级了。Bill 将 ID 0 的硬盘移除（因有坏道）并且又从收藏区中拿了一块一样的硬盘来。然后他用 Fred 神奇的网际 FTP 磁头将新版的 FreeBSD 安装在硬盘上；安装过程没什么问题。

Fred 用了新版本几天后，觉得它很合用在工程部... 是时候将以前放在旧系统的工作数据复制过来了。因此，Fred 将 ID4 的 SCSI 硬盘（里面有放着旧系统中复制来的最新数据）mount 起来，结果竟然在 ID4 的硬盘上，他以前的所有数据都不见了！

数据到哪里去了？

当初 Bill 将 ID0 硬盘的数据复制到 ID4 的时候，ID4 即成为一个“新的副本”。而当他 SCSI BIOS 设定 ID4 开机，想系统从 ID4 开机，其实只是他自己，因为大部分的系统可以直接 BIOS 而改开机顺序，但是 FreeBSD 却会把开机顺序原成正常的模式，因此，Fred 的 FreeBSD 是从原来那块 ID0 的硬盘开机的。所有的数据都在那块硬盘上，而不是在想象之中的 ID4 硬盘。

幸运的是，在我知道这件事的时候那些数据都在，我将那些数据从最早的那块 ID0 硬盘取出来并交给 Fred，而 Bill 也由此了解到计算机编号是从 0 开始的。

当然我这里的例子使用 SCSI 硬盘，但是相同的概念也可以套用在 IDE 硬盘上。

## 2.6.2. 使用 FDisk 建立分区



如果不再做改动，数据将会写入硬盘。如果你犯了一个错误想重新开始，可以按 `sysinstall` 安装程序的退出按钮(exit)。或按 `U` 键来 Undo 操作。如果操作没有效果，可以重新开机的计算机来达到目的。

当在 `sysinstall` 主菜单使用默认安装后，你会看到下面的信息：

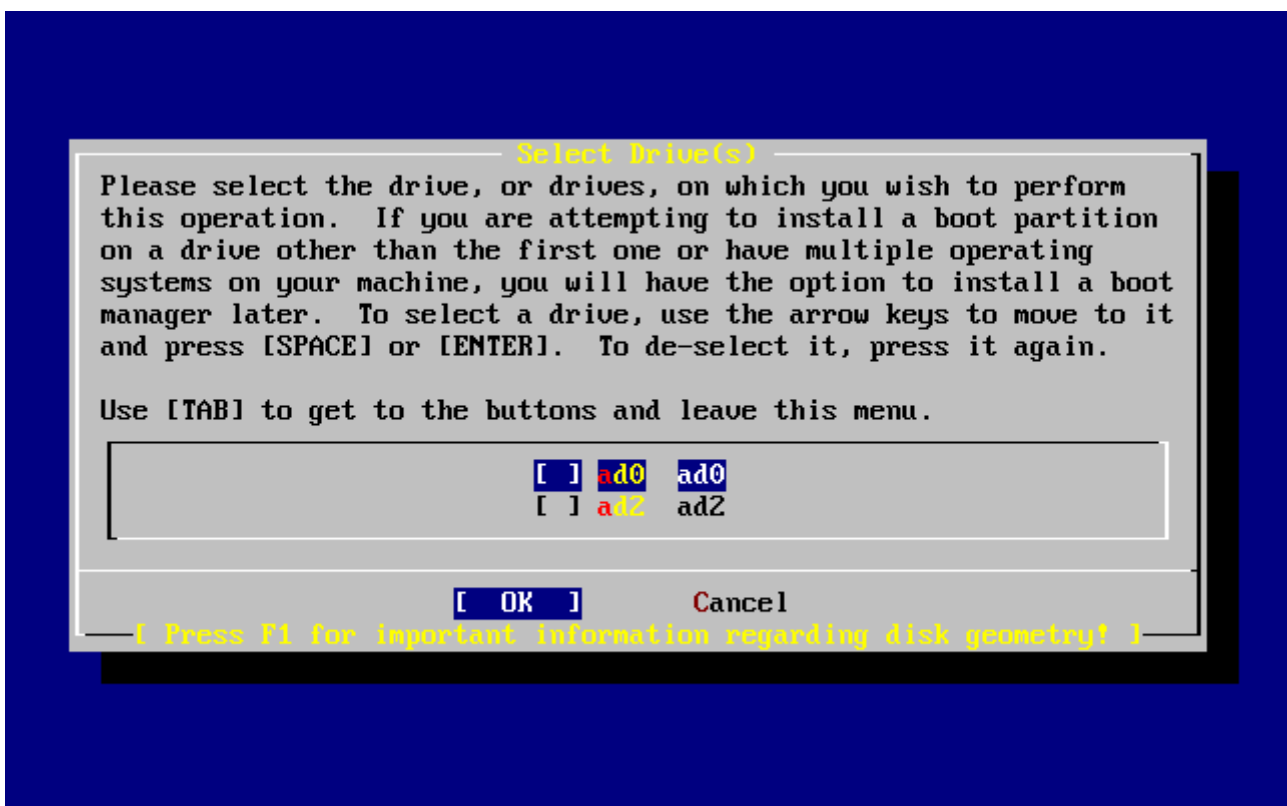
### Message

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

[ OK ]

[ Press enter or space ]

如屏幕指示, 按 **Enter** , 然后就会看到一个列表列出所有在探测到的硬盘。要分区的硬盘 示例的是有 2 个 IDE 硬盘的情形, 2 个硬盘分作 ad0 和 ad2。



13. 要分区的硬盘

可能正在奇怪, 为什么 ad1 没有列出来? 为什么失了?

想, 如果有 2 个 IDE 硬盘, 一个是在第一个 Primary master, 一个是 Secondary master, 会生什么事? 如果 FreeBSD 依照到的顺序来命名, 如 ad0 和 ad1 那就不会有什。

但是, 在来了。如果在想在 primary slave 加装第三个硬盘, 那这个硬盘的名称就会是 ad1, 之前的 ad1 就会成 ad2。会造成什? 因的名称 (如 ad1s1a) 是用来文件系的, 因此可能会, 突然, 有些文件系从此无法正确地示出来, 必须修改 FreeBSD 配置文件 (注: /etc/fstab) 才可以正示。

了解决些, 在配置内核的时候可以叫 FreeBSD 直接用 IDE 所在的位置来命名, 而不是依据到的顺序。使用方式的, 在 secondary master 的 IDE 就永远是 ad2, 即使的系中没有 ad0 或 ad1

也不受影。

此 FreeBSD 内核的默认，也是它上面的画面只示 ad0 和 ad2 的原因。画面上台机器的硬是装在 primary 及 secondary 的 master 上面；并没有任何一个硬安装在 slave 槽上。

想安装 FreeBSD 的硬，然后按下【OK】。之后 FDisk 就会始，会看到似 典型的尚未前的 Fdisk 分区表的画面。

FDisk 的示画面分三个部分。

第一部分是画面上最上面行，示的是目前所的硬的信息。包含它的 FreeBSD 名称、硬分布以及硬的容量。

第二部分示的是目前硬的硬上有些分区，个分区的始及束位置、所占容量、FreeBSD 名称、它的描述以及（sub-type）。此例示有个未使用的小分区，有一个大的 FAT 分区，（很可能是 MS-DOS® 或 Windows® 的 C: ），以及一个展分区（在 MS-DOS® 或 Windows® 里面可以包含分区）。

第三个部分示 FDisk 中可用的命令。

```

Disk name:      ad0
DISK Geometry:  16383 cyls/16 heads/63 sectors = 16514064 sectors (8063MB)

Offset      Size(ST)      End      Name  PType      Desc  Subtype  Flags
-----
0           63           62      -     6      unused     0
63      4193217      4193279      ad0s1  2         fat       14      >
4193280      1008      4194287      -     6      unused     0      >
4194288      12319776      16514063      ad0s2  4      extended    15      >

The following commands are supported (in upper or lower case):

A = Use Entire Disk      G = set Drive Geometry  C = Create Slice      F = `DD' mode
D = Delete Slice        Z = Toggle Size Units   S = Set Bootable      I = Wizard m.
T = Change Type         U = Undo All Changes    Q = Finish

Use F1 or ? to get more help, arrow keys to select.
```

图 14. 典型的尚未前的 Fdisk 分区表

接下来要做的事跟要分的硬分区有。

如果要 FreeBSD 使用整个硬（后要 sysinstall 安装后会除所有个硬上的料），那就可以按 **A**（Use Entire Disk）目前已有的分区都会被除，取而代之的是一个小的，示 unused 的分区，以及一个大的 FreeBSD 分区。之后，用方向将光标移到个 FreeBSD 分区，然后按 **S** 以将此分区分区。会看到似 Fdisk 分区使用整个硬 的画面。注意，在 Flags 中的 **A** 号表示此分区是激活的，因而将从此分区行。

要除有的分区以便 FreeBSD 出空，可以将光标移到要除的分区后按 **D**。然后就可按 **C**，并在

出的框中输入将要建的分区的大小。 输入合的大小后按 **Enter** 。 一般而言， 个框中的初始 是可以分配分区的大小。 它可能是最大的接分区或未分配的整个硬大小。

如果已建立好 FreeBSD 的分区 （使用像 PartitionMagic®似的工具）， 那可以按下 **C** 来建立一个新的分区。同的， 会有框要建立的分区的大小。

```
Disk name:      ad0      FDISK Partition Editor
DISK Geometry:  16383 cyls/16 heads/63 sectors = 16514064 sectors (8063MB)

Offset      Size(ST)      End      Name  PType      Desc  Subtype      Flags
-----
0           63           62      -      6      unused      0
63          16514001      16514063  ad0s1  3      freebsd     165      CA

The following commands are supported (in upper or lower case):

A = Use Entire Disk      G = set Drive Geometry      C = Create Slice      F = `DD' mode
D = Delete Slice         Z = Toggle Size Units       S = Set Bootable      I = Wizard m.
T = Change Type          U = Undo All Changes        Q = Finish

Use F1 or ? to get more help, arrow keys to select.
```

15. Fdisk 分区使用整个硬

完成后，按 **Q** 。 的更会存在 sysinstall 中， 但是不会真正写入的硬。

### 2.6.3. 安装多重引

在可以要不要安装一个多重引管理器。 一般而言， 如果到下列的情形， 要安装多重引管理程序。

- 有一个以上的硬， 并且 FreeBSD 并不是安装在第一个硬上。
- 除了 FreeBSD， 有其它的操作系安装在同一硬上， 所以需要在机的候要入一个系。

如果在台机器上只安装一个 FreeBSD 操作系， 并且安装在第一个硬， 那 Standard 安装就可以了。 如果已使用了一个第三方的多重引程序， 那 None。

好配置后按 **Enter**。

**Install Boot Manager for drive ad0?**

FreeBSD comes with a boot selector that allows you to easily select between FreeBSD and any other operating systems on your machine at boot time. If you have more than one drive and want to boot from the second one, the boot selector will also make it possible to do so (limitations in the PC BIOS usually prevent this otherwise). If you do not want a boot selector, or wish to replace an existing one, select "standard". If you would prefer your Master Boot Record to remain untouched then select "None".

**NOTE:** PC-DOS users will almost certainly require "None"!

|                 |  |
|-----------------|--|
| <b>BootMgr</b>  | Install the FreeBSD Boot Manager         |
| <b>Standard</b> | Install a standard MBR (no boot manager) |
| <b>None</b>     | Leave the Master Boot Record untouched   |

[ OK ]      Cancel

( Press F1 to read about drive setup )

#### □ 16. Sysinstall 多重引导管理程序

按下 **F1** 所显示的在说明中有的一些操作系统共存可能产生的。

#### 2.6.4. 在其它硬盘上建分区

如果你的系统上有一个以上的硬盘，在完成多重引导管理程序后会再回到硬盘的画面。如果你要将 FreeBSD 安装在多个硬盘上，那就可以在里其它的硬盘，然后重用 FDisk 来建立分区。



如果你想用 FreeBSD 来管理其它的硬盘，那每个硬盘都必须安装 FreeBSD 的多重引导管理程序。

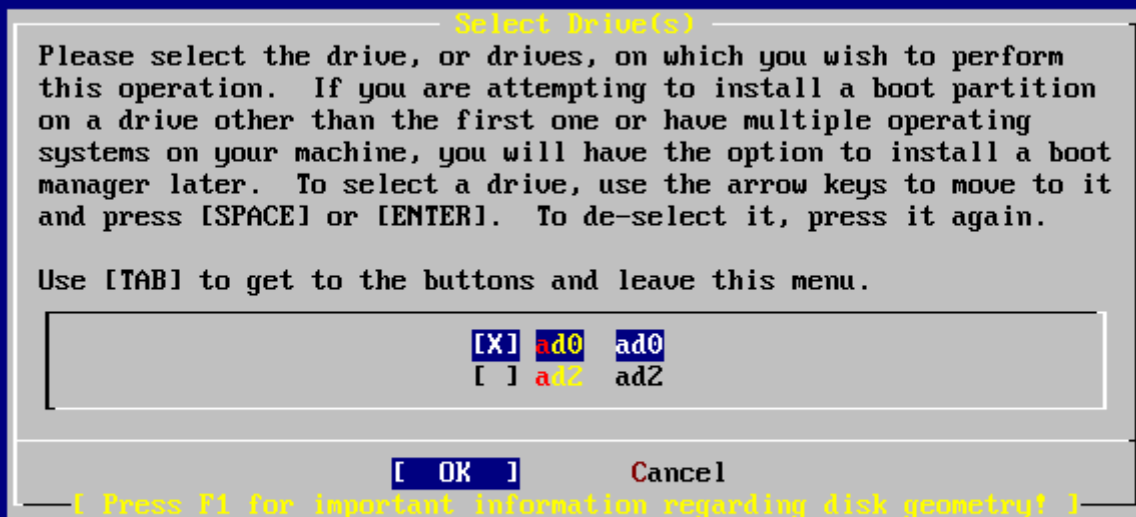


图 17. 选择硬盘画面

Tab 可以在最后列出的硬盘、[ OK ] 以及 [ Cancel ] 之间切换。

用 Tab 将光标移到 [ OK ] 然后按 Enter 开始安装程序。

## 2.6.5. 使用 bsdlabel 建立分区

必须在已经建立好的 slice 中创建一些 label。注意，每个 label 的代号是 a 到 h，另外，b、c 和 d 是有特殊用途的，不能随意使用。

某些应用程序可以利用一些特殊的分区而得到好的效果，尤其是分区分散在不同的硬盘的时候。但是，如果你是第一次安装 FreeBSD，所以不需要去考虑如何分割硬盘。最重要的是，装好 FreeBSD 然后学习如何使用它。当你熟悉 FreeBSD 有相当程度的熟悉后，可以随重新安装 FreeBSD，然后改变分区的方式。

下面的例子中有四个分区 - 一个是磁盘交换分区，另外三个是文件系统。

表 2. 第一个硬盘分区



| 分区 | 文件系统 | 大小        | 描述   |
|----|------|-----------|--|
| a  | /    | 1 GB      | <p>是一个根文件系统（root filesystem）。</p> <p>任何其它的文件系统都会挂在根目（注：用根目比确切）下面。1 GB于此目来说是合理的大小，因为往后并不会在此里存放太多的数据；在安装 FreeBSD 后会用掉 128 MB 的根目空。剩下的空是用来存放文件用的，同时，也保留一些空，因为以后的 FreeBSD 版本可能会需要多的 /（根目）空。</p>  |
| b  | N/A  | 2-3 x RAM | <p>b 分区是磁盘交换分区（swap space）。正确的交换空大小可是一学。一般来，交换空的大小是系统上内存（RAM）大小的2到3倍。交换空至少要有 64 MB。因此，如果系统上的 RAM 比 32 MB 小，将交换空大小 64 MB。</p> <p>如果有一个以上的硬盘，可以在每个硬盘上都配置交换分区。FreeBSD 会利用每个硬盘上的交换空，这样做能提高 swap 的性能。如果是情形，先算出共需要的交换空大小（如128 MB），然后除以有的硬盘数目（如2），算出的结果就是每个硬盘上要配置的交换空的大小。在这个例子中，每个硬盘的交换空 64 MB。</p> |

| 分区 | 文件系统 | 大小               | 描述   |
|----|------|------------------|--|
| e  | /var | 512 MB 至 4096 MB | /var 目录会存放不同大小的文件、日志以及其它管理用途的文件。大部分这些文件都是 FreeBSD 每天在运行的时候会读取或是写入的。当这些文件放在外的文件系统（注：即/var）可以避免影响到其它目录下面类似的文件存取机制。 |
| f  | /usr | 剩下的硬盘空间（至少 8 GB） | 所有的其它的文件通常都会存在/usr 目录以及其子目录下面。   |



上面例子中的数字限于有你的用户使用。通常我们鼓励用户使用 FreeBSD 分区器中一个叫做 **Auto Defaults** 的自动分区布局功能。

如果你要将 FreeBSD 安装在一个以上的硬盘，那你也必须在配置的其它分区上再建立分区。最简单的方式就是在每个硬盘上建立两个分区，一个是交换分区，一个是文件系统分区。

表 3. 其它磁盘分区

| 分区 | 文件系统     | 大小      | 描述  |
|----|----------|---------|---|
| b  | N/A      | 描述      | 之前提过，交换分区是可以跨硬盘的。但是，即使 a 分区没有使用，它上面是会把交换分区放在 b 分区上。   |
| e  | /disk_n_ | 剩下的硬盘空间 | 剩下的空间是一个大的分区，最简单的做法是将之作为一个 a 分区而不是 e 分区。然而，它上面 a 分区是保留根目录 (/) 用的。它不一定要遵守这个，但是 sysinstall 会，所以照着它做会使的安装比较清爽、干净。它可以将这些文件系统挂在任何地方，本例建议将它挂在 /diskn 目录，n 依据每个硬盘而有所不同，但是，喜欢的也可将它挂在别的地方。 |

分区的配置完成后，你可以用 sysinstall. 来建立它了。你会看到下面的信息：

### Message

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

[ OK ]  
[ Press enter or space ]

按下 **Enter** 开始FreeBSD分区表编辑器，称做 Disklabel。

**Sysinstall Disklabel 编辑器** 显示第一次运行 Disklabel 的画面。画面分三个区域。

前几行显示的是正在编辑的硬盘以及正在建立的 slice 位于哪个分区上。（在这里，Disklabel 使用的是 **分区名称** 而不是 slice 名）。此画面也会显示 slice 有多少空间可以使用；亦即，有多余的空间，但是尚未指派分区。

画面中区域显示已建立的分区，每个分区的文件系统名称、所占的大小以及一些用于建立这些文件系统的参数。

下方的第三区显示在 Disklabel 中可用的按钮。

```
FreeBSD Disklabel Editor
Disk: ad0      Partition name: ad0s1  Free: 16514001 blocks (8063MB)

Part      Mount      Size Newfs  Part      Mount      Size Newfs
-----
The following commands are valid here (upper or lower case):
C = Create      D = Delete    M = Mount pt.
N = Newfs Opts  Q = Finish    S = Toggle SoftUpdates  Z = Custom Newfs
T = Toggle Newfs U = Undo      A = Auto Defaults      R = Delete+Merge

Use F1 or ? to get more help, arrow keys to select.
```

### 18. Sysinstall Disklabel 编辑器

Disklabel 可以自行配置分区以及它的大小。有些默认的分区的尺寸是由内部的分区尺寸算法根据磁盘的大小算出的。可以按 **A** 使用此功能。会看到类似 **Sysinstall Disklabel 编辑器-使用自行配置** 的画面。根据硬盘的大小，自行分配所配置的大小不一定合适。但是没有关系，并不一定要使用它的大小。



默认情况下会 /tmp 目一个独立分区，而不是附属在 / 之下。 可以避免将一些 文件放到根目中（注：可能会用完根目空）。

```
FreeBSD Disklabel Editor

Disk: ad0      Partition name: ad0s1  Free: 0 blocks (0MB)

Part      Mount      Size Newfs  Part      Mount      Size Newfs
-----
ad0s1a    /              422MB UFS2  Y
ad0s1b    swap          321MB SWAP
ad0s1d    /var          710MB UFS2+S Y
ad0s1e    /tmp          377MB UFS2+S Y
ad0s1f    /usr          6232MB UFS2+S Y

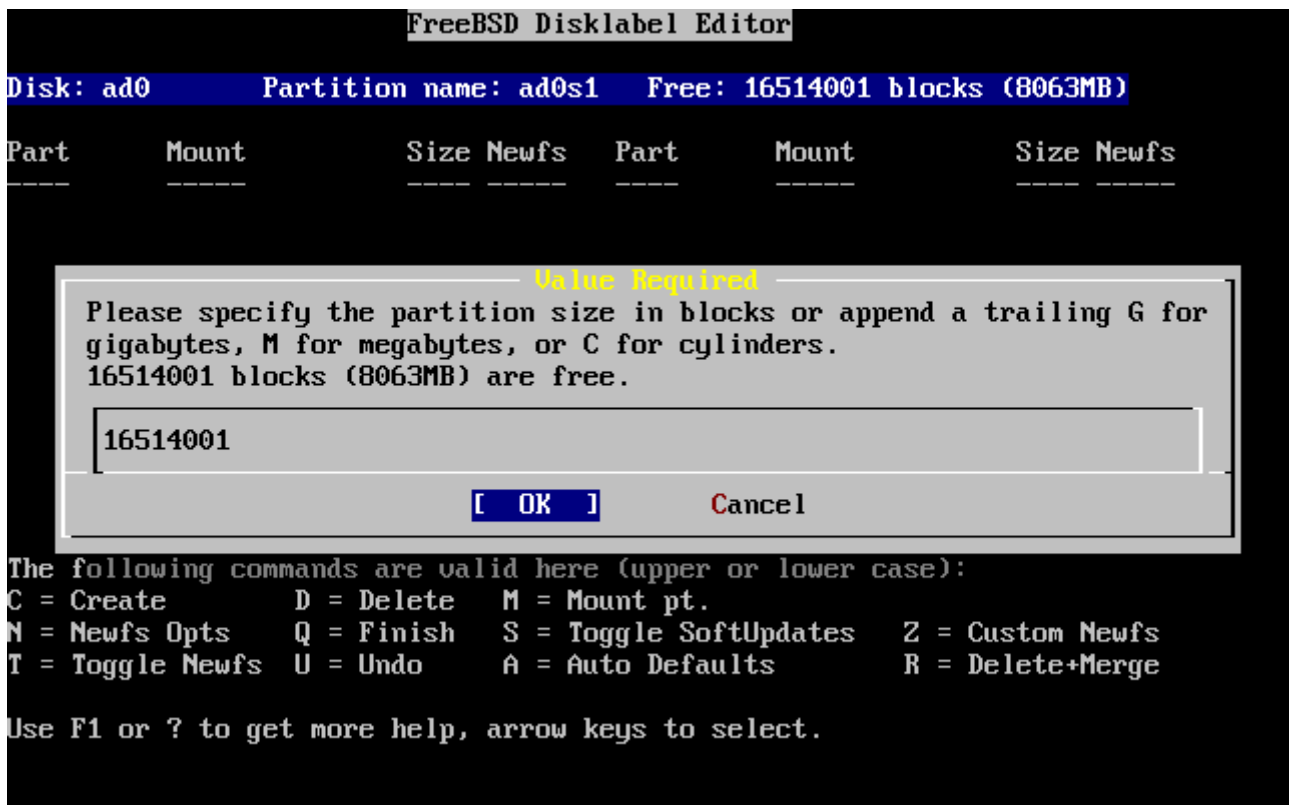
The following commands are valid here (upper or lower case):
C = Create      D = Delete    M = Mount pt.
N = Newfs Opts  Q = Finish    S = Toggle SoftUpdates  Z = Custom Newfs
T = Toggle Newfs U = Undo      A = Auto Defaults      R = Delete+Merge

Use F1 or ? to get more help, arrow keys to select.
```

#### 19. Sysinstall Disklabel 器-使用自配置

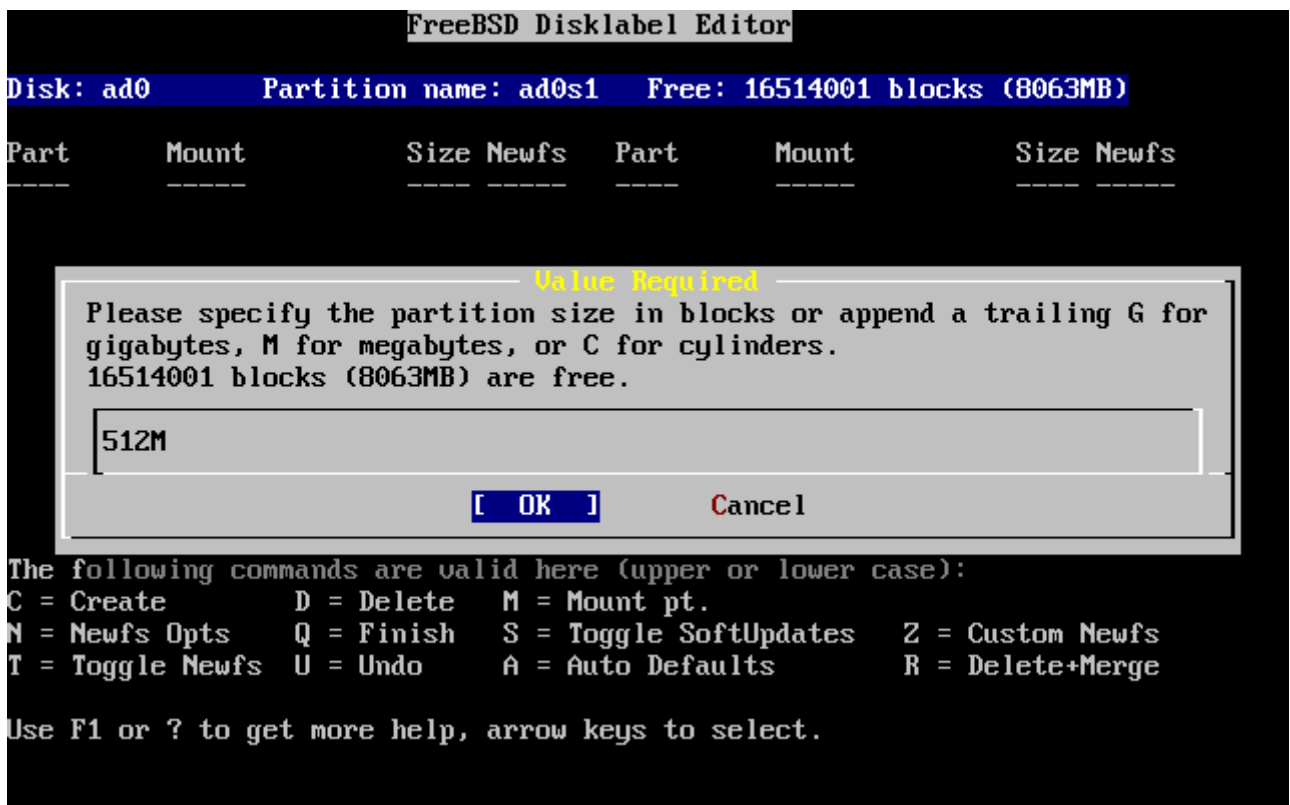
如果不想使用默认的分區布局， 需要用方向移光标并中第一个分区， 然后按 **D** 来除它。 重新一程直到除了所有推的分区。

要建立第一个分区 (**a**，作 / - 根文件系)， 已 在屏幕部中了正的 slice， 然后按 **C**。 接下来将出一个框， 要求入新分区的尺寸 (如 [根目使用空](#) 所示)。 可以入以位的尺寸， 或以 **M** 表示MB、 **G** 表示GB， 或者 **C** 表示柱面数的方式来表尺寸。



□ 20. 根目□使用空□

如果使用此□□示的默□尺寸，□□会□建一个占□整个 slice 空余空□的 partition。如果希望使用前面例子中描述的 partition 尺寸，□□按 Backspace □□除□些数字，并□入 512M，如 □□要分区大小□所示。然后，按下 [OK]。



□ 21. □□要分区大小

□入完大小后接着□□要建立的分区是文件系□□是交□空□，如 □□根分区□型□所示。第一个分区是文件系□，所以□□□□ FS后按 Enter □。

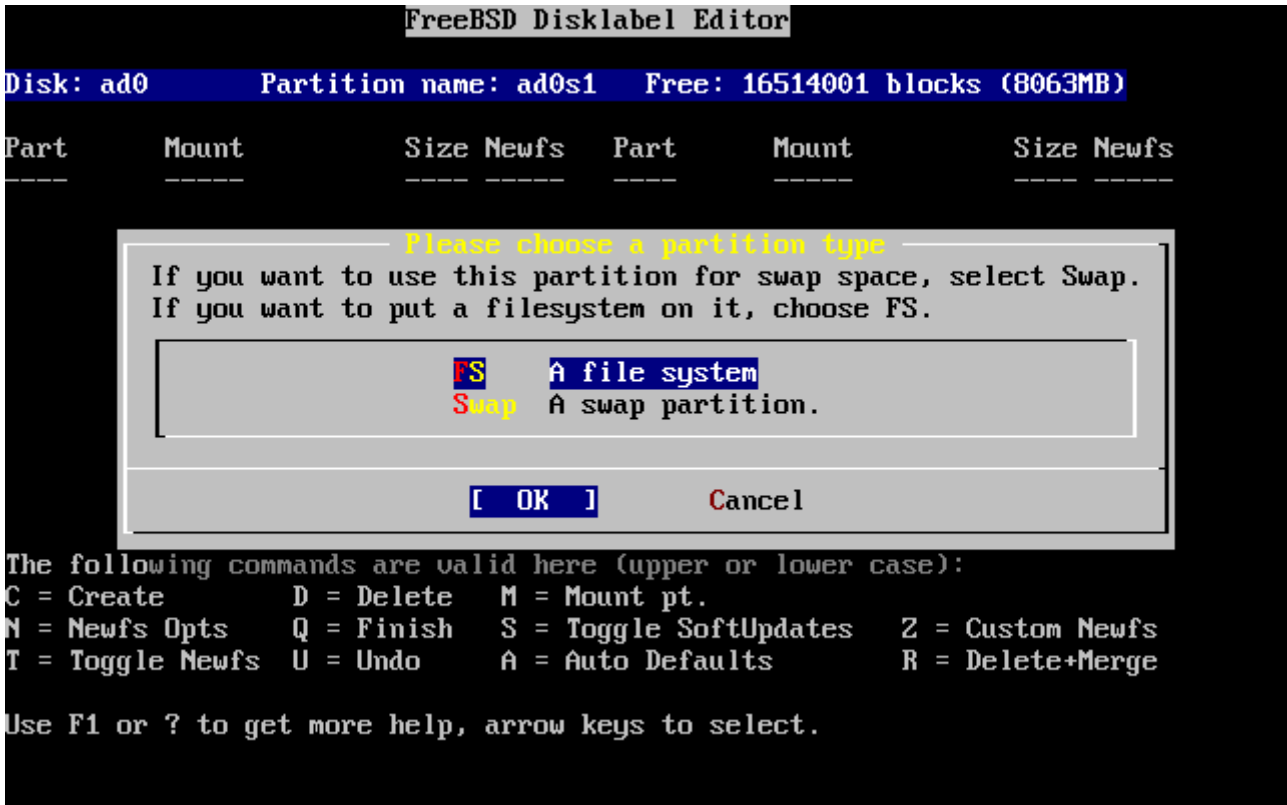


图 22. 根分区类型

最后，因我们要建立的是一个文件系统，所以必须告诉 Disklabel 这个文件系统要挂载在什么地方，如根挂载点所示。根文件系统的挂载点 /，所以我们输入 /，然后按 Enter。

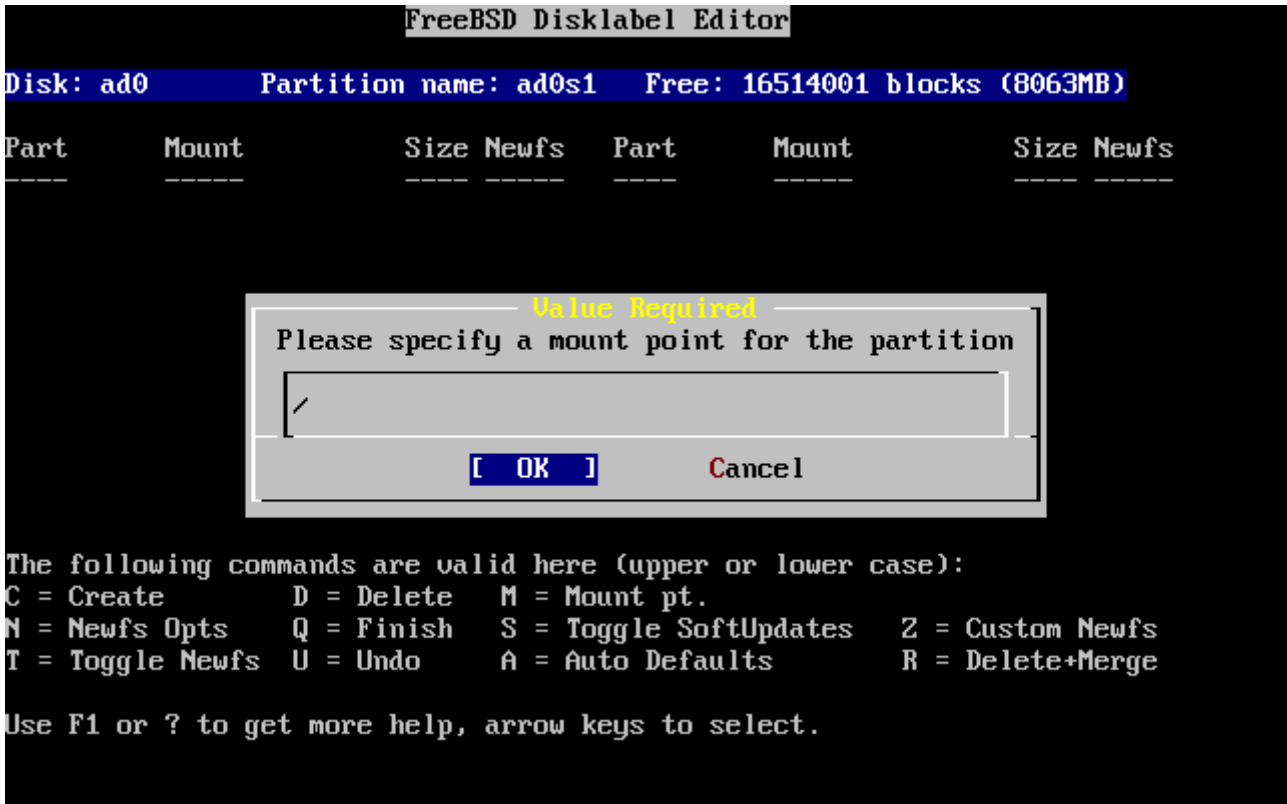


图 23. 根挂载点

制作好的分区会显示在画面上。我们重复上述的操作以建立其它的分区。当建立交空间的的时候，系统不会要将它挂载在哪里，因为交空间是不用挂在系统上的。当我们在建立最后一个分区/usr的时候，我们可以直接使用默认的大小，即所有此分区剩余的空間。

最后的 FreeBSD DiskLabel 编辑器画面会类似 [Sysinstall Disklabel 编辑器](#)，数字按钮的不同。按下 **Q** 完成分区的建立。

```
FreeBSD Disklabel Editor

Disk: ad0      Partition name: ad0s1      Free: 0 blocks (0MB)

Part      Mount      Size Newfs      Part      Mount      Size Newfs
-----
ad0s1a    /              512MB UFS2      Y
ad0s1b    swap          512MB SWAP
ad0s1d    /var          256MB UFS2+S Y
ad0s1e    /usr          6783MB UFS2+S Y

The following commands are valid here (upper or lower case):
C = Create      D = Delete      M = Mount pt.
N = Newfs Opts  Q = Finish      S = Toggle SoftUpdates  Z = Custom Newfs
T = Toggle Newfs U = Undo      A = Auto Defaults      R = Delete+Merge

Use F1 or ? to get more help, arrow keys to select.
```

图 24. Sysinstall Disklabel 编辑器

## 2.7. 要安装的软件包

### 2.7.1. 要安装的软件包

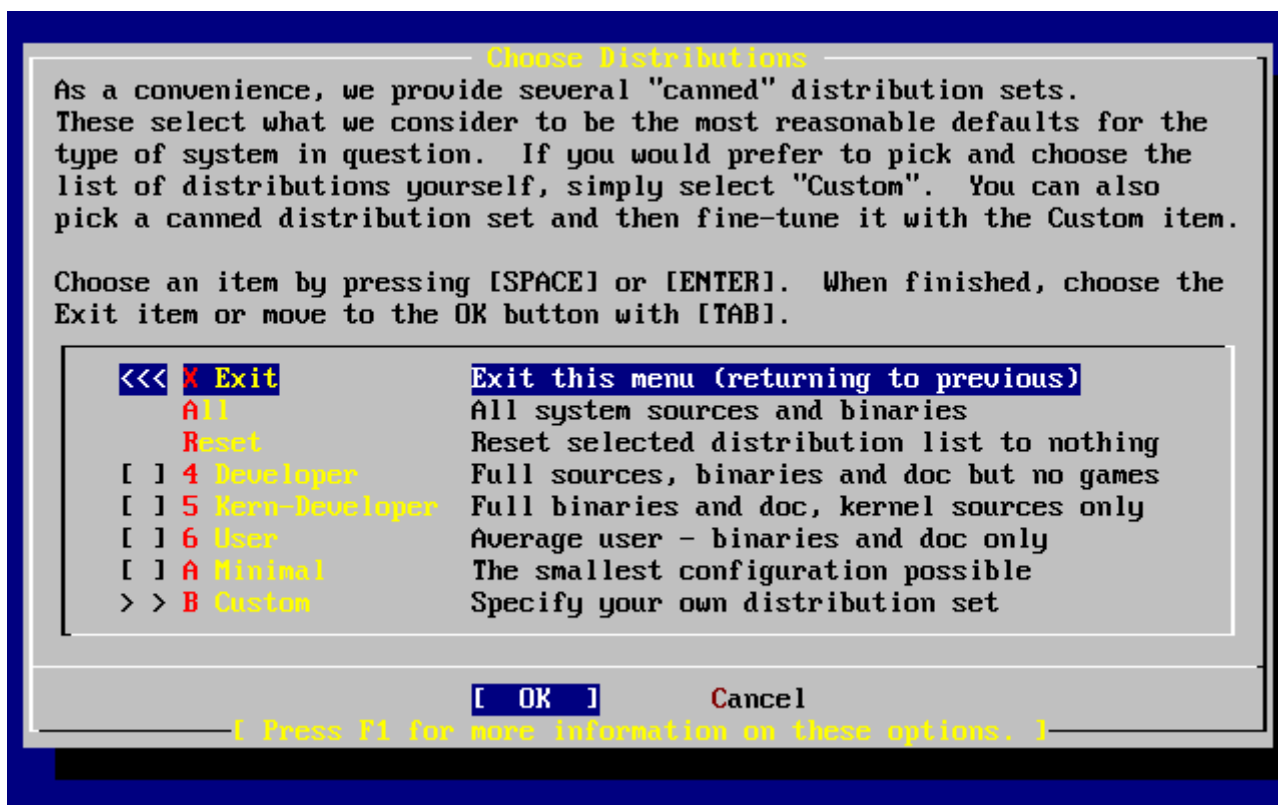
安装哪些软件包在很大程度上取决于系统将被用来做什么，以及有多少可用的磁盘空间。内建的软件包包括了系统所需要的最小系统，到把所有软件包全都装上的常用配置。UNIX® 或 FreeBSD 新手通常直接选一个定义好的软件包就可以了，而有经验的使用者可以考虑自己定制安装哪些软件包。

按下 **F1** 可以看到有软件包的更多信息，以及它们都包含了哪些软件，之后，可以按 **Enter** 回到软件包画面。

如果需要图形用户界面，配置 X 服务以及默认桌面需要在完成 FreeBSD 之后完成。关于安装和配置 X 服务的信息，可以在 [X Window 系统](#) 找到。

如果需要定制内核，您可能需要包含源代码的那个。要了解什么内核和新建新的内核，请参看 [配置 FreeBSD 的内核](#)。

当然，包含所有软件的系统是最万能的。如果磁盘空间足够，用光 [软件包](#) 中的 All 并按 **Enter**。如果担心磁盘空间不够的，选最合适的。不要担心选的是否是最合适的，因其他软件包可以在安装完后再加入来。



□ 25. □□□件包

## 2.7.2. 安装ports□件包

当□□完□想要安装的部分后，接着会□□□要不要安装FreeBSD Ports □件包；Ports□件包可以□□□□方便地安装□件包。Ports本身并不包含□□ □件所需要的程序源代码，而是一个包含自□下□、□□以及安装的文□集合。 [安装□用程序: Packages 和 Ports](#) 一章□□如何使用Ports。

安装程序并不会□□□是否有足□的硬□空□，在□□□—□之前□先□定□有足□的硬□空□。目前 FreeBSD 12.0 版本中，FreeBSD Ports Collection 大□占用 3 GB 大小的硬□空□。□于近期的版本□可能需要更多一些空□来安装他□。



User Confirmation Requested  
Would you like to **install** the FreeBSD Ports Collection?

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

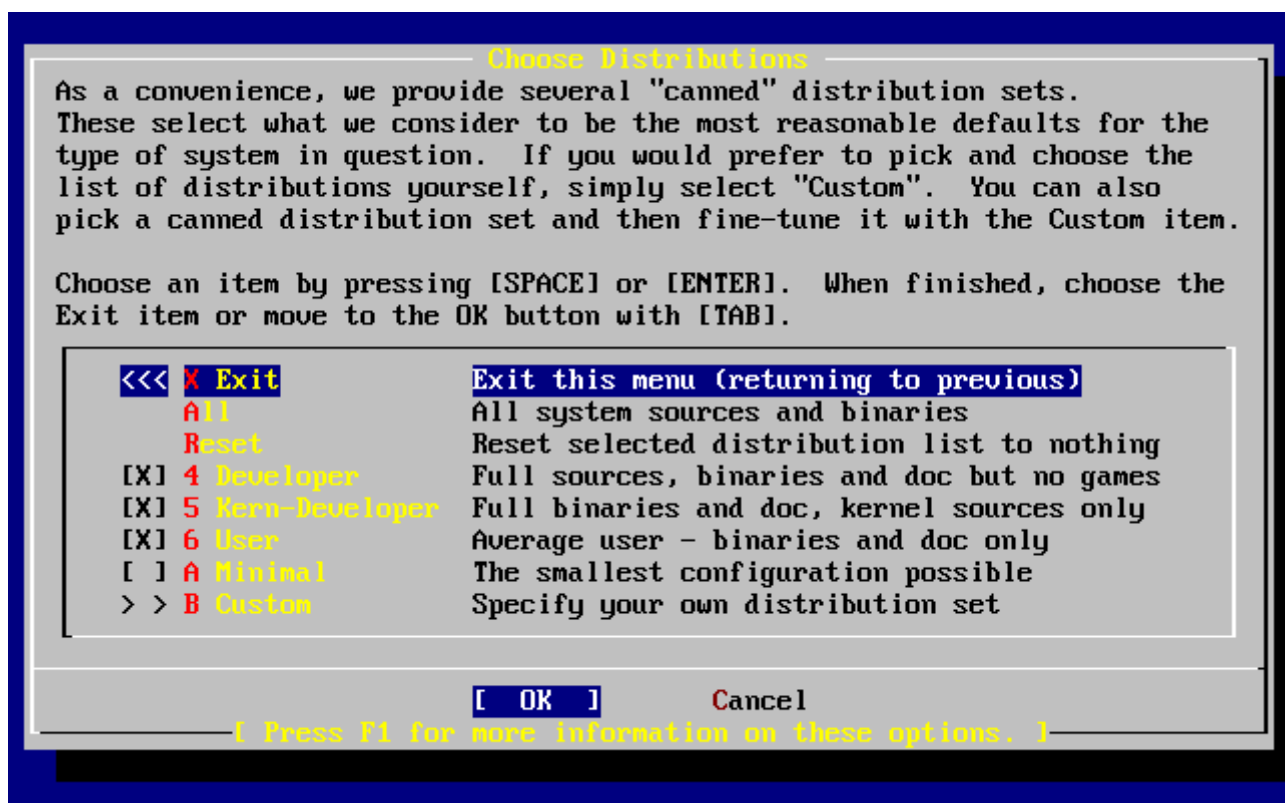
The Ports Collection is a very valuable resource and well worth having on your /usr partition, so it is advisable to say Yes to this option.

For more information on the Ports Collection & the latest ports, visit:

<http://www.FreeBSD.org/ports>

[ Yes ]      No

□□ **[yes]** 将会安装 Ports Collection, 而□□ **[no]** □将跳□它。□好后按 **Enter** □□。此后, □□安装的□件包的屏幕将再次出□。



□ 26. □□□要安装的□件包

如果□□的□□感到□意, □□□Exit 退出, □保**[OK]** 被高亮□示, 然后按 **Enter** □□。

## 2.8. 要使用的安装介

如果要从 CDROM 或 DVD 安装，使用方向键将光标移到 Install from a FreeBSD CD/DVD。按 **[OK]** 被取，然后按 **Enter** 开始安装程序。

如果要使用其它的方式安装，按 **F1** 显示安装介的在明。然后按照屏幕指示行安装。

按 **F1** 可以示安装介的在明。按一下 **Enter** 可返回安装介画面。

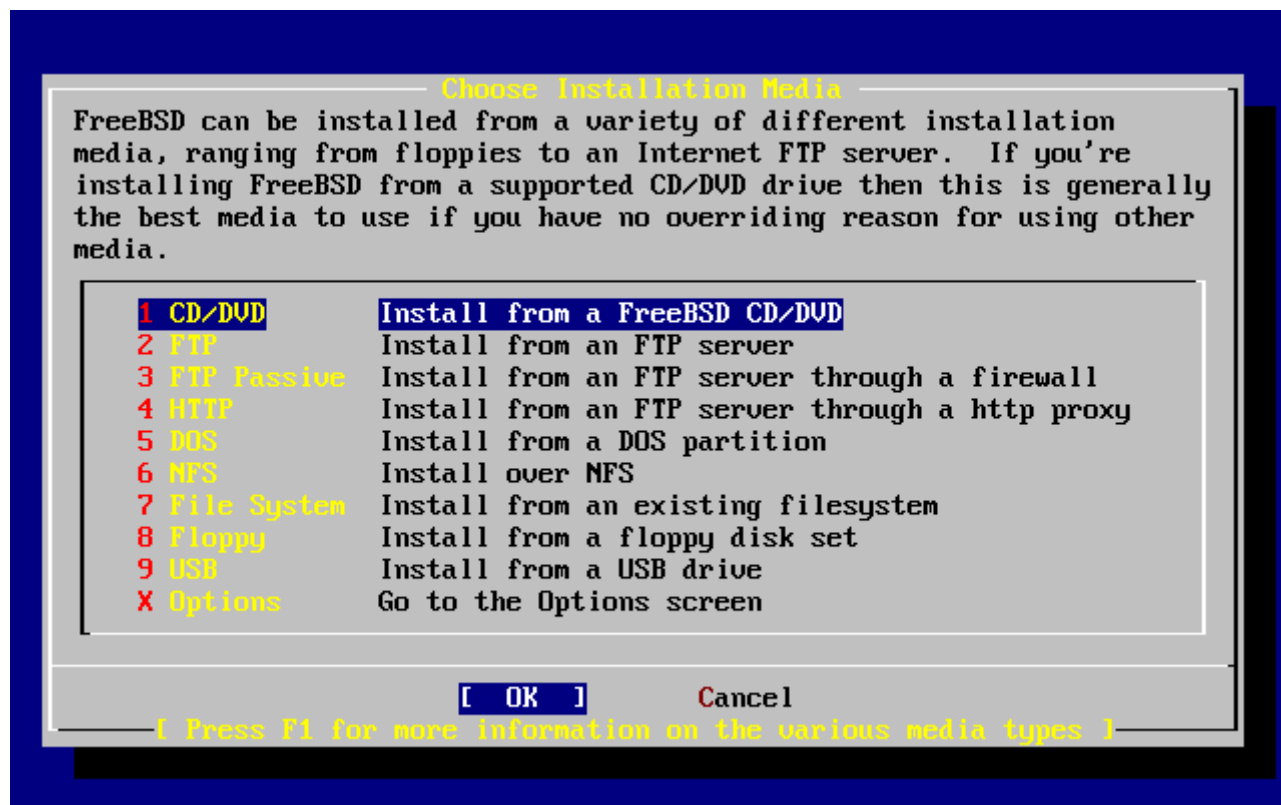


图 27. 安装介

## FTP安装模式

使用FTP安装，有三种方式：主式（active）FTP、被动式（passive）FTP 或是透过HTTP代理服务器。

### 主式FTP：从FTP服务器安装

这个将会使所有的FTP使用 "Active"模式。将无法通过防火墙，但是可以使用在那些比早期，不支持被动模式的FTP站。如果的接口在使用被动（默认）模式住了，主模式看看！

### 被动模式FTP：通过防火墙从FTP服务器安装

此会使用 `sysinstall` 使用 "Passive"模式来安装。使得使用者可以穿过 不允许用非固定TCP PORTS接入的防火墙。



### FTP 透过 HTTP 代理服务器：透过HTTP代理服务器，由FTP服务器安装

此会使用 `sysinstall` 通过HTTP（像浏览器一样）到proxy服务器。proxy服务器会解送出的请求，然后通知FTP服务器。因通过HTTP，所以可以穿防火墙。要用 方式，必须指定proxy服务器的地址。

于一个FTP代理服务器而言，通常在使用者登入名称中加入要登入的服务器的用户名，加在 "@" 符号后面。然后代理服务器就会 "假装" 成一个真的服务器。例如，假如要从 <ftp.FreeBSD.org> 安装，通过FTP代理服务器 [foo.example.com](http://foo.example.com)，使用1234端口。

在此情况下，可以到 options 菜单，将FTP username 为 `ftp@ftp.FreeBSD.org`，密码为子件地址。安装介绍部分，指定FTP（或是被动式FTP，如果代理服务器支持的）以及URL为 `ftp://foo.example.com:1234/pub/FreeBSD`。

因<ftp.FreeBSD.org>的 `/pub/FreeBSD` 目录会被取到 [foo.example.com](http://foo.example.com)之下，就可以从这台机器（会从<ftp.FreeBSD.org>取文件）安装。

## 2.9. 安装

到此为止，可以开始行安装了，也是避免更到硬的最后机会。

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

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

We can take no responsibility for lost disk contents!

[ Yes ]    No
```

按 **[yes]** 然后按下 `Enter` 安装

安装所需的时间会根据所装的件、安装介绍以及的速度而有所不同。在安装的过程中会有一些信息来示目前的度。

当您看到下面的信息表示已安装完成了：

Message

Congratulations! You now have FreeBSD installed on your system.

We will now move on to the final configuration questions.

For any option you **do** not wish to configure, simply **select** No.

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

[ OK ]

[ Press enter or space ]

按下 **Enter** 以进行安装后的配置。

按 **[no]** 然后按 **Enter** 会取消安装，不会给系统造成更改。您会看到下面的信息：

Message

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

[ OK ]

产生这个信息是因为什么也没有安装，按下

**Enter**

后会

安装程序回到主安装界面。从主安装界面可以退出安装程序。

## 2.10. 安装后的配置

安装成功后，就可以进行下一步的配置了。引导新安装的 FreeBSD 系统之后，使用 **sysinstall** 并配置。

### 2.10.1. 配置网络

如果您之前配置用 PPP 通过 FTP 安装，那么那个画面将不会出现；正像所看到的那样，您可以稍后再做配置。

如果想更多的了解网络或将 FreeBSD 配置成网络或路由器，请参考 [Advanced Networking](#) 的相关文章。

User Confirmation Requested

Would you like to configure any Ethernet or PPP network devices?

[ Yes ] No

如果要配置网口，按 **[yes]** 然后按 **Enter**。否 **[no]** 。

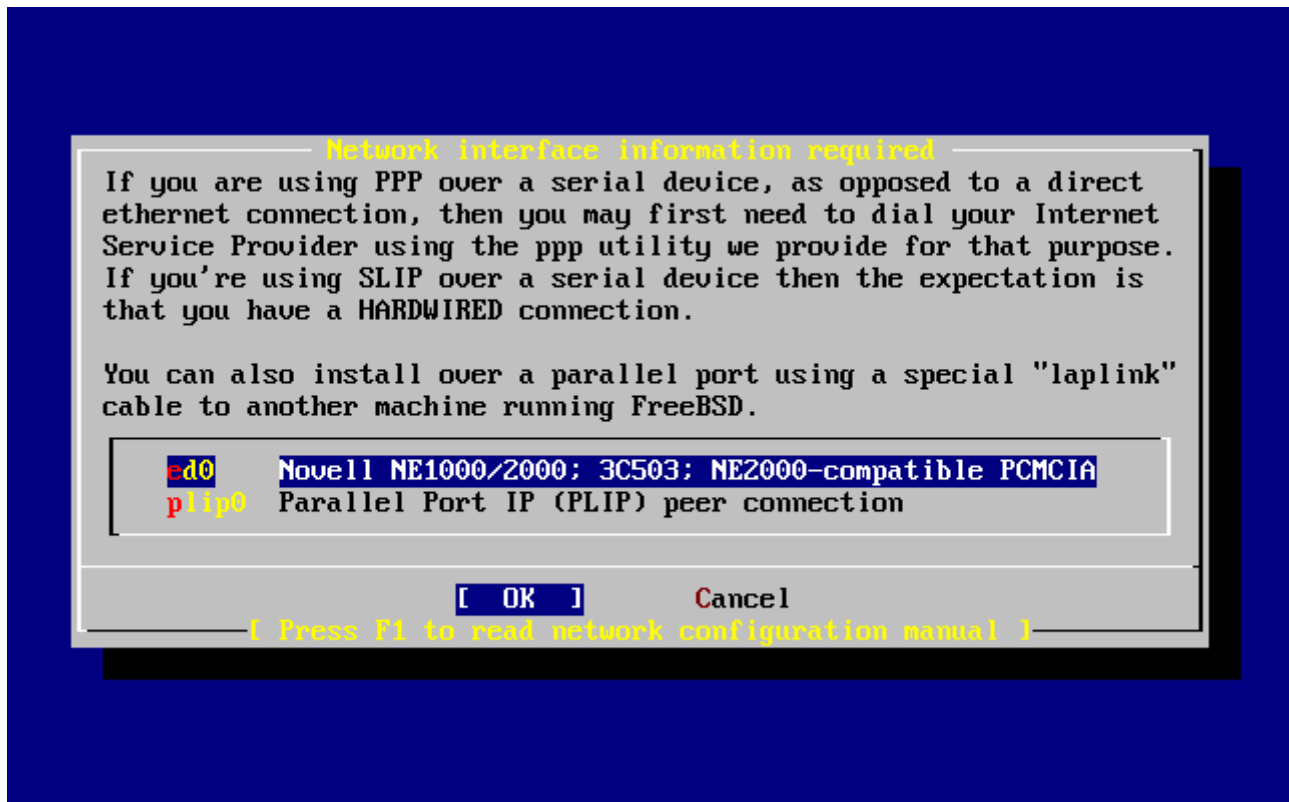
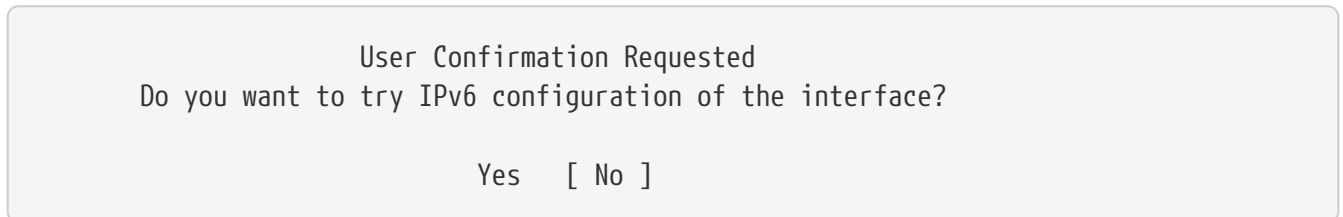


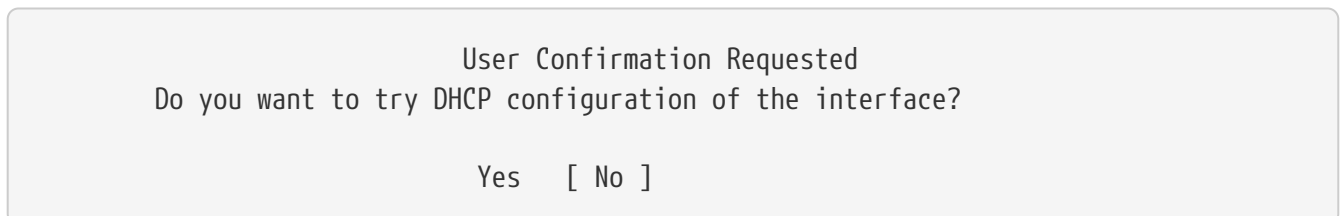
图 28. 网口配置

用方向键选择要配置的网口，然后按 **Enter**。



目前私人区域网口IPv4已足够，所以按 **[no]** 然后按 **Enter**。

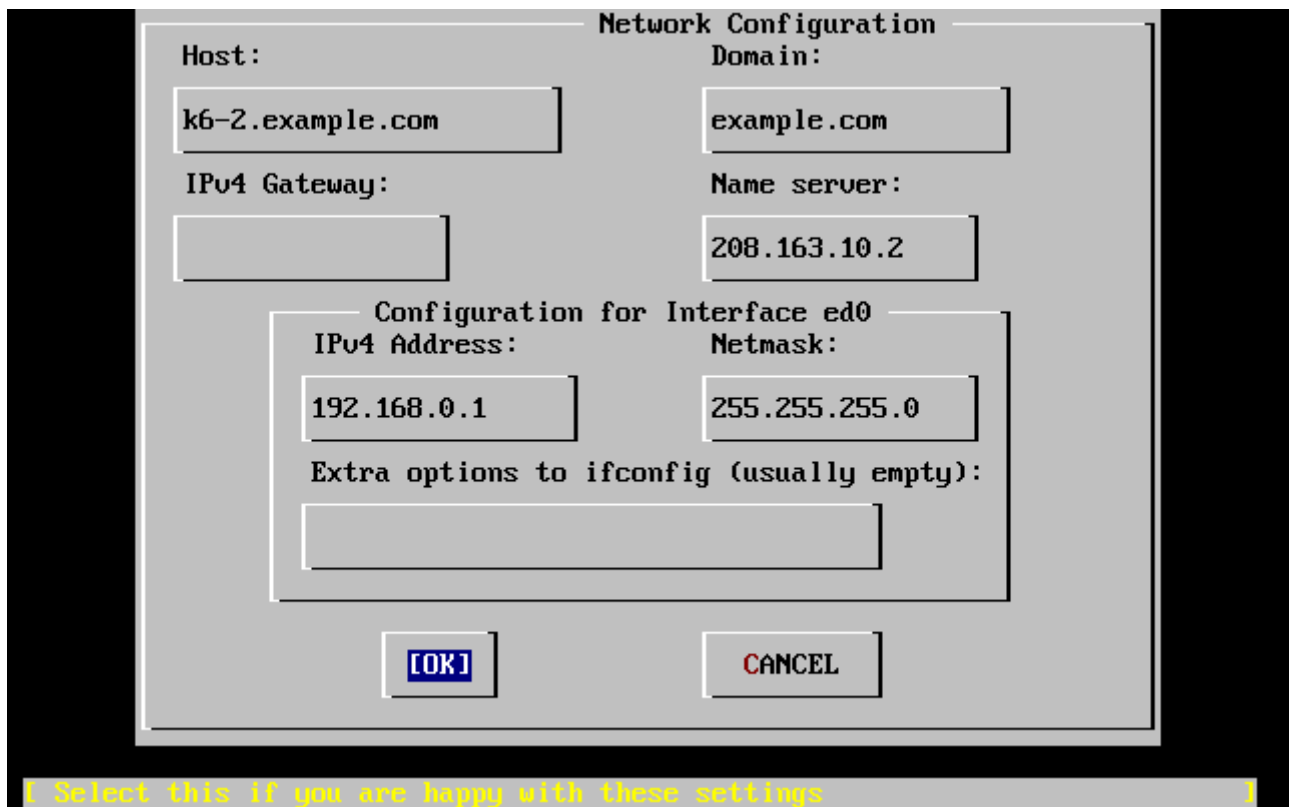
如果想新的IP通信 IPv6，使用 RA 服务，按 **[yes]** 然后按 **Enter**。RA 服务将会花几秒钟的时间。



如果不需要 DHCP (Dynamic Host Configuration Protocol 主机配置)，按 **[no]** 然后按 **Enter**。

按 **[yes]** 会运行 dhclient，如果成功，它会自将网口配置信息填上。更多的信息参考 [网口自配置 \(DHCP\)](#)。

下面的网口配置显示了把以太网口配置成区域网口的角色。



## □ 29. 配置 ed0接口

使用 **Tab** □可以在各个□目之□□行切□，□□入□当 的信息：

### Host（机器名称）

完整的机器名称，例如本例中的 **k6-2.example.com**。

### Domain（域名）

□机器所在的域名称，如本例的 **example.com**

### IPv4 Gateway（IPv4网□）

□入将数据包□送到□端网□的机器IP地址。 只有当机器是网□上的一个□点□才要□入。 如果□台机器要作□□局域网的网□，□将此□□□空白。IPv4网□， 也被称作默□网□或默□路由器。

### 域名服□器

本地网□中的域名服□器的IP地址。 本例中假□机器所在的网□中没有域名服□器， 所以填入的是ISP提供的域名服□器地址（**208.163.10.2**。）

### IPv4 地址

本机所使用的IP地址。本例□ **192.168.0.1**。

### 子网掩□

在□个局域网中所使用的地址□是 **192.168.0.0 - 192.168.0.255**， □□的子网掩□是 **255.255.255.0**。

### ifconfig □外参数□定

任何ifconfig命令跟网□接口有□的参数。 本□例中没有。

使用 **Tab** □□□ **[ OK ]**然后按 **Enter** □。

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

[ Yes ]   No
```

输入 **[yes]** 然后按 `Enter` 将会将机器的网络接口启用。机器下次启动的时候即可使用。

## 2.10.2. 配置网络

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

[ Yes ]   No
```

如果这台机器要作本地网络和其它机器之间送数据包的网关，输入 **[yes]** 然后按 `Enter`。如果这台机器只是网络上的普通点，输入 **[no]** 并按 `Enter`。

## 2.10.3. 配置网络服务

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

Yes   [ No ]
```

如果输入 **[no]**，许多网络服务，如 `telnetd` 将不会启用。那么，远端用户将无法 `telnet` 进入这台机器。本机上的用户是可以 `telnet` 到远端机器的。

有些服务可以在安装完成后修改 `/etc/inetd.conf` 配置文件来启用它。参看 [这里](#) 以得到更多的信息。

如果想现在就配置这些网络服务，输入 **[yes]**，然后会看到下面的信息：

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

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

[ Yes ]   No
```

输入 **[yes]**。

### User Confirmation Requested

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

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

[ Yes ] No

□□ [yes] 将允□添加网□服□ (或将相□网□服□□行□□的 # 除掉即可。)

```
^[ (escape) menu    ^y search prompt    ^k delete line      ^p prev li         ^g prev page
^o ascii code       ^x search           ^l undelete line    ^n next li         ^u next page
^u end of file      ^a begin of line    ^w delete word      ^b back 1 char
^t top of text      ^e end of line      ^r restore word     ^f forward 1 char
^c command          ^d delete char      ^j undelete char    ^z next word
=====line 1 col 0 lines from top 1 =====
# $FreeBSD: src/etc/inetd.conf,v 1.73.10.2.4.1 2010/06/14 02:09:06 kensmith Exp
#
# Internet server configuration database
#
# Define *both* IPv4 and IPv6 entries for dual-stack support.
# To disable a service, comment it out by prefixing the line with '#'.
# To enable a service, remove the '#' at the beginning of the line.
#
#ftp      stream  tcp      nowait  root    /usr/libexec/ftpd      ftpd -l
#ftp      stream  tcp6     nowait  root    /usr/libexec/ftpd      ftpd -l
#ssh      stream  tcp      nowait  root    /usr/sbin/sshd         sshd -i -4
#ssh      stream  tcp6     nowait  root    /usr/sbin/sshd         sshd -i -6
#telnet   stream  tcp      nowait  root    /usr/libexec/telnetd   telnetd
#telnet   stream  tcp6     nowait  root    /usr/libexec/telnetd   telnetd
#shell    stream  tcp      nowait  root    /usr/libexec/rshd      rshd
#shell    stream  tcp6     nowait  root    /usr/libexec/rshd      rshd
#login    stream  tcp      nowait  root    /usr/libexec/rlogind   rlogind
#login    stream  tcp6     nowait  root    /usr/libexec/rlogind   rlogind
file "/etc/inetd.conf", 118 lines
```

□ 30. □□ inetd.conf 配置文件

在加入□想□用的服□后，按下 **Esc** □会出□一个 □□框可以□□□□以及保存修改。

## 2.10.4. □用 SSH 登□

### User Confirmation Requested

Would you like to **enable** SSH login?

Yes [ No ]

□□ [yes] 便会□用 **sshd(8)**，也就是 OpenSSH 服□程序。它能□□□以安全的方式从□程□□机器。如欲了解□于 OpenSSH 的□一□□情，□参□ [OpenSSH](#)。



## 2.10.5. 匿名 FTP

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

Yes      [ No ]
```

### 2.10.5.1. 不允许匿名 FTP

默认的 **[no]** 并按下 **Enter** 将仍然可以在这台机器上有号的用 FTP。

### 2.10.5.2. 允许匿名 FTP

如果允许匿名 FTP 存取，那网中任何人都可以使用FTP来这台机器。在用匿名之前考虑网的安全。如果要知道更多有网安全的信息，参[安全](#)。

要用FTP匿名，用方向键 **[yes]** 并按 **Enter**。系统会出一的信息：

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

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

Do you wish to continue configuring anonymous FTP?

[ Yes ]      No
```

些信息会告诉 FTP 服需要在 /etc/inetd.conf 中用。假如希望允许匿名 FTP 接，参[配置网服](#)。 **[yes]** 并按 **Enter** ； 系统将出下列信息：

Anonymous FTP Configuration

UID: 14      Group: ftp      Comment: Anonymous FTP Admin

Path Configuration

FTP Root Directory: /var/ftp

Upload Subdirectory: incoming

OK      CANCEL

What user ID to assign to FTP Admin

### 31. 默认的匿名 FTP 配置

使用 **Tab** 在不同的信息字段之间切换，并填写必要的信息：

#### UID

用于分配匿名 FTP 用户的用户 ID。所有上传的文件的所有者都将是该 ID。

#### Group

匿名 FTP 用户所在的组。

#### Comment

用于在 `/etc/passwd` 中描述用户的说明性信息。

#### FTP Root Directory

可供匿名 FTP 用户使用的文件所在的根目录。

#### Upload Subdirectory

匿名 FTP 用户上传的文件存放位置。

默认的 FTP 根目录将放在 `/var` 目录下。如果该目录空间不足以应付您的 FTP 需求，您可以将 FTP 的根目录改到 `/usr` 目录下的 `/usr/ftp` 目录。

当一切配置都满意后，请按 **Enter** 键。

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

[ Yes ]      No

如果输入 **[yes]** 并按下 **Enter**，系统会自启动文本编辑器编辑FTP的欢迎信息。

```
^[ (escape) menu  ^y search prompt  ^k delete line    ^p prev line     ^g prev page
^o ascii code     ^x search          ^l undelete line  ^n next line     ^v next page
^u end of file    ^a begin of line   ^w delete word    ^b back char     ^z next word
^t begin of file  ^e end of line    ^r restore word   ^f forward char
^c command        ^d delete char    ^j undelete char
                                     ESC-Enter: exit
=====
Your welcome message here.

file "/var/ftp/etc/ftpmotd", 1 lines, read only
```

### 32. 编辑FTP欢迎信息

此文本编辑器叫做 **ee**。按照指示修改信息文本或是在后再用喜欢的文本编辑器来修改。记住画面下方显示的文件位置。

按 **Esc** 将出一个默认 a) leave editor 的对话框。按 **Enter** 退出并。再次按 **Enter** 将保存修改。

## 2.10.6. 配置网络文件系统

网络文件系统（NFS）可以可以在网络上共享的文件。一台机器可以配置成NFS服务器、客户端或者并存。  
参考 [网络文件系统（NFS）](#) 以得更多的信息。

### 2.10.6.1. NFS 服务器

User Confirmation Requested

Do you want to configure this machine as an NFS server?

Yes [ No ]

如果不安装网络文件系统，输入 **[no]** 然后按 **Enter**。

如果输入 **[yes]** 将会出一个对话框提醒必须先建立一个 `exports` 文件。

### Message

Operating as an NFS server means that you must first configure an /etc/exports file to indicate which hosts are allowed certain kinds of access to your **local** filesystems.

Press [Enter] now to invoke an editor on /etc/exports

[ OK ]

按 **Enter** 回车。系统会打开文本编辑器编辑 exports 文件。

```
^[ (escape) menu    ^y search prompt    ^k delete line      ^p prev li          ^g prev page
^o ascii code        ^x search            ^l undelete line    ^n next li          ^u next page
^u end of file        ^a begin of line     ^w delete word       ^b back 1 char
^t begin of file      ^e end of line       ^r restore word      ^f forward 1 char
^c command            ^d delete char       ^j undelete char     ^z next word

L: 1 C: 1 =====
#The following examples export /usr to 3 machines named after ducks,
#/usr/src and /usr/ports read-only to machines named after trouble makers
#/home and all directories under it to machines named after dead rock stars
#and, /a to a network of privileged machines allowed to write on it as root.
#/usr                                huey louie dewie
#/usr/src /usr/obj -ro               calvin hobbes
#/home -alldirs                      janice jimmy frank
#/a -maproot=0 -network 10.0.1.0 -mask 255.255.248.0
#
# You should replace these lines with your actual exported filesystems.
# Note that BSD's export syntax is 'host-centric' vs. Sun's 'FS-centric' one.

file "/etc/exports", 12 lines
```

□ 33. □ exports 文件

按照指示加入真输出的文件目录或是□后用□喜欢的□器自行□。□下画面下方□示的文件名称及位置。

按下 **Esc** 会出一具□框，默认是 a) leave editor。按下 **Enter** □并□。

### 2.10.6.2. NFS 客户端

NFS 客户端允许□的机器□NFS服务器。

### User Confirmation Requested

Do you want to configure this machine as an NFS client?

Yes [ No ]

按照□的需要，□ **[yes]** 或 **[no]** 然后按 **Enter**。

## 2.10.7. 配置系统端

系统提供了几个可以配置端的表方式。

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

```
[ Yes ] No
```

要及配置些， [yes] 并按 `Enter`。

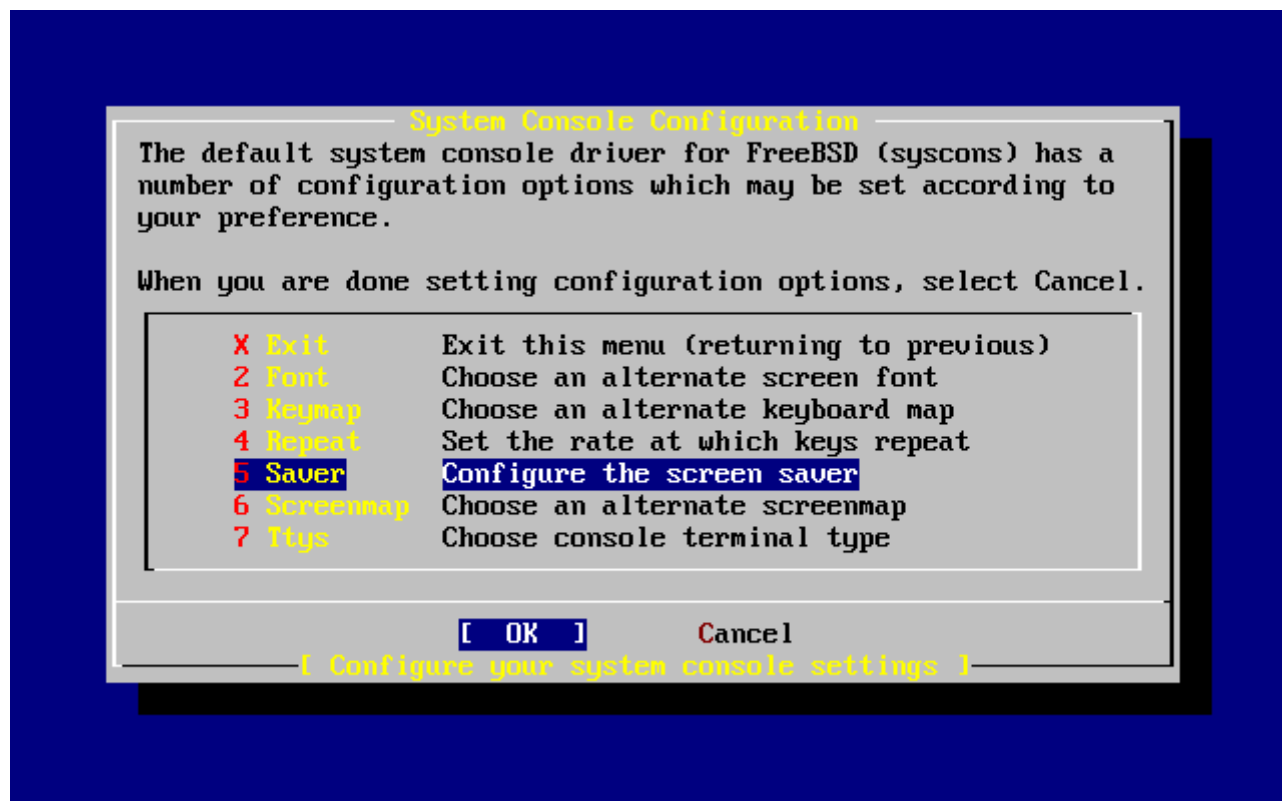
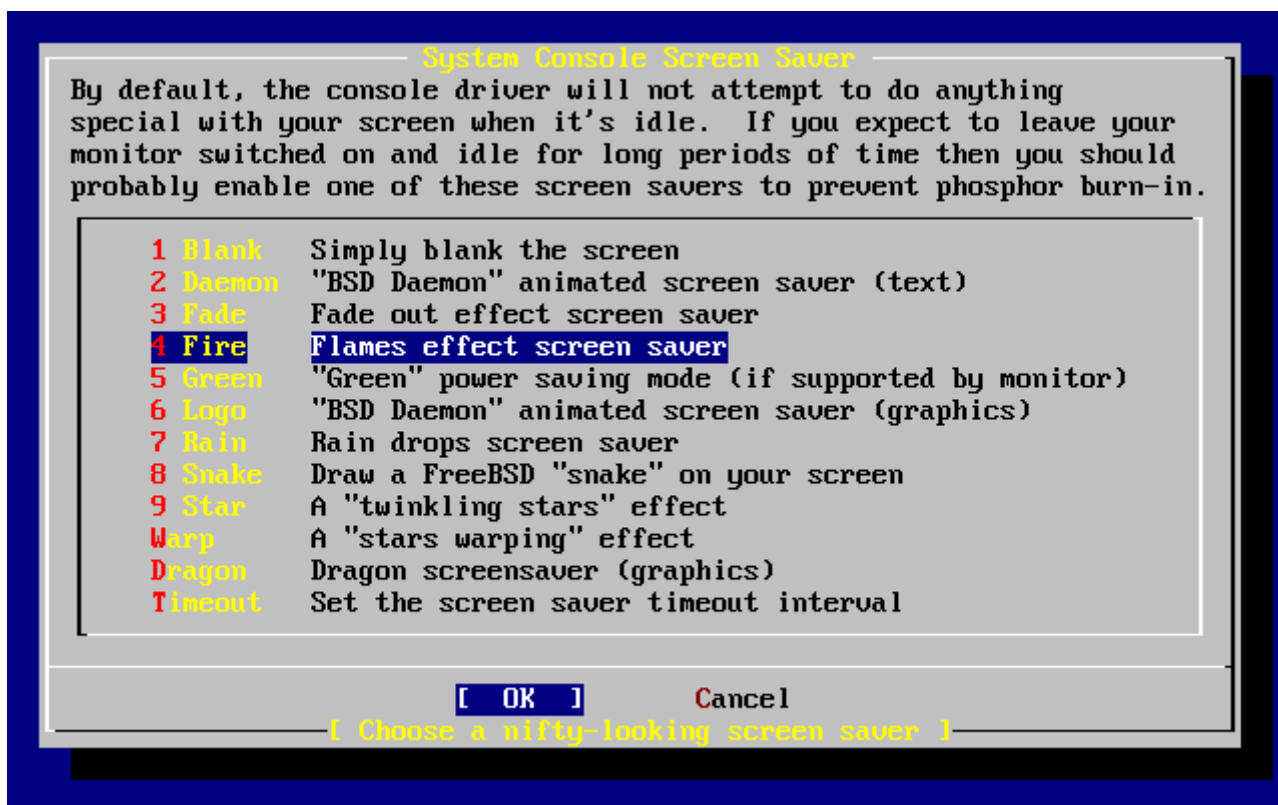


图 34. 系统端配置

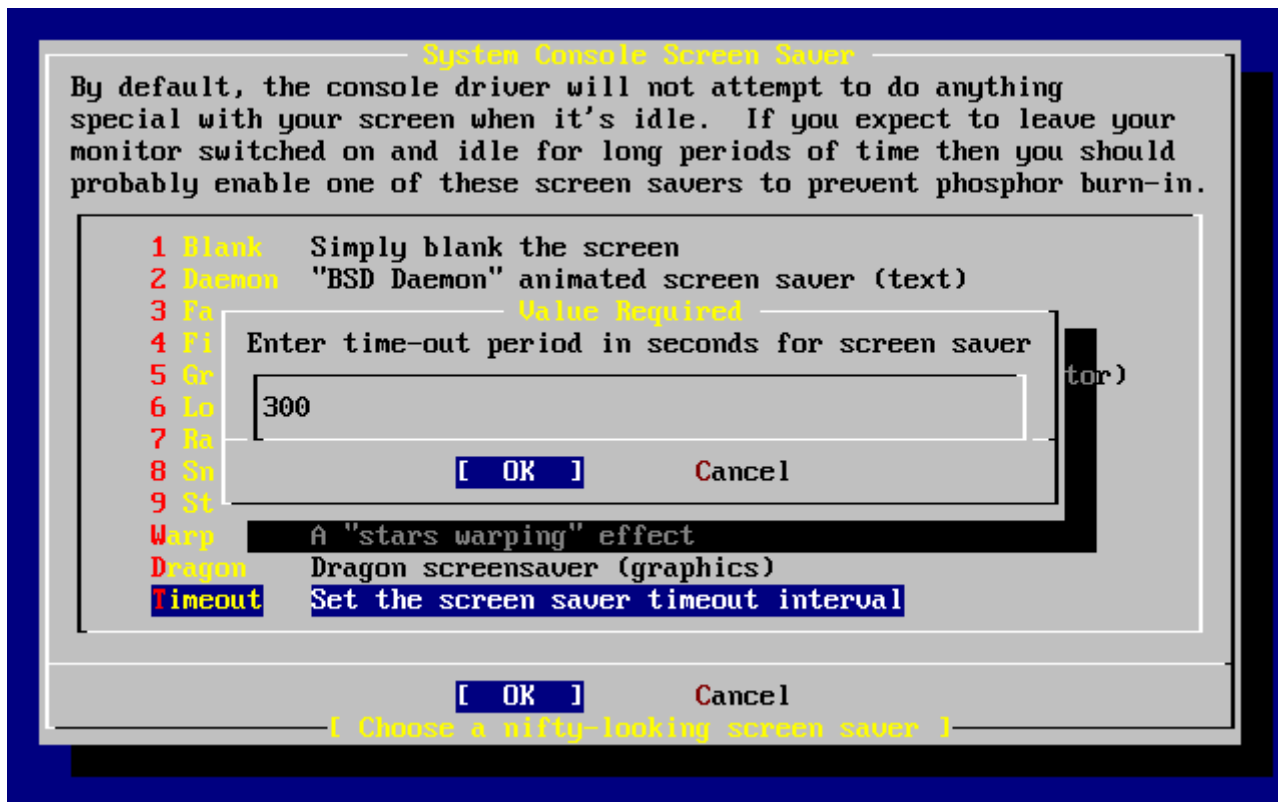
最常用的就是屏幕保护程序了。使用方向键将光标移到 Saver 然后按 `Enter`。



□ 35. 屏幕保□程序□□

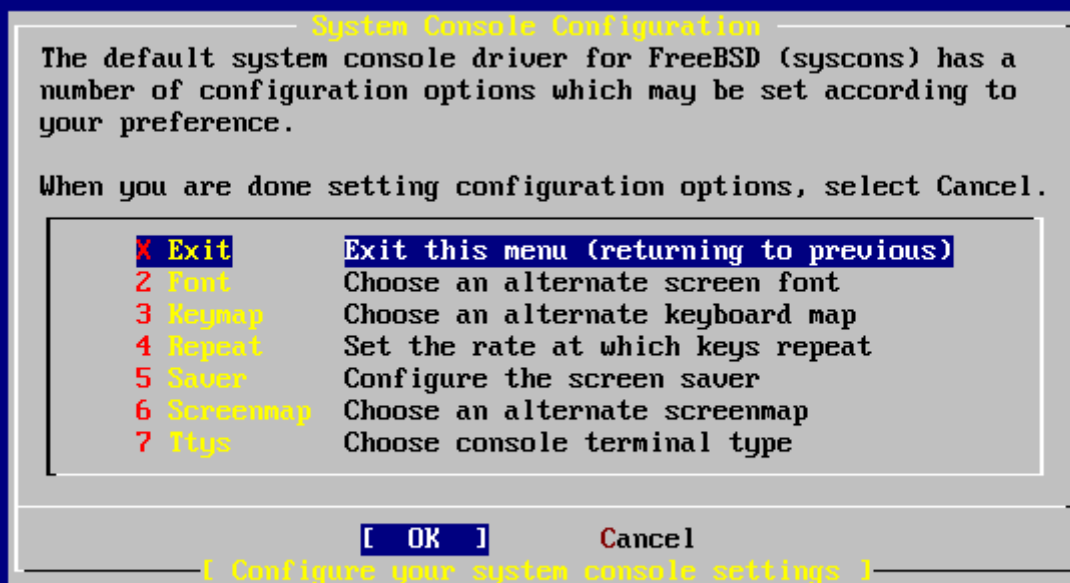
□□□想使用的屏幕保□程序，然后按 。之后回到系□□端配置画面。

默□□□屏幕保□程序的□□是300秒。如果要更改此□□，□再次□□ Saver。然后□□ Timeout 并按 。系□□会□□出一个□□框如下：



□ 36. 屏幕保□□□□□置

□□可以直接改□□个□，然后□  并按  □□回到系□□端配置画面。



□ 37. 退出系□端配置

□□ Exit 然后按下  □会回到安装后的配置画面。

## 2.10.8. 配置□区

配置□机器的□区可以□系□自□校正任何区域□□的□更，并且在□行一些跟□区相□的程序□不会出□。

例子中假□此台机器位于美国□部的□区。□参考□所在的地理位置来配置。

User Confirmation Requested  
Would you like to **set** this machine's **time zone** now?

[ **Yes** ]    No

□□ [yes] 并按下  □以配置□区。

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

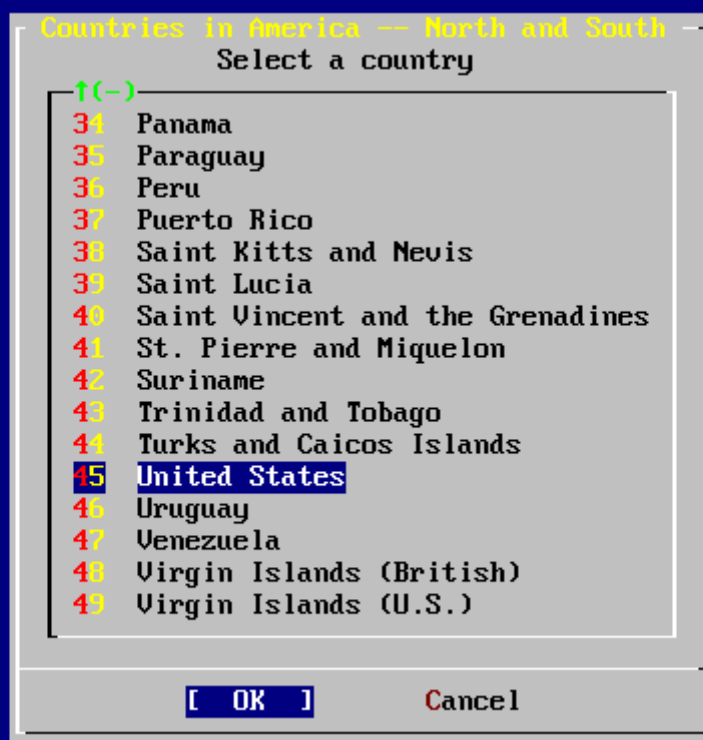
Yes    [ No ]

□里按照□机器□□的配置，□□ [yes] 或 [no] 然后按 。



□ 38. □□□所□的地理区域

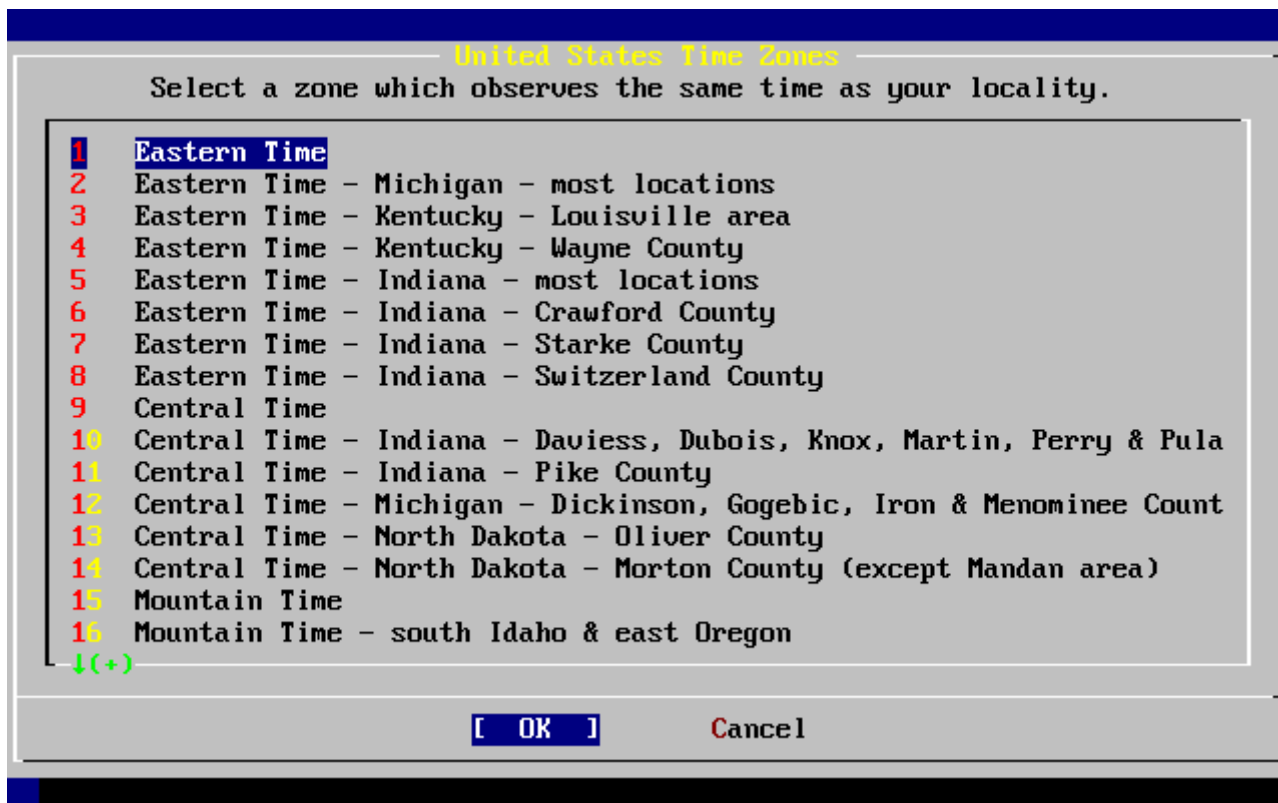
□□□□当的区域然后按 Enter。



□ 39. □□□所在的国家

□□□所在的国家然后按 Enter。





□ 40. □□□所在的□区

□□□所在的□区然后按 。



□□一下□区的□写是否正□，如果没□，□按  返回系□安装后的配置画面。

## 2.10.9. Linux 兼容性



□□内容只□用于 FreeBSD 7.X 安装□程， 如果□安装的是 FreeBSD 8.X 或更高版本， 系□不会□出□个提示。



□□ [yes] 并按下  □， 将允□□在FreeBSD中□行Linux的□件。安装程序会安装一些□了跟 Linux 兼容的□件包。

如果□是通□FTP安装， 那□□必□□到网□上。 □□□候FTP站并不会包含所有的安装□件包（例如Linux兼容□件包）； 不□， □后□□可以再安装□个□目。

## 2.10.10. 配置鼠标

此窗口可以在终端上使用三键鼠标剪贴文字。 如果您用的鼠标是个按钮，请参考手册 [moused\(8\)](#)；以取得有关模式三键鼠标的信息。 示例中使用的鼠标不是USB接口。（例如ps/2或com接口的鼠标）：

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

[ Yes ]    No

如果您使用的是 PS/2、串口或 Bus 鼠标，请按 **[yes]**，如果是 USB 鼠标，请按 **[no]** 并按 **Enter**。

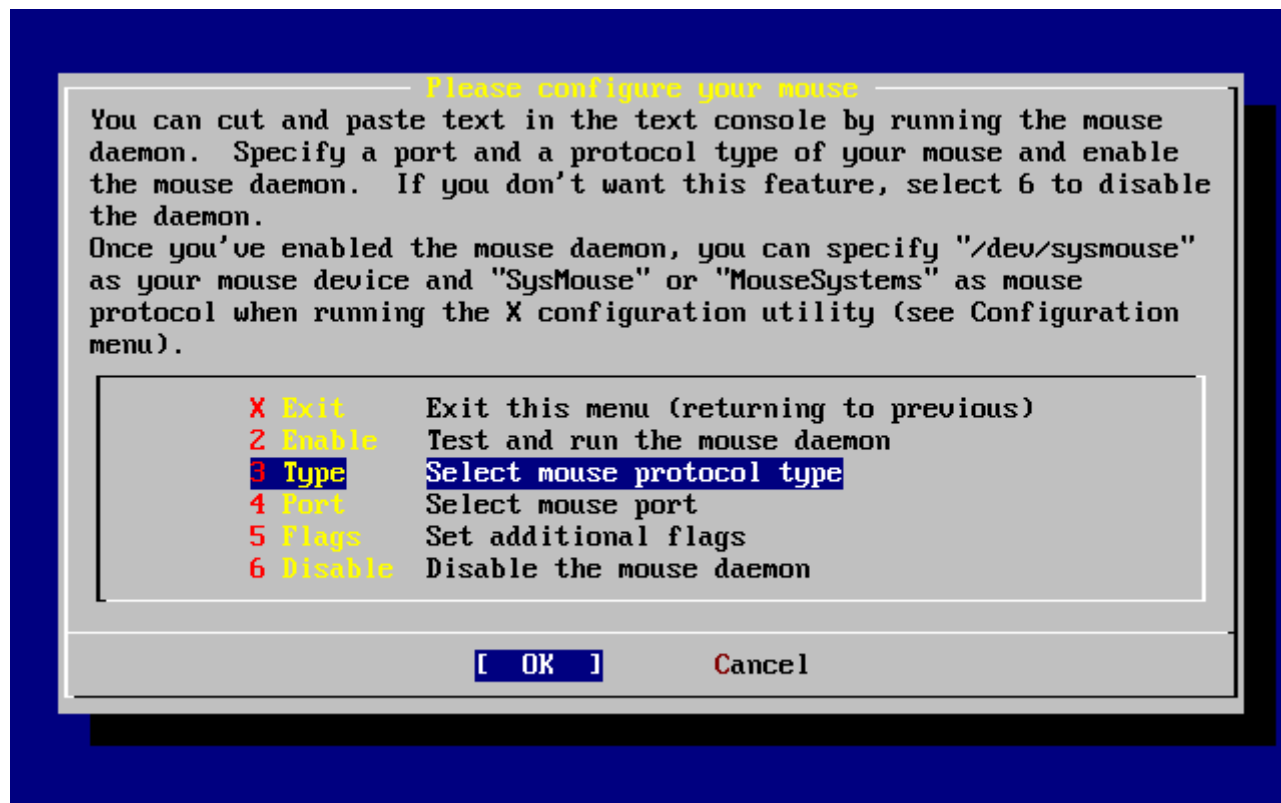
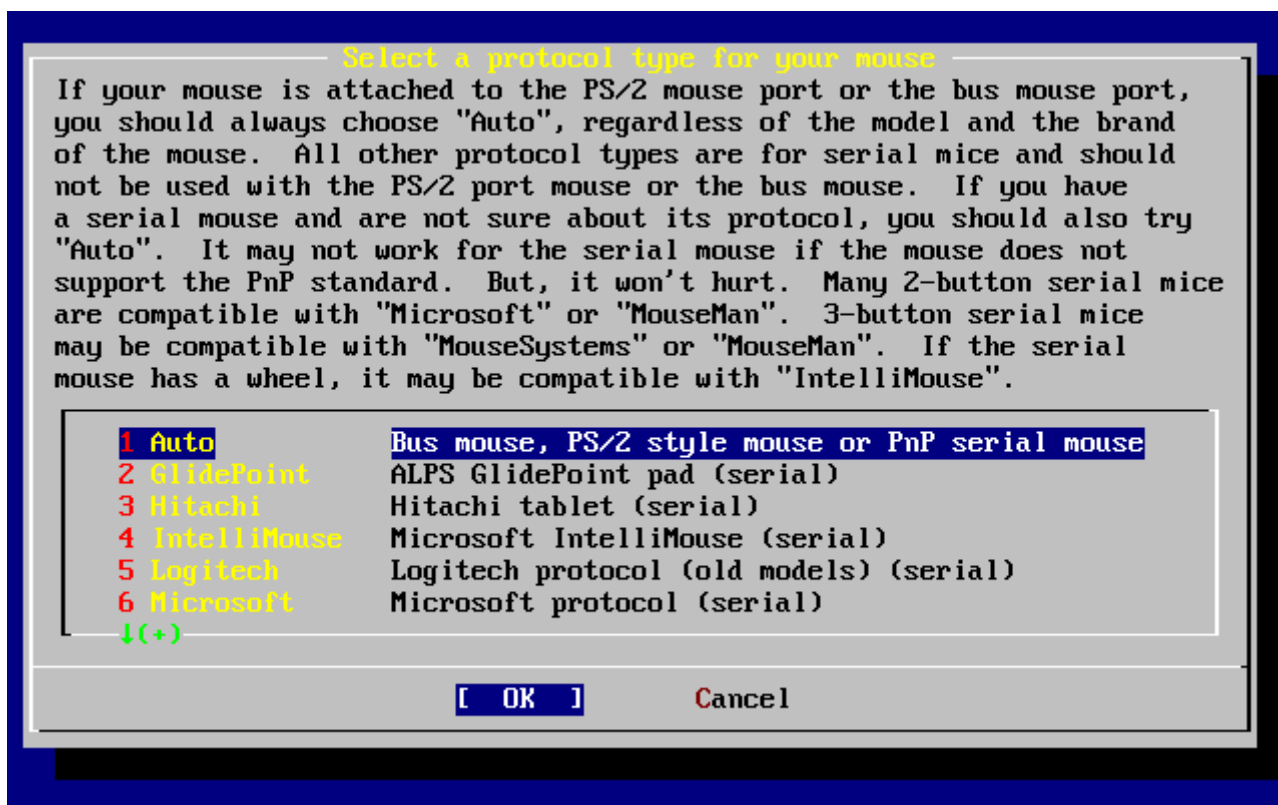


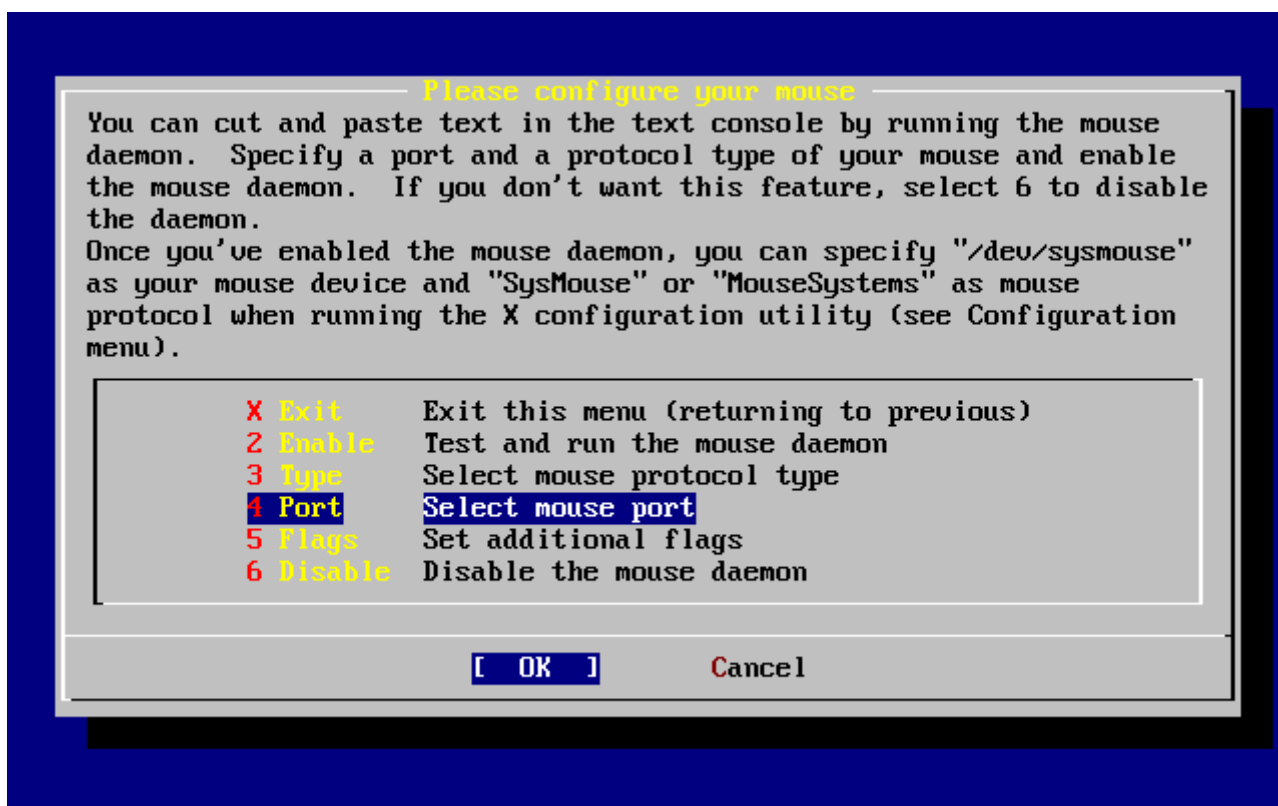
图 41. 鼠标配置型

使用方向键 **Type** 然后按 **Enter**。



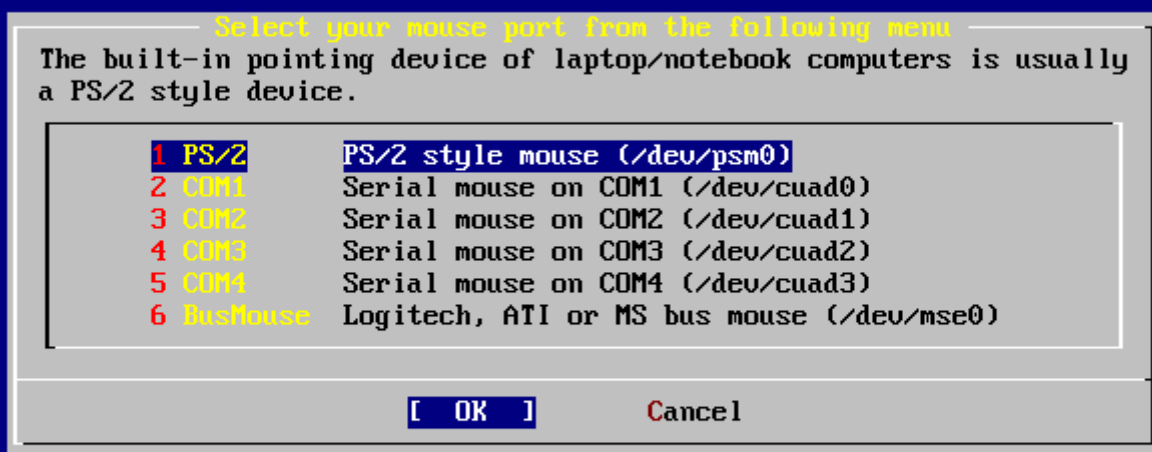
□ 42. □置鼠□□

在□个例子中使用的□型是ps/2鼠□，所以可以使用默□的 Auto（自□）。□可以用方向□□合□的□目，□定□了 [OK] 后按  □□此画面。



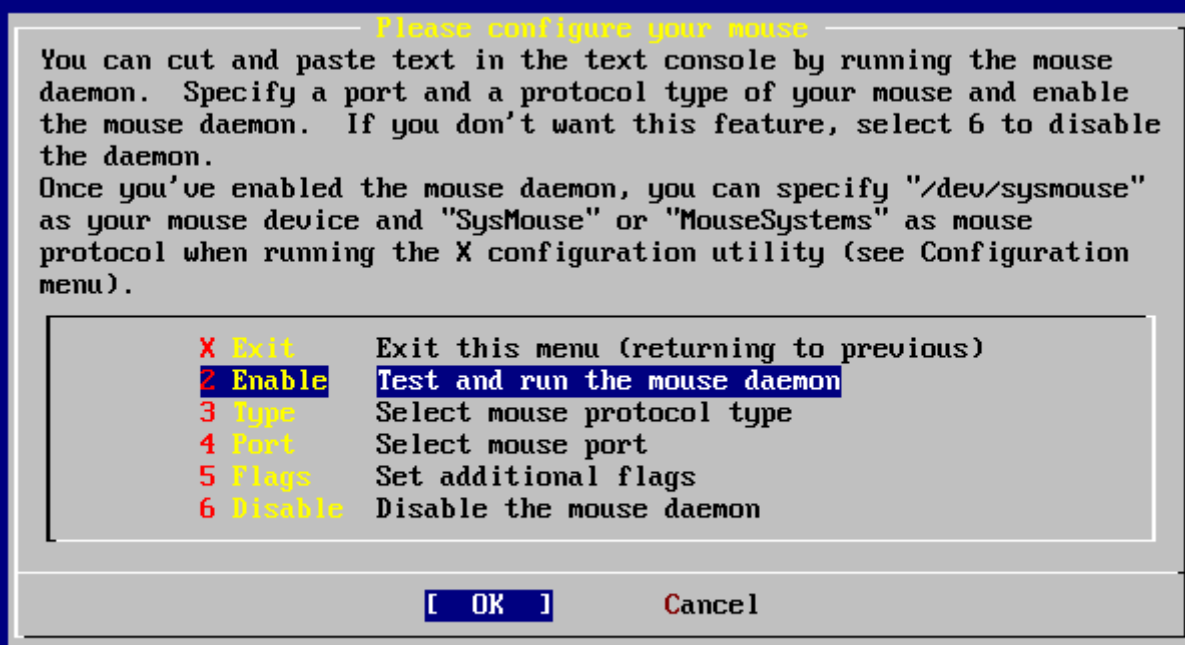
□ 43. 配置鼠□端口

□□ Port 然后按 。



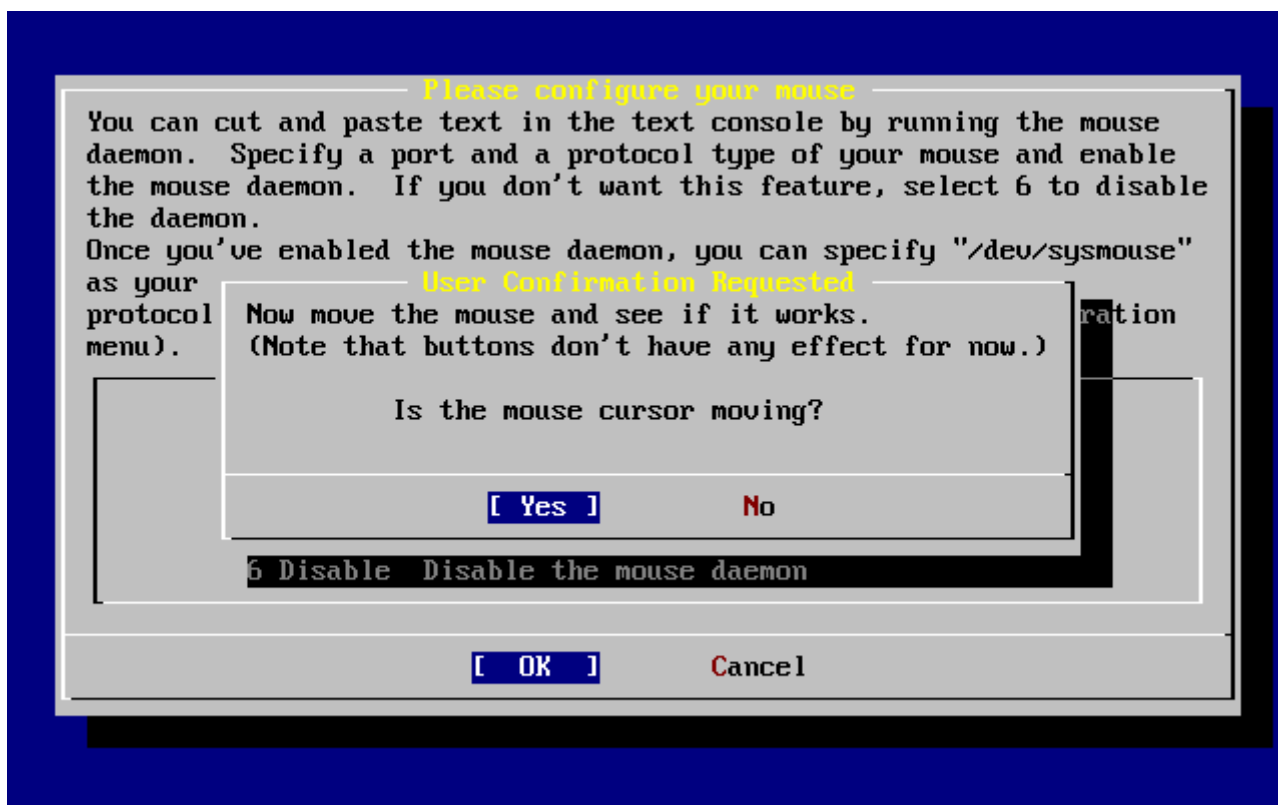
□ 44. 配置鼠标端口

假如台式机用的是ps/2鼠标，可以采用默认的 PS/2 端口。选中后按 。



□ 45. 启动鼠标服务程序

选中 Enable 然后按  来启动鼠标服务。



#### □ 46. □□鼠□功能

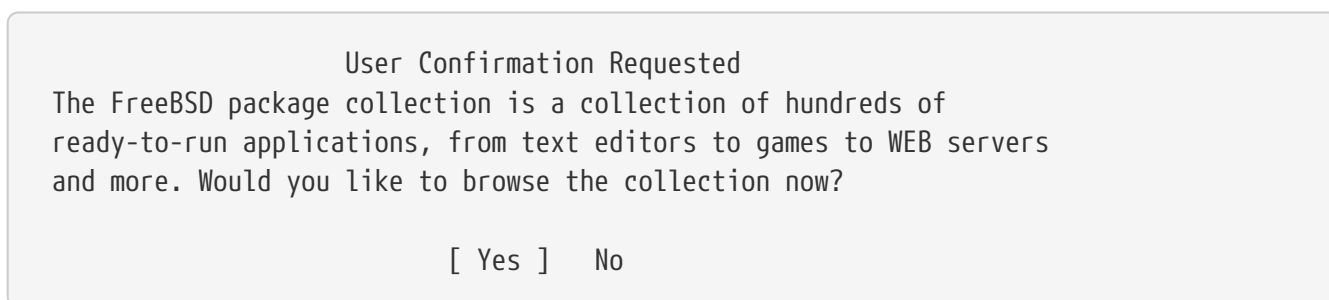
鼠□指□可以在屏幕上移□，指明鼠□服□已□正常□用。那□□□□ [yes] 按 。否□鼠□没 有配置成功 - □□ [no] 并□□不同的配置 □□。

□□ Exit 并按  退回到系□安装完成后的配置画面。

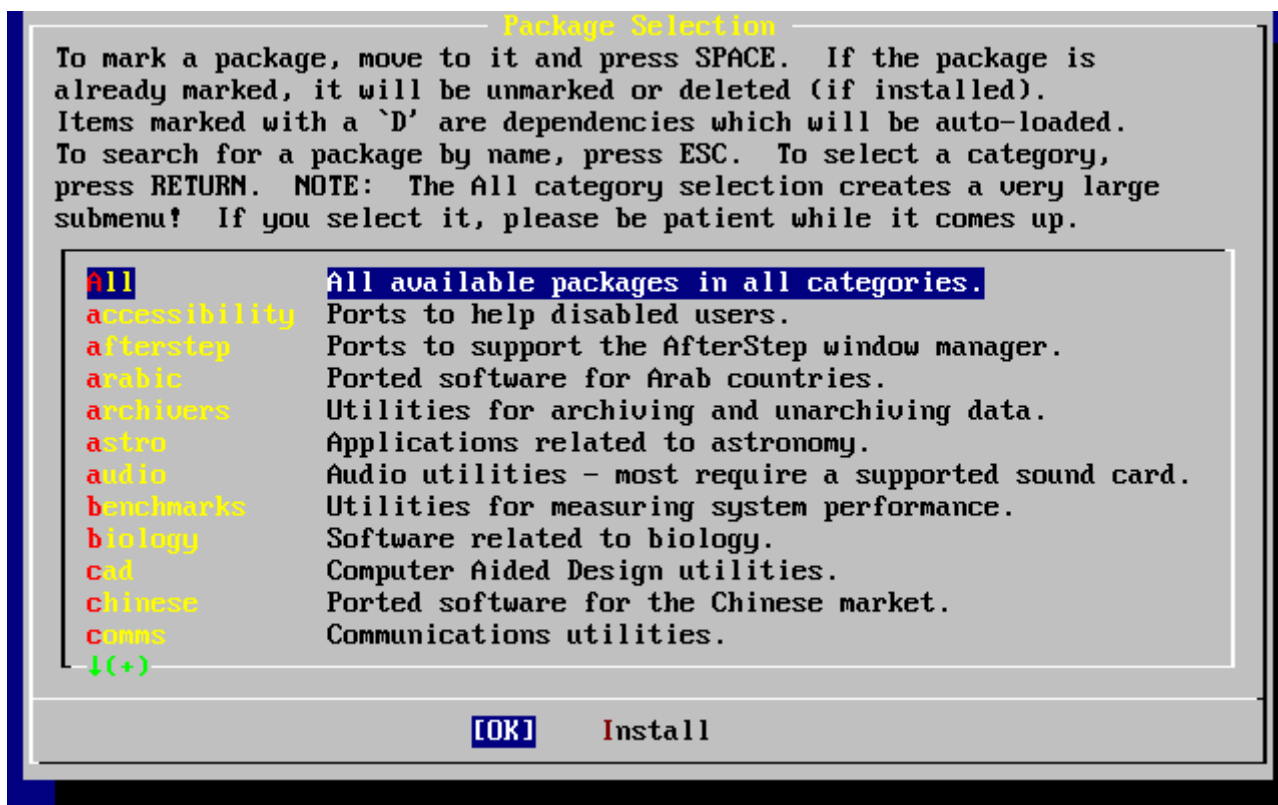
### 2.10.11. 安装□□□的□件包 (package)

Package 是事先□□好的二□制文件，因此，□是安装□件的一□便捷的方式。

在□里作□例子我□将□出安装一个 package 所需的□程。如果需要，□可以在□一□段加入其他 package。安装完成之后， `sysinstall` 依然可以用来安装其他 package。



□□ [yes] 并按  将□入 package □□界面：

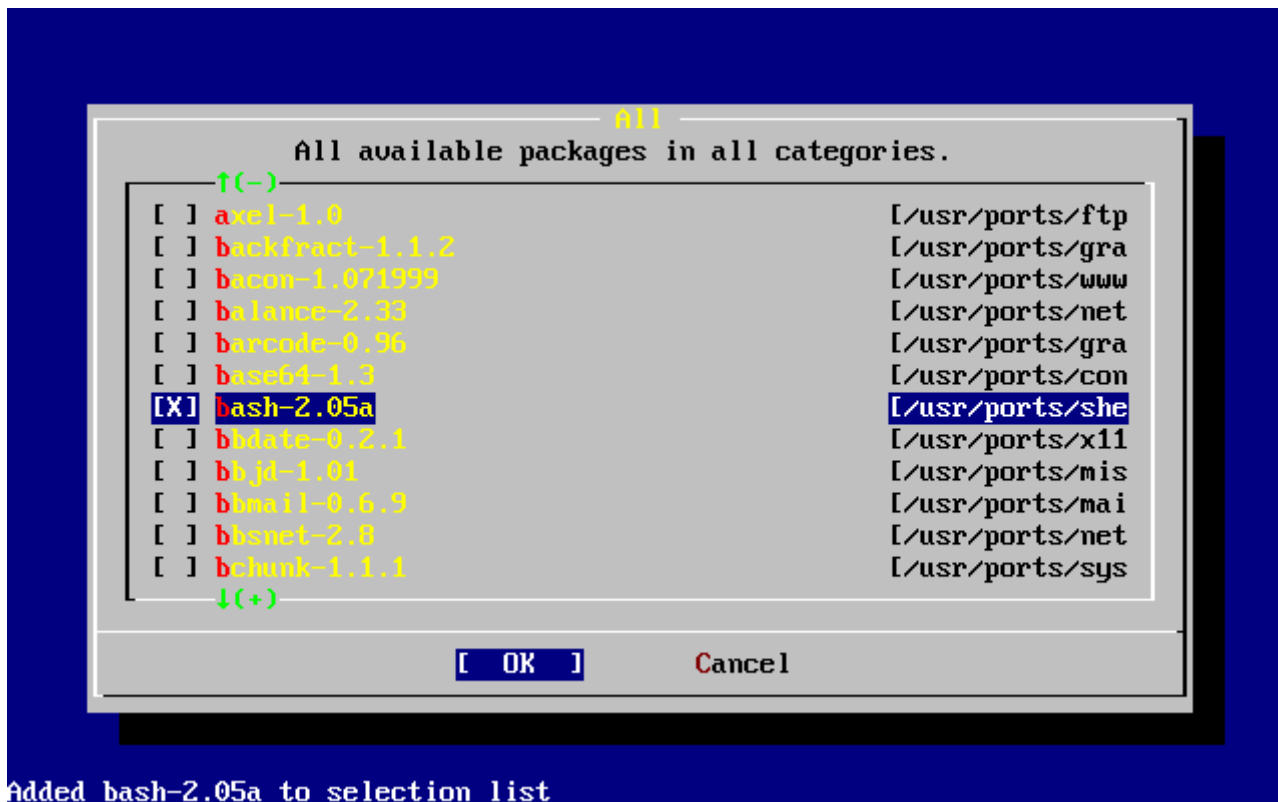


□ 47. □□ Package □□

在任何□候，只有当前安装介□上存在的 package 才可以安装。

如果□□了 All 或某个特定的分□，□系□会列出全部可用的 package。用光□□移□光棒□中需要的 package，并按 **Enter**。

系□会□示可供□□的 package：



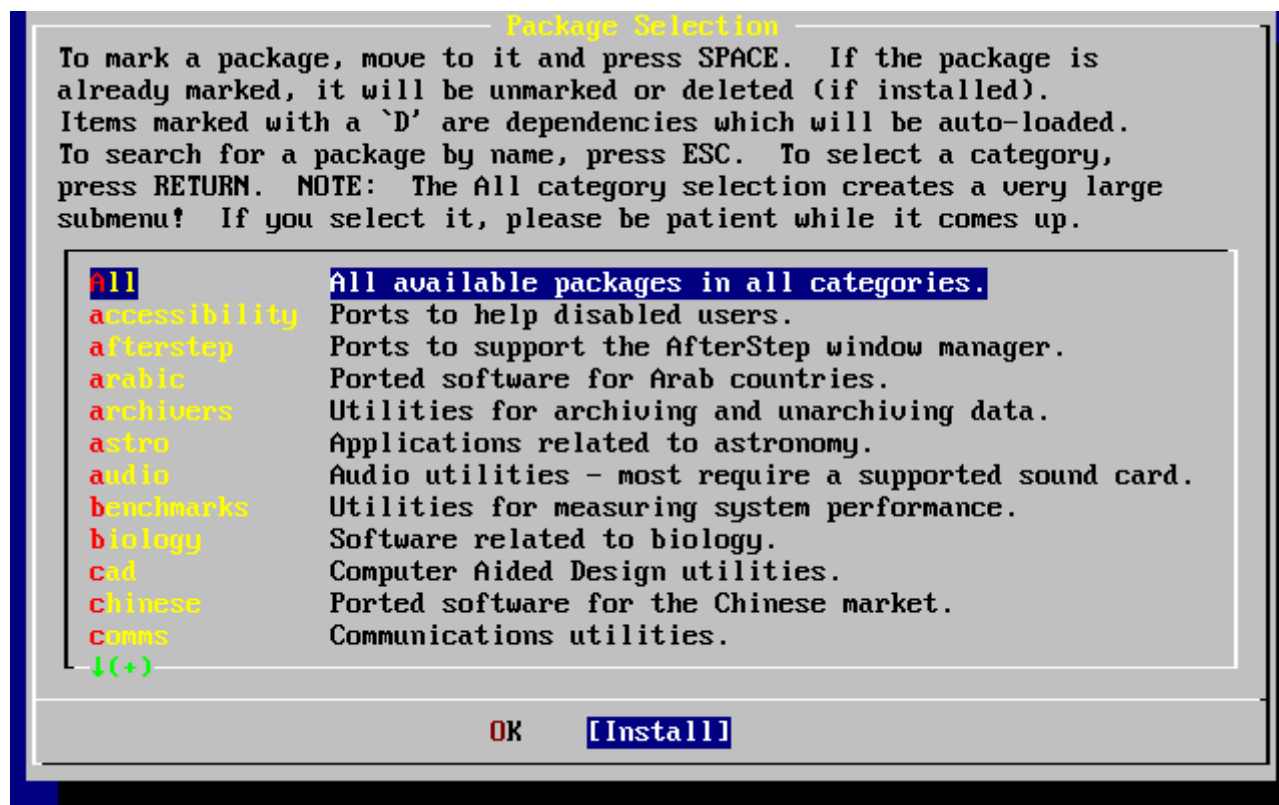
□ 48. □□ Package

如图所示，我进入了 bash shell。可以根据需要使用 `Space` 来勾选定的 package。在屏幕左下角会显示出 package 的短明。

反按下 `Tab`，可以在最后中的 package、`[OK]` 和 `[Cancel]` 之间来回切换。

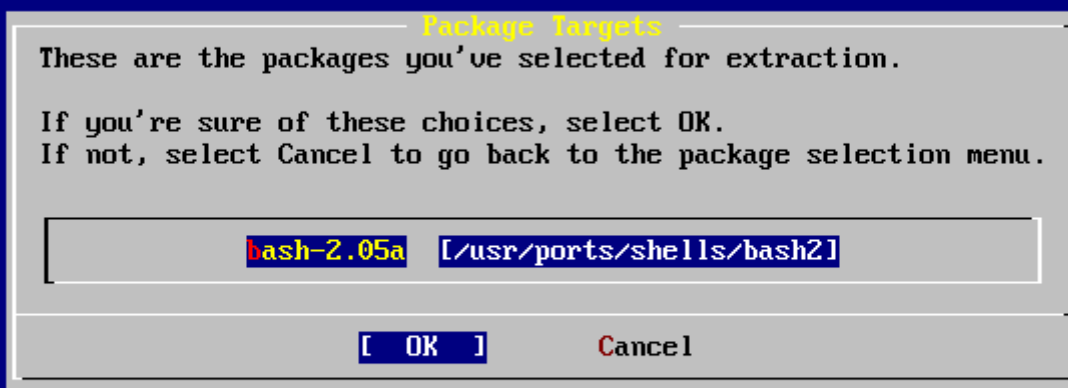
当把需要的 package 都安装之后，按一下 `Tab` 切换到 `[OK]`，随后按下 `Enter` 就可以回到 package 菜单了。

左右方向键可以用于在 `[OK]` 和 `[Cancel]` 之间切换。这个方法也可以用来 `[OK]`，随后按下 `Enter` 也可以回到 package 菜单。



#### 49. 安装软件包

使用 `Tab` 和左右方向键 `[Install]` 并按 `Enter`。接下来需要将要安装的包：



## The GNU Bourne Again Shell

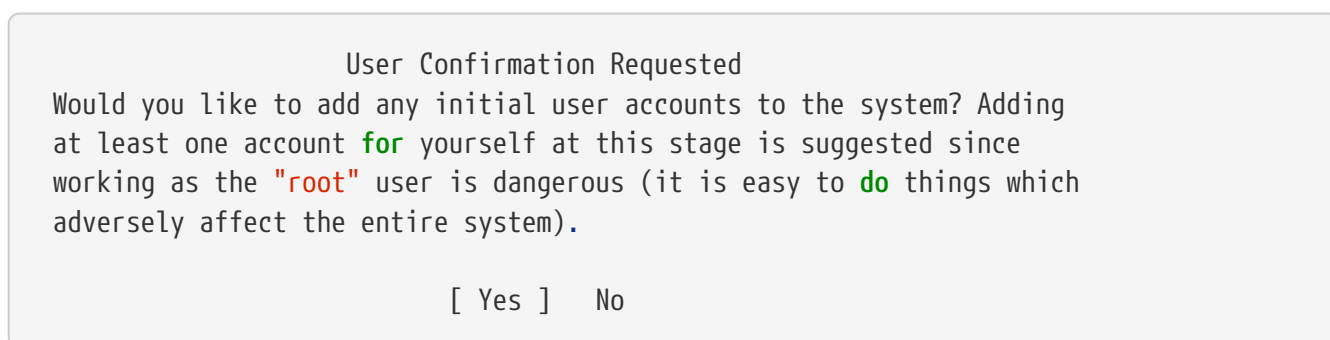
□ 50. □□将要安装的□□□包

□□ **[OK]** 并按下  就可以□始□□□包的安装了。在□个□程中□会看到安装的相□信息，直到安装完成□止。□留意□察是否有□□信息出□。

在完成□□□包的安装之后，就□入了最后的配置□段。如果□没有□□任何□□□包，并希望直接□入最后的配置□段，□可以□□ **[Install]** 来跳□。

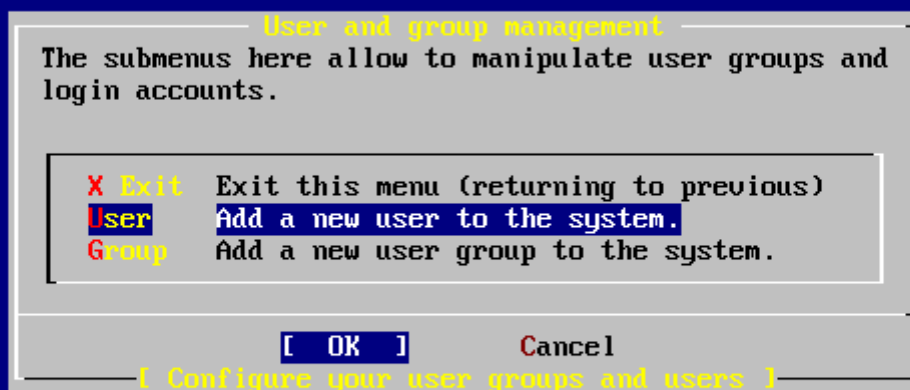
### 2.10.12. 添加用□和□

在安装系□的□程中，□□添加至少一个用□，□□以避免直接以 **root** 用□的身□登□。□□用以保存其用□数据的根分区通常很小，□□因此用 **root** 身□□□□程序可能将其迅速填□。□□下面的提示信息介□了□□做可能□□来的更大□□患：



□□ **[yes]** 并按  即可□始□建用□的□程。





□ 51. □□用□

用箭□□来□□ User 然后按 。

□ 52. 添加用□信息

下面的描述信息会出□在屏幕的下方，可以使用  □来切□不同的□目，以便□入相□信息：

### Login ID

新用□的登□名（□制性必□写）

## UID

□个用□的ID□号（如果不写，系□自□添加）

## Group

□个用□的登□名（如果不写，系□自□添加）

## Password

□个用□的密□（□入□个需要很仔□！）

## Full name

用□的全名（解□、□注）

## Member groups

□个用□所在的□

## Home directory

用□的主目□（如果不写，系□自□添加）

## Login shell

用□登□的shell（默□是/bin/sh）。

□可以将登□ shell 由 /bin/sh 改□ /usr/local/bin/bash，以便使用事先以 package 形式安装的 bash shell。不要使用一个不存在的或□不能登□的shell。最通用的shell是使用 BSD-world 的 C shell，可以通过□指定/bin/tcsh来修改。

用□也可以被添加到 wheel □中成了一个超□用□，从而□有 root □限。

当□感□□意□，□入【OK】□，用□和□管理菜□将会重新出□。

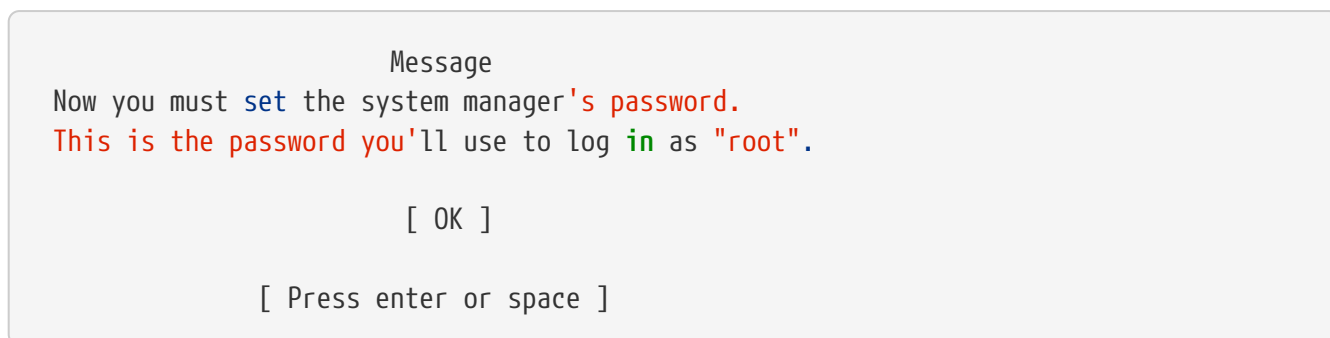


### □ 53. 退出用□和□管理

如果有其他的需要，此□□可以添加其他的□。此外，□可以通过 `sysinstall` 在安装完成之后添加它□。

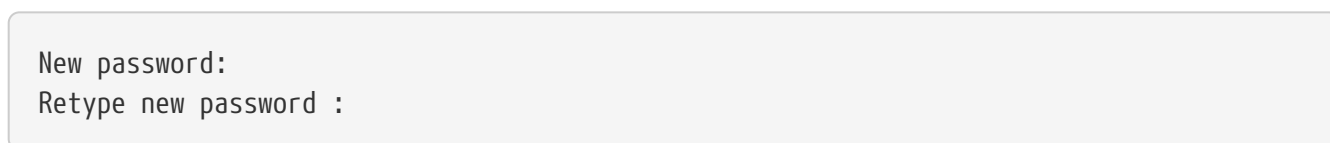
当□完成添加用□的□候，□□Exit 然后□入 `Enter` □□下面的安装。

### 2.10.13. □置 `root` 密□



□入 `Enter` 来□置 `root` 密□。

密□必□正□地□入□次。 毋庸□言， □需要□□一个不容易忘□的口令。 □注意□□入的口令不会回□， □也不会□示星号。



密□成功□入后，安装将□□。

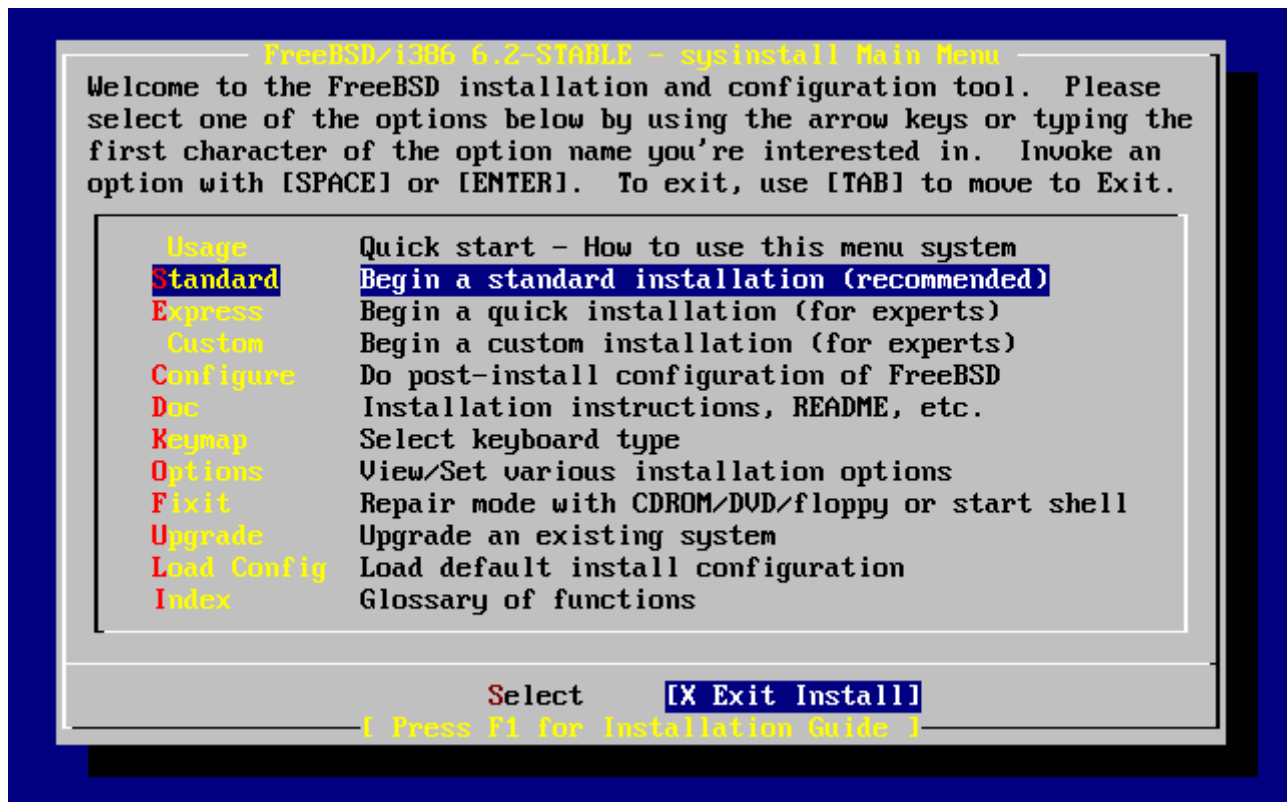
## 2.10.14. 退出安装

如果不需要置 [其他网](#)，或需要完成其他的配置工作， 可以在此或者事后通 `sysinstall` 来行配置。

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

```
Yes  [ No ]
```

按 **[no]** 然后入 `Enter` 返回到主安装菜。



54. 退出安装

按 **[X Exit Install]** 然后入 `Enter`。可能需要是否真的退出安装：

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

```
[ Yes ]  No
```

按 **[yes]**。如果是从 CDROM 引的系， 会出下面的提示信息要求取出光：

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

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

在系开始重之前，CDROM 器是住的。CDROM 解后就可以取出光了（作要快）。按 **[OK]** 重系。

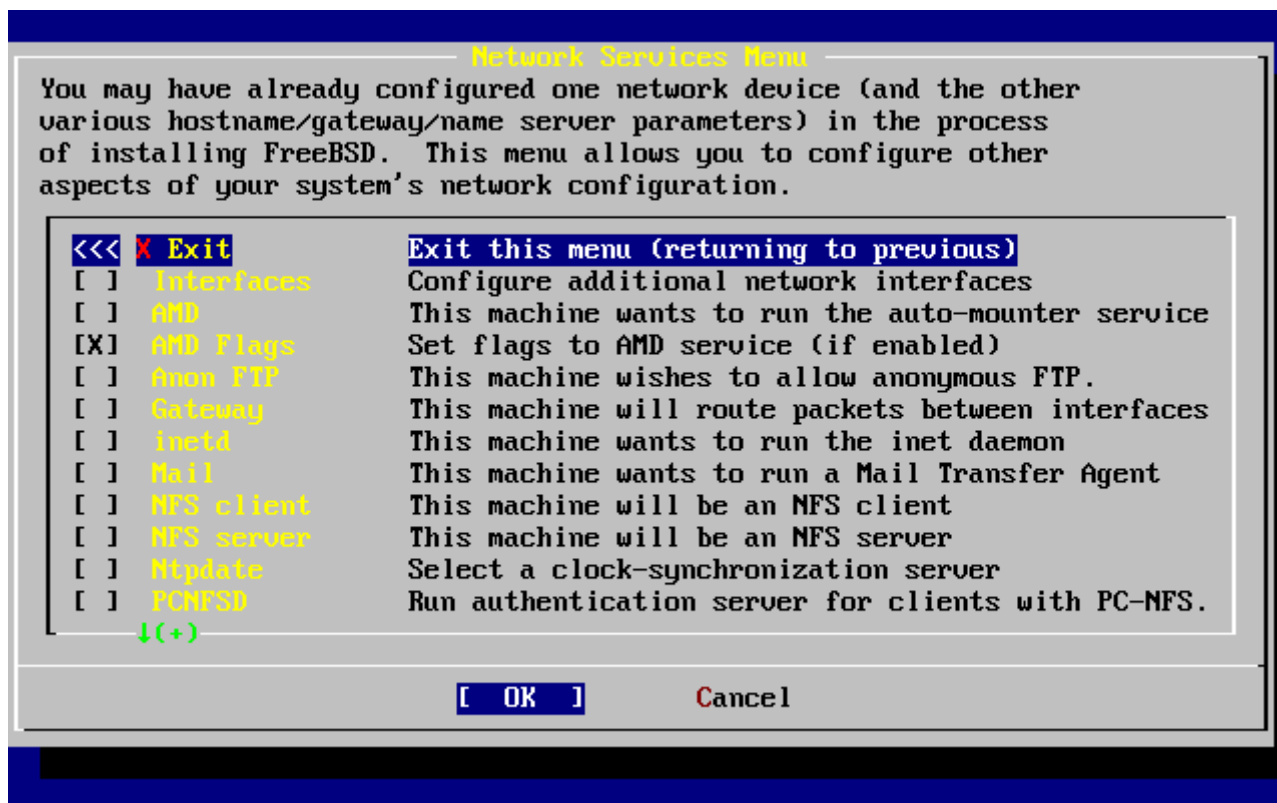
此后系将重新，因此留意是否会出一些信息。一的，参 [FreeBSD 的程](#)。

### 2.10.15. 配置其他网服

如果之前缺少一的，那配置网服于新手而言，很可能会是一件很有挑的事情。网，包括 Internet，于包括 FreeBSD 在内的所有代操作系统而言都至重要。因此，首先 FreeBSD 提供的富的网性能加以了解会很有助。在安装程中了解些知，能保用更好地理解他可以用到的各服。

网服是一些可以接收来自网上任何地方的人所提交的入信息的程序。人一直都在努力保些程序不会做任何“有害的”事情。不幸的是，程序并不是十全十美的完人，因此，网服程序中的漏洞，便有可能被攻者利用来做一些坏事。因而，只用那些知道自己需要的服就很重要了。如果存在疑，那就最好不要在需要它之前任何网服。可以事后通再次行 sysinstall 或直接手工配置 /etc/rc.conf 来随用些服。

Networking 将下示一个似下面的菜：



55. 网配置之上配置

第一个，Interfaces，已在前面的 [配置网](#) 中做配置，因此在可以略它。

AMD 将添加于 BSD 自挂接程序的支持。 个程序通常会和 NFS (详情参下文) 配合使用, 以便自挂程文件系。 用它不需要在此行特殊的 外配置。

下一行是 AMD Flags 的参数。 它之后, 会出一个 AMD 参数的子菜。 菜中包含一系列的:

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

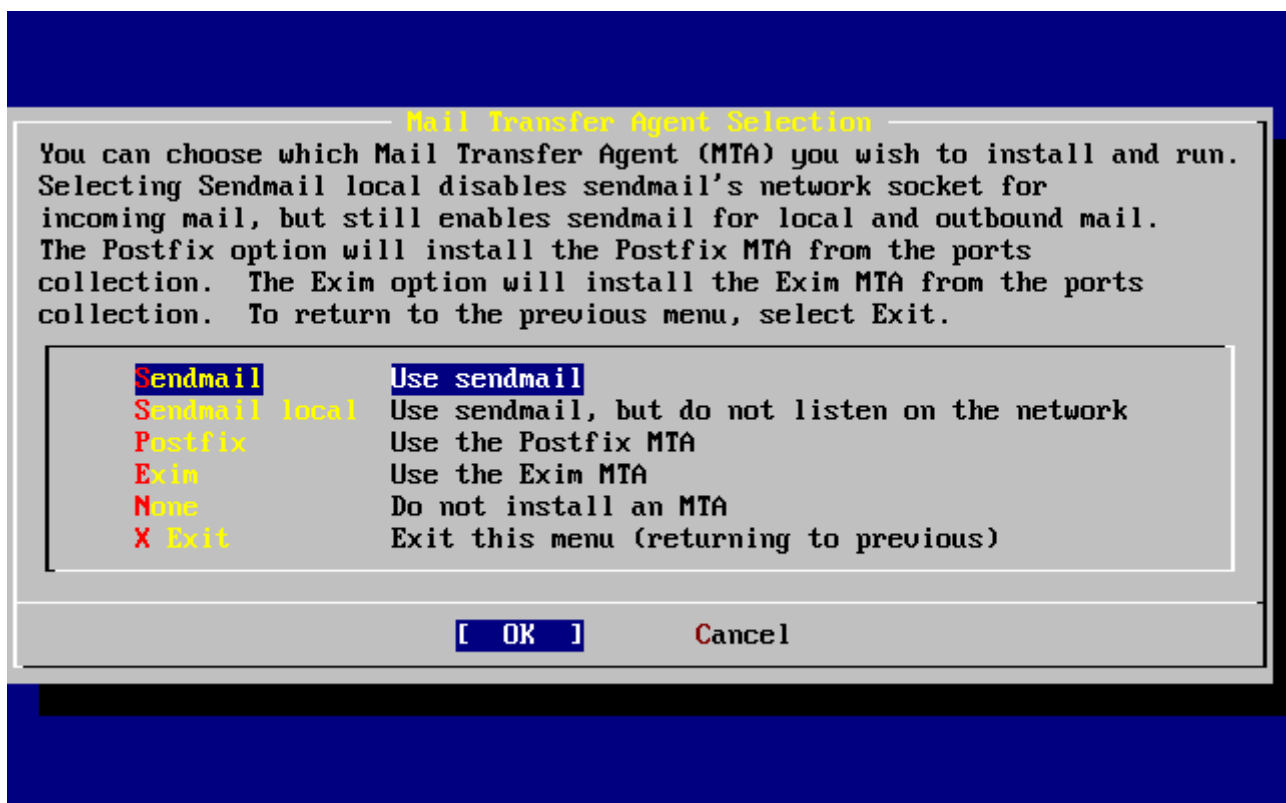
-a 用来置默的挂接位置, 里使用的是 /.amd\_mnt目。 -l 指定默的 日志 文件; 但是, 当使用 syslogd , 所有在日志中 的活, 都会 送到系 日志服 去。 /host 用来挂接 程主机上 出的文件系, 而 /net 目 用来挂接从特定 IP 地址 出的文件系。 /etc/amd.map 文件定 了用于 AMD 的默 出。

Anon FTP 允 匿名 FTP 。 中 个 , 可以使 台机器成 一台匿名 FTP 服 器。 要注意 用 个 的安全 。 系 将使用 外的菜 来 明安全 和 一的配置。

Gateway 可以使将本机配置成 一台以前我 介 的网 。 如果 在安装 程中不小心 中了 Gateway, 也可以在 里用 个 来取消。

Inetd 用来配置或完全禁用前面 的 inetd(8) 服 程序。

Mail 用来配置系 默的 MTA 或 件 代理。 个 将出 下面的菜:



## 56. 默的 MTA

里 提供了一个安装MTA 并将其配置 默 的机会。MTA 是一 能 将 件 本系 或互 网上的用 的 件服 。

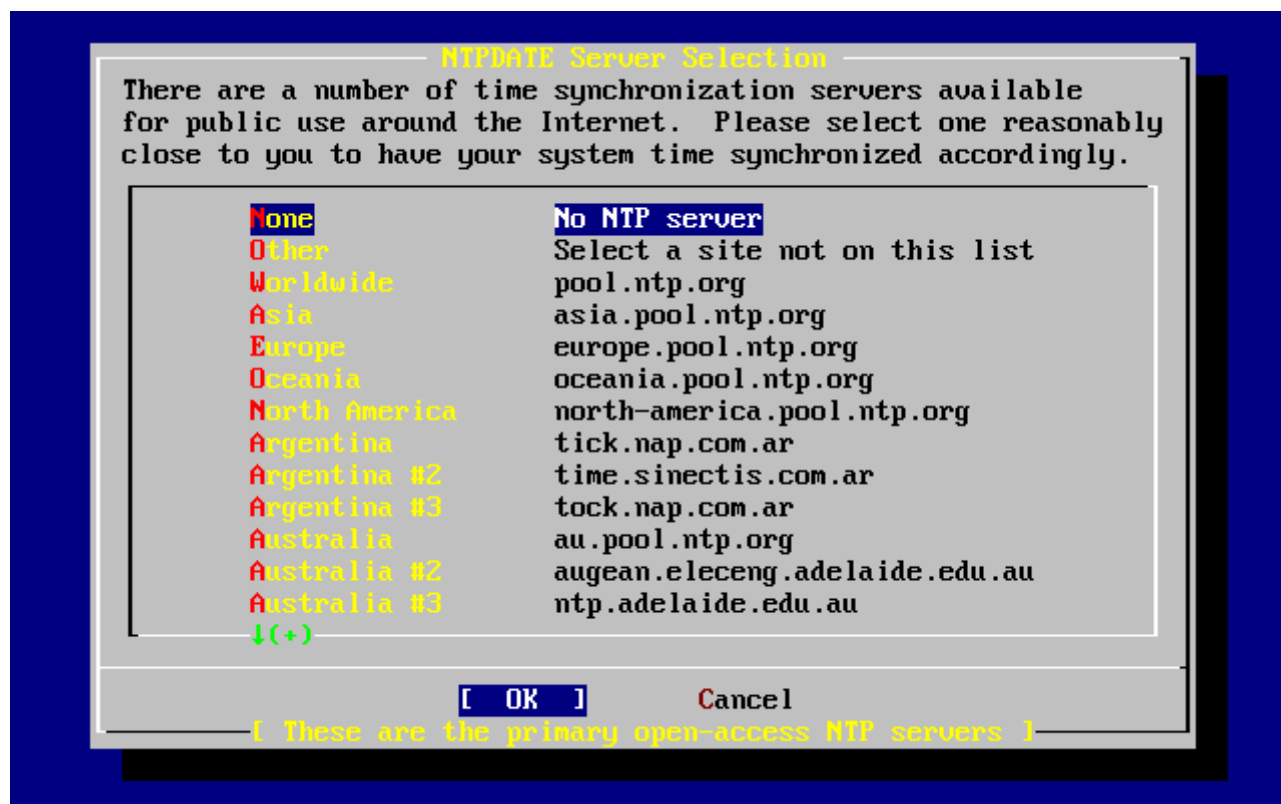
Sendmail 将会安装十分流行的 sendmail 服, 也是 FreeBSD 的默配置。Sendmail local 表示将 sendmail 默的 MTA, 但禁止其从 Internet 上接收 件的能力。此外 有一些其他 , Postfix 和 Exim 与 Sendmail 的功能 似。它 者也可以投 件; 不, 有些用 会喜 使用它 代替 sendmailMTA。

☐ MTA 或决定不挑☐ MTA 之后，网☐配置菜☐的下一☐将是 NFS client。

NFS client 客☐端可以使系☐通☐ NFS 与服☐器☐行通信。NFS 服☐器通☐ NFS ☐☐可以使其它在网☐上的机器来☐自己的文件系☐。如果☐台机器要作☐一台独立的服☐器，☐个☐☐可以保留不☐。如果☐用它，☐在之后☐需要☐行更多的其他配置；☐参☐ [网☐文件系☐ \(NFS\)](#) 以了解☐于配置客☐机和服☐器的☐一☐情。

接下来的 NFS server ☐☐，可以☐☐将本机系☐配置☐ NFS 服☐器。☐会自☐将☐☐ RPC ☐程☐程☐用的信息写入配置文件。RPC 是一☐在多个主机和程序之☐☐行☐接☐☐的机制。

下一☐是 Ntpdate ☐☐，它能☐☐理☐☐同☐。当☐☐它后，会出☐一个像下面所似的菜☐：



## ☐ 57. Ntpdate 配置

从☐个菜☐☐☐一个☐☐最近的服☐器。☐☐☐近的服☐器，有助于提高☐☐同☐的精度，因☐☐☐的服☐器的☐接延☐可能会比☐大。

下一个☐☐是 PCNFS.D。☐个☐☐将安装第三方☐件包 [net/pcnfsd](#)。它可以用来☐无法自行提供 NFS ☐☐服☐的操作系☐，如微☐的 MS-DOS® 提供服☐。

☐屏到下一☐看一下其它☐☐：





### 2.10.16.1. FreeBSD/i386 的启动过程

如果一切正常，你将看到在屏幕上有很多信息，最后你会看到登录命令行。可以通过按下 `Scroll-Lock` 和使用 `PgUp` 与 `PgDn` 来看信息，再按下 `Scroll-Lock` 回到命令行。

某些信息可能不会显示（缓冲区限制）。可以通过按下 `dmesg` 来看。

使用你在安装过程中设置的用户名/密码来登录。（例子中使用 `rpratt`）。除非必要的时候不要用 `root` 用户登录。

典型的启动信息：（忽略版本信息）

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

Timecounter "i8254" frequency 1193182 Hz
CPU: AMD-K6(tm) 3D processor (300.68-MHz 586-class CPU)
  Origin = "AuthenticAMD" Id = 0x580 Stepping = 0
  Features=0x8001bf<FPU,VME,DE,PSE,TSC,MSR,MCE,CX8,MMX>
  AMD Features=0x80000800<SYSCALL,3DNow!>
real memory = 268435456 (262144K bytes)
config> di sn0
config> di lnc0
config> di le0
config> di ie0
config> di fe0
config> di cs0
config> di bt0
config> di aic0
config> di aha0
config> di adv0
config> q
avail memory = 256311296 (250304K bytes)
Preloaded elf kernel "kernel" at 0xc0491000.
Preloaded userconfig_script "/boot/kernel.conf" at 0xc049109c.
md0: Malloc disk
Using $PIR table, 4 entries at 0xc00fde60
npx0: <math processor> on motherboard
npx0: INT 16 interface
pcib0: <Host to PCI bridge> on motherboard
pci0: <PCI bus> on pcib0
pcib1: <VIA 82C598MVP (Apollo MVP3) PCI-PCI (AGP) bridge> at device 1.0 on pci0
pci1: <PCI bus> on pcib1
pci1: <Matrox MGA G200 AGP graphics accelerator> at 0.0 irq 11
isab0: <VIA 82C586 PCI-ISA bridge> at device 7.0 on pci0
isa0: <ISA bus> on isab0
atapci0: <VIA 82C586 ATA33 controller> port 0xe000-0xe00f at device 7.1 on pci0
ata0: at 0x1f0 irq 14 on atapci0
ata1: at 0x170 irq 15 on atapci0
uhci0: <VIA 83C572 USB controller> port 0xe400-0xe41f irq 10 at device 7.2 on pci0
usb0: <VIA 83C572 USB controller> on uhci0
```

```

usb0: USB revision 1.0
uhub0: VIA UHCI root hub, class 9/0, rev 1.00/1.00, addr 1
uhub0: 2 ports with 2 removable, self powered
chip1: <VIA 82C586B ACPI interface> at device 7.3 on pci0
ed0: <NE2000 PCI Ethernet (RealTek 8029)> port 0xe800-0xe81f irq 9 at
device 10.0 on pci0
ed0: address 52:54:05:de:73:1b, type NE2000 (16 bit)
isa0: too many dependant configs (8)
isa0: unexpected small tag 14
fdc0: <NEC 72065B or clone> at port 0x3f0-0x3f5,0x3f7 irq 6 drq 2 on isa0
fdc0: FIFO enabled, 8 bytes threshold
fd0: <1440-KB 3.5" drive> on fdc0 drive 0
atkbdc0: <keyboard controller (i8042)> at port 0x60-0x64 on isa0
atkbd0: <AT Keyboard> flags 0x1 irq 1 on atkbdc0
kbd0 at atkbd0
psm0: <PS/2 Mouse> irq 12 on atkbdc0
psm0: model Generic PS/2 mouse, device ID 0
vga0: <Generic ISA VGA> at port 0x3c0-0x3df iomem 0xa0000-0xbffff on isa0
sc0: <System console> at flags 0x1 on isa0
sc0: VGA <16 virtual consoles, flags=0x300>
sio0 at port 0x3f8-0x3ff irq 4 flags 0x10 on isa0
sio0: type 16550A
sio1 at port 0x2f8-0x2ff irq 3 on isa0
sio1: type 16550A
ppc0: <Parallel port> at port 0x378-0x37f irq 7 on isa0
ppc0: SMC-like chipset (ECP/EPP/PS2/NIBBLE) in COMPATIBLE mode
ppc0: FIFO with 16/16/15 bytes threshold
ppbus0: IEEE1284 device found /NIBBLE
Probing for PnP devices on ppbus0:
plip0: <PLIP network interface> on ppbus0
lpt0: <Printer> on ppbus0
lpt0: Interrupt-driven port
ppi0: <Parallel I/O> on ppbus0
ad0: 8063MB <IBM-DHEA-38451> [16383/16/63] at ata0-master using UDMA33
ad2: 8063MB <IBM-DHEA-38451> [16383/16/63] at ata1-master using UDMA33
acd0: CDR0M <DELTA OTC-H101/ST3 F/W by OIPD> at ata0-slave using PIO4
Mounting root from ufs:/dev/ad0s1a
swapon: adding /dev/ad0s1b as swap device
Automatic boot in progress...
/dev/ad0s1a: FILESYSTEM CLEAN; SKIPPING CHECKS
/dev/ad0s1a: clean, 48752 free (552 frags, 6025 blocks, 0.9% fragmentation)
/dev/ad0s1f: FILESYSTEM CLEAN; SKIPPING CHECKS
/dev/ad0s1f: clean, 128997 free (21 frags, 16122 blocks, 0.0% fragmentation)
/dev/ad0s1g: FILESYSTEM CLEAN; SKIPPING CHECKS
/dev/ad0s1g: clean, 3036299 free (43175 frags, 374073 blocks, 1.3% fragmentation)
/dev/ad0s1e: filesystem CLEAN; SKIPPING CHECKS
/dev/ad0s1e: clean, 128193 free (17 frags, 16022 blocks, 0.0% fragmentation)
Doing initial network setup: hostname.
ed0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    inet 192.168.0.1 netmask 0xffffffff broadcast 192.168.0.255
    inet6 fe80::5054::5ff::fede:731b%ed0 prefixlen 64 tentative scopeid 0x1

```

```

ether 52:54:05:de:73:1b
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
    inet6 fe80::1%lo0 prefixlen 64 scopeid 0x8
    inet6 ::1 prefixlen 128
    inet 127.0.0.1 netmask 0xff000000
Additional routing options: IP gateway=YES TCP keepalive=YES
routing daemons:.
additional daemons: syslogd.
Doing additional network setup:.
Starting final network daemons: creating ssh RSA host key
Generating public/private rsa1 key pair.
Your identification has been saved in /etc/ssh/ssh_host_key.
Your public key has been saved in /etc/ssh/ssh_host_key.pub.
The key fingerprint is:
cd:76:89:16:69:0e:d0:6e:f8:66:d0:07:26:3c:7e:2d root@k6-2.example.com
creating ssh DSA host key
Generating public/private dsa key pair.
Your identification has been saved in /etc/ssh/ssh_host_dsa_key.
Your public key has been saved in /etc/ssh/ssh_host_dsa_key.pub.
The key fingerprint is:
f9:a1:a9:47:c4:ad:f9:8d:52:b8:b8:ff:8c:ad:2d:e6 root@k6-2.example.com.
setting ELF ldconfig path: /usr/lib /usr/lib/compat /usr/X11R6/lib
/usr/local/lib
a.out ldconfig path: /usr/lib/aout /usr/lib/compat/aout /usr/X11R6/lib/aout
starting standard daemons: inetd cron sshd usbd sendmail.
Initial rc.i386 initialization:.
rc.i386 configuring syscons: blank_time screensaver moused.
Additional ABI support: linux.
Local package initialization:.
Additional TCP options:.

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

login: rpratt
Password:

```

生成 RSA 和 DSA 密钥在比它慢的机器上可能要花很长时间。它只是一个新安装后的首次操作，以后的操作会更快一点。

如果已完成 X 服务器的配置，且指定了默认的面板窗口管理器，就可以在命令行输入 `startx` 来启动它了。

## 2.10.17. FreeBSD 关机

正确的操作系统是很重要的。不要胡乱猜测。首先，你需要成为一个超级用户，通常输入 `su` 命令来做到。然后输入 `root` 密码。你需要用 `wheel` 组的一名成员。然后，以 `root` 输入 `shutdown -h now` 命令。

```

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

```

当shutdown命令执行后，屏幕上出现 "Please press any key to reboot" 信息，您就可以安全的关机了。如果按下任意一个键，计算机将重新启动。

您也可以使用 `Ctrl + Alt + Del` 组合键来重新启动计算机，但是不推荐使用这个操作。

## 2.11. 常见问题

下面将介绍一些在安装过程中常见的问题，像如何报告生成的问题，如何双重引导 FreeBSD 和 MS-DOS® 或 Windows®。

### 2.11.1. 当您遇到问题时，应该做什么？

由于 PC 硬件的限制，硬件不可能 100% 地可靠，但是有些问题是您可以自己解决的。

首先看一下您使用的 FreeBSD 版本的 [硬件兼容说明](#) 文看看您使用的是否是被支持的硬件。

如果您使用的硬件是系统支持的，但仍然遇到了死机或其他问题，您可能需要 [定制的内核](#)。系统支持默认的 GENERIC 内核所不支持的问题。在引导上的内核假定大多数的硬件，均按出厂设置的方式配置了 IRQ、IO 地址和 DMA 通道。如果您的硬件重新进行了配置，您可能需要内核配置，并重新编译内核，以便报告 FreeBSD 到那里去。

除此之外，也可能遇到某些情况，即探测某并不存在的东西，会干扰到其他东西的东西并使其失效。某些情况下禁止某些程序可能致冲突的。



有些安装问题可以借助更新硬件的程序来解决，特别是主板的 BIOS。大部分的主板制造商都会提供网站用于下载新的 BIOS 以及提供如何更新的说明。

也有许多制造商强烈建议，除非必要否则不要轻易更新 BIOS。因为更新的程序可能会产生问题，从而损害 BIOS 芯片。

### 2.11.2. 使用 MS-DOS® 和 Windows® 文件系统

目前，FreeBSD 尚不支持通用 Double Space™ 程序的文件系统。因此，如果希望 FreeBSD 管理数据，您首先解某些文件系统。这项工作，可以通过位于 Start> Programs > System Tools 菜单的 Compression Agent 来完成。

FreeBSD 可以支持基于 MS-DOS® 的文件系统（有时被称为 FAT 文件系统）。`mount_msdosfs(8)` 命令能把您的文件系统挂载到已有的目录中，并允许访问 FAT 文件系统上的内容。通常我们并不直接使用 `mount_msdosfs(8)` 程序，它一般会在 `/etc/fstab` 中的某一行被引用或者被 `mount(8)` 工具并配合适当的参数来引用。

`/etc/fstab` 中一个典型的例子：

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



`/dos` 目录必须事先存在。更多关于 `/etc/fstab` 的内容，请参看 [fstab\(5\)](#)。

一个使用 `mount(8)` 挂载 MS-DOS® 文件系统的例子：

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

在此例子中，MS-DOS® 文件系统位于主硬盘的第一个分区。的情况可能与引不同，看命令 `dmesg` 和 `mount` 的输出。它可能可以得到足够的分区信息。



FreeBSD 可能使用和其他操作系统不同计数方法来磁 slices，特别需要指出的是，MS-DOS® 的扩展分区通常会比 MS-DOS® 主分区被更高的数。可以使用 `fdisk(8)` 工具来帮助确定些 slices 属于 FreeBSD 些是属于其他的操作系统。

NTFS 分区也可以通过似 `mount_ntfs(8)` 命令挂载在 FreeBSD 上。

### 2.11.3. 排除故障的常见问题和解决方法

#### 2.11.3.1. 我的系统在引导到探测硬件时生了死机、安装过程中行异常，或没有看到。

FreeBSD 在过程中广泛使用了 i386、amd64 及 ia64 平台提供的 ACPI 服务来配置。不幸的是，在 ACPI 和主板 BIOS 中存在一些 bug。如果遇到情况，可以在系统引导禁用 ACPI，其方法是在第三段引导加载器使用 hint `hint.acpi.0.disabled`：

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

配置会在系统重启之后失效，因此，如果需要的，请在 `/boot/loader.conf` 文件中添加 `hint.acpi.0.disabled="1"`。于引导加载器的配置情况，参 [概述](#)。

#### 2.11.3.2. 在硬盘安装 FreeBSD 之后的首次启动，内核加载并识别了硬件，但出现下列消息并停止运行：

系统在引导非系统中的第一启动有一个由来已久的。BIOS 采用的编号方式有和 FreeBSD 不一致，而方法将其一致很正确地。

因而，在生情况，FreeBSD 可能会需要一些帮助才能到磁。有常的情况，在些情况下都需要手工告诉 FreeBSD 根文件系统模的位置。是通过告诉引导加载器 BIOS 磁号、磁型以及 FreeBSD 中的磁的号来的。

第一情况是有 IDE 硬盘，分区配置 IDE 上的主 (master) 盘，并希望 FreeBSD 从第二硬盘上。BIOS 将硬盘磁 0 和磁 1，而 FreeBSD 将其分叫做 ad0 和 ad2。

FreeBSD 位于 BIOS 磁 1，其型是 `ad` 而 FreeBSD 磁号是 2，因此，输入：

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

注意，如果的主盘上有从盘，配置是不必要的 (因配置是的)。

第二情况是从 SCSI 磁，但系统中安装了一个或多个 IDE 硬盘。时，FreeBSD 磁号会比 BIOS 磁号小。如果有 IDE 硬盘，以及一 SCSI 硬盘，SCSI 硬盘将会是 BIOS 磁 2，型 `da` 而 FreeBSD 磁号是 0，因此，输入：

来告诉 FreeBSD 希望从 BIOS 磁头 2 引导，而它是系统中的第一块 SCSI 硬盘。假如只有一块 IDE 硬盘，可以以 1: 代替。

一旦确定了要用的正确配置，就可以用标准的文本编辑器把它写到 `/boot.config` 文件中了。除非另行指定，FreeBSD 将使用该文件的内容，作为 `boot:` 提示的默认回答。

**2.11.3.3. 在硬盘安装 FreeBSD 之后的首次引导，Boot Manager 只是显示了 F? 的菜单位提示，但并不引导过程。**

在安装 FreeBSD 时行到分区编辑器所设置的磁头尺寸信息不。回到分区编辑器并指定正确的磁头尺寸。该情况必须重新安装 FreeBSD。

如果无法确定在机器上的正确尺寸信息，可以用一个小技巧：在磁头起始的地方安装一个小的 DOS 分区，并在其后安装 FreeBSD。安装程序能看到该 DOS 分区，并利用它推算磁头的尺寸信息，通常会有所帮助。

下面的技巧不再推荐使用，在里面仅供参考：

如果正准备建立只运行 FreeBSD 的服务器或工作站，而无需考虑（之后）与 DOS、Linux 或其他操作系统的兼容性，也可以使用整个硬盘（分区编辑器中的 A），让 FreeBSD 独占整个硬盘的一个扇区的非标准。这会忽略磁头尺寸的一切，但会限制以后运行 FreeBSD 以外的其他操作系统的功能。

**2.11.3.4. 系统到了 `ed(4)` 网络，但是超时 (device timeout) 。**

网络可能使用了与 `/boot/device.hints` 文件中指定的 IRQ 不同的中断请求号。`ed(4)` 默认情况下并不支持 "0" 配置（在 DOS 中使用 EZSETUP 配置的），但如果网络 hints 中指定 `-1`，便会使用该配置。

使用网络的跳行硬配置（根据需要修改内核配置）或通过 `hint hint.ed.0.irq="-1"` 将 IRQ 指定为 `-1`。这会告诉内核使用该配置。

一个可能是网络使用 IRQ 9，会与 IRQ 2 共用同一中断请求，同时也是导致的一个常见原因（特别是 VGA 使用 IRQ 2 的时候！）。尽量避免使用 IRQ 2 或 9。

**2.11.3.5. 当在 X11 终端中进行 `sysinstall` 的时候，黄色的字体相对于浅灰色的背景难以阅读。有没有什么能让这个应用程序提供高对比度的方法？ `colorcontrast`**

如果已经安装了 X11 并且 `sysinstall` 在 `xterm(1)` 或者 `rxvt(1)` 中默认的颜色使得文字难以阅读，可以在 `~/.Xdefaults` 中加入 `XTerm*color7: #c0c0c0` 获得深灰色的背景。

## 2.12. 高级安装指南

本章主要描述在一些特殊情况下如何安装 FreeBSD。

### 2.12.1. 在一个没有显示器或鼠标的系统上安装 FreeBSD

这种类型的安装叫做 "headless install（无头安装）"，因为正要安装 FreeBSD 的机器不是没有显示器，就是没有鼠标。可能会问那怎么安装？可以使用一个串行控制台。串行控制台基本上是使用另一台机器来充当主显示

和。要做，只要行下面的： 建安装 USB 棒，看 [准引介](#)—明；此外， 也可下 ISO 映像文件， 具体参 [建—安装光](#)。

要将安装介改使用串口控制台， 需要按下面些来操作 (如果使用 CDROM 可跳第一)：

## 1. 安装 USB 引导并进入串口控制台

如果使用制作的 USB 引导系统，FreeBSD 会进入正常的安装模式。我希望引导到串口控制台来完成安装。为了做到这一点，需要在 FreeBSD 中使用 [mount\(8\)](#) 挂载 USB。

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



需要根据情况修改挂点的名称。

在挂好了 USB 棒，需要对其行配置令其进入串口控制台。此，需要在 USB 棒中的 loader.conf 文件中加入下面的行配置：

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

就完成了 USB 棒的配置，使用 [umount\(8\)](#) 命令将其卸下：

```
# umount /mnt
```

在就可以拔下 USB 棒并进入教程的第三步了。

## 2. 安装 CD 引导并进入串口控制台

如果直接使用 ISO 映像 (see [构建安装光碟](#)) 制作的 CD 引导，FreeBSD 会引导进入正常的安装模式。我希望引导到串口控制台来完成安装。为了做到这一点，需要展示、修改并重新生成 ISO 文件，然后再刻光盘。

在保存例如 FreeBSD-8.1-RELEASE-i386-disc1.iso ISO 的 FreeBSD 系上用 [tar\(1\)](#) 工具提取全部文件：

```
# mkdir /path/to/headless-iso
# tar -C /path/to/headless-iso -pxvf FreeBSD-8.1-RELEASE-i386-disc1.iso
```

接下来需要对其行配置令其进入串口控制台。此，需要在从 ISO 映像中提取的 loader.conf 文件中加入下面的行配置：

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

最后，从修改好的目录中构建新的 ISO 映像。这里我使用通过 [sysutils/cdrtools](#) port 安装的 [mkisofs\(8\)](#) 工具来完成：

```
# mkisofs -v -b boot/cdboot -no-emul-boot -r -J -V "Headless_install" \
-o Headless-FreeBSD-8.1-RELEASE-i386-disc1.iso /path/to/headless-iso
```



☐☐就完成了☐ ISO 映像的配置，☐可以使用☐熟悉的工具将其刻☐到 CD-R 上了。

### 3. ☐接 Null-modem ☐

☐在需要一根 [null-modem ☐](#) 来☐接☐台机器。只要☐接☐台机器的串口。☐里不能使用普通的串口☐，而必☐使用 null-modem ☐，因☐它需要一些内部交叉的☐☐。

### 4. ☐始☐☐安装

☐在可以☐始安装了。将 USB ☐棒☐到☐准☐☐行 headless 安装的机器上，然后☐机。如果☐使用的是 CDROM，☐在☐机之后立即将光☐放☐光☐。

### 5. ☐接☐的无☐机器

☐在☐已☐通☐[cu\(1\)](#)☐接到了那台机器。

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

在 FreeBSD 7.X 上☐使用下面的命令：

```
# cu -l /dev/cuad0
```

☐☐就可以了！☐☐在可以通☐ [cu](#) 会☐来控制那台 headless 的机器了。接着系☐会提示☐☐☐端☐型。☐☐ FreeBSD 彩色控制台并☐☐安装！

## 2.13. 准☐☐自己的安装介☐



☐了避免重☐ "FreeBSD disc" 在☐里指 FreeBSD CDROM or DVD 那即意味着☐要☐☐或自己制做。

有好几个原因需要☐☐建自己的FreeBSD安装介☐。☐可能是物理介☐，如磁☐，使用 `sysinstall` 程序☐到的安装文件，FTP 站点或 MS-DOS®分区。

例如：

- ☐有☐多机器☐接到本地网☐，使用一个FreeBSD光☐。☐要使用FreeBSD来☐建一个本地FTP站点，然后使用☐个FTP站点来代替☐接到Internet。
- ☐有一☐ FreeBSD 光☐，FreeBSD 不支持☐的 CD/DVD ☐☐器，但 MS-DOS®/Windows® 支持。☐要☐制安装文件到一个DOS分区，然后使用☐些文件☐行安装。
- ☐要安装的☐算机没有 CD/DVD☐☐器和网☐，但☐可以☐接一个 "Laplink-style" 串口或并☐☐☐到那台☐算机。
- ☐要通☐一个磁☐机来安装FreeBSD。

### 2.13.1. ☐建一☐安装光☐

FreeBSD 的☐个☐行版本都☐☐一支持的平台提供至少☐☐ CDROM 映像 ("ISO images")。如果☐有刻☐机，

一些映像文件可以被("burned") 成FreeBSD的安装光盘。如果没有刻盘机，而上网却很便宜，它也是一种很好的安装方式。

## 1. 下载正版的 ISO 映像文件

每个版本的 ISO 映像文件都可以从 <ftp://ftp.FreeBSD.org/pub/FreeBSD/ISO-IMAGES-架构名/版本> 或最近的镜像站点下载。符合的架构和版本。

目录中包含下面一些映像文件：

表 4. FreeBSD 7.X 和 8.X ISO 映像文件名和含义

| 文件名                                | 包含内容  |
|------------------------------------|---|
| FreeBSD-版本-RELEASE-架构-bootonly.iso | 一个 CD 映像可以刻录到光盘并输入安装程序，但它并不提供用于支持从 CD 直接安装 FreeBSD 所需的文件。在从 CD 引导之后，需要通过网络（例如从 FTP 服务器）来完成安装。               |
| FreeBSD-版本-RELEASE-架构-dvd1.iso.gz  | 一个 DVD 映像包括用于安装 FreeBSD 操作系统基本组件、软件包和文档所需的全部文件。它也支持引导基于 "livefs" 的修复模式。                                     |
| FreeBSD-版本-RELEASE-架构-memstick.img | 一个映像可以写入 USB 闪存棒，用于引导系统并完成安装。它也支持引导基于 "livefs" 的修复模式。一个版本的映像中包含了文档所需的全部文件，但不提供其他包。FreeBSD 7.3 和更早版本中没有这个文件。 |
| FreeBSD-版本-RELEASE-架构-disc1.iso    | 一个 CD 映像包含了 FreeBSD 操作系统的基本组件和文档包，但不包括其它包。  |
| FreeBSD-版本-RELEASE-架构-disc2.iso    | 一个 CD 映像包含了能填充光盘的尽可能多的第三方软件包。在 FreeBSD 8.0 和更高版本中不提供这个映像。   |
| FreeBSD-版本-RELEASE-架构-disc3.iso    | 一个包含了能填充光盘的尽可能多的第三方软件包的 CD 映像。在 FreeBSD 8.0 和更高版本中不提供这个映像。  |
| 版本-RELEASE-架构-docs.iso             | FreeBSD 文档。   |
| FreeBSD-版本-RELEASE-架构-livefs.iso   | 一个 CD 映像包含了用以支持引导基于 "livefs" 的修复模式，但不包括直接从 CD 安装所需的文件。  |



FreeBSD 7.X 系列在 FreeBSD 7.3 之前的版本，以及 FreeBSD 8.X 系列在 FreeBSD 8.1 之前的版本使用不同的命名。它们的 ISO 文件名不使用 **FreeBSD-** 前缀。

必须下载 **bootonly** ISO 映像（如果有）或 **disc1** 的映像其中的一个。没有必要都下载，因为 **disc1** 映像包含了 **bootonly** ISO 映像中的全部内容。

如果您的 Internet 连接很廉价，请使用 **bootonly** ISO。它能安装 FreeBSD，而您可以根据需要使用 ports/packages 系统来下载并安装第三方组件（参看 [安装程序: Packages 和 Ports](#)）。

如果打算安装 FreeBSD 并安装常用的软件包，请使用 **dvd1**。

其它的映像也很很有用，但不是必需的，尤其是在有高速的网络接口。

## 2. 刻录 CDs

必需把一些映像文件刻录成光盘。如果在其它的FreeBSD系统上完成此工作，[看建和使用光学介质\(CD\)](#)得到更多的信息，（特别是 [burncd](#) 和 [cdrecord](#)）

如果在其它的系统平台上运行，需要相应的刻录软件。映像文件使用的是标准的ISO格式，必需被相应的刻录软件所支持。



如果有兴趣制作一个定制的 FreeBSD 版本，[参考 Release Engineering Article](#)。

### 2.13.2. 安装并建立局域网 FTP 站点

FreeBSD 光驱的布局和 FTP 站点相同。因此，建立局域网 FTP 站点来用于网络上的其它计算机安装 FreeBSD，就十分的容易。

1. 在要作FTP站点的那台FreeBSD机器上，将FreeBSD磁碟放入光驱中并将它挂在 /cdrom 目录中。

```
# mount /cdrom
```

2. 在 /etc/passwd 文件中建立一个可匿名访问 FTP 服务器的帐号。可以利用 [vipw\(8\)](#) 命令编辑 /etc/passwd 文件，加入下面一行叙述：

```
ftp:*:99:99::0:0:FTP:/cdrom:/nonexistent
```

3. 确定在 /etc/inetd.conf 配置文件中开了FTP服务。

任何本地网络中的机器在安装 FreeBSD 时安装介绍就可以通过 FTP 站点，然后选择 "Other" 后输入 [ftp://本地FTP服务器](#) 即可以透本地的FTP站点来安装FreeBSD。



如果用作 FTP 客户端的引导介绍（通常是本地）与本地局域网的 FTP 站点上的版本不一致，sysinstall 会不允许完成安装。如果使用的版本差距不很大，并且希望一一判断，可进入 **Options** 菜单，并将安装包的名字改为 any。



此方式最好使用在有防火墙保护的内部网络。如果要将此FTP服务公开到外面的网络（非本地用），则必须承担被侵入或其它的风险。我强烈建议要有完善的安全机制才这样做。

### 2.13.3. 构建安装程序

如果从本地安装（我不推荐那样做），或者是由于不支持硬件或者更合理的理由是因为需要安装。必需准备几块。

至少有些必须 是 1.44 MB 的，用来容纳所有在 base (基本系统) 目录下的文件。如果在 DOS 操作系统下就必须使用 MS-DOS® 的 **FORMAT** 命令来格式化。如果使用的是 Windows® 操作系统，在资源管理器中就可以完成这个工作 (用右键单击 A: 驱动器，并单击 "Format")。

不要指望厂家的先格式化！最好是自行格式化。去用公告的很多都是由于不正当地使用格式化所造成的，所以我需要在里面着重提一下。

如果在外面一台 FreeBSD 的机器上做了磁盘的，行格式化是一个不错的主意。当然不需要把磁盘都做成 DOS 文件系统。也可以使用 **bsdlabel** 和 **newfs** 命令来建立一个 UFS 文件系统，具体操作按下面的顺序行：

```
# fdformat -f 1440 fd0.1440
# bsdlabel -w fd0.1440 floppy3
# newfs -t 2 -u 18 -l 1 -i 65536 /dev/fd0
```

然后就可以像其它的文件系统一样挂上和写入一些磁。

格式化一些磁后，必须把文件复制到磁中。一些行文件被分割成刚好可存 1.44 MB 的。所有的磁，出所有可能符合的文件。直到得到所有需要的配置并且将它以方式安置。第一个配置都有一个子目在磁上，例如：a:\base\base.aa、a:\base\base.ab，等等。



base.inf 文件，也放在 base 的第一磁上，因安装程序需要取这个文件，以了解在得布包需要下多少文件。

一旦进入安装介绍的屏幕，Floppy 将会看到后面的提示符。

#### 2.13.4. 从 MS-DOS® 分区安装

如果从 MS-DOS® 分区安装，需要将布文件复制到分区根目录下的 freebsd 目录中。例如：c:\freebsd。必须复制一部分 CDROM 或 FTP 上的目录，因此，如果从光盘行复制，建议使用 DOS 的 **xcopy** 命令。下面是准行 FreeBSD 最小系统安装的例子：

```
C:\> md c:\freebsd
C:\> xcopy e:\bin c:\freebsd\bin\ /s
C:\> xcopy e:\manpages c:\freebsd\manpages\ /s
```

假设 C: 是空的空，E: 是挂接的 CDROM。

如果没有光驱，可以从以下网站下行包。[ftp.FreeBSD.org](http://ftp.FreeBSD.org)。一个行包都在一个目录中，例如 base 行包可以在 [12.0/base/](http://12.0/base/) 目录中得到。

很多行包来，如果希望从 MS-DOS® 分区安装的 (有足的空)，安装 c:\freebsd - 下的个文件 - 个 **BIN** 行包只是最低限度的要求。

#### 2.13.5. 建立一个安装磁

从磁安装也是最快的方式，比在行使用 FTP 安装或使用 CDROM 快。安装的程序假设是被在磁上。在得到所有配置文件后，地解它，用下面的命令：

```
# cd /freebsd/distdir
# tar cvf /dev/rwt0 dist1 ... dist2
```

在安装的时候，要预留有足够的空间（允许）来容纳磁盘安装的全部内容。由于不是随机磁的，所以安装方法需要很多空间。



开始安装，在从磁盘之前，磁盘必须已放在磁盘中。否则，安装过程中可能会找不到它。

## 2.13.6. 网络安装

可用的网络安装类型有三种。以太网（标准的以太网控制器）、串口（PPP）以及并口（PLIP (laplink 接口)）。

如果希望以最迅速的方式完成网络安装，那么以太网适配器当然就是首选！FreeBSD 支持大多数常用 PC 以太网；系统能支持的网口（以及所需的配置）可以在 FreeBSD 发行版附带的硬件兼容表中查到。如果你使用的是系统支持的 PCMCIA 以太网，在添加之前一定要把它插好！很不幸，FreeBSD 目前并不支持在安装过程中 PCMCIA。

此外，你需要知道自己的 IP 地址、网口类型的子网掩码，以及机器名。如果你正通过 PPP 接口安装而没有固定的静态 IP，不用怕，这个 IP 地址会由你的 ISP 自动分配。你的系统管理会告诉你行网口配置所需的信息。如果你需要通名字而不是 IP 地址来连接其他主机，你需要配置一个域名服务器，可能还需要一个网口地址（在使用 PPP 时，这个地址是服务提供商的 IP 地址）。如果你希望通过 HTTP 代理服务来完成 FTP 安装，你需要知道代理服务器的地址。如果你不知道某些信息，请在行安装之前向系统管理或 ISP 咨询。

如果你使用一个 MODEM，那你就只有 PPP 一选项了。在安装的过程中，要能很容易地得到完整且快速的关于服务提供商的信息。

如果你使用 PAP 或 CHAP 方式连接到你的 ISP，（即，如果你不使用脚本在 Windows® 中连接到你的 ISP），那么你需要在 ppp 提示符下输入 dial 命令。否则，当 PPP 接口提供者只提供一最简的端模拟器，你必须知道如何使用 MODEM 的 "AT commands" 号到你的 ISP。想知道更深入的信息可以参考使用手册中的用 PPP 那章以及 FAQ。如果你有一些时间，可以使用 set log local ... 命令将日志显示在屏幕上。

你也可以通过并口连接到另一台 FreeBSD 机器上行安装，你可以考虑使用 "laplink" 并口接口行安装。通过并口安装要比通过串口（最高 50 kbytes/sec）安装快得多。

### 2.13.6.1. 通过网络安装之前

NFS 安装方式是非常方便的。只需要将 FreeBSD 文件复制到一台服务器上，然后在安装时 NFS 介绍。

如果你的服务器要“特端口”才能支持（如 SUN 的工作站），你需要在安装前在 Options 菜单中置 NFS Secure。

如果你使用了一低量的以太网口比糟糕，速度很慢，考虑 NFS Slow 的选项。

到了到 NFS 安装的目的，你的服务器必须支持 subdir 加。例如，如果你的 FreeBSD 12.0 目录存在：ziggy:/usr/archive/stuff/FreeBSD，然后 ziggy 将必须允许直接挂上 /usr/archive/stuff/FreeBSD，而不是 /usr 或 /usr/archive/stuff。

在 FreeBSD 的 `/etc/exports` 配置文件中，是由 `-alldirs` 来控制的。其它 NFS 服务器也有不同的方式。如果你从服务器得到 `permission denied` 个信息，可能是因你没有正的用它。

# Chapter 3. 安装 FreeBSD (用于 9.x 及以后版本)

## 3.1. 概述

FreeBSD 提供了一个以文字为主、便于使用的安装程序：从 FreeBSD 9.0-RELEASE 开始是指 `bsdinstall`，而在之前是指 `sysinstall`。本章介绍 `bsdinstall` 的使用，有关 `sysinstall` 的使用参见 [安装 FreeBSD](#)。

学完本章之后，你将知道：

- 如何创建 FreeBSD 安装介。
- FreeBSD 如何分目硬。
- 如何 `bsdinstall`。
- 行 `bsdinstall` 需要回答的，的具体含，以及如何回答。

本章之前，：

- 看将要安装的 FreeBSD 版本所附的硬件支持列表，以定硬件能被支持。



一般来，此安装明是 i386™ (“PC 兼容机”) 架的计算机；同也会尽可能地其他架下的安装予以明。然本文常更新，但仍可能与所安装版本上附的明文有些出入，因此建将其作常的安装指。

## 3.2. 硬件需求

### 3.2.1. 最低配置

安装 FreeBSD 所需的最低配置，随版本及硬件架而有所不同。

以下几些信息行了。根据所的安装方式，可能需要使用 FreeBSD 支持的 CDROM 或网配器，[准安装介](#)。

#### 3.2.1.1. FreeBSD/i386

FreeBSD/i386 需要 486 或更快的理器，最小 64 MB 的内存，以及至少 1.1 GB 的硬空。



通常情况下于老旧的计算机而言，安装更大的内存和出更多的硬空，会比使用更快的理器性能的提升更加明。

#### 3.2.1.2. FreeBSD/amd64

FreeBSD/amd64 支持理器。第一是 AMD64 理器，包括 AMD Athlon™64、AMD Athlon™64-FX、AMD Opteron™ 以及更高的理器。

能使用 FreeBSD/amd64 的理器是采用了 Intel® EM64 架的理器。理器包括 Intel® Core™ 2 Duo、Quad 和 Extreme 家族，包括 Intel® Xeon™ 3000、5000 和 7000 系列，以及 Intel® Core™



i3、i5 和 i7。

由于使用了 nVidia nForce3 Pro-150 的机器，必须在 BIOS 设置中禁用 IO APIC，如果没有的话就只能禁用 ACPI。因为 Pro-150 芯片存在 bug，而目前还没有能避免此 bug 的方法。

### 3.2.1.3. FreeBSD/powerpc Apple® Macintosh®

支持所有内建 USB 的 New World Apple® Macintosh® 系列，同时也配置多 CPU 的机器提供 SMP 支持。

注意 32 位的内核只能使用内存的前 2 GB，而 PowerMac G3 白机上的 FireWire® 也不被支持。

### 3.2.1.4. FreeBSD/sparc64

有 FreeBSD/sparc64 的系列支持，见 [FreeBSD/sparc64](#) 目录。

FreeBSD/sparc64 需要独占一个磁盘。目前不支持与其他操作系统共享同一个磁盘。

## 3.2.2. 支持的硬件

FreeBSD 发行版所支持的硬件架构及平台会列在硬件兼容说明文件中，此文件通常名为 `HARDWARE.TXT`，位于发行版介绍的根目录下。有些内容也可以在 FreeBSD 网站的 [发行版信息](#) 页面上得到。

## 3.3. 安装前的准备工作

### 3.3.1. 备份数据

在将 FreeBSD 安装至目标机器前，应首先备份其上的重要数据并谨慎行事。FreeBSD 安装程序硬要做任何改动前都会行提示，而一旦操作开始就无法撤回。

### 3.3.2. 决定将 FreeBSD 安装在何处

如果整个硬盘上安装 FreeBSD 一个操作系统，那将直接跳此节；但如果需要 FreeBSD 与其他操作系统并存，那将首先应了解 FreeBSD 的硬盘布局。

#### 3.3.2.1. FreeBSD/i386 与 FreeBSD/amd64 的硬盘布局

硬盘可以分割成多个区域，这些区域称作 *partition*（分区）。

有旧硬盘分区方式。旧的 *Master Boot Record* (MBR, 主引导记录) 的分区表中可以定义四个 *primary partitions* (主分区)。(由于历史原因，FreeBSD 中将主分区称作 *slice*。)为了突破四个分区的限制，可以将其中一个主分区建为 *extended partition* (扩展分区)，并在其中建立 *logical partitions* (逻辑分区)。正如看到的那样，旧方法十分笨拙。

新式的 *GUID Partition Table* (GUID 分区表) (GPT) 提供了更简便的磁盘分区方法。与旧的 MBR 分区相比，GPT 功能更大。常有的 GPT 可以在一个磁盘上支持多达 128 个分区，从而无需再采用类似分区选择床架屋的旧方法。



一些旧式的操作系统，如 Windows® XP 并不兼容 GPT 分区格式。如果需要 FreeBSD 与旧的操作系共用一个硬盘，就必须使用 MBR 分区了。

FreeBSD 的引导加载器需要使用一个主分区或 GPT 分区。(有 FreeBSD 引导程序的详情，请参考 [FreeBSD 引导程序](#)。) 如果所有的主分区或 GPT 分区都已在使用中，必须 FreeBSD 出一个来使用。

最小安装的 FreeBSD 只需 1 GB 磁盘空间。不过，这是非常基本的安装，而且也不会留下多少可用的空间。比通常的情况下，如果不使用图形界面，最小安装分配至少 3 GB 的空间，而使用图形界面，则分配至少 5 GB 的空间。此外，第三方应用程序可能需要更多的空间。

有很多 [免费或商业的分区调整工具](#) 可供使用。例如，以 Live CD 形式提供的 [GParted Live](#) 中的 GParted 分区调整器。此外，GParted 也可以在许多其它 Linux Live CD 发行版中找到。



磁盘分区程序有可能会破坏已有的数据。在修改磁盘分区之前，请先做一次完整的备份并校验其完整性。

调整 Microsoft® Vista 分区大小可能会遇到一些问题。如果要这样做，请提前做好 Vista 安装光盘。

#### 例 3. 使用已有的分区

假设一台安装了 Windows® 的计算机上有一个 40 GB 的硬盘，分成了两个 20 GB 的分区。Windows® 将它划分叫做 C: 和 D:。C: 分区包含了 10 GB 数据，而 D: 分区包含了 5 GB 数据。

将数据从 D: 移到 C:，就可将第二个分区空出来供 FreeBSD 使用了。

#### 例 4. 缩小已有的分区

假设一台安装了 Windows® 的计算机上有一个 40 GB 的硬盘，一个大的分区使用了整个磁盘的全部空间。Windows® 将这个 40 GB 分区叫做 C:。目前占用了 15 GB 空间。希望将 Windows® 分区减少到 20 GB，并将余下的 20 GB 分给 FreeBSD 使用。

可以在以下方法中任选一种：

1. 备份 Windows® 数据。接着，重新安装 Windows®，在安装过程中建立一个 20 GB 的分区。
2. 使用类似 GParted 的分区调整工具来缩小 Windows® 分区，并腾出空间供 FreeBSD 使用。

包含不同操作系统的磁盘分区令您能够在任何时候使用其中之一。可以使用在 [虚拟化](#) 中介的方法。

如果希望同时运行多个不同的操作系统，

### 3.3.3. 收集网络配置信息

某些 FreeBSD 安装方式需要通过网络接口下载相关文件。若要连接至以太网（或 PPPoE/DSL 调制解调器上的以太网接口），则需要向安装程序提供必要的网络配置信息。

DHCP 可以用来提供自动配置网络的信息。假如没有可用的 DHCP，必须从局域网管理员，或网络服务提供商那里获得必要的配置信息：

1. IP 地址
2. 子网掩码
3. 默认网关的 IP 地址

4. 本地网域名
5. DNS 服务器的 IP 地址

### 3.3.4. 了解 FreeBSD 发行勘

尽管 FreeBSD 项目会保证每个发行版尽可能地稳定，但 bug 是在所难免。少数情况下，一些 bug 甚至会影响安装。一旦这些 bug 被识别并修正后，就会列在 FreeBSD 网站的 [FreeBSD 发行勘](#) 中。在安装之前，应首先阅读这些勘，以确保安装可以顺利进行。

有关所有发行版的信息及勘，可以在 [FreeBSD 网站](#) 的 [发行版信息](#) 一文中找到。

### 3.3.5. 准备安装介质

FreeBSD 的安装介质包括 CD、DVD 及 USB 闪存棒。若要开始安装，只需使用安装介质引导计算机即可；注意不能通过在其他操作系统中行安装程序的方式行安装。

标准的安装介质中包含了 FreeBSD 安装所需的全部文件，除此之外，还有一个 *bootonly* 安装介质。该介质并不在其中直接包含安装所需的全部文件，而是在需要时通过网络行下载。因此，与标准的安装介质相比，bootonly 安装介质体积更小。

FreeBSD 安装介质的副本可以从 [FreeBSD 网站](#) 获取。



如果您已有 FreeBSD 的安装 CD、DVD 或 USB 闪存棒，可以跳过此步。

FreeBSD 的安装 CD 或 DVD 映像均可引导的 ISO 文件。只需要 CD 或 DVD 其中之一即可完成安装操作。任一介质在当前操作系统中刻录成可引导光盘即可。

若要创建可引导的闪存棒，请行以下操作：

## 1. 获取内存棒映像

FreeBSD 9.0-RELEASE 和更高版本的内存棒映像文件可以在 <ftp://ftp.FreeBSD.org/pub/FreeBSD/releases/arch/arch/ISO-IMAGES/version/FreeBSD-version-RELEASE-arch-memstick.img> 中的 ISO-IMAGES/ 目录中找到，其中，*arch* 是指要安装的架构，而 *version* 是指要安装版本号。例如，FreeBSD/i386 9.0-RELEASE 的内存棒映像位于 <ftp://ftp.FreeBSD.org/pub/FreeBSD/releases/i386/i386/ISO-IMAGES/9.0/FreeBSD-9.0-RELEASE-i386-memstick.img> 找到。



在 FreeBSD 8.X 以及更早的版本中，映像文件的下划线位置略有不同。关于 FreeBSD 8.X 和更早版本的安装操作请参考 [安装 FreeBSD](#)。

内存棒映像的扩展名为 `.img`。在 ISO-IMAGES/ 目录中提供了多个不同的映像，可以根据需要的 FreeBSD 版本，有时也包括安装映像的硬件状况进行。



进行以下操作前，请备份 USB 内存棒上的数据，因为之后的操作将擦除这些数据。

## 2. 将映像文件写入内存棒

### Procedure: 在 FreeBSD 中操作

在下面的例子中，目标内存棒的端点是 `/dev/da0`。操作前仔细检查目标是否正确，以免损坏已有的数据。

#### a. 使用 `dd(1)` 写入映像

扩展名为 `.img` 的映像文件不是一个普通的文件。它是内存棒上完整内容所做的映像，因此不能只是像普通文件一样的复制，而使用 `dd(1)` 将其直接写入目标：

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

## Procedure: 在 Windows® 中操作

操作前请确认是否已正确识别了设备的 ID 号，否则可能会覆盖并损坏您的数据。

### a. 获取 Image Writer for Windows®

Image Writer for Windows® 是一款能将映像正确写入到 U 盘中的免费应用程序。从 <https://launchpad.net/win32-image-writer/> 下载并将其提取至任意文件夹后即可开始使用。

### b. 使用 Image Writer 写入映像

双击 Win32DiskImager 程序后，确定 **Device** 下面显示的设备 ID 号所指的是 U 盘。点击文件菜单以需要写入的映像文件，然后点击 **【Save】** 接受。在所有操作无误且没有其他 U 盘插入后，点击 **【Write】** 将映像文件写入 U 盘。



系统不再支持从 U 盘行安装了。

现在可以开始安装 FreeBSD 了。

## 3.4. 开始安装

默认情况下，在看到下面信息之前，安装程序不会硬性数据做任何修改：



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，抑或担心进行了某些配置，也可以直接回源。

### 3.4.1. 虚拟机

#### 3.4.1.1. 引导 i386™ 及 amd64 系

1. 若要使用 [准安装介](#) 所述的 USB 棒引导，请在开机前将其插入计算机。

若要使用 CDROM 引导，请在开机后立刻将其放入计算机。

2. 根据所使用的安装介，从 CDROM 或 USB。在 BIOS 设置中，可以选择特定的引导。大多数系统可以在 BIOS 引导，通常需要按 **F10**、**F11**、**F12** 或 **Escape**。
3. 如果计算机正常启动并加载了已有的操作系统，那么：
  - a. USB 棒插入或 CDROM 放入，将其拔下或取出，然后重新启动计算机并再次。
  - b. BIOS 设置，重新设置。
  - c. BIOS 不支持从当前介；可以使用 [Plop Boot Manager](#)，它能老式计算机支持 CD 或 USB。
4. FreeBSD 将开始。如果使用的是 CDROM，会看到类似的提示（版本信息可以忽略）：

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

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

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

FreeBSD/i386 bootstrap loader, Revision 1.1

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

5. FreeBSD 引导加载器会提示：



图 59. FreeBSD 引导加载器菜单

您可以等待十秒或按 `Enter` 。

### 3.4.1.2. 引导 Macintosh® PowerPC®

在大多数机器上，开机时按住 `C` 可以从 CD 启动。除此之外，按住 `Command` + `Option` + `0` + `F`，在非 Apple® 电脑上则是 `Windows` + `Alt` + `0` + `F`，然后在出现的提示符 `0 >` 下输入

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

对于不兼容的 Xserve 机器，请参考 [Apple® 支持网站](#) 以了解如何引导至 Open Firmware。

### 3.4.1.3. 引导 sparc64

多数 sparc64 系统均配置成了硬盘启动。若要安装 FreeBSD，需从网络或 CDROM 启动，这就需要首先输入 PROM (OpenFirmware)。

重启系统后等待引导信息输出，虽然其具体内容取决于机器型号，但通常会类似：

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

如果此系统已启动从硬盘，那么按下 `L1` + `A` 或 `Stop` + `A` 或在串口控制台发送 `BREAK`（在 `tip(1)` 或 `cu(1)` 中是 `~#`）以输入 PROM 提示符，它如下所示：

```
ok ①
ok {0} ②
```

① 是在 CPU 系上的提示符。

② 是在 SMP 系上的提示符， 其中的数字表示可用的 CPU 个数。

在， 放入 CDROM 并在 PROM 提示符后入 `boot cdrom`。

### 3.4.2. 看探果

了便于， 屏幕上所示的最后几百行字符会始保存在缓冲区里。

若要缓冲区， 可以按下 `Scroll Lock` 来屏幕的功能； 后即可使用方向、 `PageUp` 或 `PageDown` 行翻； 再次按下 `Scroll Lock` 将功能。

将看到内核行了探， 其果似 [典型的探果](#) 中的文本， 但具体内容会因计算机中所包含的而有所不同。

典型的探果

```
Copyright (c) 1992-2011 The FreeBSD Project.
Copyright (c) 1979, 1980, 1983, 1986, 1988, 1989, 1991, 1992, 1993, 1994
    The Regents of the University of California. All rights reserved.
FreeBSD is a registered trademark of The FreeBSD Foundation.
FreeBSD 9.0-RELEASE #0 r225473M: Sun Sep 11 16:07:30 BST 2011
root@psi:/usr/obj/usr/src/sys/GENERIC amd64
CPU: Intel(R) Core(TM)2 Duo CPU      T9400 @ 2.53GHz (2527.05-MHz K8-class CPU)
  Origin = "GenuineIntel"  Id = 0x10676  Family = 6  Model = 17  Stepping = 6
  Features
=0xbfebfbff<FPU,VME,DE,PSE,TSC,MSR,PAE,MCE,CX8,APIC,SEP,MTRR,PGE,MCA,CMOV,PAT,PSE36,CL
FLUSH,DTS,ACPI,MMX,FXSR,SSE,SSE2,SS,HTT,TM,PBE>
  Features2
=0x8e3fd<SSE3,DTES64,MON,DS_CPL,VMX,SMX,EST,TM2,SSSE3,CX16,xTPR,PDCM,SSE4.1>
  AMD Features=0x20100800<SYSCALL,NX,LM>
  AMD Features2=0x1<LAHF>
  TSC: P-state invariant, performance statistics
real memory = 3221225472 (3072 MB)
avail memory = 2926649344 (2791 MB)
Event timer "LAPIC" quality 400
ACPI APIC Table: <TOSHIB A0064  >
FreeBSD/SMP: Multiprocessor System Detected: 2 CPUs
FreeBSD/SMP: 1 package(s) x 2 core(s)
  cpu0 (BSP): APIC ID: 0
  cpu1 (AP): APIC ID: 1
ioapic0: Changing APIC ID to 1
ioapic0 <Version 2.0> irqs 0-23 on motherboard
kbd1 at kbdmux0
acpi0: <TOSHIB A0064> on motherboard
acpi0: Power Button (fixed)
acpi0: reservation of 0, a0000 (3) failed
```



```

acpi0: reservation of 100000, b6690000 (3) failed
Timecounter "ACPI-safe" frequency 3579545 Hz quality 850
acpi_timer0: <24-bit timer at 3.579545MHz> port 0xd808-0xd80b on acpi0
cpu0: <ACPI CPU> on acpi0
ACPI Warning: Incorrect checksum in table [ASF!] - 0xFE, should be 0x9A
(20110527/tbutils-282)
cpu1: <ACPI CPU> on acpi0
pcib0: <ACPI Host-PCI bridge> port 0xcf8-0xcff on acpi0
pci0: <ACPI PCI bus> on pcib0
vgapci0: <VGA-compatible display> port 0xcff8-0xcfff mem 0xff400000-
0xff7fffff,0xe0000000-0xffffffff irq 16 at device 2.0 on pci0
agp0: <Intel GM45 SVGA controller> on vgapci0
agp0: aperture size is 256M, detected 131068k stolen memory
vgapci1: <VGA-compatible display> mem 0xffc00000-0xffcfffff at device 2.1 on pci0
pci0: <simple comms> at device 3.0 (no driver attached)
em0: <Intel(R) PRO/1000 Network Connection 7.2.3> port 0xcf80-0xcf9f mem 0xff9c0000-
0xff9dffff,0xff9fe000-0xff9fefff irq 20 at device 25.0 on pci0
em0: Using an MSI interrupt
em0: Ethernet address: 00:1c:7e:6a:ca:b0
uhci0: <Intel 82801I (ICH9) USB controller> port 0xcf60-0xcf7f irq 16 at device 26.0
on pci0
usb0: <Intel 82801I (ICH9) USB controller> on uhci0
uhci1: <Intel 82801I (ICH9) USB controller> port 0xcf40-0xcf5f irq 21 at device 26.1
on pci0
usb1: <Intel 82801I (ICH9) USB controller> on uhci1
uhci2: <Intel 82801I (ICH9) USB controller> port 0xcf20-0xcf3f irq 19 at device 26.2
on pci0
usb2: <Intel 82801I (ICH9) USB controller> on uhci2
ehci0: <Intel 82801I (ICH9) USB 2.0 controller> mem 0xff9ff800-0xff9ffbff irq 19 at
device 26.7 on pci0
usb3: EHCI version 1.0
usb3: <Intel 82801I (ICH9) USB 2.0 controller> on ehci0
hdac0: <Intel 82801I High Definition Audio Controller> mem 0xff9f8000-0xff9fbfff irq
22 at device 27.0 on pci0
pcib1: <ACPI PCI-PCI bridge> irq 17 at device 28.0 on pci0
pci1: <ACPI PCI bus> on pcib1
iwn0: <Intel(R) WiFi Link 5100> mem 0xff8fe000-0xff8fffff irq 16 at device 0.0 on pci1
pcib2: <ACPI PCI-PCI bridge> irq 16 at device 28.1 on pci0
pci2: <ACPI PCI bus> on pcib2
pcib3: <ACPI PCI-PCI bridge> irq 18 at device 28.2 on pci0
pci4: <ACPI PCI bus> on pcib3
pcib4: <ACPI PCI-PCI bridge> at device 30.0 on pci0
pci5: <ACPI PCI bus> on pcib4
cbb0: <RF5C476 PCI-CardBus Bridge> at device 11.0 on pci5
cardbus0: <CardBus bus> on cbb0
pccard0: <16-bit PCCard bus> on cbb0
isab0: <PCI-ISA bridge> at device 31.0 on pci0
isa0: <ISA bus> on isab0
ahci0: <Intel ICH9M AHCI SATA controller> port 0x8f58-0x8f5f,0x8f54-0x8f57,0x8f48-
0x8f4f,0x8f44-0x8f47,0x8f20-0x8f3f mem 0xff9fd800-0xff9fdfff irq 19 at device 31.2 on
pci0

```

```

ahci0: AHCI v1.20 with 4 3Gbps ports, Port Multiplier not supported
ahcich0: <AHCI channel> at channel 0 on ahci0
ahcich1: <AHCI channel> at channel 1 on ahci0
ahcich2: <AHCI channel> at channel 4 on ahci0
acpi_lid0: <Control Method Lid Switch> on acpi0
battery0: <ACPI Control Method Battery> on acpi0
acpi_button0: <Power Button> on acpi0
acpi_acad0: <AC Adapter> on acpi0
acpi_toshiba0: <Toshiba HCI Extras> on acpi0
acpi_tz0: <Thermal Zone> on acpi0
attimer0: <AT timer> port 0x40-0x43 irq 0 on acpi0
Timecounter "i8254" frequency 1193182 Hz quality 0
Event timer "i8254" frequency 1193182 Hz quality 100
atkbdc0: <Keyboard controller (i8042)> port 0x60,0x64 irq 1 on acpi0
atkbd0: <AT Keyboard> irq 1 on atkbdc0
kbd0 at atkbd0
atkbd0: [GIANT-LOCKED]
psm0: <PS/2 Mouse> irq 12 on atkbdc0
psm0: [GIANT-LOCKED]
psm0: model GlidePoint, device ID 0
atrtc0: <AT realtime clock> port 0x70-0x71 irq 8 on acpi0
Event timer "RTC" frequency 32768 Hz quality 0
hpet0: <High Precision Event Timer> iomem 0xfed00000-0xfed003ff on acpi0
Timecounter "HPET" frequency 14318180 Hz quality 950
Event timer "HPET" frequency 14318180 Hz quality 450
Event timer "HPET1" frequency 14318180 Hz quality 440
Event timer "HPET2" frequency 14318180 Hz quality 440
Event timer "HPET3" frequency 14318180 Hz quality 440
uart0: <16550 or compatible> port 0x3f8-0x3ff irq 4 flags 0x10 on acpi0
sc0: <System console> at flags 0x100 on isa0
sc0: VGA <16 virtual consoles, flags=0x300>
vga0: <Generic ISA VGA> at port 0x3c0-0x3df iomem 0xa0000-0xbffff on isa0
ppc0: cannot reserve I/O port range
est0: <Enhanced SpeedStep Frequency Control> on cpu0
p4tcc0: <CPU Frequency Thermal Control> on cpu0
est1: <Enhanced SpeedStep Frequency Control> on cpu1
p4tcc1: <CPU Frequency Thermal Control> on cpu1
Timecounters tick every 1.000 msec
hdac0: HDA Codec #0: Realtek ALC268
hdac0: HDA Codec #1: Lucent/Agere Systems (Unknown)
pcm0: <HDA Realtek ALC268 PCM #0 Analog> at cad 0 nid 1 on hdac0
pcm1: <HDA Realtek ALC268 PCM #1 Analog> at cad 0 nid 1 on hdac0
usb0: 12Mbps Full Speed USB v1.0
usb1: 12Mbps Full Speed USB v1.0
usb2: 12Mbps Full Speed USB v1.0
usb3: 480Mbps High Speed USB v2.0
ugen0.1: <Intel> at usb0
uhub0: <Intel UHCI root HUB, class 9/0, rev 1.00/1.00, addr 1> on usb0
ugen1.1: <Intel> at usb1
uhub1: <Intel UHCI root HUB, class 9/0, rev 1.00/1.00, addr 1> on usb1
ugen2.1: <Intel> at usb2

```

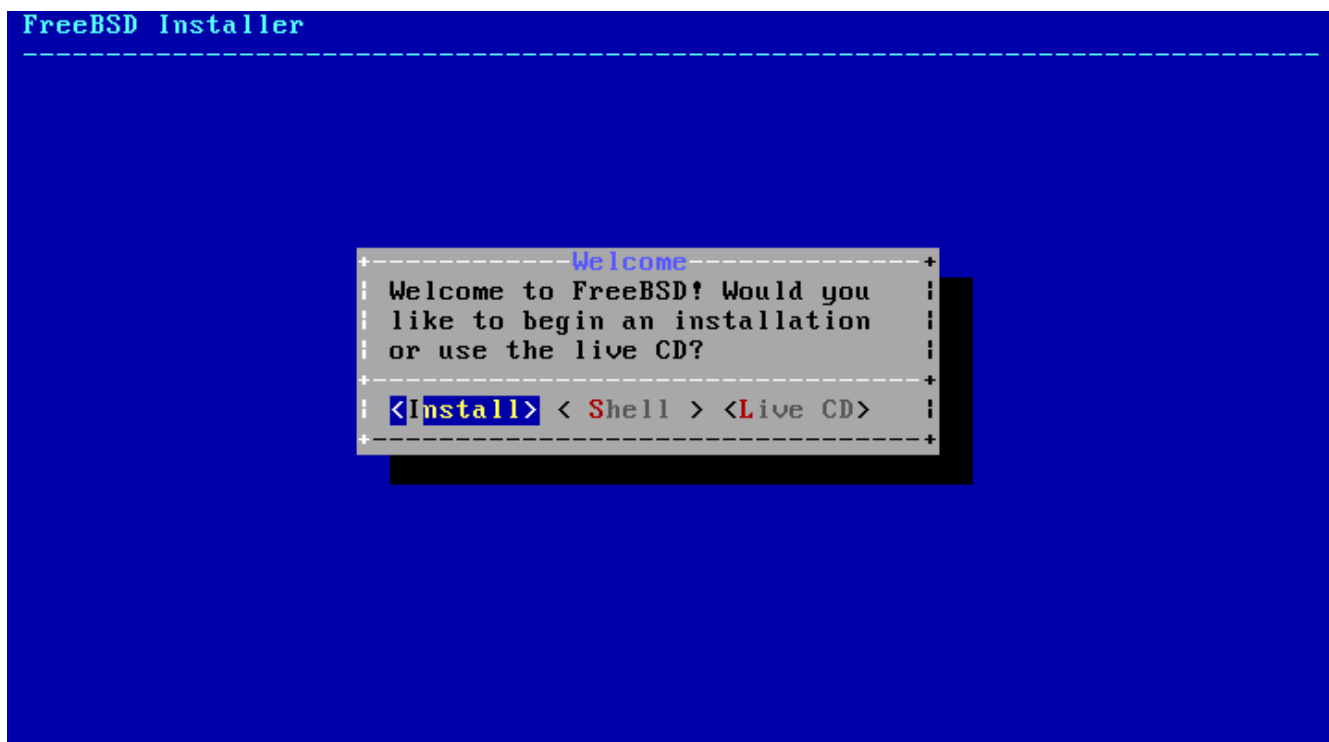
```

uhub2: <Intel UHCI root HUB, class 9/0, rev 1.00/1.00, addr 1> on usb2
ugen3.1: <Intel> at usb3
uhub3: <Intel EHCI root HUB, class 9/0, rev 2.00/1.00, addr 1> on usb3
uhub0: 2 ports with 2 removable, self powered
uhub1: 2 ports with 2 removable, self powered
uhub2: 2 ports with 2 removable, self powered
uhub3: 6 ports with 6 removable, self powered
ugen2.2: <vendor 0x0b97> at usb2
uhub8: <vendor 0x0b97 product 0x7761, class 9/0, rev 1.10/1.10, addr 2> on usb2
ugen1.2: <Microsoft> at usb1
ada0 at ahcich0 bus 0 scbus1 target 0 lun 0
ada0: <Hitachi HTS543225L9SA00 FBE0C43C> ATA-8 SATA 1.x device
ada0: 150.000MB/s transfers (SATA 1.x, UDMA6, PIO 8192bytes)
ada0: Command Queueing enabled
ada0: 238475MB (488397168 512 byte sectors: 16H 63S/T 16383C)
ada0: Previously was known as ad4
ums0: <Microsoft Microsoft 3-Button Mouse with IntelliEyeTM, class 0/0, rev 1.10/3.00,
addr 2> on usb1
SMP: AP CPU #1 Launched!
cd0 at ahcich1 bus 0 scbus2 target 0 lun 0
cd0: <TEAC DV-W28S-RT 7.0C> Removable CD-ROM SCSI-0 device
cd0: 150.000MB/s transfers (SATA 1.x, ums0: 3 buttons and [XYZ] coordinates ID=0
UDMA2, ATAPI 12bytes, PIO 8192bytes)
cd0: cd present [1 x 2048 byte records]
ugen0.2: <Microsoft> at usb0
ukbd0: <Microsoft Natural Ergonomic Keyboard 4000, class 0/0, rev 2.00/1.73, addr 2>
on usb0
kbd2 at ukbd0
uhid0: <Microsoft Natural Ergonomic Keyboard 4000, class 0/0, rev 2.00/1.73, addr 2>
on usb0
Trying to mount root from cd9660:/dev/iso9660/FREEBSD_INSTALL [ro]...

```

仔细探索结果，以确定 FreeBSD 看到了所有希望使用的设备。没有看到的设备并不会在列表里列出，因为默认的 GENERIC 内核中不包含它们；可以通过 [内核模块](#) 为某些设备提供支持。

探索完成后，您将看到 [安装介绍的使用方式](#)，表明安装介绍共有三种用途：安装 FreeBSD、制作“Live CD”或引导至 FreeBSD 的命令行界面。使用方向键——后按 **Enter**。



□ 60. □□安装介□的使用方式

在□里，□□□ [ **Install** ] 以□行安装程序。

## 3.5. 介□ **bsdinstall**

**bsdinstall** 是一个基于文本的 FreeBSD 安装程序，作者是 Nathan Whitehorn <[nwhitehorn@FreeBSD.org](mailto:nwhitehorn@FreeBSD.org)>，于 2011 年被 FreeBSD 9.0 采用。

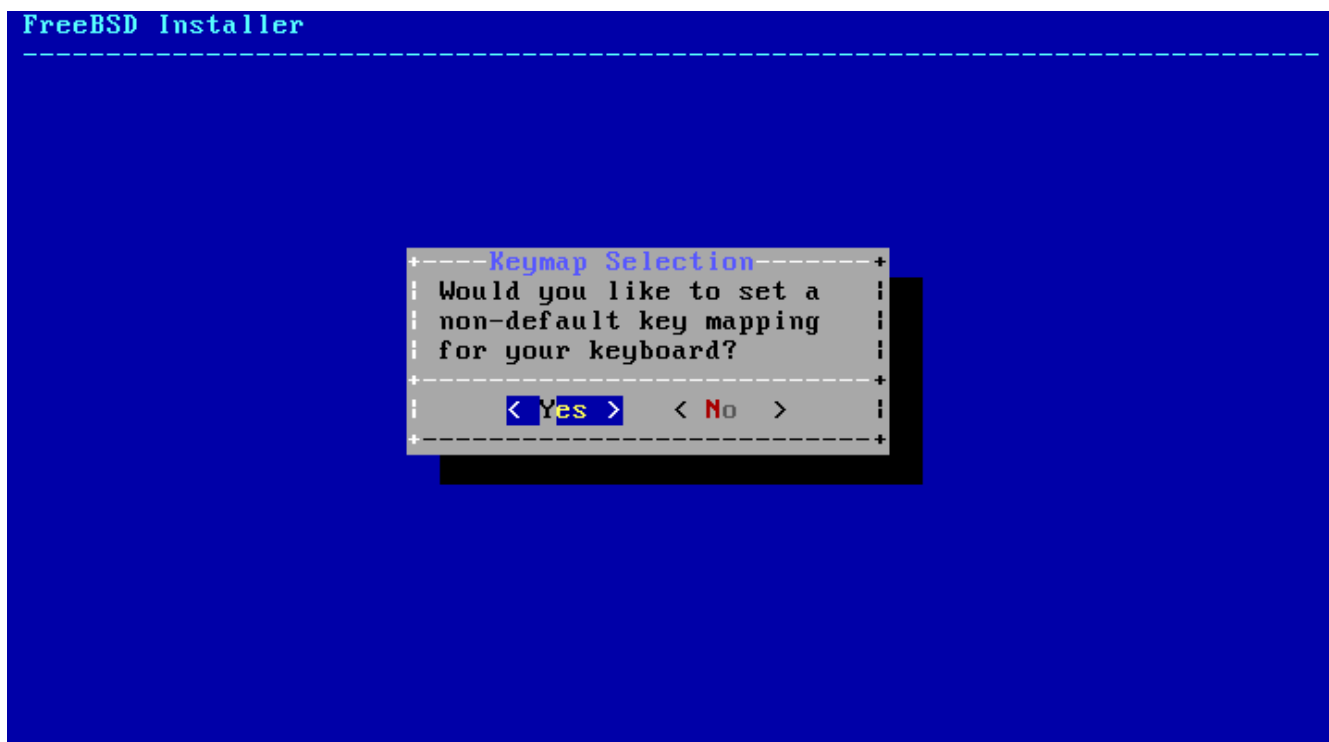


Kris Moore <[kmoore@FreeBSD.org](mailto:kmoore@FreeBSD.org)> □ **PC-BSD** □写的 **pc-sysinstall** 也可以用于 安装 **FreeBSD**。□然有□会同 **bsdinstall** 混□，但□□□者并不相□。

**bsdinstall** 菜□系□的主要控制□包括方向□、**Enter** □、**Tab** □、**Space** □等。

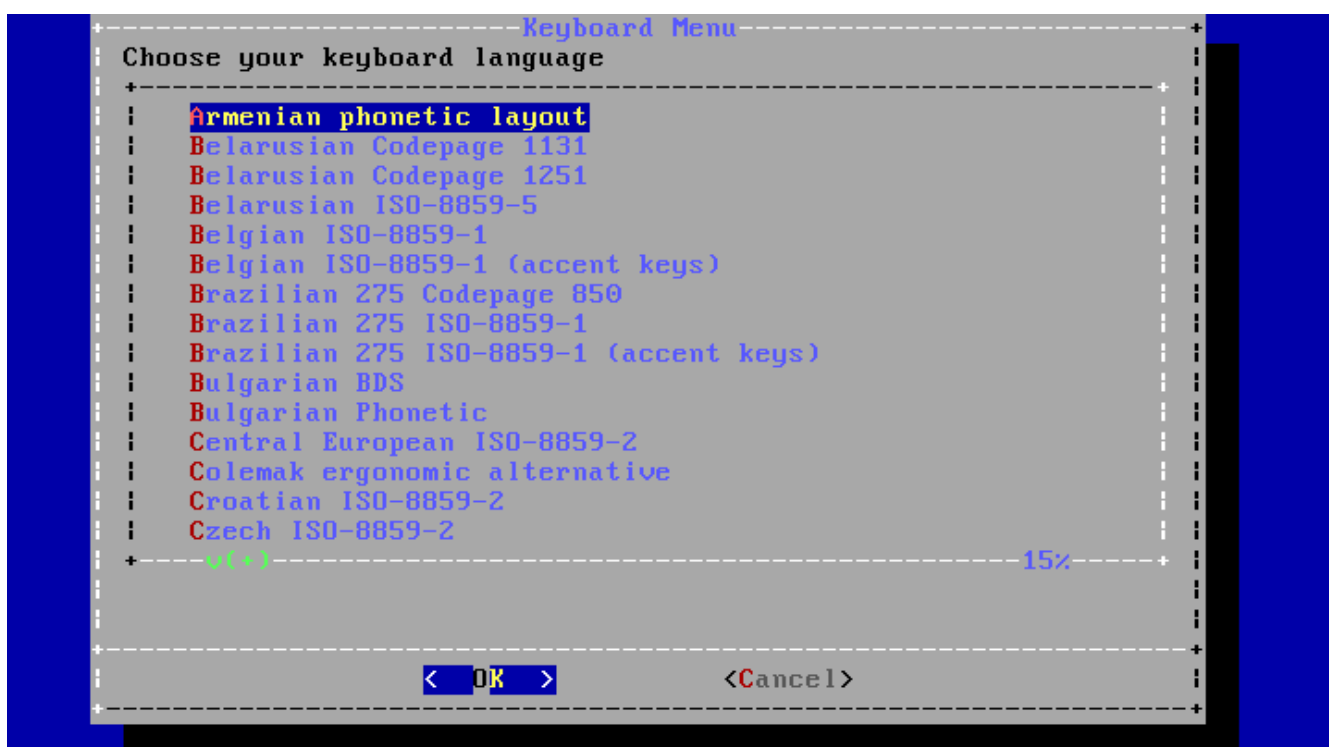
### 3.5.1. □□□□映射

根据当前正在使用的系□控制台，**bsdinstall** 可能会首先提示□□—□非默□的□□布局。



▢ 61. 映射

按了 [YES] 后，将显示下面的画面；否则将不显示此画面而直接使用默认映射。



▢ 62. 菜单

使用上/下方向键最合当前系的映射后，按 **Enter**。



按 **Esc** 以使用默认的映射。如果不清楚——，推 United States of America ISO-8859-1。

### 3.5.2. 设置主机名

下面，bsdinstall 将提示新安装的系设置主机名。

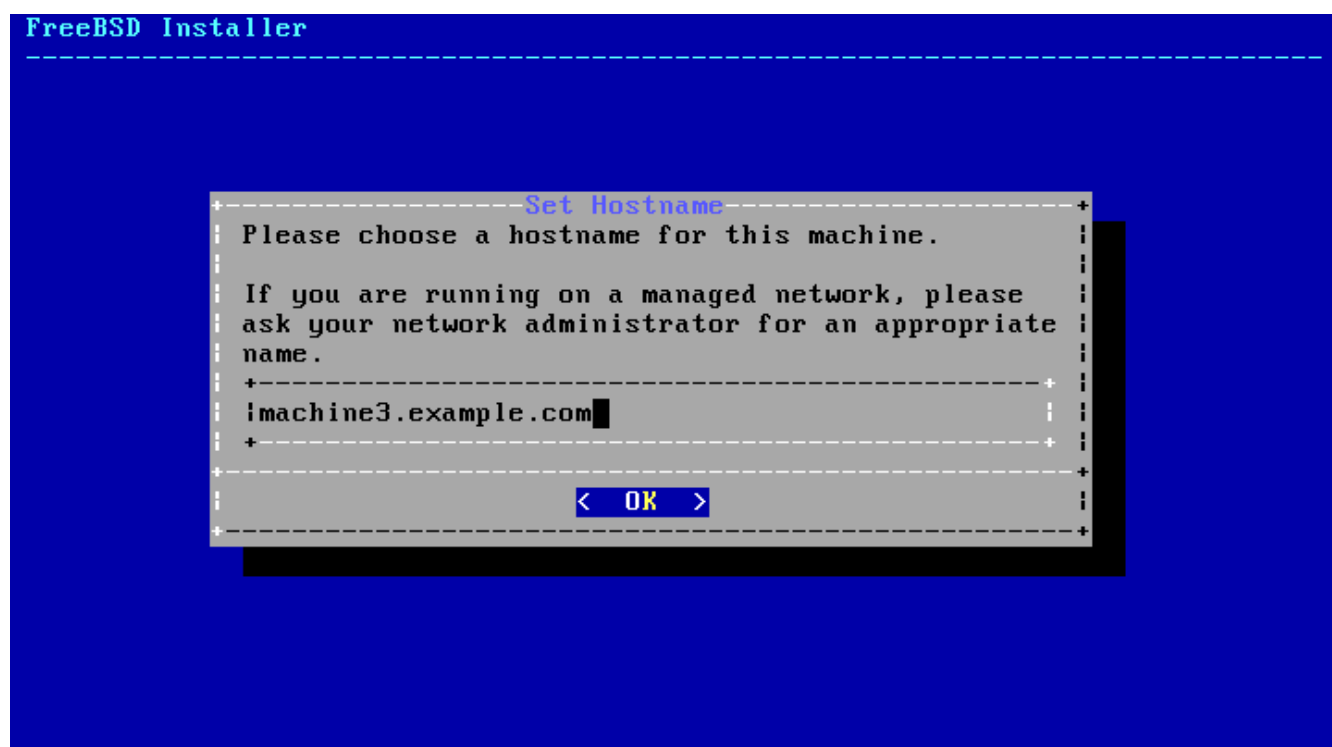


图 63. 设置主机名

输入完整的主机名，例如 `machine3.example.com`。

### 3.5.3. 需要安装的软件

下面，bsdinstall 将提示需要安装的软件。

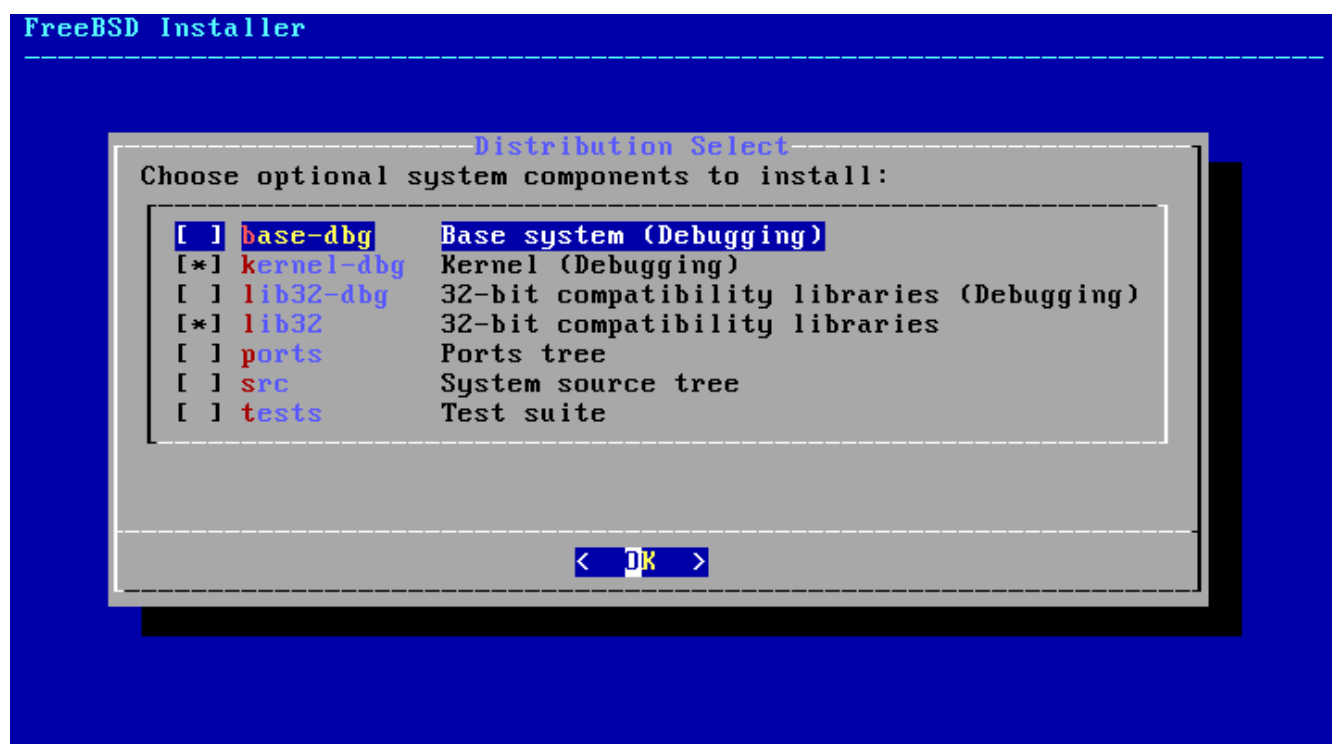


图 64. 需要安装的软件

安装哪些组件很大程度上取决于系统用途及可用磁盘空间。注意，任何情况下都会安装 FreeBSD 内核及用户空间（即称“基系统”）。

根据安装类型的不同，某些组件可能不会显示。

可选项

- **doc** - 附加文档，主要是与项目历史相关的内容。之后可以安装 FreeBSD 文档所提供的文档。
- **games** - 一些经典的 BSD 游戏，包括 fortune 与 rot13 等。
- **lib32** - 兼容库文件，用于在 64 位版本的 FreeBSD 上运行 32 位程序。
- **ports** - FreeBSD 的 ports 集。

ports 集提供了一简单而方便的途径来安装组件。在 ports 集中，并不包含组件所需的源代码，取而代之的是一能够自行下载、编译并安装第三方组件包的文件。[安装应用程序](#)、[Packages](#) 和 [Ports](#) 会描述如何使用 ports 集。



因此，必须确保有充足的硬盘空间，注意安装程序并不会对此进行检查。FreeBSD 9.0 的 ports 集需 3 GB 的磁盘空间；您也可以安装后的版本预留更大的空间。

- **src** - 系统源代码。

FreeBSD 提供了与内核及用户空间有关的完整源代码。大部分程序并不需要这些源代码，它主要用于特定组件（例如内核或内核模块）或者 FreeBSD 本身的开发。

完整的源代码需要 1 GB 的磁盘空间，而重新安装整个 FreeBSD 系统外还需要 5 GB 的空间。

## 3.6. 通过网络安装

*bootonly* 安装介绍中并不会包含所有的安装文件。如果使用介绍进行安装，那么需要的文件就必须通过网络下载。



## 65. 通网安装

根据 [配置网口](#) 配置了网口接后，即可开始镜像站点。镜像站点上存有 FreeBSD 的安装文件，如果一个更近的镜像站点有助于更快的取些文件，从而少安装。



## 66. 一个镜像站点

接至所镜像站点并到所需文件后，安装将行。

## 3.7. 分配磁盘

FreeBSD 提供了三种方式来分配磁盘： *Guided*（向式）分区能自置磁分区；而 *Manual*（手式）



分区允许高级用户建立自定义分区；用户可以入 shell 中直接使用类似 gpart(8)、fdisk(8) 与 bsdlabel(8) 的命令执行程序。

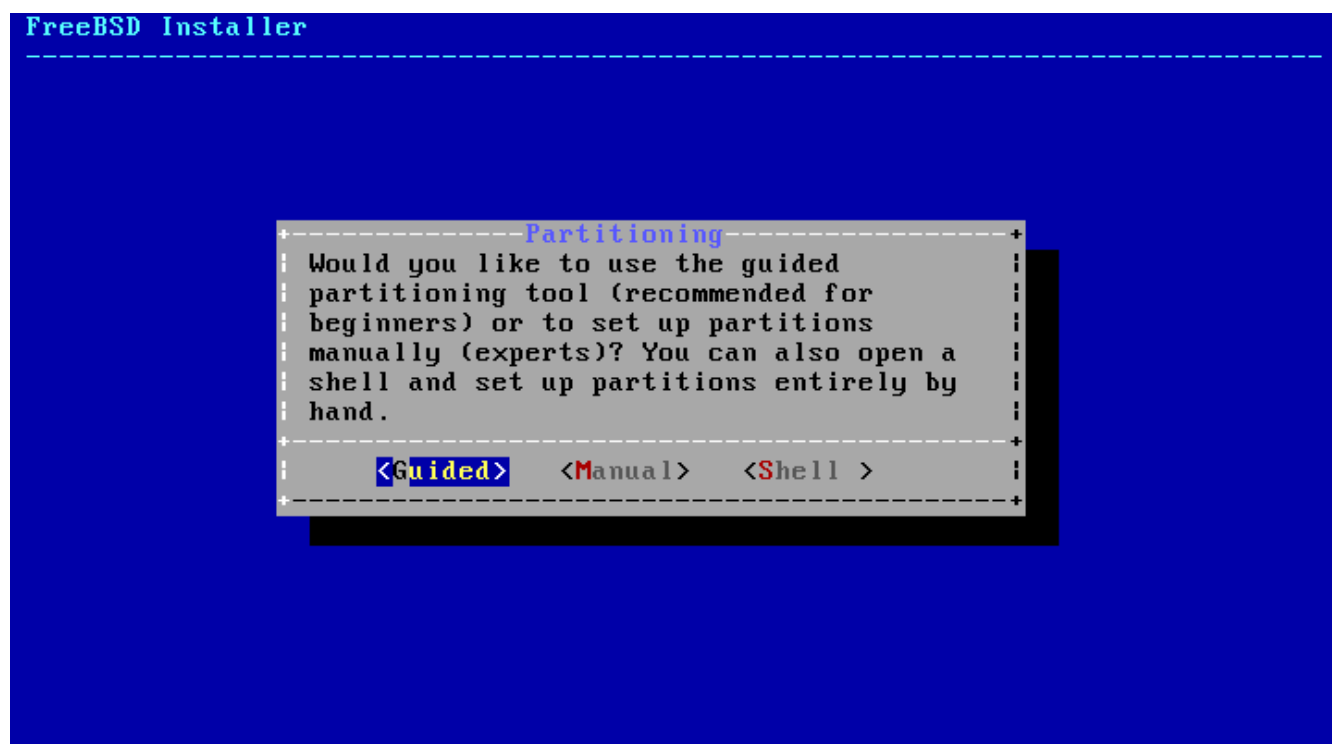


图 67. 磁盘分配的方式

### 3.7.1. 向磁盘分区

如果机器上配有多块磁盘，则需要 FreeBSD 的安装指定目标磁盘。

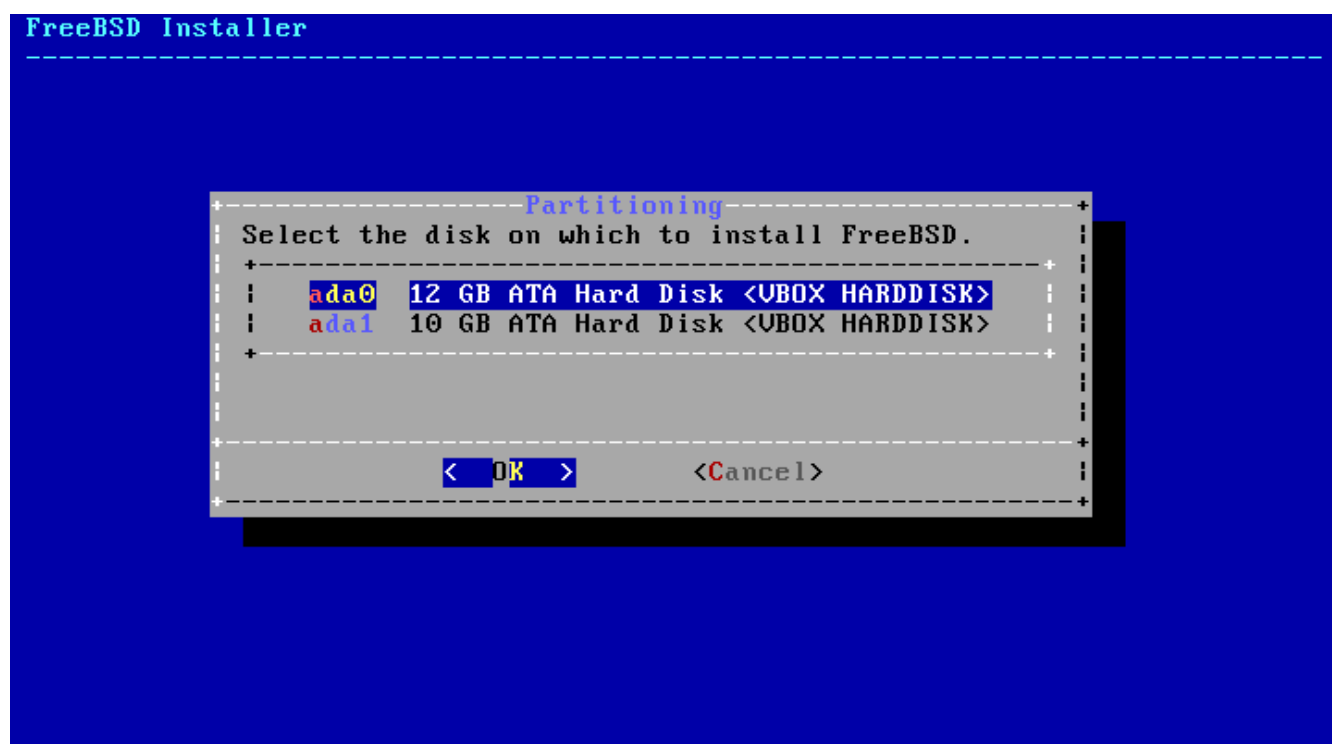


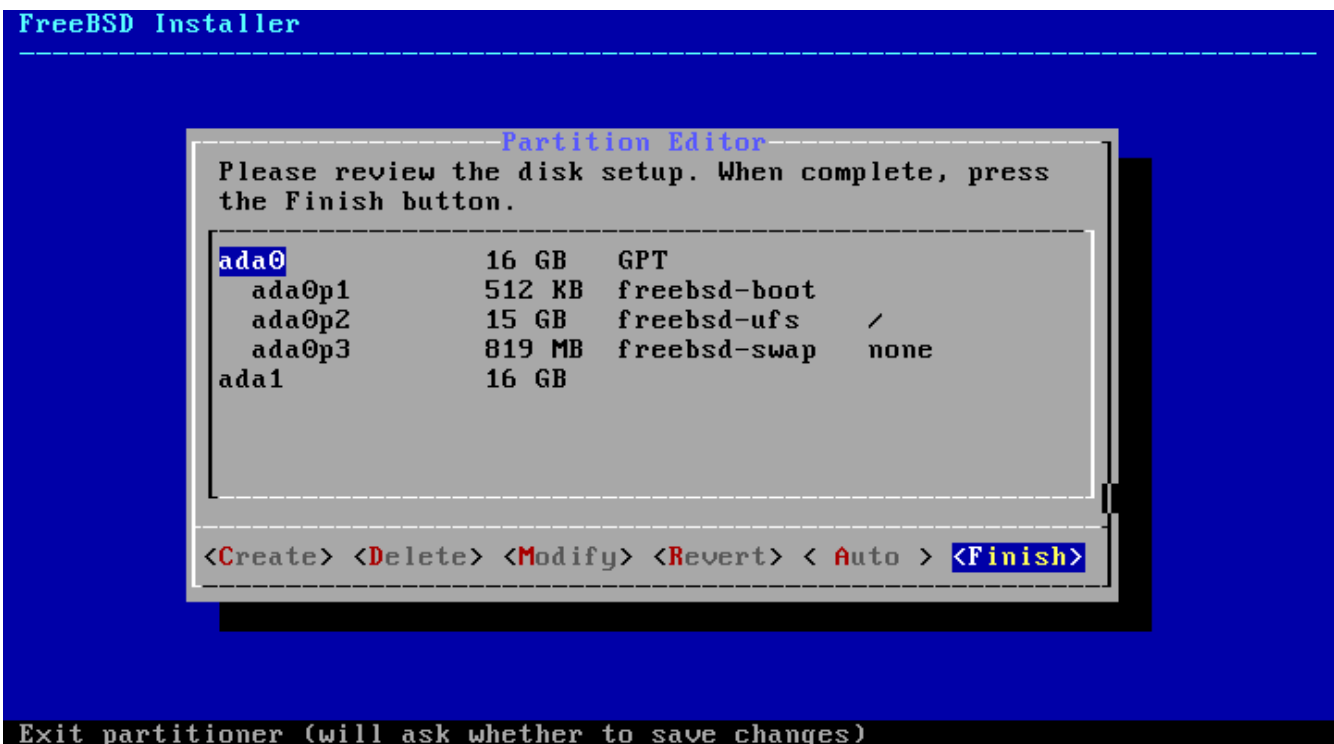
图 68. 从多块磁盘中进行选择

可以将整个磁盘都分配给 FreeBSD，也可以只分配其中的一部分。若选择的是 **[Entire Disk]**，则建立分区布局时会直接使用整个磁盘；若选择的是 **[Partition]**，则建立分区时会使用磁盘上的空闲空间。



□ 69. □□如何□建分区布局

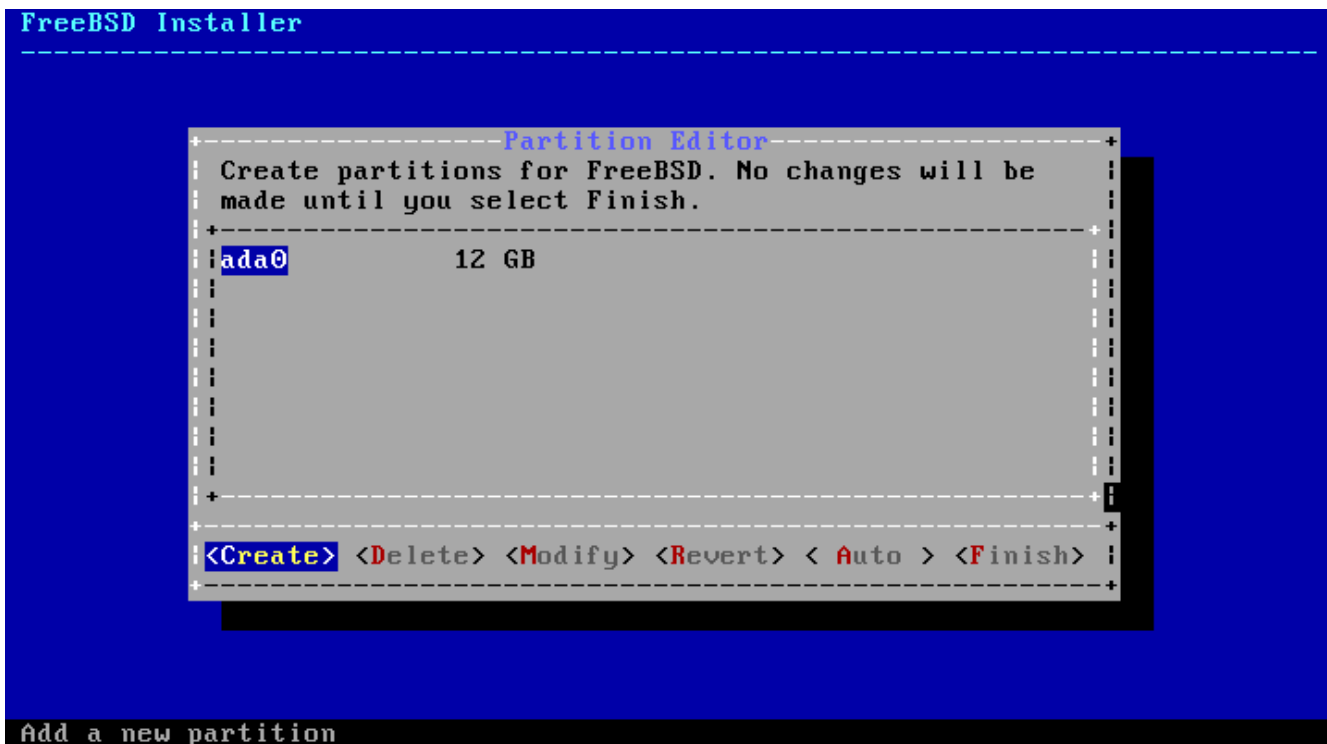
□仔□□分区布局的□建□果。 如果□□有□□之□， 可以□□ **[Revert]** 来□原之前的分区； 此外， 也可以□□ **[Auto]** 重新□ FreeBSD 自□□建分区。 也可以手□□建、 修改或□除分区。 正□□建了分区之后， □□□ **[Finish]** 以□□安装。



□ 70. □□已□建分区

### 3.7.2. 手□式分区

手□式分区将直接使用分区□□器□行操作。



## 71. 手动建分区

高亮目标设备（本例中为 `ada0`）并点击 **[ Create ]** 以显示 *partitioning scheme*（分区方案）菜单。



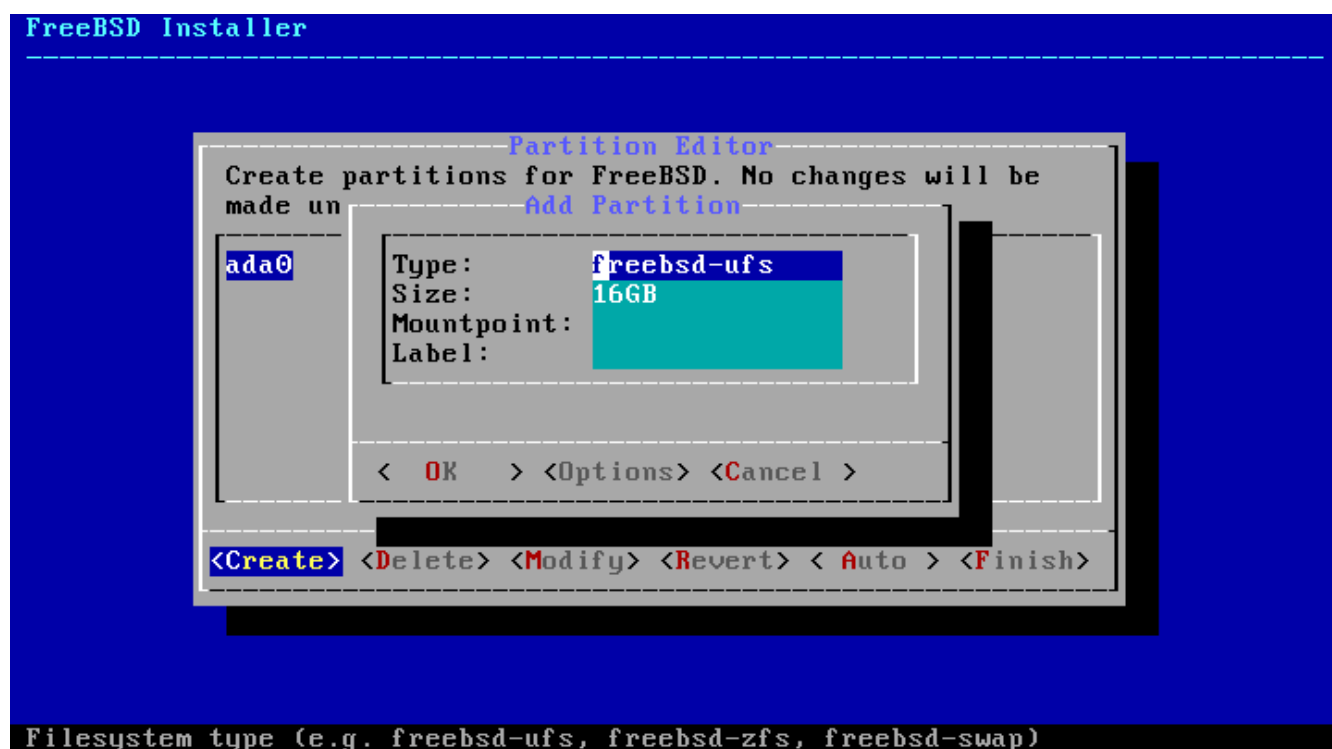
## 72. 手动建分区

对于 PC 兼容机来说，GPT 分区通常是最合适的，而某些不兼容 GPT 的老式操作系统可能需要使用 MBR 分区。除此之外的分区方案用于一些不常见的或其他的老式操作系统。

表 5. 分区方案

| 缩写    | 说明  |
|-------|---|
| APM   | Apple Partition Map, 用于 PowerPC® Macintosh®。  |
| BSD   | 不是 MBR 的 BSD Label, 有时也称为危险的专用模式, “dangerously dedicated mode”。参见 <a href="#">bsdlabeled(8)</a> 。 |
| GPT   | GUID 分区表。   |
| MBR   | Master Boot Record, 主引导记录。  |
| PC98  | MBR 格式, 用于 NEC PC-98 计算机。   |
| VTOD8 | Volume Table Of Contents, 用于 Sun SPARC64 和 UltraSPARC 计算机。  |

定义了分区方案并创建完成后，可再次按 **[ Create ]** 以创建新的分区。



### 73. 手动创建分区

FreeBSD 的标准 GPT 安装至少会使用三个分区：

标准 *FreeBSD* GPT 分区

- **freebsd-boot** - FreeBSD 引导分区，它必须位于首位。
- **freebsd-ufs** - FreeBSD 的 UFS 文件系统。
- **freebsd-swap** - FreeBSD 的交换空间。

也可以同时创建多个文件系统分区。有些用户会喜欢这样的分区格局，即 `/`、`/var`、`/tmp`，以及 `/usr` 文件系统分区创建分区。参见 [创建分区的分割式文件系统分区](#) 中的例子。

可用的 GPT 分区类型可以在 [gpart\(8\)](#) 中找到。

在指定尺寸时，可以使用常用的缩写：K 表示 kilobytes、M 表示 megabytes，而 G 表示 gigabytes。



正确地磁扇区能取得最佳性能。无论磁的个扇区 512 字节是 4K 字节，将分区大小置 4K 字节的倍数都能保证。操作中，只要使分区的大小等于 1M 或 1G 的倍数即可。唯一的例外是 *freebsd-boot* 分区，目前由于引代所限，此分区不能大于 512K。

若分区包含文件系统，需要在 Mountpoint 中其入挂点；若建了一个 UFS 分区，在此中入 /。

最后需要入的是 *Label* ( )，用于命名所建的分区。如果将器接至不同的控制器或端口，其名称或号会生改，但的并不会化。在似 */etc/fstab* 的文件中，通引用分区比通器名加分区号引用更加活，因引用使系硬件的改更加容。GPT 的会在磁接后出在 */dev/gpt/* 中；而其他分区方案中的也有不同的功能，它会出在 */dev/* 中的不同目里。



避免冲突，个文件系统指定独一无二的。与算机的名称、用途或位置相的字符均可添加至。例如，室算机的 UFS 根目可以命名 “labroot” 或 “rootfs-lab”。

例 5. 建的分割式文件系统分区

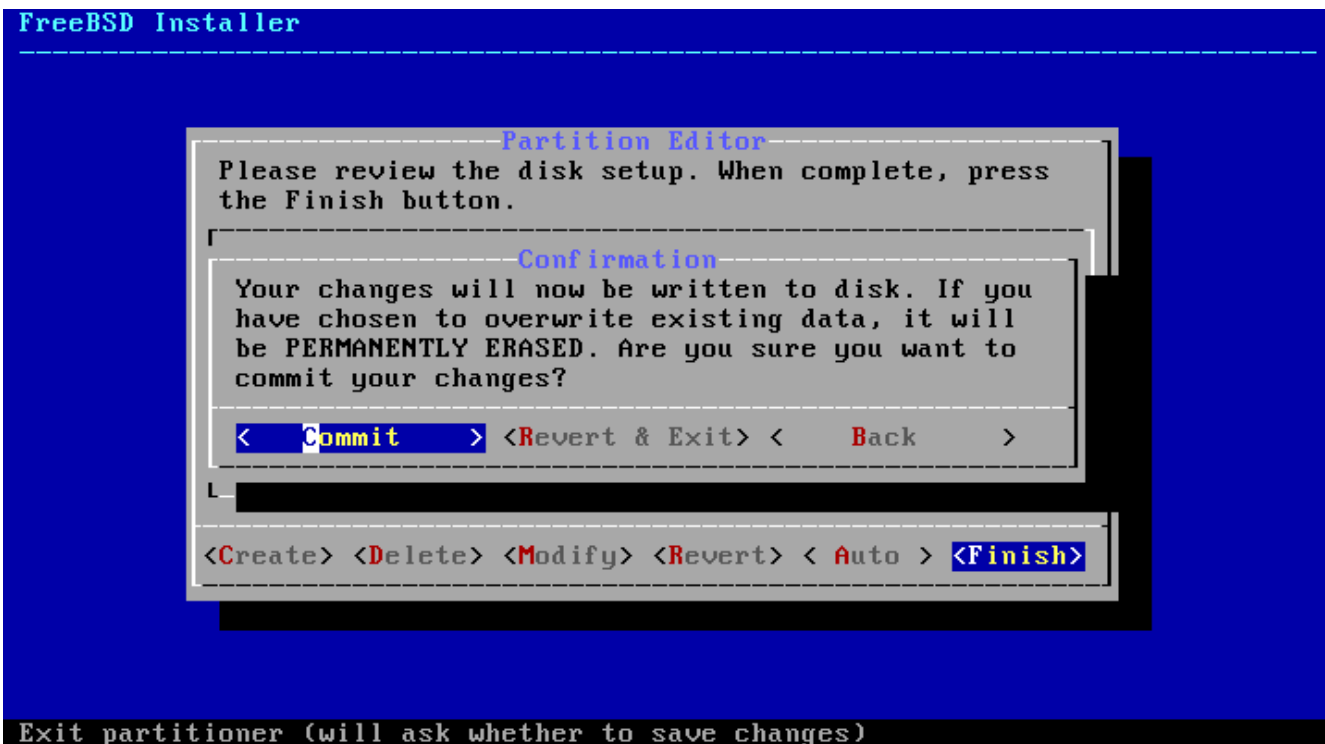
在的分区布局中，目 /、/var、/tmp 及 /user 都是位于自己分区上的独立文件系统；在 GPT 分区方案中也可以建的分区布局。本例中所使用的是一 20G 的硬，如果使用更大的硬，建更大的交或 /var 分区。的前 ex 是指 “example”，具体操作可以使用任何独一无二的字符。

| 分区型          | 大小        | 挂点   |          |
|--------------|-----------|------|----------|
| freebsd-boot | 512K      |      |          |
| freebsd-ufs  | 2G        | /    | exrootfs |
| freebsd-swap | 4G        |      | exswap   |
| freebsd-ufs  | 2G        | /var | exvarfs  |
| freebsd-ufs  | 1G        | /tmp | extmpfs  |
| freebsd-ufs  | 接受默认（剩余空） | /usr | exusrfs  |

建了自定分区后， [ Finish ] 以安装。

3.8. 安装

下面，安装程序将真正硬行写操作，也是取消安装的最后机会。



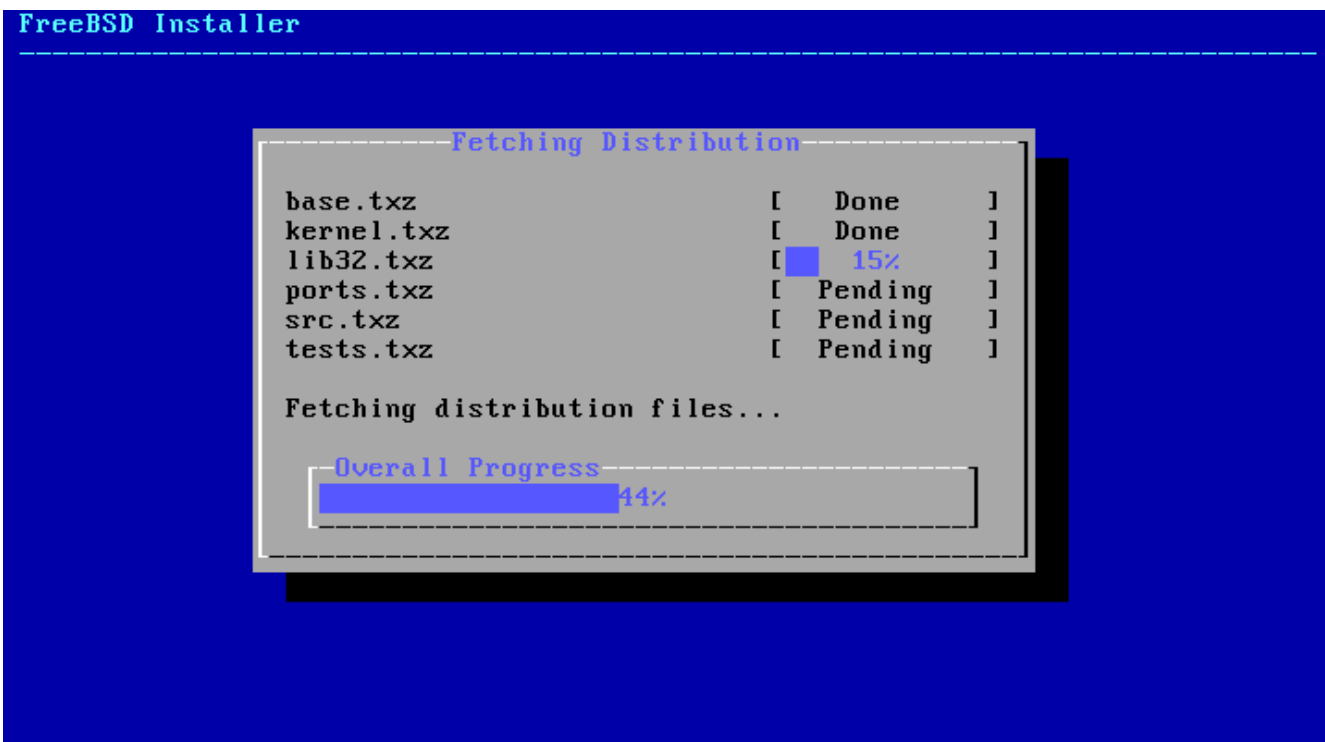
□ 74. 最后□□

□□ **[Commit]** 并按 **Enter** □□□安装； □□ **[Back]** 以返回分区□□器□行修改； □□ **[Revert & Exit]** 以退出安装而不修改任何硬□数据。

根据所□□件、安装介□和机器速度的不同，需要的□□会有所□化。安装□□会有一系列信息□示目前的□度。

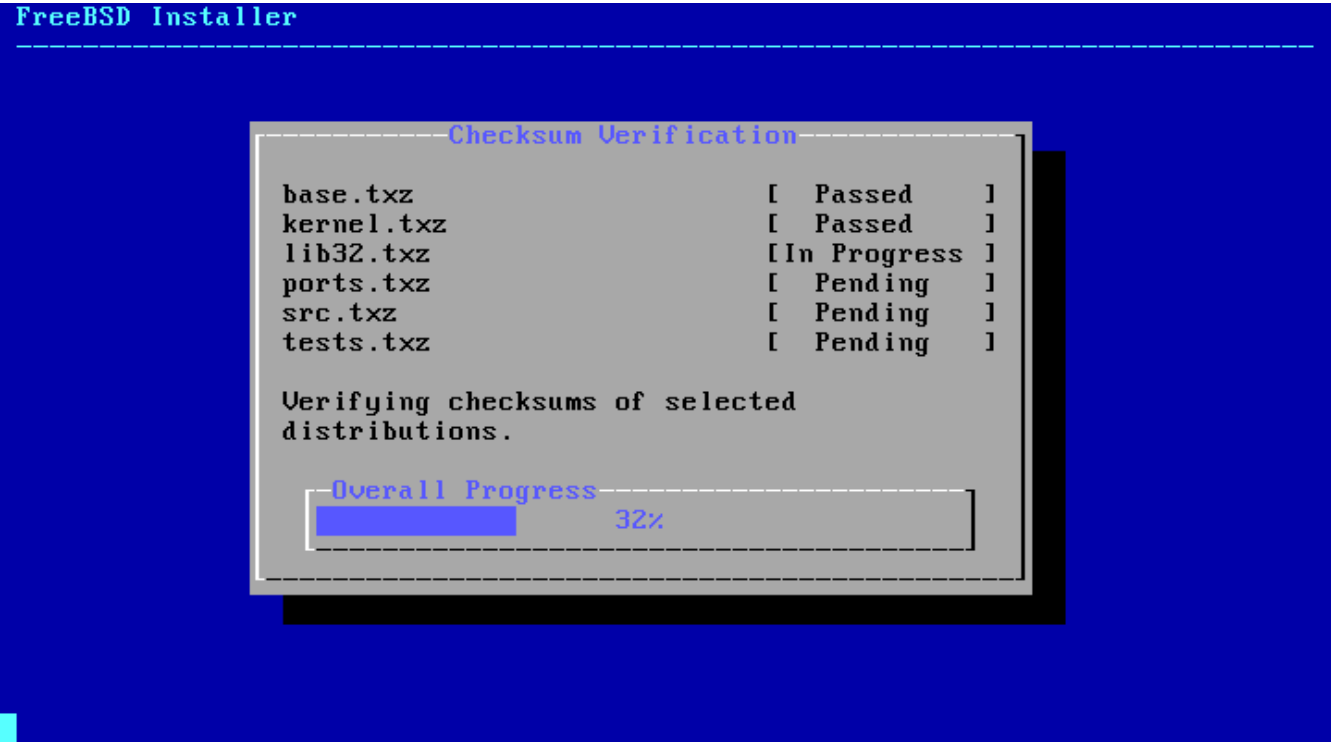
首先，安装程序会将分区布局写入磁□，并□行 **newfs** 初始化分区。

如果是通□网□安装，**bsdinstall** 将根据之前所□的□□件下□□□的文件。



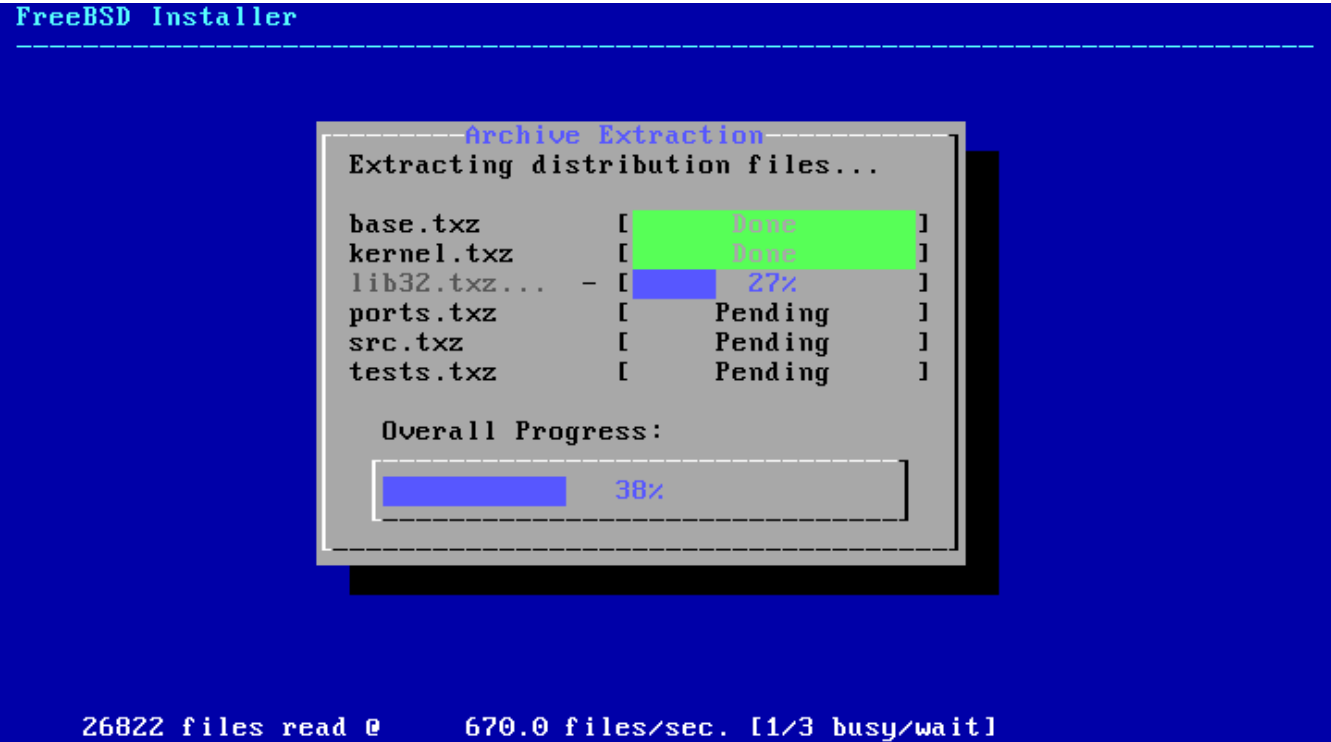
□ 75. □取□□件□□的文件

接下来，会检查文件的完整性，以防止其在下一次损坏或从安装介质中删除。



76. 检查文件的完整性

最后，提取的文件会被提取至磁盘。



77. 提取文件

文件提取全部完成后，bsdinstall 将开始安装后的配置任务（参看 [安装后的配置](#)）。

### 3.9. 安装后的配置

成功安装 FreeBSD 后，需要依次进行一些配置。在重新输入新系统前，一些配置始终可以通过最后的配置菜单

行修改。

### 3.9.1. 设置 root 密码

必须设置 root 密码。注意输入密码时，被输入的字符并不会在屏幕上显示，因此防止输入，必须再次输入相同的字符。

```
FreeBSD Installer
=====

Please select a password for the system management account (root):
Typed characters will not be visible.
Changing local password for root
New Password:
Retype New Password:
```

78. 设置 root 密码

成功设置密码后，安装将继续进行。

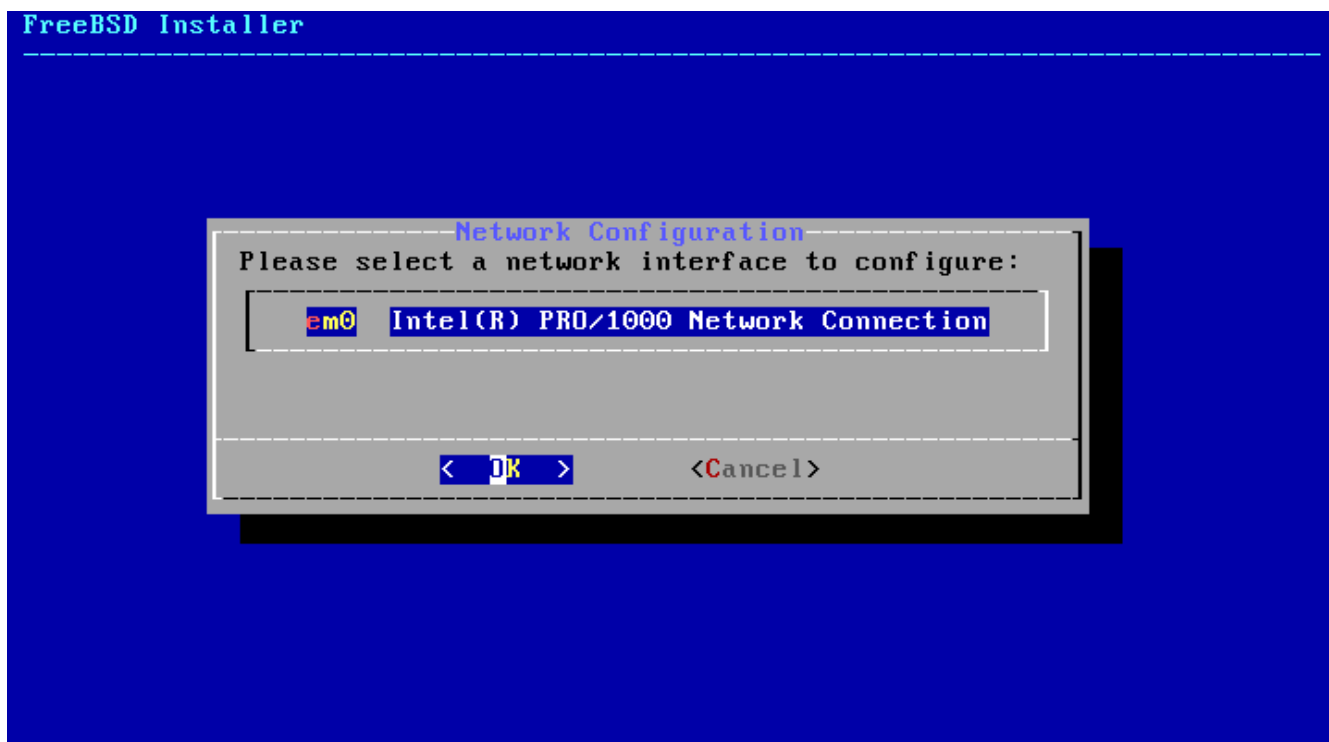
### 3.9.2. 配置网络接口



如果已在 *bootonly* 安装配置网络接口，可略此。

这里将显示一个网络接口列表，其中的接口都是在当前计算机上检测到的，可以一个配置。





□ 79. □□一个网□接口

### 3.9.2.1. 配置无□网□接口

如果□□了无□网□接口，□必□□入相□的无□网□□□及安全参数，以允□其□接至特定的网□。

无□网□是通□ Service Set Identifier（服□集□□符，□写□ SSID）来表示的，它是唯一表示无□网□的短字符串。

大多数无□网□都会以加密方式□□数据，藉此保□信息不被未□授□者□看。□烈建□采用 WPA2 □□加密。旧式的加密□型，如 WEP，几乎没有任何安全性可言。

若要□接至一个无□网□，首先需要□描无□接入点。

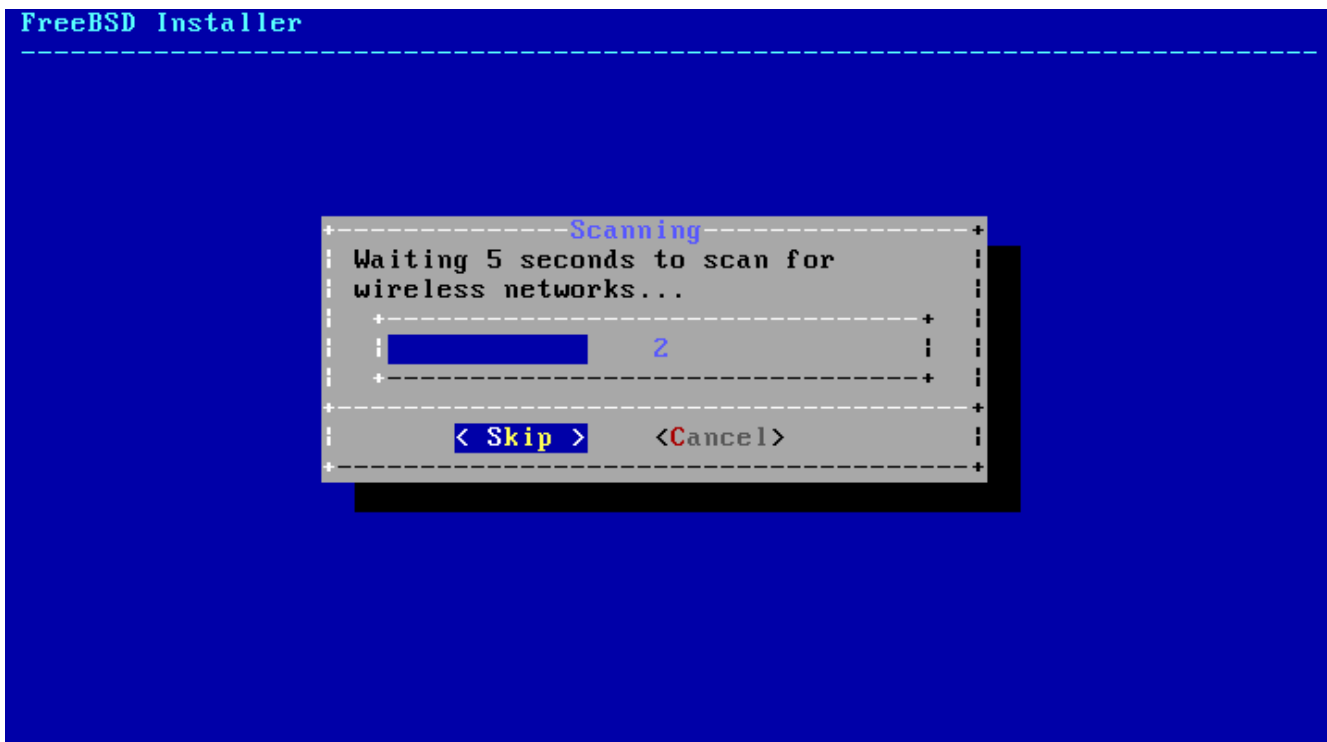


图 80. 扫描无线接入点

扫描完成后，会列出所有的 SSID 以及它支持的加密类型。如果需要连接的 SSID 没有列出，可以按 **[ Rescan ]** 再次扫描。如果没有输出，可能是天线问题，或将计算机移至更接近接入点的地方。在做一些改善措施之后，再重新扫描。

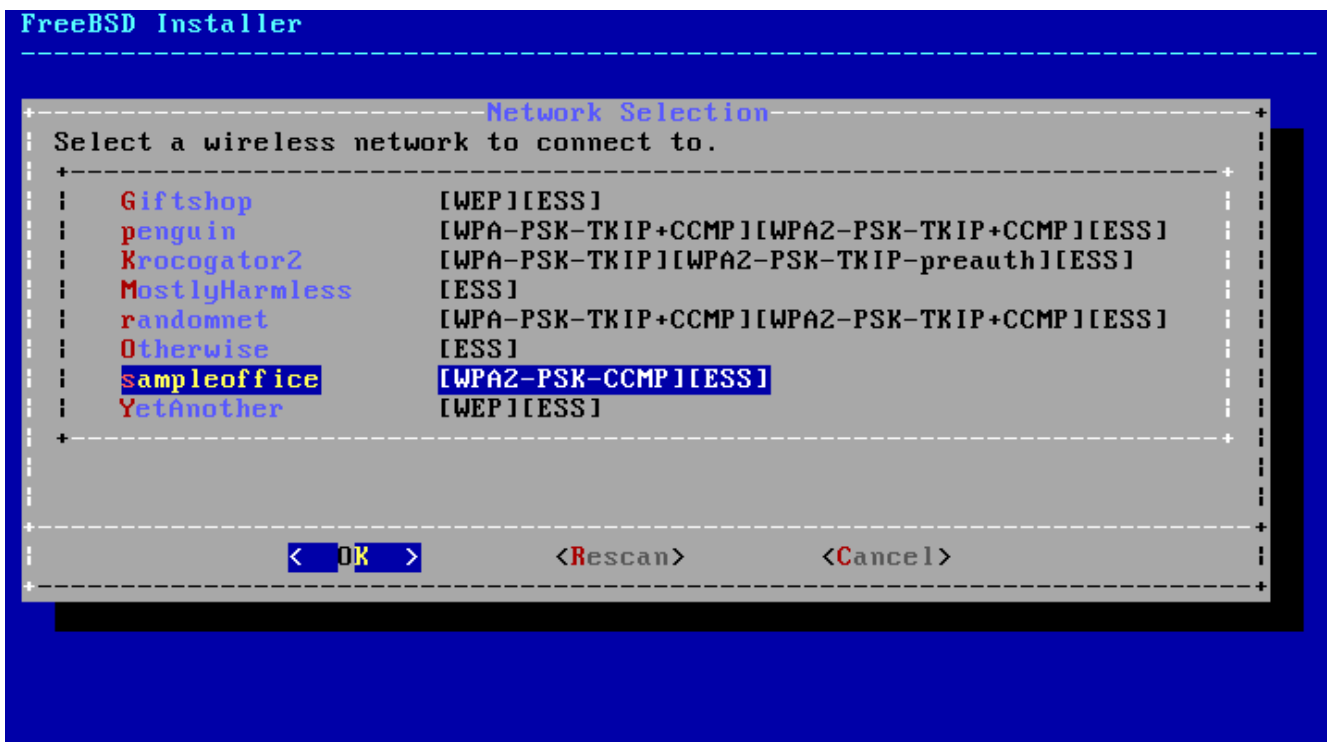


图 81. 选择一个无线网络

当所要连接的无线网络，即可输入连接所需的加密信息。对于 WPA2，只需输入一个密码（也叫共享密码，（即称 PSK）。为安全起见，在输入框中输入的字符将显示为星号。

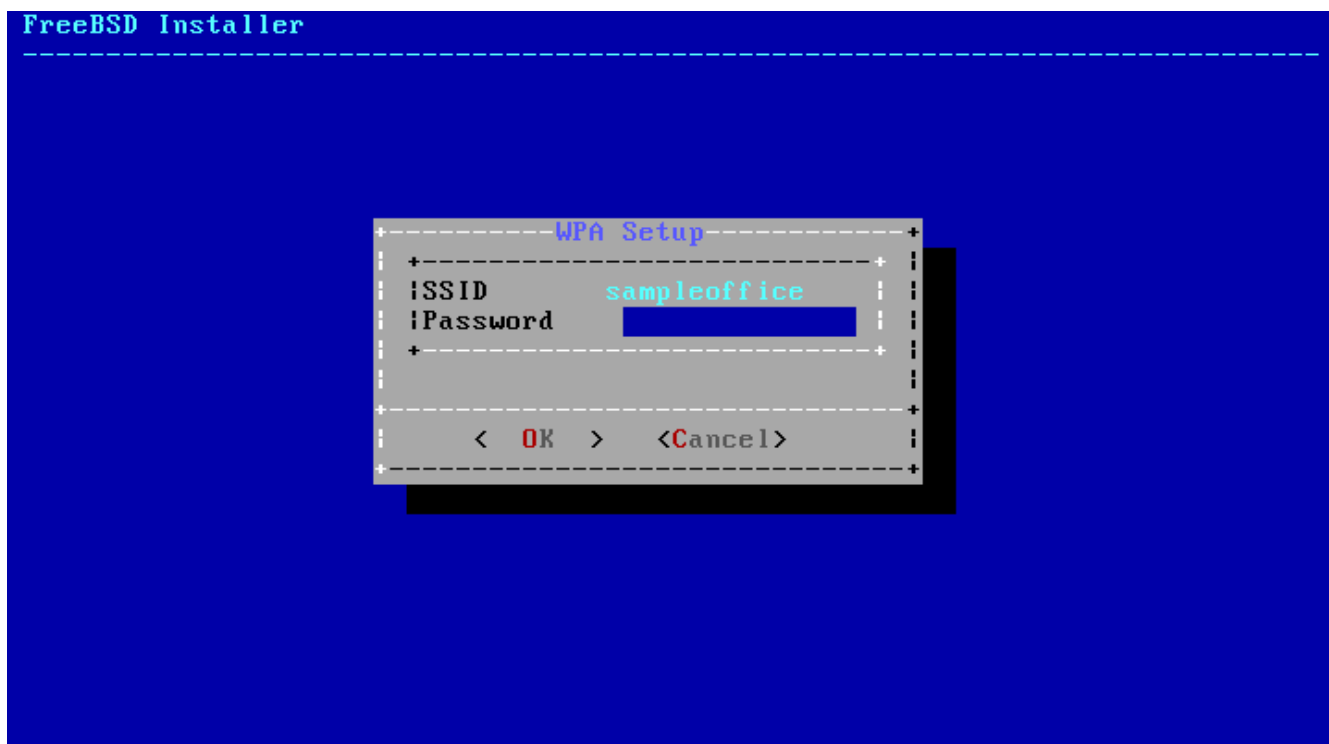


图 82. WPA2 配置

在输入了无线网卡并输入了连接所需的信息后，网卡配置将完成。

### 3.9.2.2. 配置 IPv4 网络

您是否使用 IPv4 网络。它是最常见的网络接口类型。

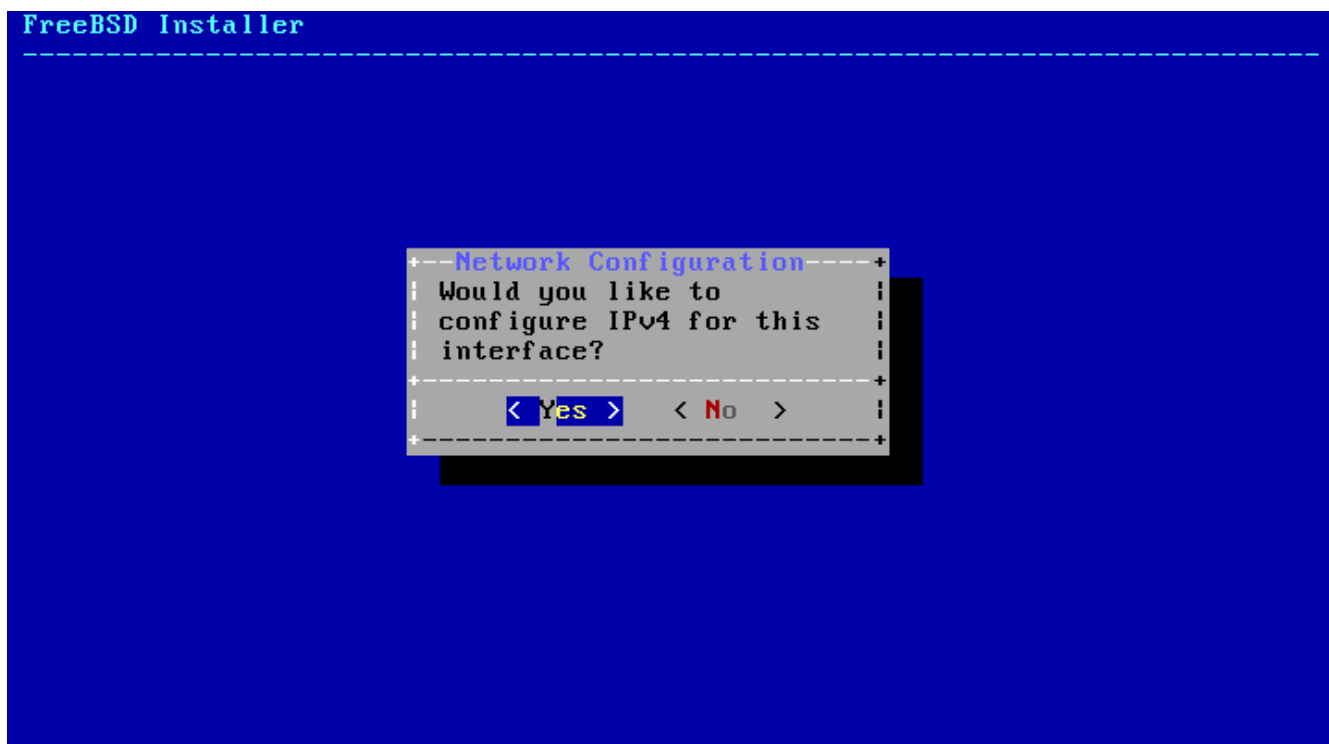


图 83. 配置 IPv4 网络

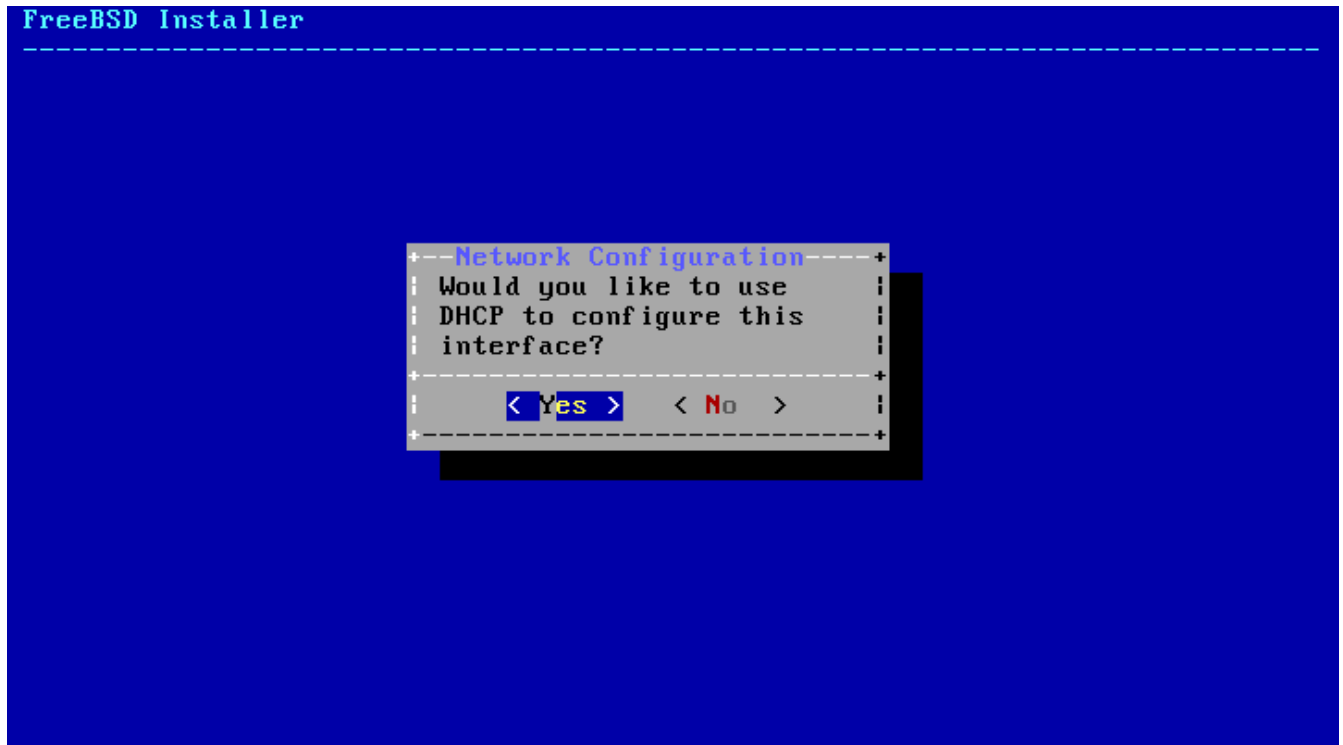
有两种配置 IPv4 的方式。DHCP 会自动地为网络接口进行正确的配置，通常情况下，它是首选的方式。而 Static（静态）方式则需要手工输入网络的配置信息。



不要随意输入网卡的配置信息，因错误的网卡就无法正常工作。向网卡管理或服务提供商那里取得 [收集网卡配置信息](#) 所列出的配置信息。

#### 3.9.2.2.1. 使用 DHCP 方式

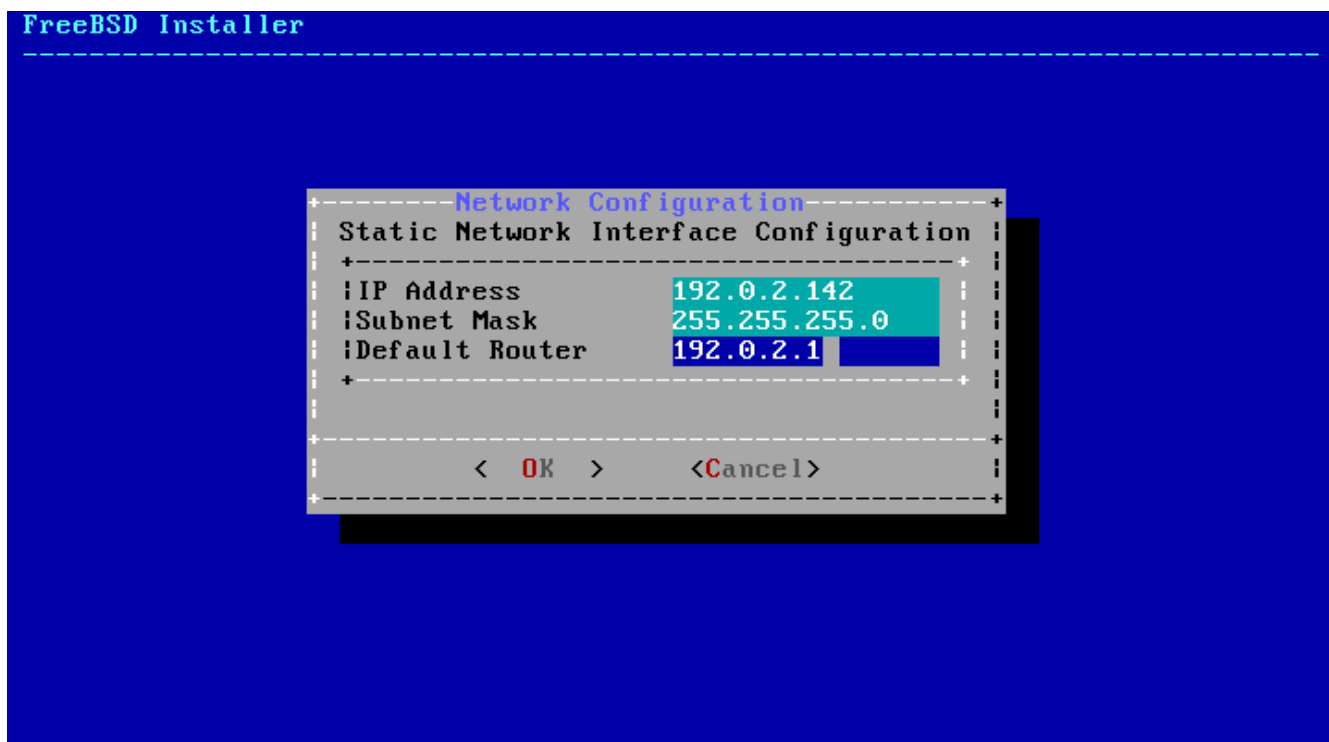
若存在可用的 DHCP 服务器，输入 **[ Yes ]** 以自动配置网卡接口。



84. DHCP 配置 IPv4

#### 3.9.2.2.2. 使用静态配置方式

网卡接口的静态配置需要输入相关的 IPv4 配置信息。



#### □ 85. 静□配置 IPv4

- **IP Address** - IP 地址， 即□当前□计算机手□分配的 IPv4 地址。 此地址必□是唯一的， 并且在本地网□上□没有被其他□□使用。
- **Subnet Mask** - 子网掩□， 用于本地网□。 通常是 **255.255.255.0**。
- **Default Router** (默□路由) - 网□上默□路由的 IP 地址。 通常， □是将本地网□□接至 Internet 的路由器或其他网□□□的地址。 也称作 *default gateway* (默□网□)。

#### 3.9.2.3. 配置 IPv6 网□

IPv6 是一□新的网□配置方式。 如果□有可用的 IPv6 □接， 并需要使用它， □□ **[ Yes ]** 来□始配置。



图 86. 配置 IPv6 网络

IPv6 也有两种配置方式。SLAAC，或 *Stateless Address AutoConfiguration*（无状态地址自动配置）方式能自动配置正确的网络接口，而 *Static*（静态）配置方式则需要手动输入网络信息。

### 3.9.2.3.1. 使用 Stateless Address Autoconfiguration 方式

SLAAC 允许 IPv6 组件从本地路由器请求自动配置信息，详情参见 [RFC4862](#)。

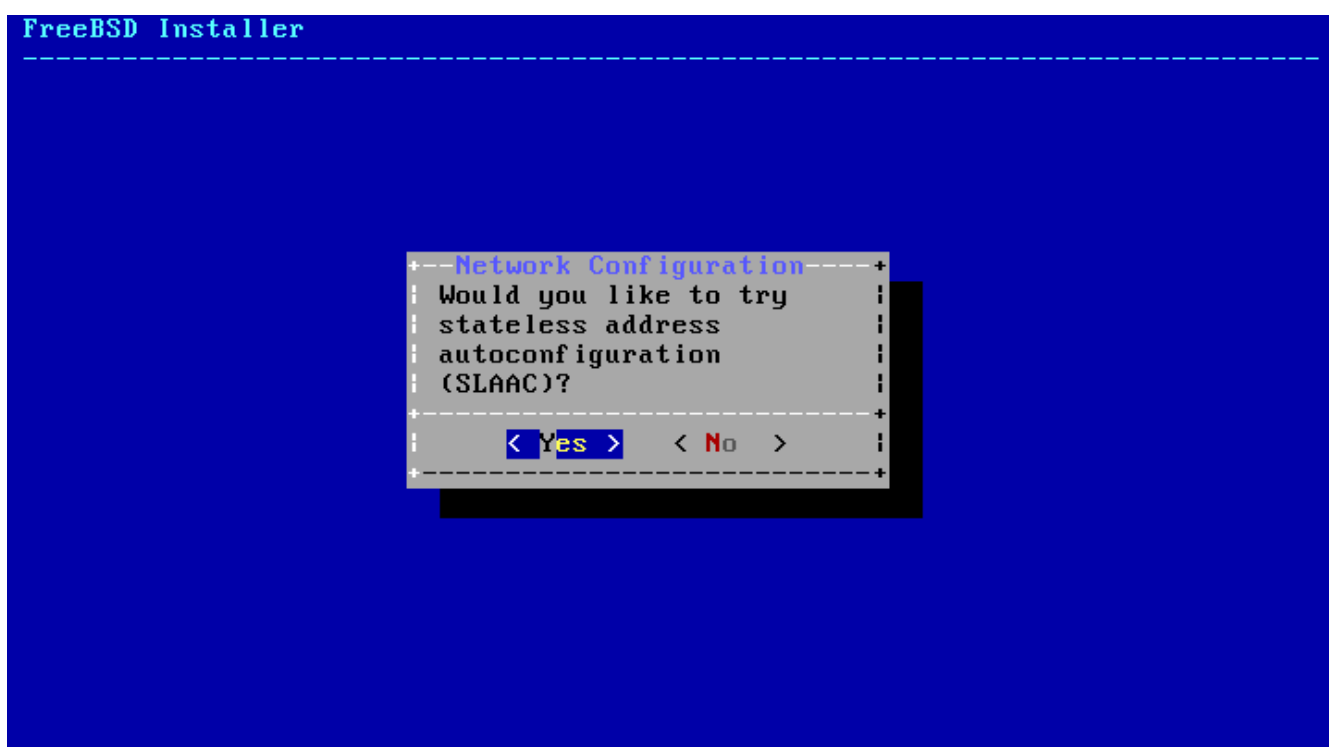
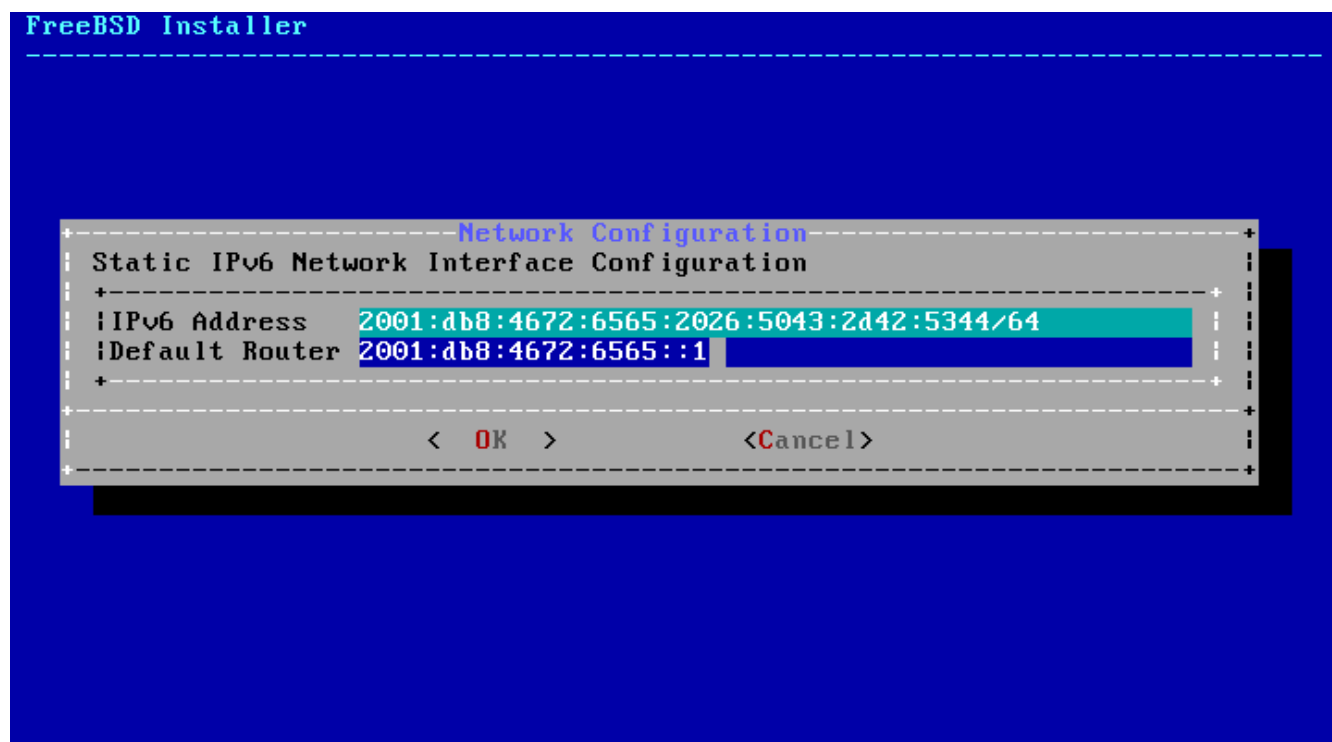


图 87. 配置 SLAAC 配置 IPv6

### 3.9.2.3.2. 使用静态配置方式

网络接口的静态配置需要输入相关的 IPv6 配置信息。

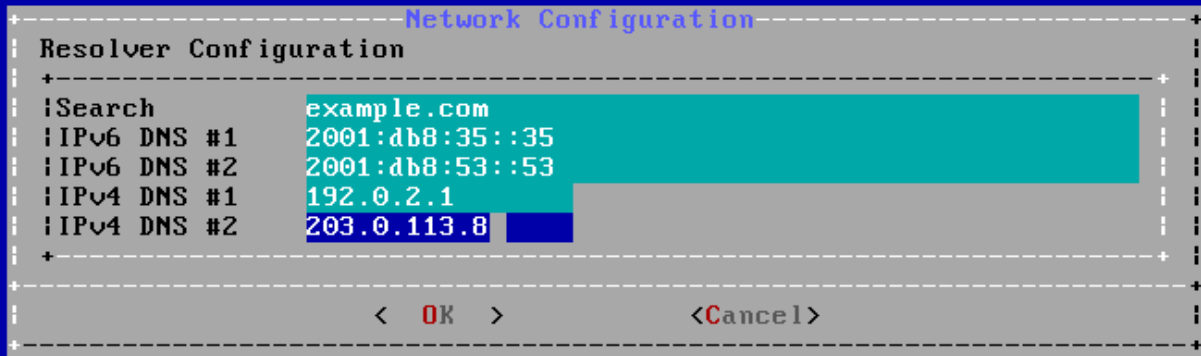


#### 图 88. 静态配置 IPv6

- **IPv6 Address** (IPv6地址) - 当前计算机手工分配的 IP 地址。 该地址必须是唯一的, 并且没有被其他本地网络使用。
- **Default Router** (默认路由) - 网络上默认路由的地址。 通常, 是将本地网络接至 Internet 的路由器或其他网络的地址。 也称作 *default gateway* (默认网关)。

### 3.9.2.4. 配置 DNS

*Domain Name System* (域名系统, 简称 *DNS*) 解析器用于主机名和网络地址的相互映射。 如果使用的是 DHCP 或 SLAAC, 那么其配置很可能已存在; 否则, 在 *Search* 字段中输入本地网络的域名, 在 *DNS #1* 和 *DNS #2* 中输入本地 DNS 服务器的 IP 地址。 至少需要配置一个 DNS 服务器。

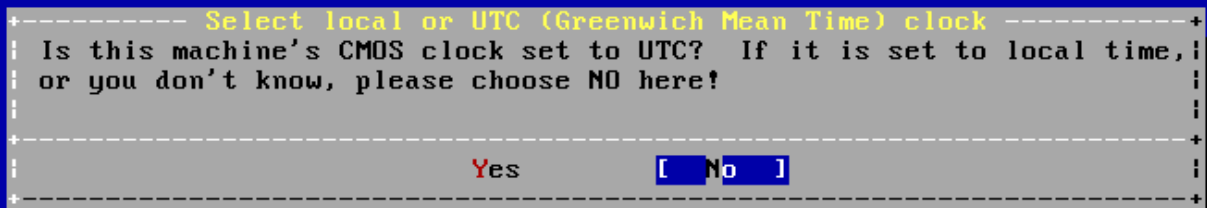


□ 89. DNS 配置

### 3.9.3. □置□区

□□的机器□置□区将允□其自□校□，并正□□行一些与□区相□的操作。

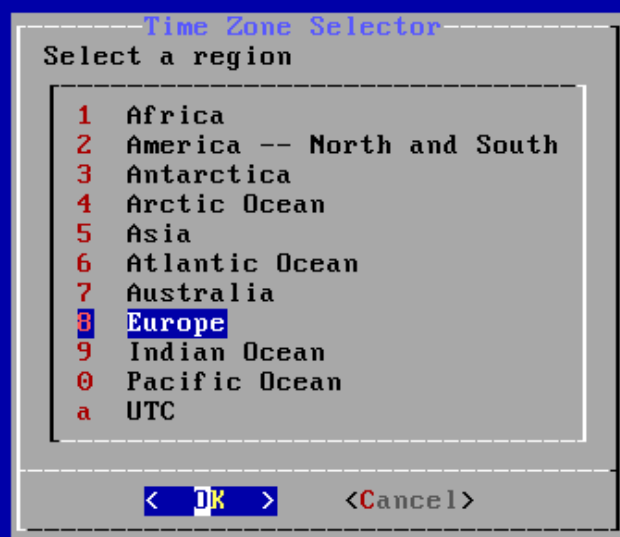
示例中的机器位于美国□部□区。根据所□的地理位置，□的□□可能会有所不同。



□ 90. □□本地或 UTC □□

□□ [ Yes ] 或 [ No ] 以□定机器□□的配置方式，然后按  □。如果□并不知道系□使用的是 UTC □是本地□□，□□□ [ No ] 以使用更□常□的本地□□。





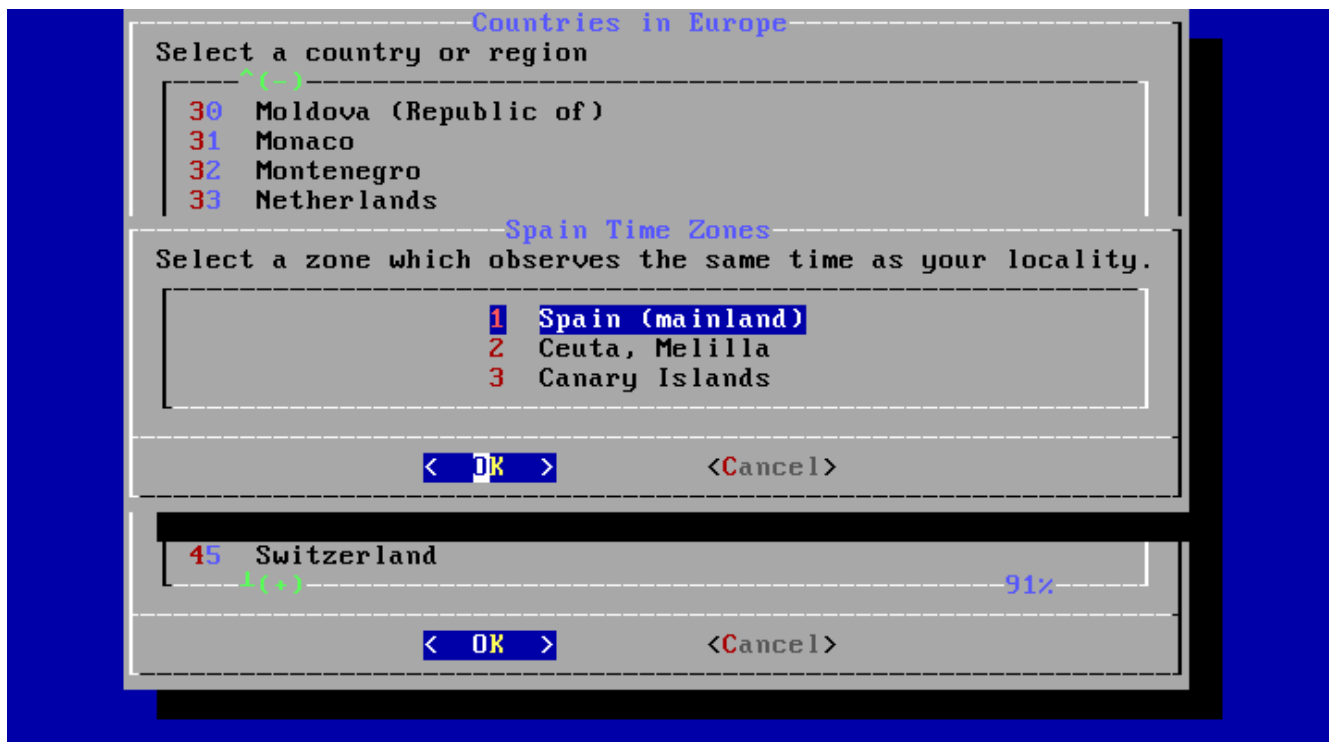
## 91. 地区

使用方向键合的地区后按下 **Enter** 。



## 92. 国家

用方向键合的国家后按下 **Enter** 。



□ 93. □□□□

用方向□□合□的□□后按下  □。



□ 94. □□□□□□

□□□□的□写是正□的，然后按  □以□□安装后的配置。

### 3.9.4. □□需要□□的服□

可以□□□外的系□服□，它□会在系□□□□自□□行。所有□些服□都是可□的。

## System Configuration

Choose the services you would like to be started at boot:

|  |   |
|--|---|
| <input type="checkbox"/> <b>local_unbound</b>      | Local caching validating resolver               |
| <input checked="" type="checkbox"/> <b>sshd</b>    | Secure shell daemon                             |
| <input type="checkbox"/> <b>moused</b>             | PS/2 mouse pointer on console                   |
| <input type="checkbox"/> <b>ntptime</b>            | Synchronize system and network time at boottime |
| <input type="checkbox"/> <b>ntpd</b>               | Synchronize system and network time             |
| <input type="checkbox"/> <b>powerd</b>             | Adjust CPU frequency dynamically if supported   |
| <input checked="" type="checkbox"/> <b>dumpdev</b> | Enable kernel crash dumps to /var/crash         |

&lt; OK &gt;

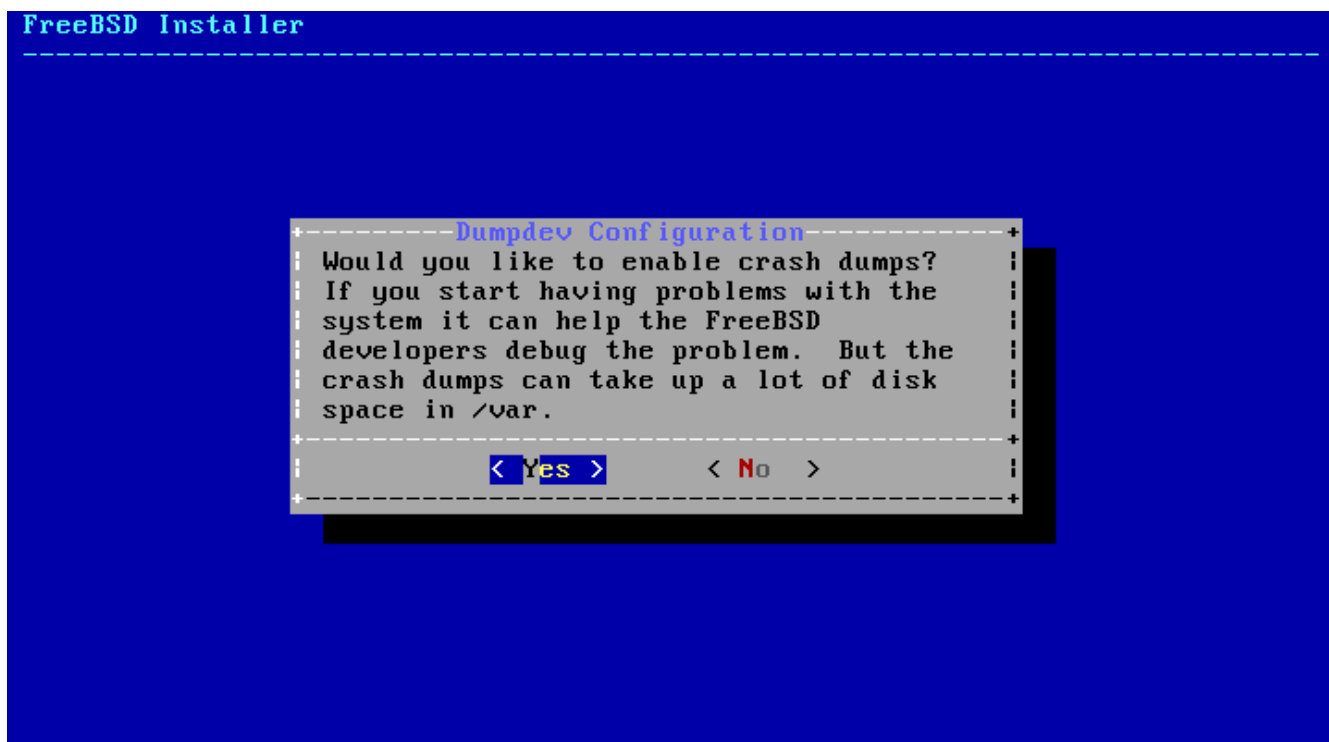
□ 95. □□需要□□的服□

□外的系□服□

- **sshd** - Secure Shell（即 SSH）守□□程，提供安全的□□程□□。
- **moused** - 支持在系□控制台中使用鼠□。
- **ntpd** - Network Time Protocol（网□□□□□，□称 NTP）守□□程，提供□□自□□同□。
- **powerd** - 系□□量控制程序，用于控制□量及□能。

### 3.9.5. □用崩□□□

bsdinstall 将□□是否在目□系□上□用崩□□□。由于在□□系□□非常有用，因此鼓励用□尽可能地□用崩□□□。□□  
**[ Yes ]** 以□用崩□□□，或□□**[ No ]** 以不□用崩□□□。

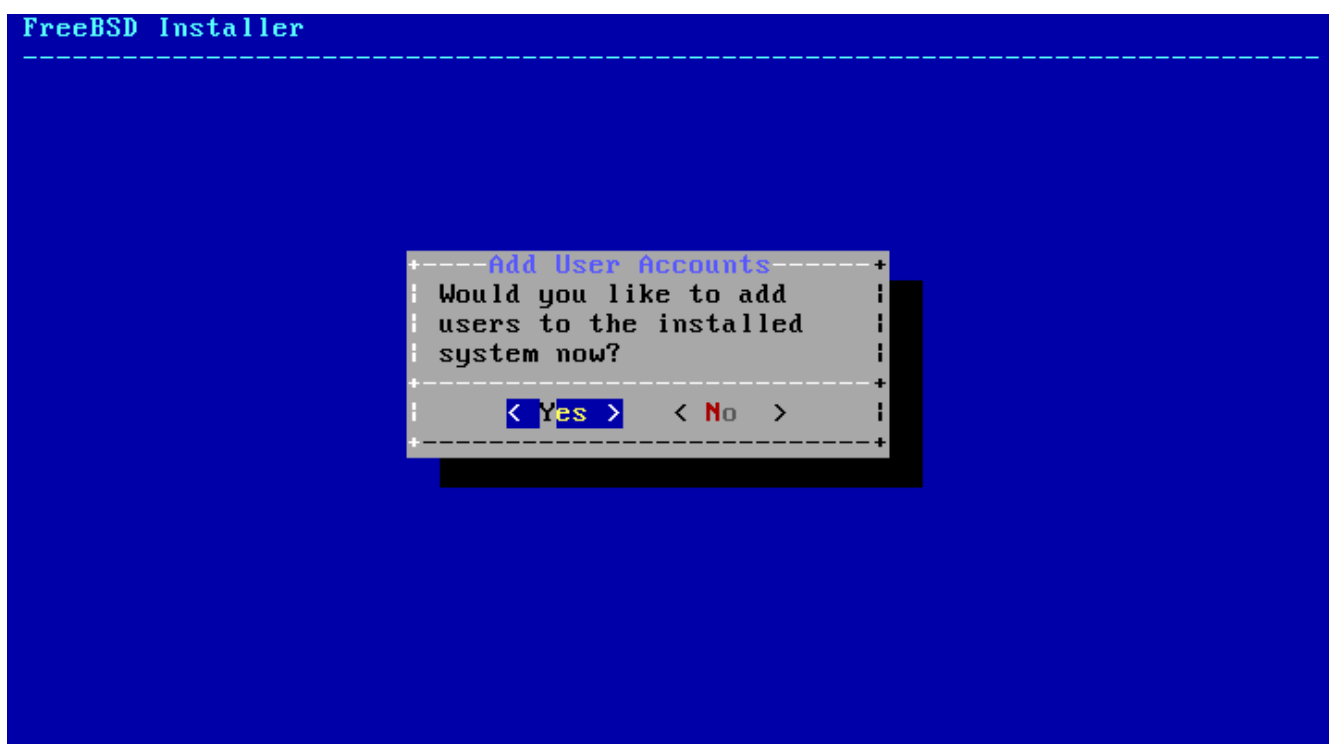


□ 96. □用崩□□

### 3.9.6. 添加用□

在安装□程中，□至少添加一位普通用□，□而不要始□以 **root** 身□登入。当以 **root** 身□登入系□，系□几乎不会□其操作提供任何限制或保□。以普通用□身□登□更□安全。

□□ [ **Yes** ] 来添加新用□。



□ 97. 添加用□□号

□需要添加的用□□入信息。

```

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]: █

```

## ▢ 98. ▢入用▢信息

### 用▢信息

- **Username** - 用▢名，即登入▢用▢所▢入的名称。通常是名的首字母加姓的▢合。
- **Full name** - 用▢的全名。
- **Uid** - 用▢ ID。通常留空以自▢分配。
- **Login group** - 用▢。通常留空以接受默▢取▢。
- **Invite user into other groups?** - 是否同▢将用▢加入其他▢限▢？如果需要，▢▢入▢限▢名称。
- **Login class** - 登▢▢▢。通常留空以接受默▢取▢。
- **Shell** - 用▢ shell。在本例中▢▢的是 **csh(1)**。
- **Home directory** - 用▢主目▢。通常留空以接受默▢取▢。
- **Home directory permissions** - 用▢主目▢的▢限。通常留空以接受默▢取▢。
- **Use password-based authentication?** - 是否使用基于密▢的▢▢？通常▢“yes”。
- **Use an empty password?** - 是否使用空密▢？通常▢“no”。
- **Use a random password?** - 是否使用随机密▢？通常▢“no”。
- **Enter password** - 用▢的▢▢密▢。▢入的字符不会在屏幕上▢示。
- **Enter password again** - 必▢再次▢入密▢以▢行▢▢。
- **Lock out the account after creation?** - ▢建后▢定▢号？通常▢“no”。

全部信息▢入完成后，系▢会▢示摘要并▢▢是否正▢。如果▢▢了▢▢，可以▢入 **no** 后▢行修改；如果没有▢▢，▢▢入 **yes** 以▢建新用▢。

```

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):

```

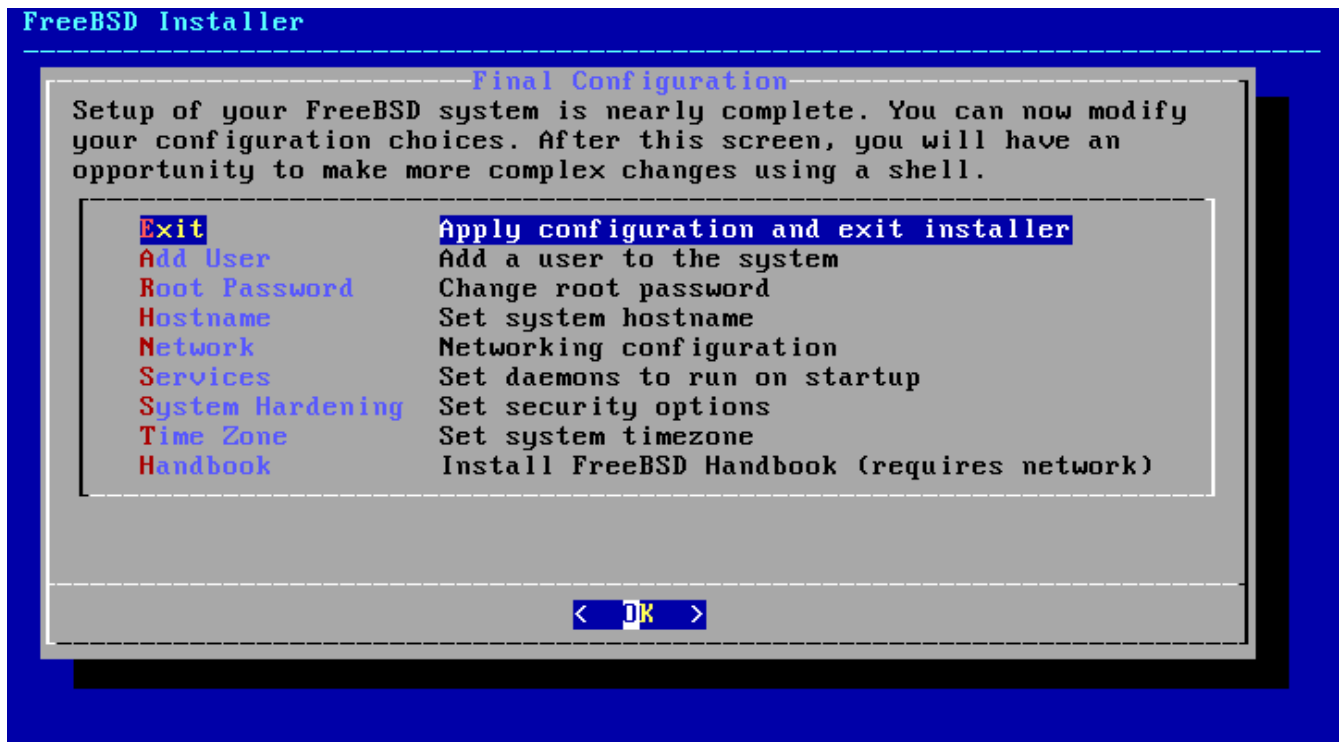
▢ 99. 退出用▢与▢管理

若需添加更多用▢，▢在▢▢"Add another user?"后▢入 `yes`；▢入 `no` 以完成用▢添加并▢▢安装。

更多有▢用▢添加及管理的信息，▢参▢ [用▢和基本的▢▢管理](#)。

### 3.9.7. 最▢配置

所有的安装及配置完成后，仍有机会▢其▢行修改。



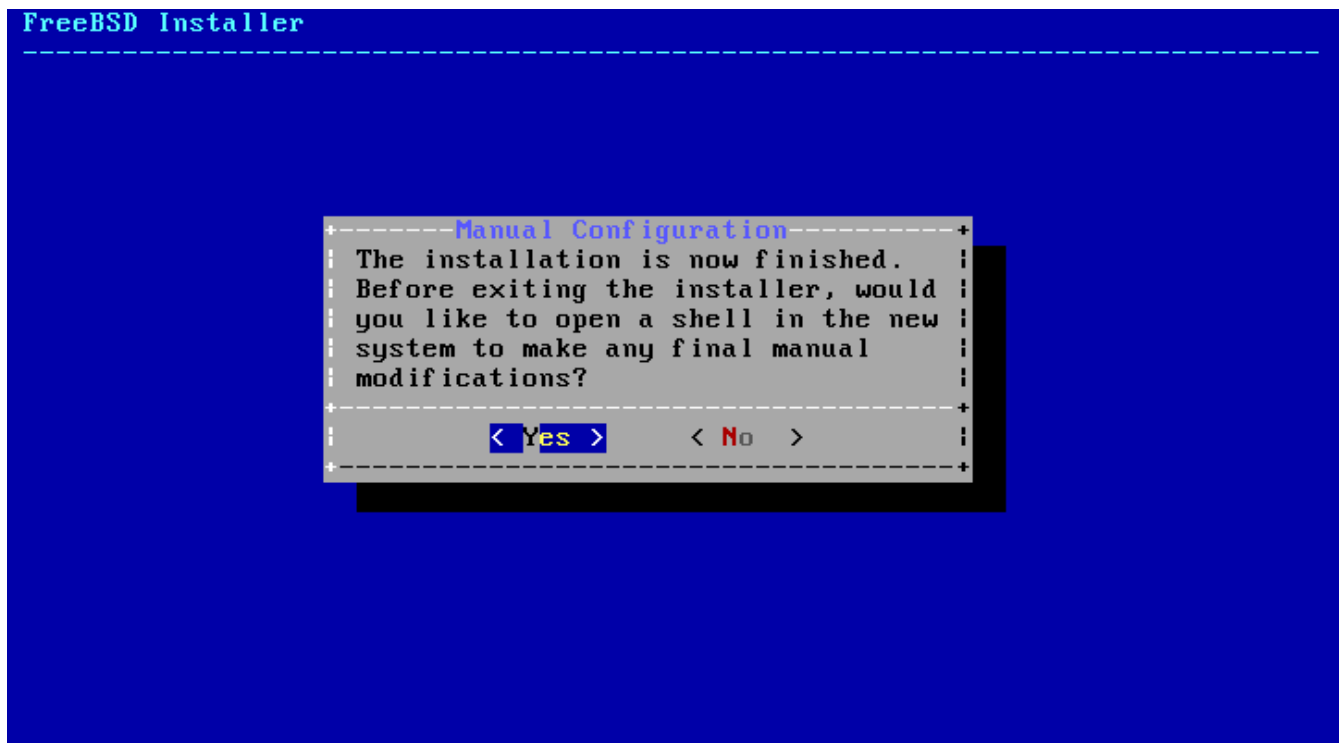
▢ 100. 最▢的配置菜▢

使用此菜▢，可以在完成安装前添加或修改任何配置。

最后的配置

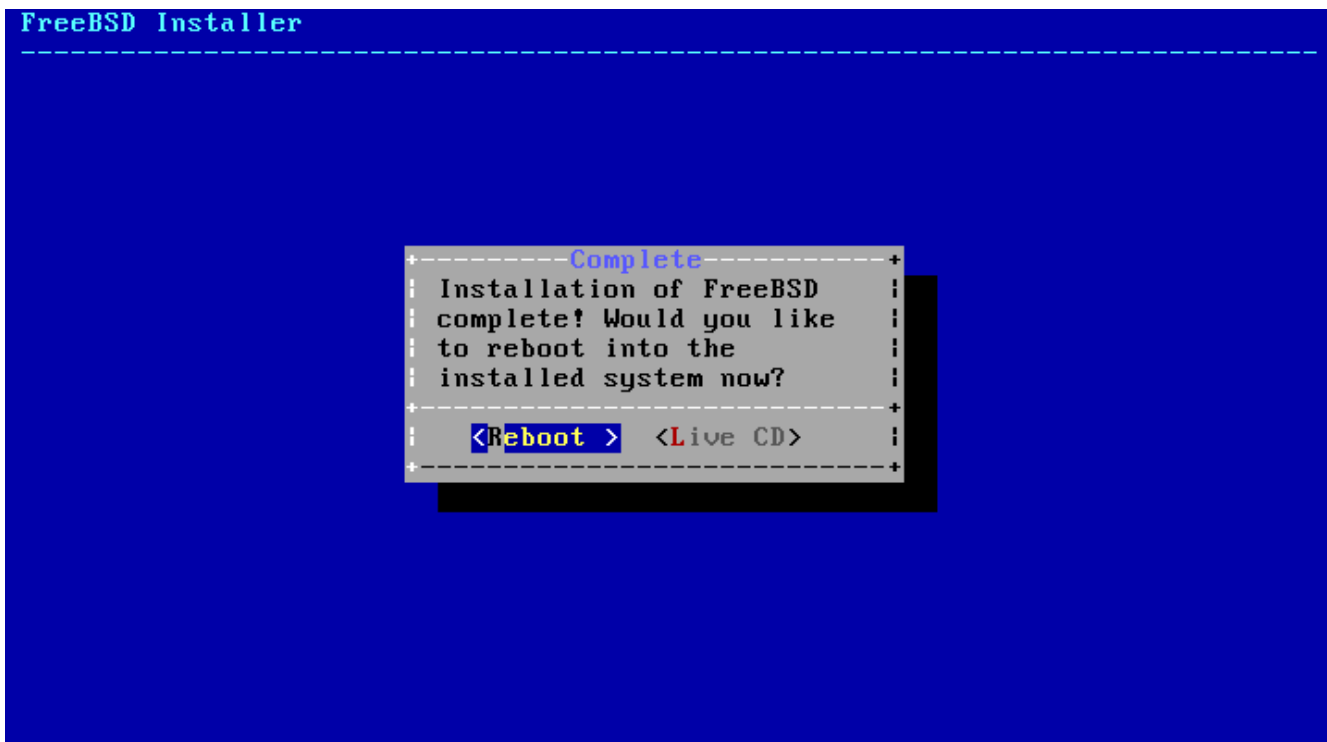
- **Add User** - 添加用户， 添加用户。
- **Root Password** - root 密码， 设置 root 密码。
- **Hostname** - 主机名， 设置主机名。
- **Network** - 网， 配置网接口。
- **Services** - 服务， 需要哪些的服务。
- **Time Zone** - 时区， 设置时区。
- **Handbook** - 手册， 将下并安装 FreeBSD 使用手册（即本）。

完成了最后配置后， 按 **[Exit]** 以安装。



101. 手动配置

bsdinstall 会重问前是否需要额外的配置： 按 **[Yes]** 入 shell 做一些配置， 按 **[No]** 以行安装的最后一。



## □ 102. 完成安装

如果需要□一的配置或特殊的□置，可以□□ **[Live CD]** 来□入安装介□的 Live CD 模式。

安装完成后，□□ **[Reboot]** 重□算机，并□始使用全新的 FreeBSD 系□。□不要忘□移除 FreeBSD 的安装 CD、DVD 或 USB □□棒，否□算机可能会再次从□些介□□□。

## 3.9.8. FreeBSD 的□□与□□

### 3.9.8.1. FreeBSD/i386 的□□

FreeBSD □□□会□示□多相□信息，正常情况下屏幕会不断□□，而□□完成后□会□示一个登□提示符。如果需要□看□□□的相□信息，可以按下 `Scroll-Lock` □□□ *scroll-back buffer* (回□□存)，然后使用 `PageUp` □、`PageDown` □与方向□行翻□；再次按下 `Scroll Lock` □将□□回□□存并返回正常的屏幕。

在 `login:` 提示符□□入安装□添加的用□名来登□系□，本例中是 `asample`。除非有必要，否□□勿作□ `root` 登□。

上述的回□□存大小有限，因而未必全部可□。登入系□后，在提示符□□入 `dmesg | less`，能□□看到□大部分的□□信息，□看后按 `q` □返回命令行。

典型的□□信息（此□略去了版本信息）：

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

root@farrell.cse.buffalo.edu:/usr/obj/usr/src/sys/GENERIC amd64
CPU: Intel(R) Core(TM)2 Duo CPU      E8400  @ 3.00GHz (3007.77-MHz K8-class CPU)
Origin = "GenuineIntel" Id = 0x10676  Family = 6  Model = 17  Stepping = 6
Features
```



```

=0x783fbfff<FPU,VME,DE,PSE,TSC,MSR,PAE,MCE,CX8,APIC,SEP,MTRR,PGE,MCA,CMOV,PAT,PSE36,MMX
,FXSR,SSE,SSE2>
Features2=0x209<SSE3,MON,SSSE3>
AMD Features=0x20100800<SYSCALL,NX,LM>
AMD Features2=0x1<LAHF>
real memory = 536805376 (511 MB)
avail memory = 491819008 (469 MB)
Event timer "LAPIC" quality 400
ACPI APIC Table: <VBOX VBOXAPIC>
ioapic0: Changing APIC ID to 1
ioapic0 <Version 1.1> irqs 0-23 on motherboard
kbd1 at kbdmux0
acpi0: <VBOX VBOXXSDT> on motherboard
acpi0: Power Button (fixed)
acpi0: Sleep Button (fixed)
Timecounter "ACPI-fast" frequency 3579545 Hz quality 900
acpi_timer0: <32-bit timer at 3.579545MHz> port 0x4008-0x400b on acpi0
cpu0: <ACPI CPU> on acpi0
pcib0: <ACPI Host-PCI bridge> port 0xcf8-0xcff on acpi0
pci0: <ACPI PCI bus> on pcib0
isab0: <PCI-ISA bridge> at device 1.0 on pci0
isa0: <ISA bus> on isab0
atapci0: <Intel PIIX4 UDMA33 controller> port 0x1f0-0x1f7,0x3f6,0x170-
0x177,0x376,0xd000-0xd00f at device 1.1 on pci0
ata0: <ATA channel 0> on atapci0
ata1: <ATA channel 1> on atapci0
vgapci0: <VGA-compatible display> mem 0xe0000000-0xe0ffffff irq 18 at device 2.0 on
pci0
em0: <Intel(R) PRO/1000 Legacy Network Connection 1.0.3> port 0xd010-0xd017 mem
0xf0000000-0xf001ffff irq 19 at device 3.0 on pci0
em0: Ethernet address: 08:00:27:9f:e0:92
pci0: <base peripheral> at device 4.0 (no driver attached)
pcm0: <Intel ICH (82801AA)> port 0xd100-0xd1ff,0xd200-0xd23f irq 21 at device 5.0 on
pci0
pcm0: <SigmaTel STAC9700/83/84 AC97 Codec>
ohci0: <OHCI (generic) USB controller> mem 0xf0804000-0xf0804fff irq 22 at device 6.0
on pci0
usb0: <OHCI (generic) USB controller> on ohci0
pci0: <bridge> at device 7.0 (no driver attached)
acpi_acad0: <AC Adapter> on acpi0
atkbdc0: <Keyboard controller (i8042)> port 0x60,0x64 irq 1 on acpi0
atkbd0: <AT Keyboard> irq 1 on atkbdc0
kbd0 at atkbd0
atkbd0: [GIANT-LOCKED]
psm0: <PS/2 Mouse> irq 12 on atkbdc0
psm0: [GIANT-LOCKED]
psm0: model IntelliMouse Explorer, device ID 4
attimer0: <AT timer> port 0x40-0x43,0x50-0x53 on acpi0
Timecounter "i8254" frequency 1193182 Hz quality 0
Event timer "i8254" frequency 1193182 Hz quality 100
sc0: <System console> at flags 0x100 on isa0

```

```

sc0: VGA <16 virtual consoles, flags=0x300>
vga0: <Generic ISA VGA> at port 0x3c0-0x3df iomem 0xa0000-0xbffff on isa0
atrtc0: <AT realtime clock> at port 0x70 irq 8 on isa0
Event timer "RTC" frequency 32768 Hz quality 0
ppc0: cannot reserve I/O port range
Timecounters tick every 10.000 msec
pcm0: measured ac97 link rate at 485193 Hz
em0: link state changed to UP
usb0: 12Mbps Full Speed USB v1.0
ugen0.1: <Apple> at usb0
uhub0: <Apple OHCI root HUB, class 9/0, rev 1.00/1.00, addr 1> on usb0
cd0 at ata1 bus 0 scbus1 target 0 lun 0
cd0: <VBOX CD-ROM 1.0> Removable CD-ROM SCSI-0 device
cd0: 33.300MB/s transfers (UDMA2, ATAPI 12bytes, PIO 65534bytes)
cd0: Attempt to query device size failed: NOT READY, Medium not present
ada0 at ata0 bus 0 scbus0 target 0 lun 0
ada0: <VBOX HARDDISK 1.0> ATA-6 device
ada0: 33.300MB/s transfers (UDMA2, PIO 65536bytes)
ada0: 12546MB (25694208 512 byte sectors: 16H 63S/T 16383C)
ada0: Previously was known as ad0
Timecounter "TSC" frequency 3007772192 Hz quality 800
Root mount waiting for: usb0
uhub0: 8 ports with 8 removable, self powered
Trying to mount root from ufs:/dev/ada0p2 [rw]...
Setting hostuuid: 1848d7bf-e6a4-4ed4-b782-bd3f1685d551.
Setting hostid: 0xa03479b2.
Entropy harvesting: interrupts ethernet point_to_point kickstart.
Starting file system checks:
/dev/ada0p2: FILE SYSTEM CLEAN; SKIPPING CHECKS
/dev/ada0p2: clean, 2620402 free (714 frags, 327461 blocks, 0.0% fragmentation)
Mounting local file systems:.
vboxguest0 port 0xd020-0xd03f mem 0xf0400000-0xf07fffff,0xf0800000-0xf0803fff irq 20
at device 4.0 on pci0
vboxguest: loaded successfully
Setting hostname: machine3.example.com.
Starting Network: lo0 em0.
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> metric 0 mtu 16384
options=3<RXCSUM,TXCSUM>
inet6 ::1 prefixlen 128
inet6 fe80::1%lo0 prefixlen 64 scopeid 0x3
inet 127.0.0.1 netmask 0xff000000
nd6 options=21<PERFORMNUD,AUTO_LINKLOCAL>
em0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
options=9b<RXCSUM,TXCSUM,VLAN_MTU,VLAN_HWTAGGING,VLAN_HWCSUM>
ether 08:00:27:9f:e0:92
nd6 options=29<PERFORMNUD,IFDISABLED,AUTO_LINKLOCAL>
media: Ethernet autoselect (1000baseT <full-duplex>)
status: active

Starting devd.
Starting Network: usb0.
DHCPREQUEST on em0 to 255.255.255.255 port 67

```

```

DHCPACK from 10.0.2.2
bound to 192.168.1.142 -- renewal in 43200 seconds.
add net ::ffff:0.0.0.0: gateway ::1
add net ::0.0.0.0: gateway ::1
add net fe80:: gateway ::1
add net ff02:: gateway ::1
ELF ldconfig path: /lib /usr/lib /usr/lib/compat /usr/local/lib
32-bit compatibility ldconfig path: /usr/lib32
Creating and/or trimming log files.
Starting syslogd.
No core dumps found.
Clearing /tmp (X related).
Updating motd:.
Configuring syscons: blanktime.
Generating public/private rsa1 key pair.
Your identification has been saved in /etc/ssh/ssh_host_key.
Your public key has been saved in /etc/ssh/ssh_host_key.pub.
The key fingerprint is:
10:a0:f5:af:93:ae:a3:1a:b2:bb:3c:35:d9:5a:b3:f3 root@machine3.example.com
The key's randomart image is:
+--[RSA1 1024]-----+
|    o..          |
|    o . .        |
|    .  o         |
|        o        |
|    o  S         |
|    + + o        |
|o . + *         |
|o+ ..+ .        |
|==o..o+E        |
+-----+
Generating public/private dsa key pair.
Your identification has been saved in /etc/ssh/ssh_host_dsa_key.
Your public key has been saved in /etc/ssh/ssh_host_dsa_key.pub.
The key fingerprint is:
7e:1c:ce:dc:8a:3a:18:13:5b:34:b5:cf:d9:d1:47:b2 root@machine3.example.com
The key's randomart image is:
+--[ DSA 1024]-----+
|      ..      . . |
|      o . . . +   |
|      . . . . E . |
|      . . o o . . |
|      + S = .     |
|      + . = o     |
|      + . * .     |
|      . . o .     |
|      .o. .       |
+-----+
Starting sshd.
Starting cron.
Starting background file system checks in 60 seconds.

```

Thu Oct 6 19:15:31 MDT 2011

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

login:

在慢的机器上，生成 RSA 和 DSA 密钥可能需要一些时间。这种情况只会在安装了 sshd 的新系统首次启动时发生，之后的启动速度不受影响。

FreeBSD 默认情况下并不会安装图形环境，但提供了多种不同的选择。请参考 [X Window 系](#) 了解详情。

### 3.9.9. 重启 FreeBSD

正常重启 FreeBSD 有助于保护数据及系统硬件不受损坏。不要直接断电。如果用的是 `wheel` 的发行版，首先在命令行中输入 `su` 后输入 `root` 密码或超用户。此外，也可作 `root` 登录，然后使用命令 `shutdown -p now`。系统将安全地自行重启。

当然也可以使用组合键 `Ctrl + Alt + Del` 重启系统，但正常情况下并不推荐这样做。

## 3.10. 故障排除

下面将介绍如何排除基本的安装故障，例如用通常警告的信息。

### 3.10.1. 遇到错误时如何处理

由于 PC 架构的各种限制，硬件不可能 100% 地可探测，然而，当此现象发生时，有可能可以通过一些操作来自行解决它。

首先应根据所安装的 FreeBSD 版本核对 [硬件兼容说明](#)，以确保其支持的硬件。

如果使用被支持的硬件仍遇到了死机或其他问题，需要一个 [自定义内核](#)，即可为那些 GENERIC 内核中不存在的设备提供支持。引导上的内核假定大多数硬件的 IRQ、IO 地址和 DMA 通道均出厂设置，如果硬件被重新配置，就很可能需要修改内核配置文件并重新编译内核，以支持 FreeBSD 某些硬件。

可能出现一种情况，某个不存在的设备会导致后其他存在的设备失效。在这种情况下，禁止引起冲突的设备所运行的程序。



有些安装问题可以通过更新硬件固件来避免或改善，尤其是主板。主板固件通常被称作 BIOS，大多数主板和计算机制造商都有提供升级和相关信息网站。

制造商通常建议，除非有似乎更新必要的理由，否则避免升级主板 BIOS。升级过程一旦出错，BIOS 信息将遭到破坏，从而导致计算机无法工作。

### 3.10.2. 故障排除问答

**3.10.2.1. 在引导时，我的系统在硬件上挂起，或在安装过程中行不通。**

在 i386、amd64 和 ia64 平台的安装过程中，FreeBSD 广泛使用了 ACPI 服务来配置系统，不幸的是 ACPI

BIOS 和主板 BIOS 中仍存在一些 bug。在第三阶段引导加载器中，可以通过设置 `hint.acpi.0.disabled` 来禁用 ACPI：

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

此设置会在系统重启后失效，因此必须将 `hint.acpi.0.disabled="1"` 添加至文件 `/boot/loader.conf` 中。关于引导加载器的更多信息，参见 [概述](#)。

# Chapter 4. UNIX 基

## 4.1. 概述

下列章的命令和功能用于FreeBSD操作系统。同里多内容和一些 UNIX® 操作系统相。假如已熟悉些内容可跳不。假如是FreeBSD新手，那真地从头到尾一遍些章。

取些内容，将了解：

- 在FreeBSD使用 "虚控制台"。
- 在 UNIX® 中文件限如何作，以及理解 FreeBSD 中的文件志。
- FreeBSD 默文件系统架。
- FreeBSD磁架。
- 挂接或卸下文件系统。
- 什是程、守程、信号。
- 什是shell，当去改登入的默境。
- 使用基本的文本器。
- 什是，什是点。
- FreeBSD 下，使用的是什可行文件格式。
- 使用 man 手册并取得更多。

## 4.2. 虚控制台和端

可以用多不同的方式使用 FreeBSD，在文本端入命令是其中之一。通使用方式，可以容易地使用 FreeBSD 来得 UNIX® 操作系的活而大的功能。一将介 "端" 和 "控制台"，以及如何在 FreeBSD 中使用它。

### 4.2.1. 控制台

假如没有置 FreeBSD 在期形登界面，那系将在引和脚本正行完成后，一个登的提示。会看到似界面：

```
Additional ABI support:.
Local package initialization:.
Additional TCP options:.

Fri Sep 20 13:01:06 EEST 2002

FreeBSD/i386 (pc3.example.org) (ttyv0)

login:
```

些信息可能和的系微有点不同，但不会有很大差。最后行是我感兴趣的，理解一行：

```
FreeBSD/i386 (pc3.example.org) (ttyv0)
```

□一行是□才□的系□信息其中一□，□所看到的是一个"FreeBSD"控制台，□行在一个Intel或兼容的x86体系架□上面。□台□算机的名字(□台 UNIX® □计算机都有自己的名字)叫 `pc3.example.org`，就是□在□个系□控制台-□个 ttyv0 □端的□子。

在最后，最后一行一直保持□□：

```
login:
```

□里，□将可以□入用□名 "username" 并登□到 FreeBSD 系□中。接下来的一□，将介□如何登□系□。

### 4.2.2. □入FreeBSD

FreeBSD是一个多用□多任□的系□，□句□来□就是一个系□中可以容□多不同的用□，而□些用□都可以同□在□台机器中□行大量的程序。

□一个多用□系□都必□在某方面去区分 "user"，在 FreeBSD 里 (以及 □-UNIX® 操作系□)，完成□方面工作是有必要的，因而，□位使用者在□行程序之前都必□首先 "登□"，而□位用□都有与之□□的用□名 ("username") 和密□ ("password")。FreeBSD 会在用□□入之前作出□□□□□信息。

当 FreeBSD 引□并□行完□□脚本之后，□，它会□出一个提示，并要求□入有效的用□名：

```
login:
```

□个例子更容易理解，我□假□□的用□名叫 `john`。在提示符下□入 `john` 并按 `Enter`，此□□□□看到□个提示 "password"：

```
login: john
Password:
```

□在□入 `john`的密□并按下 `Enter`。□入密□□是不回□的！不必□此担心，□□做是出于安全考□。

假如□□入的密□是正□的，□□□□□已□入 FreeBSD，并可以□始□□可用的命令了。

□□□看□ MOTD 或者出□一个命令提示符 (`#`、`$` 或 `%` 字符)。□表明□已成功登□□入FreeBSD。

### 4.2.3. 多个控制台

在一个控制台□行 UNIX® 命令□□很好，但 FreeBSD 具有一次□行 多个程序的能力。□使用一个控制台只会浪□ FreeBSD 同□□行多任□的能力。而 "虚□控制台" 在□方面□□□大的功能。

FreeBSD 能配置出□足□不同需求的虚□控制台，在□□上□用一□□就能从各个虚□控制台之□切□。各个控制台有自己的□□通道，当□在各个控制台切□□ FreeBSD 会切□到合□的□□□□通道和□示器□□通道。

FreeBSD 各个控制台之□可利用特殊□□切□并保留原有控制台，□可□□做：`Alt + F1`，`Alt + F2`，一直到 `Alt`

+ **F8** 在FreeBSD里切换到其中一个虚拟控制台。

同时，正在从其中某个控制台切换到一个控制台的时候，FreeBSD 会保存正在使用和恢复将要使用屏幕通道。结果形成一“”，有“多”虚拟屏幕和可以输入很多的命令。有些程序需要在一个虚拟控制台不能停止运行而又不需要观察它，它运行而可以切换到其他的虚拟控制台。

#### 4.2.4. /etc/ttys文件

FreeBSD 虚拟控制台的默认配置是8个，但并不是硬性配置，可以很容易配置虚拟控制台的个数多或少。虚拟控制台的编号和位置在 /etc/ttys 文件里。

可以使用 /etc/ttys 文件在 FreeBSD 下配置虚拟控制台。文件里一未加注释的行都能配置一个终端或虚拟控制台（当行里含有 # 个字符不能使用）。FreeBSD 默认配置是配置出9个虚拟控制台而只能有8个，以下有些行是 **tttyv** 一起：

| # | name              | getty                          | type   | status | comments |
|---|-------------------|--------------------------------|--------|--------|----------|
| # | tttyv0            | "/usr/libexec/getty Pc"        | cons25 | on     | secure   |
| # | Virtual terminals |                                |        |        |          |
|   | tttyv1            | "/usr/libexec/getty Pc"        | cons25 | on     | secure   |
|   | tttyv2            | "/usr/libexec/getty Pc"        | cons25 | on     | secure   |
|   | tttyv3            | "/usr/libexec/getty Pc"        | cons25 | on     | secure   |
|   | tttyv4            | "/usr/libexec/getty Pc"        | cons25 | on     | secure   |
|   | tttyv5            | "/usr/libexec/getty Pc"        | cons25 | on     | secure   |
|   | tttyv6            | "/usr/libexec/getty Pc"        | cons25 | on     | secure   |
|   | tttyv7            | "/usr/libexec/getty Pc"        | cons25 | on     | secure   |
|   | tttyv8            | "/usr/X11R6/bin/xdm -nodaemon" | xterm  | off    | secure   |

如果要了解个文件中一系列的简介，以及虚拟控制台上所能使用的配置，参考机手册 [ttys\(5\)](#)。

#### 4.2.5. 单用户模式的控制台

关于“单用户模式”简介在 [单用户模式](#) 里可以得到。当运行单用户模式只能使用一个控制台，没有多个虚拟控制台可使用。单用户模式的控制台同也可以在 /etc/ttys 文件配置，可在行得到要的控制台：

| # | 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    | secure   |



这个 **console** 已注释掉，可运行把 **secure** 改为 **insecure**。当，当用单用户模式进入 FreeBSD 时，它仍然要求提供 **root** 用户的密码。

在把这个改为 **insecure** 的时候一定要小心，如果忘记了 **root** 用户的密码，进入单用户模式会有点麻烦。尽管仍然能进入单用户模式，但如果不熟悉它就会非常令人头疼。



4.2.6. 改控制台的模式

FreeBSD 控制台默认的模式可以被调整 1024x768, 1280x1024, 或者任何你的芯片和显示器所支持的其他尺寸。要使用一个不同的模式，你必须首先重新内核并包含以下2个：

```
options VESA
options SC_PIXEL_MODE
```

在内核用2个选项完成后，你就可以使用 [vidcontrol\(1\)](#) 工具来定义硬件支持何模式了。以 root 身在控制台输入以下命令来得到一个所支持的模式列表。

```
# vidcontrol -i mode
```

这个命令的输出是一个硬件所支持的模式列表。你可以在以 root 身在控制台上输入 [vidcontrol\(1\)](#) 命令来改模式：

```
# vidcontrol MODE_279
```

如果对于新的模式满意，那你可以把它加入到 /etc/rc.conf 使机器在每次启动的时候都能生效，我使用了上一个例子中的模式：

```
allscreens_flags="MODE_279"
```

4.3. 权限

FreeBSD, 是 BSD UNIX® 的延伸，并基于几个UNIX® 概念。从一开始就多提到 FreeBSD 是一个多用户的操作系统，它能分配几个同时工作的用户所分配的毫无责任。并管理位用的硬件、外、内存和 CPU 作出合理安排。

因系统有能力支持多用户，在一方面系统都会作出能力、写和行的源力限制。点以三个八位元的方式存着，一个是表示文件所有者，一个是表示文件所属群，一个是表示其他人。些数字以下列方式表示：

| 数字 | 权限          | 目录列表 |
|----|-------------|------|
| 0  | 不能读，不能写，不能行 | ---  |
| 1  | 不能读，不能写，可行  | --X  |
| 2  | 不能读，可写，不能行  | -W-  |
| 3  | 不能读，可写，可行   | -WX  |
| 4  | 可读，不能写，不能行  | r--  |
| 5  | 可读，不能写，可行   | r-X  |
| 6  | 可读，可写，不能行   | rW-  |

| 数 | 限        | 目列表                                  |
|---|----------|--------------------------------------|
| 7 | 可, 可写, 可 | <span style="color: red;">rwx</span> |

使用命令的 -l (ls(1)) 参数可以示出文件的所属者、所属和其他人等属性。看以下的例子：

```
% ls -l
total 530
-rw-r--r--  1 root  wheel    512 Sep  5 12:31 myfile
-rw-r--r--  1 root  wheel    512 Sep  5 12:31 otherfile
-rw-r--r--  1 root  wheel   7680 Sep  5 12:31 email.txt
...
```

使用 ls -l 在行的始出了：

```
-rw-r--r--
```

从左起的第一个字，告我个文件是一的文件：普通文件？目？特殊？socket？或是文件？在个例子，- 表示一个普通文件。接下来三个字是 rw- 是文件有者的限。再接下来的三个字是 r-- 是文件所属群的限。最後三个字是 --- 是其他人的限。以个文件例，他的限定是有者可以写个文件、群可以取、其他使用者也能取个文件。根据上面的表格，用数字表示个文件其三部分的限是 644。

很好，但系行限控制的？事上 FreeBSD 将大部硬件当作一个文件看待，用程序能打、取、写入数据就如其他的文件一。而文件放在 `/dev` 目。

目也是个文件，也有取、写入、行的限。但目的行限意并不与普通文件相同，上行限是入限。当一个目是被示可以行的，表示可以入它，或者言之，利用 `"cd"` (改当前目) 入它。此外，也表示有入目的用，可以其下的已知名字的文件 (当然目下的文件也受到限制)。

方面，想取一个目的列表就必可限，同想除一个已知的文件，就必把目下个文件可写 和行限。

有更多限定，但是他大多用在特殊状况下如一个setuid的行文件和粘性目，如果想要得知有文件限和如何定的更多，看手册chmod(1)。

### 4.3.1. 限的符号化表示

限符号，某些候就是指符号表式，使用八制的字符目或文件分配限。限符号的使用法是 ( ) (作用) (限)。看看下列数的在那些地方所起什的作用：

|      | 字母 | 介          |
|------|----|------------|
| ( )  | u  | 用          |
| ( )  | g  | 所属群体       |
| ( )  | o  | 其他人        |
| ( )  | a  | 所有人 ("全部") |
| (作用) | +  | 加限         |

| 符号   | 字母 | 介绍           |
|------|----|--------------|
| (作用) | -  | 最少权限         |
| (作用) | =  | 指定权限         |
| (权限) | r  | 可读           |
| (权限) | w  | 可写           |
| (权限) | x  | 可执行          |
| (权限) | t  | 粘滞位          |
| (权限) | s  | 设置 UID 或 GID |

有些数字 `chmod(1)` 以符号指定的。举个例子，用以下命令阻止其他人对 `FILE` 文件：

```
% chmod go= FILE
```

如果需要文件一次可执行多操作，可用逗号分隔，在下面的例子中，将去掉 `FILE` 文件的群体和 " 全体其他用户" 可写权限，并对所有人增加可执行权限：

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

### 4.3.2. FreeBSD 文件标志

在前面所介绍的文件权限的基础上，FreeBSD 还支持使用 "文件标志"。有些标志文件提供了另一套的安全控制机制，但有些控制并不用于目的。

有些文件标志提供了对文件的另一套控制，帮助确保即使是 `root` 用户也无法删除或修改文件。

文件标志可以通过使用 `chflags(1)` 工具来修改，其用户界面很简单。例如，要在文件 `file1` 上设置系禁标志，使用下述命令：

```
# chflags sunlink file1
```

要禁用系禁标志，只需在前述命令中的 `sunlink` 标志前加 "no"。例如：

```
# chflags nosunlink file1
```

要显示文件上的标志，使用命令 `ls(1)` 的 `-lo` 参数：

```
# ls -lo file1
```

输出结果类似于：

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

多标志只可以由 **root** 用 `chmod` 来添加，而一些，可以由文件的所有者来添加。建议管理仔细看看 [chflags\(1\)](#) 和 [chflags\(2\)](#) 的手册，以进一步加深理解。

### 4.3.3. setuid、setgid 和 sticky 权限

除了前面已看到的那些权限之外，还有三个管理知道的权限配置。它们是 **setuid**、**setgid** 和 **sticky**。

这些配置对于某些 UNIX® 操作而言很重要，因为它们能提供一些一般情况下不会授予普通用户的功能。为了便于理解，我们首先介绍真实用户 ID (real user ID) 和生效用户 ID (effective user ID)。

真实用户 ID 是用户或进程的 UID。生效 UID 是进程以其自身运行的用户 ID。例如，[passwd\(1\)](#) 工具通常是以超修改密码的用户身份运行，也就是其进程的真实用户 ID 是那个用户的 ID；但是，由于需要修改密码数据，它会以 **root** 用户作为生效用户 ID 的身份运行。因此，普通的非特权用户就可以修改口令，而不是看到 **Permission Denied** 了。



[mount\(8\)](#) 的 **nosuid** 选项可以令系统在不出任何提示的情况下不运行某些程序。一方面，这个选项并不是万无一失的，正如 [mount\(8\)](#) 的手册所提到的那样，如果系统中安装了 **nosuid** 的封装程序，那它就可以被绕过了。

**setuid** 权限可以通过在普通权限前面加上一个数字四 (4) 来设置，如下面的例子所示：

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

这样一来，`suidexample.sh` 的权限如下面：

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

我们会注意到，在原先的属主权限的位置成了 **s**。因此，需要提升特权的可执行文件，例如 **passwd** 就可以正常运行了。

可以打开两个终端来观察这一情形。在其中一个终端里面，以普通用户身份运行 **passwd** 进程。在它等待输入新口令时，在另一个终端中查看进程表中关于 **passwd** 命令的信息。

在终端 A 中：

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

在终端 B 中：

```
# 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
```

正如前面所看到的那，`passwd` 是以普通用户的身体的，但其生效 UID 是 `root`。

与此，`setgid` 权限的作用，与 `setuid` 权限似，只是当应用程序配合一确定行，它会被授予有文件的那个的权限。

如果需要在文件上配置 `setgid` 权限，可以在权限数前面加数字二 (2) 来行 `chmod` 命令，如下面的例子所示：

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

可以用与前面似的方法来新确定的生效情况，在权限的地方的 `s` 表示一配置已生效：

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



在些例子中，尽管 shell 脚本也属于可执行文件的一，但它不会以配置的 EUID 或生效用 ID 的身体行。是因 shell 脚本可能无法直接呼叫 `setuid(2)` 用。

我已有了个特殊权限位 (`setuid` 和 `setgid` 权限位)，它用在使用程序能用到更高的权限，有会削弱系的安全性。除了个之外，有第三个特殊权限位：`sticky bit`，它能安全性。

当在目录上置了 `sticky bit` 之后，其下的文件就只能由文件的所有者除了。个权限置能防止用除似 `/tmp` 的公共目录中不属于他的文件。要用权限，可以在权限置前面加上数字一 (1)。例如：

```
# chmod 1777 /tmp
```

在，可以用 `ls` 命令来看效果：

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

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

里的尾的 `t` 表示了 `sticky bit` 权限。

## 4.4. 目录架

理解 FreeBSD 的目录次于建立系整体的理解十分重要的基。其中，最重要的概念是根目录，`/`。个目录是引挂接的第一个目录，它包含了用以准多用操作所需的操作系统基件。根目录中也包含了用于在到多用模式之前挂接其他文件系统所需的挂接点。

挂接点 (mount point) 是新文件系统接入有系的起点位置 (通常是根目录)。在磁此行了的

述。 准的挂载点包括 /usr、 /var、 /tmp、 /mnt， 以及 /cdrom。 些目录通常会在 /etc/fstab 文件中提及。 /etc/fstab 是一包含系中各个文件系统及挂载点的表。 在 /etc/fstab 中的大多数文件系统都会在由 [rc\(8\)](#) 脚本自动挂载， 除非特指定了 **noauto** 。 更多参考 [fstab 文件](#)。

可以通过 [hier\(7\)](#) 来了解完整的文件系统层次。 在， 我先来查看一下大多数的常的目录以供参考。

| 目录              | 介绍  |
|-----------------|---|
| /               | 文件系统的根目录。   |
| /bin/           | 在单个用户和多用户环境下的基本工具目录。  |
| /boot/          | 在操作系统在添加期间所用的程序和配置。   |
| /boot/defaults/ | 默认引导的配置内容， 参见 <a href="#">loader.conf(5)</a> 。  |
| /dev/           | 设备点， 参见 <a href="#">intro(4)</a> 。  |
| /etc/           | 系统配置和脚本。  |
| /etc/defaults/  | 系统默认的配置和脚本， 参见 <a href="#">rc(8)</a> 。  |
| /etc/mail/      | 邮件系统操作的配置， 参见 <a href="#">sendmail(8)</a> 。   |
| /etc/namedb/    | <b>named</b> 配置文件， 参见 <a href="#">named(8)</a> 。  |
| /etc/periodic/  | 每天、每周和每月周期性地运行的脚本， 通过 <a href="#">cron(8)</a> 参见 <a href="#">periodic(8)</a> 。  |
| /etc/ppp/       | <b>ppp</b> 配置文件， 参见 <a href="#">ppp(8)</a> 。  |
| /mnt/           | 由管理使用挂载点的空目录。   |
| /proc/          | 运行中的文件系统， 参见 <a href="#">procfs(5)</a> 和 <a href="#">mount_procfs(8)</a> 。  |
| /rescue/        | 用于急救的一静默的程序； 参见 <a href="#">rescue(8)</a> 。   |
| /root/          | <b>root</b> 用户的Home(主)目录。   |
| /sbin/          | 在单个用户和多用户环境下的存放系统程序和管理所需的基本用目录。   |
| /tmp/           | 临时文件。 /tmp 目录中的内容， 一般不在系统重新之后保留。 通常会将基于内存的文件系统挂在 /tmp 上。 一工作可以用一系列 tmpmfs 相关的 <a href="#">rc.conf(5)</a> 变量来自完成。(或者， 也可以在 /etc/fstab 添加； 参见 <a href="#">mdmfs(8)</a> )。 |
| /usr/           | 存放大多数用的用文件。   |
| /usr/bin/       | 存放用命令， 程序工具， 和用文件。  |
| /usr/include/   | 存放标准 C include 文件。  |
| /usr/lib/       | 存放文件。   |
| /usr/libdata/   | 存放各用工具的数据文件。  |
| /usr/libexec/   | 存放系统用或后台程序 (从外的程序运行)。   |

| 目            | 介   |
|--------------|---|
| /usr/local/  | 存放本地可执行文件、配置文件等等，同也是 FreeBSD ports 安装的默认安装目录。/usr/local 在 /usr 中的目录布局大体相同，参见 <a href="#">hier(7)</a> 。但 man 目录例外，它是直接放在 /usr/local 而不是 /usr/local/share 下的，而 ports 的明文在 share/doc/port。 |
| /usr/obj/    | 通过 /usr/src 得到的目录文件。  |
| /usr/ports/  | 存放 FreeBSD 的 Ports Collection (可)。  |
| /usr/sbin/   | 存放系统后台程序和系统工具 (由用户运行)。  |
| /usr/shared/ | 存放架构独立的文件。  |
| /usr/src/    | 存放 BSD 或者本地源代码文件。   |
| /usr/X11R6/  | 存放 X11R6 可执行文件、配置文件、配置文件等的目录(可)。  |
| /var/        | 多用途日志、或短期存放的，以及打印假脱机系统文件。有会将基于内存的文件系统挂在 /var 上。一工作可以通过在 <a href="#">rc.conf(5)</a> 中设置一系列 varmfs 量 (或在 /etc/fstab 中加入一行配置；参见 <a href="#">mdmfs(8)</a> ) 来完成。                            |
| /var/log/    | 存放各的系文件。  |
| /var/mail/   | 存放用 mailbox(一邮件存放格式)文件。   |
| /var/spool/  | 各打印机和邮件系统 spooling(回)的目录。   |
| /var/tmp/    | 文件。些文件在系统重新通常会保留，除非 /var 是一个内存中的文件系统。   |
| /var/yp/     | NIS 映射。   |

## 4.5. 磁

FreeBSD 文件的最小单位是文件名。而文件名区分大小写，就意味着 readme.txt 和 README.TXT 是两个不相同的文件。FreeBSD 不凭文件扩展名 (.txt) 去判断文件是程序、文，或是其他格式的数据。

各文件存放在目录里。一个目录可以空，也可以含有多个的文件。一个目录同可以包含其他的目录，允许在一个目录里建立多个不同次的目录。将帮助轻松地的数据。

文件或目录是由文件名或目录名，加上斜符号 /，再根据需要在目录名后面加上其他目录的名称。如果有一个名 foo 的目录，它包含一个目录 bar，后者包括一个叫 readme.txt 的文件，全名，或者到文件的路径就是 foo/bar/readme.txt。

在文件系统里目录和文件的作用是存数据。一个文件系统都有且只有一个目录根目录，一个根目录可以容纳其他目录。

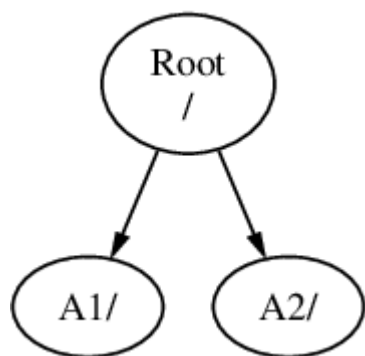
也在其他的一些操作系统到类似里的情况，当然也有不同的情况。些例子，MS-DOS® 是用 \ 分隔文件名或目录名，而 Mac OS® 使用 :。

FreeBSD在路径方面不使用驱动器名符号或驱动器名称，在FreeBSD里不能使用：c:/foo/bar/readme.txt。

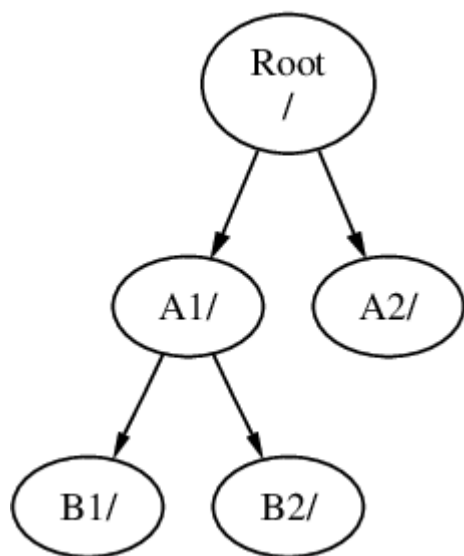
了代替(驱动器名符号)，一个文件系统会指定根文件系统，根文件系统的根目录是 /。其他一个文件系统挂载在根文件系统下。无论有多少磁盘在FreeBSD系统里，每个磁盘都会以目录的方式加上。

假设有三个文件系统，名A、B和C。每个文件系统有一个根目录，而各自含有两个其他的目录，名A1, A2 (B1, B2和C1, C2)。

看看A的根文件系统。假如用ls命令来看这个目录会看到两个子目录：A1和A2。这个目录是个子目录：



一个文件系统必须挂到一个文件系统的某一目，所以在假把B文件系统挂到A1目录，那B根目录因此代替了A1，而显示出B目录的内容：

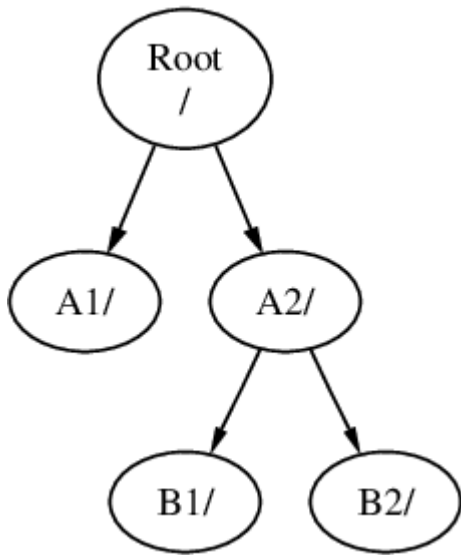


无论B1或B2目录在那里而延伸出来的路径必须是/A1/B1或/A1/B2。而在/A1里原有的文件会被隐藏。想这些文件再出把B从A挂载放。

所有在B1或B2目录里的文件都可以通/A1/B1或/A1/B2访问。而在/A1中原有的文件会被隐藏，直到B从A上被卸载(unmount)止。

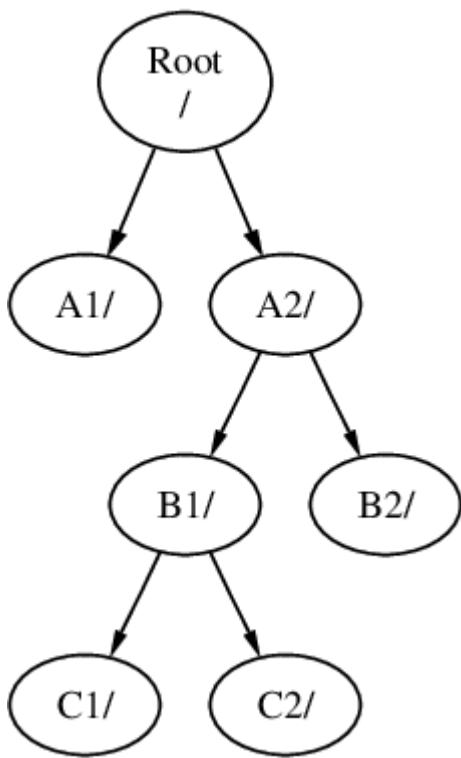
把B挂载在A2那表现的子就是子：



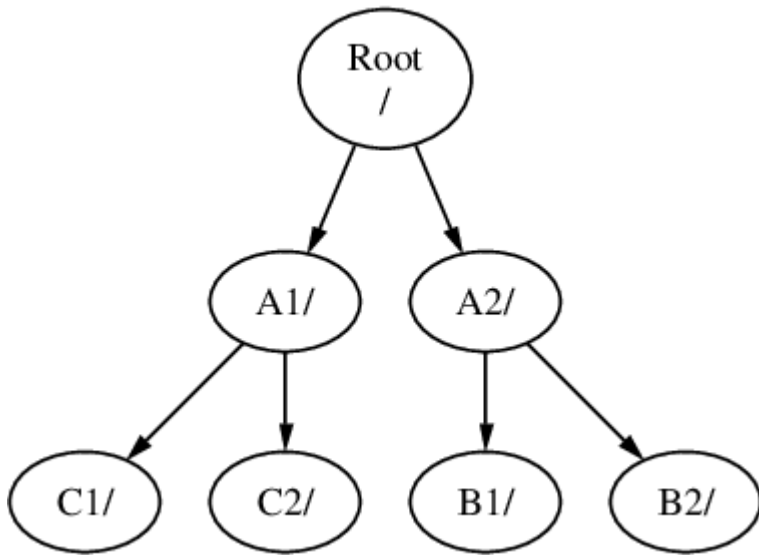


两个路径分别是 /A2/B1 和 /A2/B2 。

文件系统能把一部挂载在一个文件系统上。 举个例子， 把 C 文件系统挂载在 B 文件系统里的 B1 目录，排列如下：



或者把 C 文件系统挂载在 A 文件系统里的A1目录：



假如熟悉 MS-DOS® 并知道 `join` 命令，尽管不相同，其功能是相似的。

这方面不是普通知识而且涉及到自己所关心的，当安装 FreeBSD 并在以后添加新磁盘，必须知道如何新建文件系统并挂接上。

(FreeBSD 系统) 它有一个主要的根文件系统，不需要另外新建，但当需要手工管理，是一个有用的知识。

多个文件系统的益处

- 不同的文件系统可用不同的挂接参数。 有些例子， 仔细想一下， 根文件系统能用只读的方式挂接上，防止不意删除或写到一个危险的文件。 把各用读能写入的文件系统分开， 像 `/home`， 由读外的文件系统分用 `nosuid` 参数挂接， 这个参数防止 `suid/guid` 在运行这个文件系统文件生效， 从而了解了一些安全问题。
- FreeBSD 能根据一个文件系统使用的情况自动化 这个文件系统上的文件布局。 所以一个存了大量小文件并会被频繁写入文件系统的自动化与一个存了少量大文件的自动化是不同的。 而在一个大的单一文件系统上无法体现的自动化。
- FreeBSD 的文件系统能在断时尽可能避免丢失。 然而， 在时点的源失效仍然可能会破坏文件系统的。 将这个文件系统分成多个有助于分散， 并方便备份和恢复。

单一文件系统的益处

- 文件系统是固定大小的。 当安装 FreeBSD 新建一个文件系统并定一个大小， 会在以后时到必须去建一个大的分区。 如果配置不当， 需要时、 重新建文件系统， 然后再恢复数据。



FreeBSD 提供了 `growfs(8)` 命令。 这使得能时地调整文件系统的大小， 因而不受其限制。

文件系统是和分区一一的。 这里的分区和常用的分区 (例如， MS-DOS® 分区) 的意思并不一样， 是由于 FreeBSD 的 UNIX® 造成的。 一个分区使用一个从 `a` 到 `h` 的字母来表示。 一个分区只能包含一个文件系统， 意味着文件系统通常可以由它在文件系统目录中的挂接点， 或时的分区字母来表示。

FreeBSD 的交换分区也需要使用磁盘空间。 交换分区是 FreeBSD 作虚拟内存使用的， 能令时的计算机有更多的内存可使用， 当 FreeBSD 在运行而内存不时的候， 它会把其他一些可移的数据移到交换分区， 空出内存的位置以供使用。

某些 partitions 的用途是固定的。

| 分区 | 定义  |
|----|---|
| a  | 通常指定根文件系统   |
| b  | 通常指定交换分区  |
| c  | 通常它和所在的 slice 大小相同。c 分区上工作必定会影响到整个 slice (例如，坏道扫描器)。通常不意在单个 partition 建立文件系统。 |
| d  | 分区 d 曾有特殊的含义，不意味着在它的系统上已不再使用，因此 d 可以和任何其它普通的分区一起使用了。                          |

一个包含了文件系统的分区被保存在 FreeBSD 称 slice 的部分上。Slice 是一个 FreeBSD 磁盘，通常被叫做分区，再次说，这是由于 FreeBSD 的 UNIX® 背景。Slices 有其编号，从1到4。

Slice 编号在磁盘名后面，并有一个 s 前缀，从 1 开始。因此 "da0s1" 是第一个 SCSI 磁盘的第一个 slice。一个磁盘上只能有四个物理的 slices，但可以在物理 slice 中使用虚拟的类型来创建 slice。有些扩展 slice 编号从 5 开始，因此 "ad0s5" 是第一个 IDE 磁盘中的第一个扩展 slice。文件系统所使用的磁盘占 slice。

Slices, "用指定" 物理磁盘，和其他磁盘都包含 partitions，那几个的 partitions 都是用字母从 a 到 h 来指定的，而有些字母都在磁盘名字之后，所以 "da0a" 是指首个 da 磁盘的 a partition，而那个就是 "指定"。"ad1s3e" 是指 IDE 磁盘上第三个 slice 的第五个 partition。

最后，每个磁盘都被系统识别。一个磁盘名字是用磁盘类型代码和编号来识别的，它不像 slices，磁盘的编号是由 0 开始的。代码可以看这里所列出的磁盘代码。

当在 FreeBSD 中指定 partition 名字时，必须同时包含该分区的 slice 和磁盘的名字；类似地，在指定 slice 时，也需指出包含该 slice 的磁盘名字。可列出：磁盘名称，s，slice 编号，和 partition 指定字母。例子看列磁盘, Slice, 和 Partition 它的命名。

一个磁盘的布局 图里显示了一个磁盘的布局，有更清楚的帮助。

在安装 FreeBSD 时，首先要配置好磁盘 slices，然后在 FreeBSD 使用的 slice 上建立 partitions。并在一个 partition 上建立一个文件系统(或交换分区)，和指定文件系统的挂载位置。

表 6. 磁盘代码

| 代码  | 说明             |
|-----|----------------|
| ad  | ATAPI (IDE) 磁盘 |
| da  | SCSI 直接存取磁盘    |
| acd | ATAPI (IDE) 光盘 |
| cd  | SCSI 光盘        |
| fd  | 软盘             |

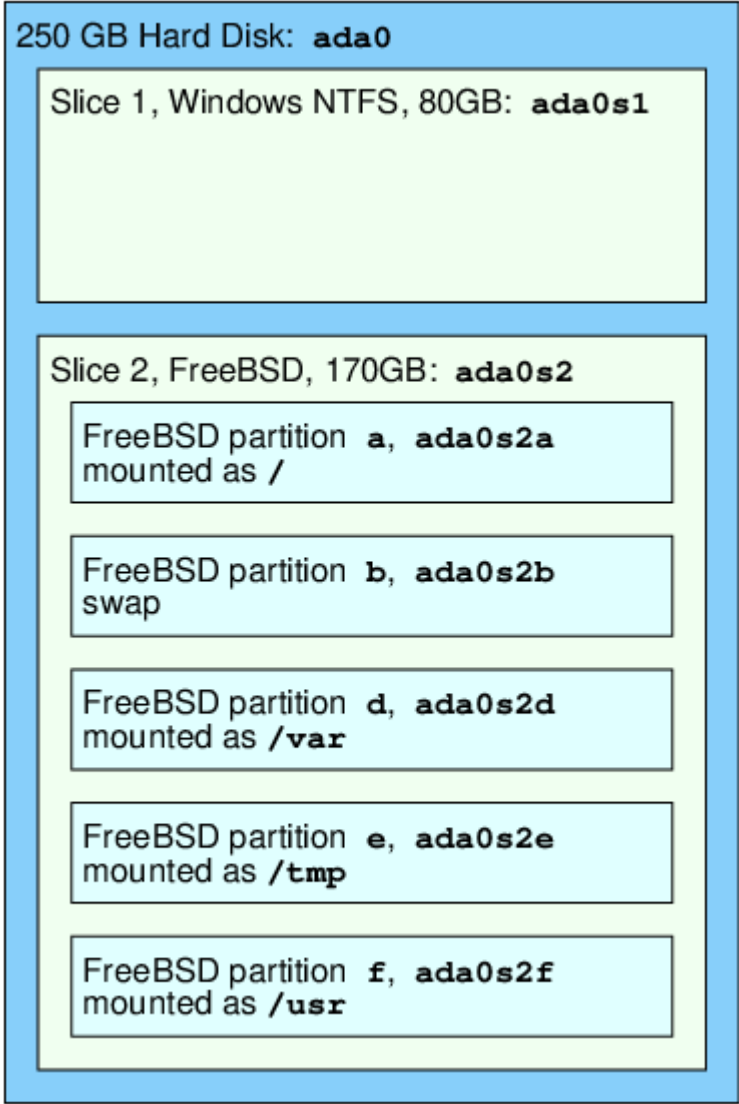
例 6. 磁盘, Slice, 和 Partition 它们的命名

| 命名     | 说明  |
|--------|---|
| ad0s1a | 在首个IDE磁盘(ad0)上的 第一个slice(s1)里的 第一个partition(a)。   |
| da1s2e | 在第二个SCSI磁盘(da1)上的 第二个slice(s2)里的 第五个partition(e)。 |

例 7. 一个磁盘的布局

从在系统里的首个IDE磁盘表可以显示出FreeBSD的分解。 假设磁盘大小4 GB, 它里面包含了2 GB 大小的slices (但在MS-DOS®叫partitions)。 首个slice是一个MS-DOS®磁盘叫C:, 而第二个slice是FreeBSD配置好的slice。 FreeBSD配置好的slice有三个partitions和一个交换分区。

三个partitions各自控制一个文件系统。 partitiona 用于根文件系统, partitione 用于 /var 目录, partitionf 用于 /usr 目录。



## 4.6. 文件系统的挂载和卸下

文件系统就像一棵那用/立根部， 是比理想的文件系统。 而/dev、 /usr 和其他目就是根目的分枝， 外些目可以再分枝， 例如/usr/local。

考虑某些目一些空从而分散文件系统。 /var 之下包含目 log/, 目spool/, 和不同型的文件， 很可能把它塞满。 把什么都塞根文件系统不是一个好主意， 好的做法是把 /var 从/分出去。

一个要考的是， 物理或虚磁些自空的文件系统定目。 例如 [网文件系统](#) 或光的挂载。

### 4.6.1. fstab 文件

在 [引进程](#) 期， 自挂上/etc/fstab所列出的文件系统。(除非他注明noauto )。

/etc/fstab 文件包含的各行的列表格式如下：

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

#### device

名称(必存在)， 明在 [命名](#)。

#### mount-point

目 (目必存在)， 用在那个挂载上的文件系统上。

#### fstype

文件系统型， 通[mount\(8\)](#)。 默的FreeBSD文件系统型是ufs。

#### options

可写文件系统的rw， 或只读文件系统的ro， 或其他一些， 可随意一个。 一个常用的 noauto 用在不需在引进程期挂载的文件系统。 其他的在 [mount\(8\)](#) 手册里列出。

#### dumpfreq

[dump\(8\)](#) 使用去决定那个文件系统必移。 假如缺少， 默的数0。

#### passno

一决定文件系统的序， 文件系统想跳将passno0。 根文件系统(那个是在方面始之前必的) 将它的 passno 1， 其他文件系统的 passno 必把数到大于1。 假如多个文件系统的passno的相同， 那 [fsck\(8\)](#) 在允的情况下将并行地去文件系统。

参 [fstab\(5\)](#) 机手册， 以得于 /etc/fstab 文件格式， 以及其中所包含的的信息。

### 4.6.2. mount 命令

个 [mount\(8\)](#) 命令是挂载文件系统的基本用。

使用最多的基本格式：

```
# mount device mountpoint
```

它的选项非常多，而`mount(8)`手册同它提及，但常用的都在它里：

挂载的各选项

**-a**

挂载/etc/fstab里所有列出的文件系统。除非选项 "noauto" 或作了排除在外的 **-t** 选项型，或者在选项之前已挂上。

**-d**

除了选项上系统调用以外，可以完成任何事情，选项是和 **-v** 参数一起在选项使用，可以决定 `mount(8)` 所做的事情。

**-f**

强制去挂载一个未知的文件系统(会有危险)，或当把一个文件系统挂载状态由可写降为只读，强制撤消可写通道。

**-r**

以只读方式挂载文件系统。和在指定了 **-o** 选项配合 **ro** 参数的效果是一样的。

**-t fstype**

根据选项出的文件系统类型挂载文件系统，假如选项于 **-a**，则挂载选项型的文件系统。

"ufs" 是默认的文件系统类型。

**-u**

在文件系统上修改挂载选项。

**-v**

版本模式。

**-w**

以可写方式挂载文件系统。

The **-o** 选项采用一个逗号分隔以下多个选项：

**noexec**

不允许文件系统上的二进制程序运行。也是一个有用的安全选项。

**nosuid**

不允许文件系统上的 setuid 或 setgid 选项生效。也是一个有用的安全选项。

### 4.6.3. `umount` 命令

`umount(8)` 命令同它采用一个参数、一个挂载点、一个选项名。或采用 **-a**，又或采用 **-A**。

所有格式都可采用 `-f` 去行卸下， 或采用 `-v` 用那当的版本。 但警告，采用 `-f`并不是一个好主意，行卸下文件系可能坏计算机或破坏文件系上的数据。

`-a` 和 `-A` 会卸下所有已挂接的文件系， 可能通`-t`后面列出的文件系行修改， 但无论如何，`-A`都不会去卸下根文件系。

## 4.7. 程

FreeBSD 是一个多任务操作系统。 就意味着好像一次可以行一个以上的程序。 一个占用一定行的程序就叫 程 (process)。 行的一个命令会至少一个新程， 有很多一直行着的系程，用以持系的正常作。

一个程用来的一个号就叫 程 ID， 或叫 PID。 而且，就像文件那，一个程也有所属用和所属群体。所属用和所属群体使用在方面：定一个程可以打那些文件和那些， 从而在初期使用文件的限。多数的程都有一个父程， 而程是依父程来的。 例如，假如把命令入到shell里那shell是一个程，而行的各个命令同是程， 那，shell就是各个行程的父程。 而方面有一个例外的程就叫init(8)。init始是首个程，所以他的PID始是1，而init在FreeBSD起由内核自。

在系上，有个命令程察非常有用：`ps(1)` 和 `top(1)`。 个`ps`命令作用是察当前行程的状， 示他的PID，使用了多少内存，它命的命令行。 而`top`命令是示所有行程，并在以秒的短内更新数据。 能交互式的察计算机的工作。

默情况下，`ps`示出自己所行的命令。 例如：

```
% ps
  PID TT  STAT      TIME COMMAND
   298 p0  Ss      0:01.10 tcsh
  7078 p0  S        2:40.88 xemacs mdoc.xsl (xemacs-21.1.14)
37393 p0  I        0:03.11 xemacs freebsd.dsl (xemacs-21.1.14)
48630 p0  S        2:50.89 /usr/local/lib/netcape-linux/navigator-linux-4.77.bi
48730 p0  IW       0:00.00 (dns helper) (navigator-linux-)
72210 p0  R+       0:00.00 ps
   390 p1  Is       0:01.14 tcsh
  7059 p2  Is+      1:36.18 /usr/local/bin/mutt -y
  6688 p3  IWs      0:00.00 tcsh
10735 p4  IWs      0:00.00 tcsh
20256 p5  IWs      0:00.00 tcsh
   262 v0  IWs      0:00.00 -tcsh (tcsh)
   270 v0  IW+      0:00.00 /bin/sh /usr/X11R6/bin/startx -- -bpp 16
   280 v0  IW+      0:00.00 xinit /home/nik/.xinitrc -- -bpp 16
   284 v0  IW       0:00.00 /bin/sh /home/nik/.xinitrc
   285 v0  S        0:38.45 /usr/X11R6/bin/sawfish
```

在个例子里可看到，从 `ps(1)` 出的列是有律的。PID 就是程ID， 个早前已了。PID号的分配由1一直上升直到99999， 当行到超限制， 些号会回分配 (仍在使用的 PID 不会分配其他程)。TT 列示了程序行所在的端， 目前可以安全地忽略。STAT 示程序的状，也可以安全地被忽略。TIME是程序在CPU理行的量， 并不是指程序到在的所用的。 多程序巧遇到某方面在他之前要花大量CPU理，他就必等候。最后，COMMAND 是行程序使所用的命令行。

`ps(1)`支持使用各选项去改显示出来的内容，最有用的一个就是`auxww`。`a`显示所有进程的内容，而不是`ps`的进程。`u`显示进程所属的用户名字以及内存使用，`x`显示后台进程。而`ww`表示`ps(1)`把整个进程的整个命令行全部显示完，而不是由于命令行长就把它从屏幕上截去。

下面和从`top(1)`出是类似的，一个示例式子就象这样子：

```
% top
last pid: 72257; load averages: 0.13, 0.09, 0.03 up 0+13:38:33 22:39:10
47 processes: 1 running, 46 sleeping
CPU states: 12.6% user, 0.0% nice, 7.8% system, 0.0% interrupt, 79.7% idle
Mem: 36M Active, 5256K Inact, 13M Wired, 6312K Cache, 15M Buf, 408K Free
Swap: 256M Total, 38M Used, 217M Free, 15% Inuse

  PID USERNAME PRI NICE  SIZE  RES STATE   TIME  WCPU   CPU COMMAND
 72257 nik      28  0 1960K 1044K RUN      0:00 14.86% 1.42% top
  7078 nik       2  0 15280K 10960K select   2:54  0.88%  0.88% xemacs-21.1.14
   281 nik       2  0 18636K  7112K select   5:36  0.73%  0.73% XF86_SVGA
   296 nik       2  0  3240K  1644K select   0:12  0.05%  0.05% xterm
48630 nik       2  0 29816K  9148K select   3:18  0.00%  0.00% navigator-linu
   175 root       2  0   924K   252K select   1:41  0.00%  0.00% syslogd
  7059 nik       2  0  7260K  4644K poll    1:38  0.00%  0.00% mutt
...
```

这个输出分成四部分。前面部分(起始前五行)显示了：运行于最后进程的PID、系统均衡（那个是指系统繁忙的方式）、系统正常运行（指从开始算起所用的时间）和当前时间。前面部分外的表格涉及：多少进程在运行（这个情况是47），多少内存和多少交换分区在使用，和在不同CPU状态里系统消耗多少时间。

在那下面一串的列表和从`ps(1)`出的内存是相似的。如以前`ps(1)`一样，能看到：PID、用户名、CPU利用率、运行的命令。`top(1)`默认是显示进程的内存空间的合计。内存空间里分成四列，一列是程序体大小，一列是必须保留大小是多少内存-程序体大小。而保留大小上是瞬间使用的多少。在以上那个例子，你会看到那`getenv(3)`需要30 MB内存，但只用了9 MB。

`top(1)`每秒自动刷新一次，可以用`s`改刷新的秒数。

## 4.8. 守护进程，信号和僵尸进程

当运行一个进程它是很容易控制的，告诉它去加文件它就加。之所以能这样做，是因为系统提供便利去做，和因有系统去附上的端口。一些程序在运行中不需要用户的输入，一有机会就从端口里分到后台去。例如，一个web系统整天都在作web请求的，他不需要输入任何东西就能完成，一个例子就是把email的发送。

我把那些程序叫 守护进程。守护神是希腊神中的一些人物，非正非邪，他是一些守护小精灵，大体上是人作出贡献。多类似web服务或mail服务的系统于今天仍有用途，就是什么在那里的，BSD的吉祥物保持一双鞋加一把叉的守护神模式。

守护进程的程序命名通常在最后加一个“d”。BIND 是伯克利互联网域名服务（而运行的程序名称是 `named`），Apache web系统的程序就叫 `httpd`，在行式打印机上的打印守护进程就是 `lpd`。只是一例，不是标准或硬性规定。例如，Sendmail而用的主要mail守护进程就叫`sendmail`，却不叫`maild`，和它推的一。



有可能会需要与守护进程通信。而信号是其中的一种通信机制。可以发送信号给守护进程（或相关的某些进程）来与它通信，不同的信号都有自己的数字编号-其中一些有特殊的含义，其它的可以被应用程序自行解释，而一般来，应用程序的文档会告诉某些信号会被如何处理。只能向所属于的进程发送信号，假如其他人的进程信号，进程就会用`kill(1)`或`kill(2)`限制行拒。当然，`root`用户会例外，它能把各信号送给个进程。

在某些情况下，FreeBSD也会向用户进程发送信号。假如一个用户进程含有恶意写入并去访问内存，那是不可想象的，FreeBSD会向那个进程发送段式错误信号（`SIGSEGV`）。假如一个用户进程使用`alarm(3)`系去进行周期性用户功能，当到时间，FreeBSD会向用户进程发送信号（`SIGALRM`）。

有个信号可以停止进程：`SIGTERM`和`SIGKILL`。`SIGTERM`比友好，进程能捕捉个信号，根据的需要来结束程序。在结束程序之前，可以结束打开的文件和完成正在做的任务。在某些情况下，假如进程正在运行而且不能中断，那进程可以忽略个`SIGTERM`信号。

至于`SIGKILL`信号，进程是不能忽略的。它是一个“我不管我在做什么，立刻停止”的信号。假如发送`SIGKILL`信号给进程，FreeBSD就将进程停止在那里。

可能会去使用`SIGHUP`、`SIGUSR1`和`SIGUSR2`信号。都是些通用的信号，各应用程序都可以用在各方面的信号发送。

假如修改了web系统的配置文件-并想web系统去重新它的配置，可以停止然后再启动`httpd`。但这样做web系统会导致一个短暂的中断周期，那是不受欢迎的。几乎所有的守护进程在编写时，都会指定`SIGHUP`信号进行从而重新配置。所以，最好的方法，就不是死并重启`httpd`，而是发送一个`SIGHUP`信号给它。因为在这方面没有一个标准，不同的守护进程有不同的用法，所以不了解一下守护进程的文档。

发送信号可用`kill(1)`命令，参考`kill(1)`所列出的例子。

## Procedure: 发送一个信号给进程

这个例子显示了如何去发送一个信号给inetd(8)。inetd配置文件是/etc/inetd.conf， 如果想inetd去重文件系统的，可以给它发送一个SIGHUP信号。

1. 需要发送信号的进程ID，可以用ps(1) 加 grep(1)来完成。grep(1)命令被用在搜索输出方面，搜索指定的字符串。命令是由普通用户来执行的，而inetd(8)是root用户执行的，所以必须用ps(1)上ax。

```
% ps -ax | grep inetd
198 ?? IwS    0:00.00 inetd -wW
```

得出inetd(8) PID号是198。有grep inetd 命令也输出在输出中，是因为在输出方面ps(1)也是列表中执行的。

2. 使用kill(1)去发送信号。因inetd(8)是由root用户的，必须使用su(1)去root用户。

```
% su
Password:
# /bin/kill -s HUP 198
```

和大多数UNIX®命令一样，kill(1)如果完成了任务，就不会输出任何消息。假如发送信号给一个不属于用户的进程，会看到kill: PID: Operation not permitted.假如给了PID号，把信号送到其他进程，那是坏事。或者不幸，把信号送到不存在的进程，会看到kill: PID: No such process.



为什么使用/bin/kill?

许多shell提供了内建kill命令，用户，shell就能直接发送信号，而不是执行/bin/kill。这点非常有用，但不同shell有不同的方法来指定发送信号的名字，与其把它学完倒不如直接地使用/bin/kill …。

发送其他的信号也很相似，只要在命令行替换TERM或KILL就行了。



在系统上随意杀死进程是个坏主意，特别是init(8)，它的进程ID是1，它非常特殊。可以执行/bin/kill -s KILL 1 命令来系迅速关机。当按下Return（回车）之前，一定要记得指定kill(1)所指定的参数。

## 4.9. Shells

在FreeBSD里，日常有一大堆工作是在命令行的界面完成的，那就叫做shell。一个shell的主要功能就是从输入取得命令然后去执行他。多的shell同时能完成内建的功能，例如：文件管理、文件、命令行、宏指令和环境量。FreeBSD内含了一些shell，例如：sh、Bourne Shell、tcsh和改良的C-shell。外也有些shell也可在FreeBSD的Ports得到，例如：zsh和bash。

你想使用哪一个shell取决于你的喜好，假如你是C程序员，你可能需要一个C-like shell例如tcsh。假如你是从Linux来的或是一个命令行的新手，你可能会喜欢bash。有一点告诉我一个shell都有各自的特性，可能用于你的工作环境，也可能不用于你的工作环境。

每个shell都有一个共通点就是文件名补全。输入命令或文件名的前几个字，然后按Tab，就能让shell的自动补全功能得出命令或文件名。这里有一个例子，假如有两个文件叫foobar和foo.bar，而我想删除foo.bar，可以在提示符后输入rm fo[Tab].[Tab]。

那个shell就会输出rm foo[BEEP].bar。

那个[BEEP]是控制台响声，那个是告诉我它不能完成文件名补全，因为有多文件名符合。foobar和foo.bar都是以fo开头，它只可以补全到foo。输入.并再按一次Tab，shell才把其余的文件名全部显示出来。

一个特点就是shell利用环境变量运行。环境变量是存在shell环境空间上相等的键和值，每个空值都能让程序从shell里取出，而且包含了多程序的配置。下一个常用环境变量列和其值的列表：

| 变量       | 说明                      |
|----------|-------------------------|
| USER     | 当前登录用户的用户名。             |
| PATH     | 搜索程序路径，以点的冒号分隔。         |
| DISPLAY  | 假如有个变量的值，就是X11显示器的网络名称。 |
| SHELL    | 当前所用的shell。             |
| TERM     | 用户终端的名字，通常用在设定终端的能力。    |
| TERMCAP  | 各终端功能所用终端分行的基本数据项目。     |
| OSTYPE   | 操作系统型，默认是FreeBSD。       |
| MACHTYPE | 是指系上运行的CPU体系。           |
| EDITOR   | 用户首选的文本编辑器。             |
| PAGER    | 用户首选的文本页面度程序。           |
| MANPATH  | 搜索手册路径，以点的冒号分隔。         |

不同的shell设置环境变量也不相同。个例子，在如tcsh和csh的C-Style shell，必须使用setenv去设置环境变量。而在如sh和bash的Bourne shell，必须使用export去设置当前环境变量。再个例子，要去设置或改EDITOR环境变量，在csh或tcsh下将EDITOR设为/usr/local/bin/emacs：

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

而在Bourne shell下，则是：

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

也可以在命令行上加一个\$字符在变量之前从而取得环境变量。个例子，用echo \$TERM就会显示出\$TERM的值，其值就是shell取得\$TERM并用echo来显示的。

shell里有多特殊的字符代表着特殊的材料，我们叫做meta-characters。

最常用的就是

`*`字符，它可代表文件名的任何字符。 有些特殊字符用到文件名全域方面。假如，`echo *`和`ls`的效果是相同的，其就是 shell 取得了全部符合 `*`的文件名，并 `echo` 在命令行下示出来。

为了防止shell去分析些特殊字符， 我可在它之前加一个 `\`字符去明它只是普通字符。 `echo $TERM`就会示出的端情况， 而 `echo \$TERM` 就会示出 `$TERM` 几个字。

### 4.9.1. 改用的Shell

改的Shell的最方法是使用 `chsh` 命令。 行 `chsh` 将根据定的 `EDITOR` 境量入到那个器，假如没有定，就会入 `vi`器。 改"Shell:"行。

可使用 `chsh` 的 `-s`， 就能置的shell却又不用器。假如想把shell改 `bash` 可用下面的技巧。

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

使用的shells必在 `/etc/shells` 文件里列出。 假如从 [ports](#)里装一个shell，那就不用做了。 假如手工装一个shell，那就要手工添加去。

个例子，假如手工把 `bash`装到 `/usr/local/bin`里，要行一：

```
# echo "/usr/local/bin/bash" >> /etc/shells
```

然后行 `chsh`。

## 4.10. 文本器

FreeBSD 的很多配置都可以通文本文件来完成。 因此， 最好能熟悉某文本器。 FreeBSD 基本系中提供了一些， 也可以从 Ports Collection 安装其它器。

最容易学的而又的器是 `ee`器， 是个准的易器。 要 `ee`， 首先就要在命令行入 `ee filename`， `filename` 是一个要的文件名。 例如， 要 `/etc/rc.conf`就要入 `ee /etc/rc.conf`， 在 `ee`的控制内， 器所有功能的操作方法都示在最上方。 个 `^` 字符代表 上的 `Ctrl`， 所以 `^e` 就是 `Ctrl + e`合。 假如想 `ee`， 按 `Esc`， 就可器。 当修改了内容的候， 器会提示保存。

FreeBSD本身也可多有大功能的文本器， 例如 `vi`。 有其它在FreeBSD Ports里几， 像 `emacs` 和 `vim`。 些器有着大的功能， 但同学起来比。 不管， 假如从事文字方面的工作， 学如 `vim` 或 `emacs` 些有大功能的器用法， 在里工作里会省不少的。

很多需要修改文件或打字入的用程序都会自打一个文本器。 更改默使用的器， 置 `EDITOR` 境量。 参 [shells](#) 以取更多信息。

## 4.11. 和点

在一个系里， 硬件描述通常用法就是一个一个， 包括磁、 打印机、 和。 当 FreeBSD 程中， 大多数的都能探到并示出来， 也可以 `/var/run/dmesg.boot`， 引所有信息都在里面。

例如， `acd0` 就是 首个 IDE 光， 而 `kbd0` 代表。

在UNIX®操作系统里，大多数存在的特殊文件就是叫做设备点，他都定位在/dev目录里。

### 4.11.1. 建立设备点

当在系中添加新设备或将附加设备的支持加入内核之后，都必须建立设备点。

#### 4.11.1.1. DEVFS (DEVIce 文件系统)

这个文件系统，或叫 DEVFS，内核的设备命名在整体文件系统命名里提供通道，并不是建立或更改设备点，DEVFS只是设备的特文件系统行。

参 devfs(5) 手册以了解更多。

## 4.12. 二进制文件格式

要理解什么 FreeBSD 使用 elf(5) 格式，你必须首先了解一些 UNIX® 系中的三 "主要" 可执行文件格式的有知：

- a.out(5)

是最古老和"经典的" UNIX® 目标文件格式，格式在其文件的始有一个短小而又的首部，首部有一个魔幻数字，用来具体的格式(更多详情参 a.out(5))。格式包含3个要装入内存的段：.text, .data, 和 .bss, 以及一个符号表和一个字符串表。

- COFF

SVR3目标文件格式。其文件包括一个区段表(section table), 因此除了.text, .data, 和 .bss区段以外，可以包含其它的区段。

- elf(5)

COFF 的后，其特点是可以有多个区段，并可以使用32位或64位的。它有一个主要的缺点：ELF在其假设每个体系只有一 ABI。假设事上相当，甚至在商业化的SYSV世界中都是的（它至少有3 ABI: SVR4, Solaris, SCO）。

FreeBSD在某程度上解决这个，它提供一个工具，可以 一个已知的ELF可执行文件 它所遵从的ABI的信息。更多方面的知可以参手册 brandelf(1)

FreeBSD从"经典"中来，因此使用了a.out(5)格式，多BSD版本的行(直到3.X分支的始)也明了格式的有效性。然在那以前的某段，在FreeBSD系上建和行ELF格式的二进制可执行文件(和内核)也是可能的，但FreeBSD一始并不"到使用ELF作其缺省的格式。什么？，当Linux完成了到ELF格式的痛苦程后，却并不足以由此而放 a.out可执行文件格式，因正是由于它不活的，基于跳表的共享机制，使得售商和者建共享非常困。直到已有的ELF工具提供了一解决共享的法，并被普遍是"前方向"以后，的代价在FreeBSD界才被接受，并由此完成了。FreeBSD的共享机制其基更似于Sun SunOS™的共享机制，并且正因，其易用性很好。

那，什么会有多不同的格式？

回溯到蒙昧和暗的去，那只有硬件。硬件支了一个和小型的系。在的系上(PDP-11)a.out格式足以任表二进制文件的任。当人将UNIX®从的系中移植出来的候，

a.out格式被保留了下来，因于早期将UNIX®移植到 Motorola 68k, VAXen等系来，它是足可用的。

然后，一些明的硬件工程，如果可以件完成一些的明操作，那他就可以在硬件中少若干路，并可以CPU核心行得更快。当a.out格式用于新型的硬件系(在我叫它 RISC)，得并不合。因此，人多了多新的格式以便在的硬件系上能得比a.out格式更越的性能。如COFF，ECOFF，有其它一些晦的格式正是在个段被明出来的，人也研究了些格式的局限性，慢慢地最落到ELF格式。

同，程序的大小得越来越大，磁空(以及物理内存)相来却仍然小，因此共享的概念便生了。VM系也得越来越了。当所有些都建立在a.out格式的基上的时候，它的可用性随着个新特性的生就受到了重考。并且，人希望可以在行装某些西，或者在初始化代行以后可以部分程序代，以便主存器和交区。程言也得越来越，人希望可以在main()函数行之之前自行某些代。了所有些功能，人a.out格式作了很多改(hack)，他在某个段里基本也是可行的。随着的推移，a.out格式不得不加大量的代和度来足些需求。然ELF格式解决了多，但是从一个可用的系移到一个系却是痛苦的。因此直到保留a.out格式的代价比移到ELF格式的代价大的时候，人才会最到ELF格式。

然而，随着的推移，FreeBSD系本身的工具(特是器和装器)以派生的工具，其展却形成了个平行的分支。FreeBSD个分支加了共享，并修改了一些。而原先写了些工具的GNU人重写了些工具，并交叉提供了更化的支持，随意入了不同格式的支持，等等。然很多人希望建FreeBSD的交叉器，但他却并未如以，因FreeBSD的as和ld的源代更老旧，所以无法完成个任。新的GNU工具(binutils)支持交叉，ELF格式，共享，C++展，等等。并且，由于很多供商都布ELF格式的二进制文件，因而FreeBSD能行它将是一个很好的事情。

ELF格式比a.out格式要大些，同也允基系有更好的展性。ELF格式的有工具有着更好的，并且提供交叉支持，多人来是很重要的。ELF格式可能会微慢一些，但很量出来。外，在者之，有多也是不同的，比如它映射面的方式，理初始化代的方式，等等。所有些都不太重要，但也不同之。在将来的当时候，GENERIC内核将不再支持a.out格式，并且，当不再需要行留的a.out格式程序，内核也将不再提供其的支持。

## 4.13. 取得更多的

### 4.13.1. 机手册

最的使用明文莫于FreeBSD里的机手册了。几乎一个程序都会附上一短明，以介个程序的的基本功能以及参数的用法。我能通man命令来些明，而使用man命令却是的事情：

```
% man command
```

command 就是要了解的命令名称。个例子，想了解ls命令就入：

```
% man ls
```

些在手册分下列章：

1. 用命令。

2. 系统调用以及库代码。
3. C 源文件里的函数说明。
4. 编译程序。
5. 文件格式。
6. 游戏以及其他程序。
7. 各库。
8. 系统库以及命令。
9. 内核运行情况。

在某些情况下，同一主题的主页也会出现在手册的不同章节。 举个例子，系统里有 `chmod` 个用户命令，而又有 `chmod()` 系统调用。在这种情形下，应当向 `man` 命令指定需要的内容：

```
% man 1 chmod
```

就会显示出手册里的用户 `chmod` 命令。 通常，我在写入文档把特定参考内容在手册括号里注明。所以 `chmod(1)` 是指 `chmod` 用户命令，而 `chmod(2)` 是指系统调用。

如果已经知道命令的名字，只是不知道要使用的选项，那就比较好办。但名字都不知道的时候就可以利用 `man` 的搜索功能，它会在手册的介绍部分搜索指定的字，它的选项是 `-k`：

```
% man -k mail
```

当使用这个命令的时候，`man` 会把介绍里含有 "mail" 字样的命令列出来，通常上和 `apropos` 命令的功能是相同的。

有可能会看到 `/usr/bin` 下有好多命令但不知他的用途，只需这样做：

```
% cd /usr/bin
% man -f *
```

或者这样做

```
% cd /usr/bin
% whatis *
```

这个命令是一样的。

### 4.13.2. GNU Info 文件

FreeBSD 许多用户程序以及工具来自 Free 软件基金会 (FSF)。 作为手册的补充，有些程序提供了一组更具有活力的超文本说明 `info`， 它可用 `info` 命令来阅读他。 假如装上 `emacs`，也能利用 `emacs` 的 `info` 模式来阅读。

使用 `info(1)` 这个命令只需简单地输入:

```
% info
```

想得到简介, 请按 `h`。想快速得到的命令说明, 请按 `?`。



# Chapter 5. 安装应用程序: Packages 和 Ports

## 5.1. 概述

FreeBSD 将许多系统工具作为基本系统的一部分。然而，要完成更多的工作，可能需要安装更多的第三方应用。FreeBSD 提供了丰富的技术，用以在系统中安装第三方软件：FreeBSD Ports 套件（用于从源代码安装），以及 `packages`（用以从源代码的二进制版本安装）。这两种方法都可以用于从本地介质，或从网上直接安装喜爱的应用程序的最新版本。

读完本章，你将了解到：

- 如何安装第三方的二进制软件包。
- 如何使用 ports 套件从源代码构建第三方软件。
- 如何删除先前安装的软件包。
- 如何改变 Ports Collection 里面的一些参数，定制软件使用。
- 如何找到需要的软件包。
- 如何升级的软件包。

## 5.2. 软件安装

如果你以前使用过 UNIX® 系统，那典型的第三方软件安装的过程是像下面描述的：

1. 下载软件，软件的行版可能是源代码格式，或是一个二进制包。
2. 解软件(其中代表性的是用 `compress(1)`, `gzip(1)`, 或 `bzip2(1)` 解压的tar包)。
3. 阅读文档，了解如何安装。(多半一个文件名是INSTALL或README，或在doc/目录下的一些文档)
4. 如果软件是以源代码形式发布的，那就需要编译它。可能需要编写一个 `Makefile` 文件，或运行 `configure` 脚本，和其他的一些工作。
5. 编译和安装软件。

如果一切顺利的话，就很简单。如果你在安装一个软件包时产生一些错误，你可能需要检查一下它的代码，以使它能正常工作。

你可以使用“旧”的方式安装软件。然而，FreeBSD 提供了新技术：`packages` 和 `ports`。就在写这篇文章的时候，已有超过 36000 个第三方的应用程序可以使用了。

对于任意一个应用程序包，是一个可以下载的FreeBSD `package`文件。每个FreeBSD `package`包含了软件好的副本，它有一些配置文件或文档。一个下载的包文件可以用FreeBSD的包管理命令来操作，例如 `pkg_add(1)`, `pkg_delete(1)`, `pkg_info(1)` 等等。可以使用一个命令安装一个新的应用程序。

一个FreeBSD的port是一个可以从源代码生成应用程序的文件集合。

记住，如果你自己来编译的话，需要进行很多的操作（解压，编译，安装）。有些整理 port 的文件集合包含了系统需要完成的工作的必需信息。你可以运行一些命令，那些源代码就可以自动地下，

解，打丁，，直至安装完成。

上，ports 系也能做出被 `pkg_add` 的程序包和不久就要到的其他包管理命令来安装的件包。

Packages 和 ports 是互相依的。假想安装一个依赖于已安装的特定应用程序。应用程序和那个都已用于 FreeBSD ports 和 packages。如果使用命令或 ports 系来添加应用程序，个都必须注意是否被安装，如果没有，它会自先安装。

里出的技是很相似的，可能会奇怪什 FreeBSD 会弄出技。其，packages 和 ports 都有它自己的，使用一完全取决于自己的喜好。

### Package Benefits

- 一个 package 通常要比一个包含源代码的应用程序小得多。
- package 不需要行外的。于大型应用程序如 Mozilla，KDE 或 GNOME 来，得尤重要，特别是在的系源比差的情况下。
- package 不需要知道如何在 FreeBSD 上件的程。

### Ports Benefits

- package 在通常使用比保守的，是了保它能行在大多数的系上。通从 port 安装，可以微整来生合于理器的代（于 Pentium 4 或 AMD 的 Athlon CPU）。
- 一些件包已把与它相的能做和不能做的事情的都去了。例如，Apache 可能就配置了很多的。从 port 中安装，不一定要接受默的，可以自己来置。

在一些例子中，一个件有不同的配置存在多个 package。例如，Ghostscript 存在 ghostscript package 和 ghostscript-nox11 package 个配置 package，取决于是否安装了 X11 服器。的整 package 是可能的，但如果一个应用程序有超一个或个不同的，就不行了。

- 一些件的许可条件禁止采用二进制形式行。它必上源代码。
- 一些人不信任二进制行形式。至少有了源代码，（理上）可以自它，潜在的。
- 如果要自己件打丁，就需要有源代码。
- 一些人喜整天着源代码，所以他喜自源代码，修改源代码等等。

保持更新 ports，件列表 [FreeBSD ports 件列表](#) 和交告 [FreeBSD ports bugs 件列表](#)。



安装任何应用程序之前，首先 <http://vuxml.freebsd.org/> 上是否有于所安装的应用程序的安全告。

也可以安装 `ports-mgmt/portaudit`，它能自地已安装的应用程序的漏洞；此外，在安装程序之前它也会首先是否存在已知的漏洞。外，也可以使用 `portaudit -F -a` 个命令在安装某个件包之后作出。

章的其余部分将介在 FreeBSD 上如何使用 packages 和 ports 来安装和管理第三方件。

## 5.3. 要的应用程序

在安装任何应用程序之前，需要知道需要什么，那个应用程序叫什。

FreeBSD中可用的应用程序正在不断地增加着。幸运的是，有多种方法可以找到所需要的程序：

- FreeBSD站点上有一个可以搜索到的当前所有可用的应用程序列表，在 <http://www.FreeBSD.org/ports/>。它分很多类，既可以通程序的名称来搜索(如果知道名字)，也可以在分类中列出所有可用的应用程序。
- Dan Langille 运营着网站 FreshPorts，在 <http://www.FreshPorts.org/>。FreshPort时刻“追踪”着在 ports 中应用程序的变化。当有任何程序被升级，他就会 email 提醒你。
- 如果你不知道想要的应用程序的名字，可以通过 (<http://www.freshmeat.net/>) 网站来查，如果你到了应用程序，可以回 FreeBSD 的主站去看一下这个应用程序是否已被 port 去了。
- 如果你知道一个port的准确名字，但需要知道在哪个包里能找到它，可以使用 `whereis(1)` 命令。输入 `whereis file`, `file` 就是想安装的程序名字。如果系统找到了它，将被告知它在哪儿，例如：

```
# whereis lsof
lsof: /usr/ports/sysutils/lsof
```

报告我这个命令 `lsof` (一个系统配置程序)可以在 `/usr/ports/sysutils/lsof` 目录中找到。

- 可以使用 `echo(1)` 语句来查某个 port 是否存在于 ports 中。例如：

```
# echo /usr/ports/*/lsof*
/usr/ports/sysutils/lsof
```

Note that this will return any matched files downloaded into the `/usr/ports/distfiles` directory.

注意这条命令将会返回下载到 `/usr/ports/distfiles` 目录中所有符合条件的文件。

- 有另外一个查找需要的port的方法—是用ports collection 内嵌的搜索机制。要使用这个搜索，需要先到 `/usr/ports` 目录下面。在那个目录里面，运行 `make search name=program-name`, `program-name` 就是想查的程序名字。举个例子，如果你想查 `lsof`：

```
# cd /usr/ports
# make search name=lsof
Port:  lsof-4.56.4
Path:  /usr/ports/sysutils/lsof
Info:  Lists information about open files (similar to fstat(1))
Maint: obrien@FreeBSD.org
Index: sysutils
B-deps:
R-deps:
```

在输出的内容里面要特别注意包含 "Path:" 的行将告诉你在哪儿可以找到这个 port。如果要安装此 port，那其他输出的信息不是必需的，但是是显示了。

为了更深入的搜索，可以用 `make search key=string`, `string` 就是想搜索的部分内容。它将搜索 port 的名字、注释、描述和从属关系，如果你不知道想搜索的程序名字，可以利用它搜索一些包主来找到需要的。

上面的一些方法，搜索的关键词没有大小写区分的。搜索 "LSOF"的结果将和搜索"lsof"的结果一样。

## 5.4. 使用 Package 系统

在 FreeBSD 系统上有几个不同的工具用来管理 package：

- **sysinstall** 工具可以在正在运行的系统上运行，以完成安装、删除和列出可用的以及已安装的软件包的任一。如欲了解更多信息，请参考 [安装系统的软件包 \(package\)](#)。
- 其余的部分将介绍用于管理软件包的命令行工具。

### 5.4.1. 一个 package 的安装

可以用 `pkg_add(1)` 命令从本地文件或网上的服务器来安装一个 FreeBSD 软件包。

例 8. 在本地手下下一个 package, 并安装它

```
# ftp -a ftp2.FreeBSD.org
Connected to ftp2.FreeBSD.org.
220 ftp2.FreeBSD.org FTP server (Version 6.00LS) ready.
331 Guest login ok, send your email address as password.
230-
230-    This machine is in Vienna, VA, USA, hosted by Verio.
230-    Questions? E-mail freebsd@vienna.verio.net.
230-
230-
230 Guest login ok, access restrictions apply.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd /pub/FreeBSD/ports/packages/sysutils/
250 CWD command successful.
ftp> get lsof-4.56.4.tgz
local: lsof-4.56.4.tgz remote: lsof-4.56.4.tgz
200 PORT command successful.
150 Opening BINARY mode data connection for 'lsof-4.56.4.tgz' (92375 bytes).
100% |*****| 92375      00:00 ETA
226 Transfer complete.
92375 bytes received in 5.60 seconds (16.11 KB/s)
ftp> exit
# pkg_add lsof-4.56.4.tgz
```

如果没有本地 package 的安装 (如 FreeBSD CD-ROM)，可以运行 `pkg_add(1)` 命令并加上 `-r` 选项。这将迫使程序自行决定目标文件的正确格式和版本，然后自行从一个 FTP 站点检索和安装 package。

```
# pkg_add -r lsof
```

上面的例子将下载正确的 package，而不需要用到的干预就可以安装。如果你想指定 FreeBSD package 的

像站点，替主站点，就必相地置 `PACKAGESITE` 个境量，覆原来的置。 `pkg_add(1)` 使用 `fetch(3)` 下文件， 可以使用多境量， 包含 `FTP_PASSIVE_MODE`、 `FTP_PROXY`， 和 `FTP_PASSWORD`。 如果使用 FTP/HTTP 代理或在防火后面， 可能需要置些境量。 的列表参考 `fetch(3)`。 上述例子中用 `lsof` 替代了 `lsof-4.56.4`。 当使用程安装 Package 的候件名字不需要包含版本号。 `pkg_add(1)` 将自的到 个件最新的版本。



如果使用 FreeBSD-CURRENT 或 FreeBSD-STABLE版本的FreeBSD, `pkg_add(1)` 将下的用件的最新版本。 如果使用 -RELEASE 版本的 FreeBSD, 它将会得与 的版本相的件包版本。 可以通修改境量 `PACKAGESITE` 来改一行。 例如, 如果行 FreeBSD 8.1-RELEASE 系, 默情况下 `pkg_add(1)` 将从 `ftp://ftp.freebsd.org/pub/FreeBSD/ports/i386/packages-8.1-release/Latest/` 下的件包。 如果希望制 `pkg_add(1)` 下 FreeBSD 8-STABLE 的件包, 可以将 `PACKAGESITE` 置 `ftp://ftp.freebsd.org/pub/FreeBSD/ports/i386/packages-8-stable/Latest/`。

件包采用 .tgz 和 .tbz 格式。 可以在 `ftp://ftp.FreeBSD.org/pub/FreeBSD/ports/packages/` 下面或从 FreeBSD 的行光到, 它在 一个 4CD 的 FreeBSD 行版的 /packages目。 件包的与 /usr/ports 一致。 个分都有自己的目, 所有的件包可以在目 All中到。

件包系的目与ports的一致; 它共同成了整个 package/port。

## 5.4.2. 件包的管理

`pkg_info(1)` 是用于列出已安装的所有件包列表和描述的程序。

```
# pkg_info
cvsup-16.1      A general network file distribution system optimized for CV
docbook-1.2    Meta-port for the different versions of the DocBook DTD
...
```

`pkg_version(1)`是一个用来所有安装的件包版本的工具。 它可以用来比本地 package 的版本与 ports 目中的当前版本是否一致。

```
# pkg_version
cvsup              =
docbook            =
...
```

在第二列的符号指出了安装版本的相和本地ports目中可用的版本。

| 符号 | 含                                     |
|----|---------------------------------------|
| =  | 在本地ports中与已安装的件包版本相匹配。                |
| <  | 已安装的版本要比在ports中的版本旧。                  |
| >  | 已安装的版本要比在ports中的版本新 (本地的 port可能没有更新)。 |

| 符号 | 含义   |
|----|--|
| ?  | 已安装的软件包无法在ports索引中找到。(可能发生的事情，举个例子，原先安装的一个 port 从 port 中移出或改名了)  |
| *  | 软件包有很多版本。  |
| !  | 已安装的软件包在索引中存有，但是由于某些原因 <b>pkg_version</b> 无法比较已安装的软件包与索引中相应的版本号。 |

### 5.4.3. 删除一个软件包

要删除先前安装的软件 package，只要使用 **pkg\_delete(1)** 工具。

```
# pkg_delete xchat-1.7.1
```

需要注意的是， **pkg\_delete(1)** 需要提供完整的包名；如果只是指定了类似 *xchat* 而不是 *xchat-1.7.1* 的名字，它将拒绝进行操作。不过，可以使用 **pkg\_version(1)** 来了解安装的 package 的版本。除此之外，也可以使用通配符：

```
# pkg_delete xchat\*
```

这样，所有名字以 *xchat* 的 package 都会被删掉。

### 5.4.4. 其它

所有已安装的 package 信息都保存在 */var/db/pkg* 目录下。安装文件的列表和每个 package 的内容和描述都能在这个目录的相应文件中找到。

## 5.5. 使用Ports Collection

下面的几个小节中，给出了如何使用 Ports 套件来在您的系统中安装或卸载程序的介绍。关于可用的 **make targets** 以及环境变量的介绍，可以在 **ports(7)** 中找到。

### 5.5.1. 获得Ports Collection

在能使用 ports 之前，必须先获得 Ports Collection - 本上是 */usr/ports* 目录下的一堆 Makefile、补丁和描述文件。

在安装 FreeBSD 系统的时候，**sysinstall** 会询问是否需要安装 Ports Collection。如果回答 **no**，那可以用下面的指令来安装 Ports Collection：

## Procedure: CVSup 方法

保持本地 Ports 套件最新的一个快捷的方法，是使用 CVSup 来行更新。如果你希望了解更多关于 CVSup 的，参 使用 CVSup。



在 FreeBSD 系里 CVSup 的 叫作 csup。

在首次行 csup 之前，必须 /usr/ports 是空的！如果你之前已用其他地方安装了一个 Ports 套件，csup 可能不会自除已在上游服务器上除掉的丁文件。

### 1. 行 csup:

```
# csup -L 2 -h cvsup.FreeBSD.org /usr/shared/examples/cvsup/ports-supfile
```

将 *cvsup.FreeBSD.org* 改最近的 CVSup 服务器。参 CVSup 像 (CVSup 站点) 中的 像站点完整列表。

有可能希望使用自己的 ports-supfile, 比如, 不想次都通 命令行来指定所使用的 CVSup 服务器。



- 情况下，需要以 root 身将 /usr/shared/examples/cvsup/ports-supfile 制到新的位置，例如 /root 或的主目。
- ports-supfile。
- 把 *CHANGE\_THIS.FreeBSD.org* 修改成最近的 CVSup 服务器。可以参考 CVSup 像 (CVSup 站点) 中的 像站点完整列表。
- 接下来按如下的方式行 csup :

```
# csup -L 2 /root/ports-supfile
```

- 此后行 **csup(1)** 命令将下最近所行的改，并将它用到 Ports Collection 上，不 一 程并不重新系上的 ports。

### Procedure: Portsnap 方式

Portsnap 是用于分布 Ports 套件的唯一一套系。参[使用 Portsnap](#) 以了解关于 Portsnap 功能更详细的介绍。

1. 下载最近的 Ports 套件快照到 /var/db/portsnap。可以根据需要在之后连接 Internet。

```
# portsnap fetch
```

2. 假如是首次运行 Portsnap，需要将快照放到 /usr/ports：

```
# portsnap extract
```

如果已有装好的 /usr/ports 而只想更新，运行下面的命令：

```
# portsnap update
```

### Procedure: Sysinstall 方式

该方法需要使用 sysinstall 从安装介质上安装 Ports 套件。注意，安装的将是分布行版的旧版 Ports 套件。如果能连接 Internet，使用前面介绍的方法之一。

1. 以 root 身份运行 **sysinstall**：

```
# sysinstall
```

2. 用光标向下配置 Configure，并按 **Enter**。
3. 向下并配置 Distributions，按 **Enter**。
4. 配置 ports，并按 **Space**。
5. 配置 Exit，并按 **Enter**。
6. 配置所希望的安装介质，例如 CDROM、FTP，等等。
7. 配置 Exit 并按 **Enter**。
8. 按 **X** 退出 sysinstall。

## 5.5.2. 安装 Ports

当提到 Ports Collection 时，第一个要说明的就是何谓 "skeleton"。简单地，port skeleton 是一个程序在 FreeBSD 上简单地并安装的所需文件的最小集合。一个 port skeleton 包含：

- 一个 Makefile。Makefile 包括好几个部分，指出应用程序是如何编译以及将被安装在系统的哪些地方。



- 一个 `distinfo` 文件。每个文件包括一些信息：一些文件用来下载后的文件校验和行（使用 `sha256(1)`），来保证在下过程中文件没有被破坏。
- 一个 `files` 目录。这个目录包括在 FreeBSD 系统上和安装程序需要用到的补丁。这些补丁基本上都是些小文件，指出特定文件作了些修正。它们都是文本的格式，基本上是行的“删除第 10 行”或“将第 26 行改...”，补丁文件也被称作“diffs”，它们由 `diff(1)` 程序生成。

这个目录也包含了在 `port` 要用到的其它文件。

- 一个 `pkg-descr` 文件。它是一个提供更多，有件的多行描述。
- 一个 `pkg-plist` 文件。它是即将被安装的所有文件的列表。它告诉 `ports` 系统在卸时需要删除哪些文件。

一些 `ports` 有些其它的文件，例如 `pkg-message`。 `ports` 系统在一些特殊情况下会用到些文件。如果你想知道些文件更多的以及 `ports` 的概要，参考 [FreeBSD Porter's Handbook](#)。

`port` 里面包含着如何源代的指令，但不包含真正的源代。可以在网上或 CD-ROM 上得源代。源代可能被布成任何格式。一般来是一个被 `tar` 和 `gzip` 的文件，或者是被一些其他的工具或未的文件。 `ports` 中个程序源代示文件叫“`distfile`”，安装 FreeBSD `port` 的方法不止。



必须使用 `root` 用登录后安装 `ports`。

在安装任何 `port` 之前，首先保证已更新到了最新的 `Ports Collection`，并 <http://vuxml.freebsd.org/> 中是否有与那个 `port` 有的安全。



在安装用程序之前，可以使用 `portaudit` 来自地是否存在已知的安全。这个工具同可以在 `Ports Collection` ([ports-mgmt/portaudit](#)) 中找到。在安装新的 `port` 之前，可以考先运行一下 `portaudit -f` 来取最新的漏洞数据。在天的周期性系安全察，数据会被自更新，并且会在之后施安全。欲了解一的情况，参考 `portaudit(1)` 和 `periodic(8)`。

`Ports` 套件假定有可用的 Internet 接。如果没有，需要将 `distfile` 手工放到 `/usr/ports/distfiles` 中。

要始操作，首先入要安装 `port` 的目：

```
# cd /usr/ports/sysutils/lsof
```

一旦入了 `lsof` 的目，将会看到个 `port` 的。下一就是 `make`，或“”个 `port`。只需在命令行地入 `make` 命令就可松完成一工作。做好之后，可以看到下面的信息：

```
# make
>> lsof_4.57D.freebsd.tar.gz doesn't seem to exist in /usr/ports/distfiles/.
>> Attempting to fetch from ftp://lsof.itap.purdue.edu/pub/tools/unix/lsof/.
===> Extracting for lsof-4.57
...
[extraction output snipped]
...
>> Checksum OK for lsof_4.57D.freebsd.tar.gz.
===> Patching for lsof-4.57
===> Applying FreeBSD patches for lsof-4.57
===> Configuring for lsof-4.57
...
[configure output snipped]
...
===> Building for lsof-4.57
...
[compilation output snipped]
...
#
```

注意，一旦完成，就会回到命令行。下一安装 port，要安装它只需要在 `make` 命令后跟上一个 `install` 即可：

```
# make install
===> Installing for lsof-4.57
...
[installation output snipped]
...
===> Generating temporary packing list
===> Compressing manual pages for lsof-4.57
===> Registering installation for lsof-4.57
===> SECURITY NOTE:
      This port has installed the following binaries which execute with
      increased privileges.
#
```

一旦返回到提示符，就可以运行安装的程序了。因为 `lsof` 是一个给予特殊权限的程序，因此显示了一个安全警告。在和安装 ports 的时候，留意任何输出的警告。

删除工作目录是个好主意，这个目录中包含了全部在程序中用到的文件。有些文件不会占用宝贵的磁盘空间，而且可能会提升新版本的 port 安装来麻烦。

```
# make clean
===> Cleaning for lsof-4.57
#
```



使用 `make install clean` 可以一完成 `make`、`make install` 和 `make clean` 三个分的工作。



一些 shell 会存环境量 `PATH` 中指定的目录里的可执行文件，以加速它的速度。如果你使用的是 `tcsh` shell，在安装 port 之后可能需要行 `rehash` 命令，然后才能行新安装的那些命令。这个命令可以在类似 `tcsh` 的 shell 中使用。对于类似 `sh` 的 shell，的命令是 `hash -r`。参的 shell 的文以了解一的情况。

某些第三方 DVD-ROM 品，如 [FreeBSD Mall](#) 的 FreeBSD Toolkit 中包含了 distfiles。些文件可以与 Ports 套件配合使用。将 DVD-ROM 挂接到 `/cdrom`。如果你使用不同的挂接点，置 `make` 量 `CD_MOUNTPTS`。如果你上有需要的 distfiles，会自使用。



注意，少数 ports 并不允通 CD-ROM 行。可能是由于下之前需要填写注册表格，或者不允再次布，或者有一些其它原因。如果你希望安装在 CD-ROM 上没有的 port，就需要在操作了。

ports 系使用 `fetch(1)` 去下文件，它有很多可以置的环境量，其中包括 `FTP_PASSIVE_MODE`、`FTP_PROXY`，和 `FTP_PASSWORD`。如果你在防火之后，或使用 FTP/HTTP代理，就可能需要置它。完整的明看 `fetch(3)`。

当使用者不是所有都能接上网，可以利用 `make fetch`。只要在目 (`/usr/ports`) 下行个命令，所有需要的文件都将被下。个命令也同可以在下目中使用，例如：`/usr/ports/net`。注意，如果一个 port 有一些依的或其他 port，它将不下些依的 port 的 distfile 文件，如果你想取所有依的 port 的所有 distfile，用 `fetch-recursive` 命令代替 `fetch` 命令。



可以在一个或在目所有的 port，或者使用上述提到的 `make fetch` 命令。是非常危的，因有一些 port 不能并存。或者有—可能，一些 port 会安装个不同的文件，但是却是相同的文件名。

在一些的例子中，用可能需要在除了 `MASTER_SITES` 以外的一个站点(本地已下下来的文件)去得一个文件包。可以用以下命令不使用 `MASTER_SITES`:

```
# cd /usr/ports/directory
# make MASTER_SITE_OVERRIDE= \
ftp://ftp.FreeBSD.org/pub/FreeBSD/ports/distfiles/ fetch
```

在个例子中，我把 `MASTER_SITES` 个改为了 `ftp.FreeBSD.org/pub/FreeBSD/ports/distfiles/`。



一些 port 允(或甚至要求)指定来用/禁用用程序中非必需的功能，一些安全，以及其他可以制的内容。具有代表性的包括 [www/mozilla](#)、[security/gpgme](#)、以及 [mail/sylpheed-claws](#)。如果存在，通常会在出提示。

### 5.5.2.1. 改默的 Ports 目

有，使用不同的工作目和目可能很有用(甚至是必要的)。可以用 `WRKDIRPREFIX` 和 `PREFIX` 个量来改默的目。例如：

```
# make WRKDIRPREFIX=/usr/home/example/ports install
```

将在 `/usr/home/example/ports` 中安装 port 并把所有的文件安装到 `/usr/local`。

```
# make PREFIX=/usr/home/example/local install
```

将在 `/usr/ports` 安装它并安装到 `/usr/home/example/local`。

当然,

```
# make WRKDIRPREFIX=../ports PREFIX=../local install
```

将包含配置 (没有办法在一行把它写完, 但用户已知道怎么回事了)。

另外, 这些变量也可以作环境变量来设置。参考的 shell 的机手册上于如何设置环境变量的说明。

#### 5.5.2.2. 处理 `imake`

一些 port 使用 `imake` (是 X Window 系的一部分) 不能正常地配合 `PREFIX`, 它会坚持把文件安装到 `/usr/X11R6` 下面。类似地, 一些 Perl port 会忽略 `PREFIX` 并把文件安装到 Perl 的目录中。这些 port 尊重 `PREFIX` 是因甚至是不可能的东西。

#### 5.5.2.3. 重新配置 Ports

当在有些 ports 的时候, 可能会出一个基于 ncurses 的菜单来安装一些软件。通常我们都希望能在一个 port 被安装了以后能再次安装菜单以添加或删除或修改些东西。上面有很多方法来做这件事情。一个方法是入那个 port 的目录后输入 `make config`, 之后便会再次显示出菜单和已装的目录。另一个方法是用 `make showconfig`, 它会显示出所有的配置。有一个方法是运行 `make rmconfig`, 它将删除所有已装的目录。有些更详细的内容参看 [ports\(7\)](#)。

#### 5.5.3. 卸载已安装的 Ports

在已了解了如何安装 ports, 并希望进一步了解如何卸载, 特别是在成功地安装了某个 port 之后。我们将卸载前面例子 (假如没有注意的, 是 `lsOf`) 中安装的 port。Ports 可以同 packages 以完全相同的方式 (在 [Packages](#) 一章中进行了介绍) 卸载, 方法是使用 `pkg_delete(1)` 命令:

```
# pkg_delete lsOf-4.57
```

#### 5.5.4. 升级 Ports

首先, 使用 `pkg_version(1)` 命令来列出 Ports Collection 中提供了更新版本的那些 port:

```
# pkg_version -v
```

#### 5.5.4.1. /usr/ports/UPDATING

在更新了 Ports 套件之后，在升 port 之前，看看 /usr/ports/UPDATING。这个文件中介绍了在升时需要注意的，以及一些可能需要行的操作。可能包括更改文件格式、配置文件位置的，以及和先前版本的兼容性等等。

如果 UPDATING 与本文中介绍的内容不同，以 UPDATING 为准。

#### 5.5.4.2. 使用 Portupgrade 来更新 Ports

portupgrade 工具是用来简化升已安装的 port 的操作的。它通过 [ports-mgmt/portupgrade](#) port 来提供。可以像其它 port 那样，使用 `make install clean` 命令来安装它：

```
# cd /usr/ports/ports-mgmt/portupgrade
# make install clean
```

使用 `pkgdb -F` 命令来扫描已安装的 port 的列表，并修正其所告的不一致。在每次升之前，有规律地行它是个好主意。

行 `portupgrade -a`，portupgrade 将开始并升系中所安装的所有 ports。如果希望在升操作中得到，指定 `-i` 参数。

```
# portupgrade -ai
```

如果只希望升某个特定的应用程序，而非全部可用的 port，使用 `portupgrade pkgname`。如果 portupgrade 首先升指定应用程序的，指定 `-R` 参数。

```
# portupgrade -R firefox
```

要使用 package 而不是 ports 来行安装，需要指定 `-P`。如果指定了个，portupgrade 会搜索 `PKG_PATH` 中指定的本地目录，如果没有到，从程站点下。如果本地没有到，而且程站点也没有成功地下包，portupgrade 将使用 ports。要禁止使用 port，可以指定 `-PP`。

```
# portupgrade -PP gnome2
```

如果只想下 distfiles (或者，如果指定了 `-P` 的，是 packages) 而不想建或安装任何东西，可以使用 `-F`。要了解更多信息，参考 [portupgrade\(1\)](#)。

#### 5.5.4.3. 使用 Portmanager 来升 Ports

Portmanager 是一个用以简化已安装 port 升操作的工具。它可以通过 [ports-mgmt/portmanager](#) port 安装：

```
# cd /usr/ports/ports-mgmt/portmanager
# make install clean
```

可以通过下面的命令来升级所有已安装的 port：

```
# portmanager -u
```

如果希望 Portmanager 在执行操作之前都给出提示，使用 `-ui` 参数。Portmanager 也可以用来在系统中安装新的 ports。与通常的 `make install clean` 命令不同，它会在升级和安装所依赖的 port 之前升级所有依赖包。

```
# portmanager x11/gnome2
```

如果对于所依赖 port 的依赖有任何问题，可以用 Portmanager 来以正确的顺序重新构建它们。完成之后，所有的 port 也将被重新构建。

```
# portmanager graphics/gimp -f
```

要了解更多信息，参考 [portmanager\(1\)](#)。

#### 5.5.4.4. 使用 Portmaster 升级 Ports

Portmaster 是另外一个用来升级已安装的 ports 的工具。Portmaster 被设计成尽可能使用 "基本" 系统中能找到的工具（它不依赖于其他的 ports）和 `/var/db/pkg/` 中的信息来找出需要升级的 ports。可以在 [ports-mgmt/portmaster](#) 找到它：

```
# cd /usr/ports/ports-mgmt/portmaster
# make install clean
```

Portmaster groups ports into four categories:

Portmaster 把 ports 分成4类：

- Root ports (不依赖其他的 ports，也不被依赖)
- Trunk ports (不依赖其他的 ports，但是被其他的 ports 依赖)
- Branch ports (依赖于其他的 ports，同时也被依赖)
- Leaf ports (依赖于其他的 ports，但不被依赖)

可以使用 `-L` 列出所有已安装的 ports 和存在更新的 ports：

```
# 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)
==>>> apache-2.2.3
      ==>>> New version available: apache-2.2.8
...
==>>> Leaf ports (Have dependencies, not depended on)
==>>> automake-1.9.6_2
==>>> bash-3.1.17
      ==>>> New version available: bash-3.2.33
...
==>>> 32 leaf ports

==>>> 137 total installed ports
      ==>>> 83 have new versions available
```

可以使用 `portmaster -L` 的命令来升所有已安装的 ports：

```
# portmaster -a
```



Portmaster 默认在除一个已有的 port 前会做一个包。如果新的版本能被成功安装，Portmaster 将删除它。使用 `-b` 后 Portmaster 便不会自动删除。加上 `-i` 之后 Portmaster 将进入交互模式，在升一个 port 以前提示给予。

如果在升的过程中出了错，可以使用 `-f` 来升/重新所有的 ports：

```
# portmaster -af
```

同样也可以使用 Portmaster 往系统里安装新的 ports，升所有的依赖系之后并安装新的 port：

```
# portmaster shells/bash
```

更多的信息参 [portmaster\(8\)](#)

### 5.5.5. Ports 和磁盘空间

使用 Ports 套件会最用完磁盘空间。在通 ports 和安装件之后，得清理的 work 目，其方法是使用 `make clean` 命令。可以使用下面的命令来清理整个 Ports 套件：

```
# portsclean -C
```

随着时间的推移，`distfiles` 目录中可能会积累下大量源代码文件。你可以手工删除这些文件，也可以使用下面的命令来删除所有 port 都不引用的文件：

```
# portsclean -D
```

除此之外，也可以用下列命令去目前安装的 port 没有使用的源包文件：

```
# portsclean -DD
```



这个 `portsclean` 工具是 `portupgrade` 套件的一部分。

不要忘了删除那些已安装，但已不再使用的 ports。用于自动完成这项工作的好工具是 `ports-mgmt/pkg_cutleaves` port。

## 5.6. 安装之后要做点什？

通常，通过 port 安装完一个软件后，可以看看它的一些文档（如果它包含文档的），或需要看看它的配置文件，来保证这个软件的运行，或在机器重启的时候（如果它是一个服务的），等等。

对于不同的软件有着不同的配置。不管如何，如果你装好了一个软件，但是不知道下一步该做什么的时候，一些小技巧可能可以帮助你：

- 使用 `pkg_info(1)` 命令，它能查到安装了哪些文件，以及装在哪儿。举个例子，如果你安装了 `FooPackage version 1.0.0`，那么该命令

```
# pkg_info -L foopackage-1.0.0 | less
```

将显示这个软件包安装的所有文件，要特别注意在 `man/` 目录里面的文件，它可能是手册，`etc/` 目录里面的配置文件，以及 `doc/` 目录下面更多的文档。

如果你不确定已安装好的软件版本，可以使用下面的命令

```
# pkg_info | grep -i foopackage
```

它将会查到所有已安装的软件包名字中包含 `foopackage` 的软件包。对于其他的，只需要在命令行中替换 `foopackage`。

- 一旦一些软件手册已被安装，你可以使用 `man(1)` 来看它。同样的，如果有的，你可以完整的看一看配置文件的示例，以及任何其他的文档。
- 如果你用软件有网站，你可以从网站上得到文档，常有的解答，或其他更多。如果你不知道它的网站地址，使用下面的命令



```
# pkg_info foopackage-1.0.0
```

一个 **WWW:** 行, 如果它存在, 它将提供一个程序的网站URL.

- **Ports** 如果需要在服务器上运行(就像互联网服务器), 它通常会把一个脚本的示例放入 `/usr/local/etc/rc.d` 目录。为了保证性, 可以查看脚本, 并可能更改脚本的名字。详情看[脚本](#)。

## 5.7. 如何处理坏掉的 Ports

如果某个 port 无法正常工作, 有几件事得做, 包括:

1. 在 [报告数据](#) 中是否有尚未提交的修正。如果有, 可以使用所提的修正。
2. 要求 port 的维护人 (maintainer) 提供帮助。输入 `make maintainer` 或 Makefile 中维护人的电子邮件地址。得把 port 的名字和版本写在邮件里 (Makefile 中的 `$FreeBSD:` 一行) 并把输出的几行给 maintainer。



某些 ports 并非一个人做, 而是写了一个 [邮件列表](#)。多, 但并非所有 port, 使用类似 `freebsd-listname@FreeBSD.org` 的地址。在提出前考虑一点。

特地, 由 [ports@FreeBSD.org](#) 做的 port, 网上并没有人做。这个邮件列表的人会觉得提供的修正和支持。我一直都需要更多志愿者!

如果没有得到回答, 可以使用 `send-pr(1)` 来提交报告 (参看 [如何撰写 FreeBSD 报告](#))。

3. 修正它! [Porter 手册](#) 中提供了关于 "Ports" 基础的信息, 通晓了解些内容, 就能修正偶然坏掉的 port, 或甚至提交自己的 port 了!
4. 从最近的 FTP 站点下一个好的安装包。"中央的" package collection 在 [ftp.FreeBSD.org](#) 的 [packages](#) 目录中, 但在此之前 事先 问一下是否存在最近的 [镜像网站](#)! 通常情况下 些安装包都可以直接使用, 而且比自行快一些。安装程序本身可以通过 `pkg_add(1)` 来完成。

# Chapter 6. X Window 系

## 6.1. 概述

FreeBSD 使用 X11 来提供功能强大的图形用户界面。X11 是一个可以免费使用的 X 系，其包括 Xorg。FreeBSD 中默认使用并受官方支持的 X11 即是 Xorg，它是由 X.Org 基金会提供的 X11 服务，采用与 FreeBSD 类似的授权。此外，也有一些用于 FreeBSD 的商业 X 服务器。

欲了解 X11 所支持的显示等硬件，看看 [Xorg](#) 网站。

在看完一章后，你将了解：

- X 系的不同组件，它们是如何协同工作的。
- 如何安装和配置 X11。
- 如何安装和使用不同的窗口管理器。
- 如何在 X11 中使用 TrueType® 字体。
- 如何配置系统置入形登录 (XDM)。

在看完一章之前，看看：

- 知道如何安装额外的第三方应用程序([安装应用程序: Packages 和 Ports](#))。

## 6.2. 理解 X

对于那些熟悉其他图形环境，比如 Microsoft® Windows® 或者 Mac OS® 的用户来说，第一次使用 X 可能会感觉很陌生。

通常并不需要深入了解各个 X 组件的作用以及它们之间的相互影响，不过，了解一些关于它的基础知识，有助于更好地利用 X 的强大功能。

### 6.2.1. 为什么要使用 X?

X 不是第一个 UNIX® 而有的系，但它是最流行的。X 的原始版本在 X 之前就已经在另外一个系上工作了。那个系的名字叫做 "W" (就是 "Window")。X 只是字母 W 后面的一个。

X 可以被叫做 "X", "X Window 系", "X11", 等等。把 X11 称做 "X Windows" 可能会冒犯某些人；看看 [X\(7\)](#) 可以了解更多的信息。

### 6.2.2. X 客户机/服务器模型

X 一开始就是网络而有的，所以采用了 "client-server" 模型。在 X 模型中，"X server" 运行在有显示器，鼠标的计算机上。服务器用来管理显示信息，处理来自鼠标和鼠标的输入信息，并与其他输入输出交互（比如作为输入的 "tablet"，或者作为输出的投影）。一个 X 应用程序（比如 XTerm，或者 [getenv\(3\)](#)）就是一个 "客户程序 (client)"。客户程序向服务器发送信息，如 "请在某些座位上画一个窗口"，而服务器返回处理信息，如 "用鼠标点了 OK 按钮"。

如果家或公司环境中只有一台使用 FreeBSD 的计算机，就只能在同一台计算机上运行 X server 和 X client 了。

然而，如果有许多运行 FreeBSD 的机器，可以在它们的计算机上运行 X server，而在比它们高的服务器上运行 X 应用程序。在这样的环境中，X server 和 X client 之间的通信就可以通过网路来进行。

可能会有一些人感到困惑，因为 X 的架构和他料想的有些不同。他以 "X server" 是运行在功能强大的大型机上的，而 "X client" 是运行在他面前的计算机上的。

记住，X server 是有窗口和显示器的那台计算机，而 X client 是那些显示窗口的程序。

Client 和 server 不一定都要运行在同一操作系统上，它甚至无需在同一型号的计算机上运行。在 Microsoft® Windows® 或 Apple 公司的 Mac OS® 上运行 X server 也是可以的，在它上面也有很多免费的和商业化的应用程序。

### 6.2.3. 窗口管理器

X 的哲学很像 UNIX® 的哲学，"tools, not policy"。就意味着 X 不会去规定任何事物如何去完成，而是，只为用户提供一些工具，至于决定如何使用这些工具是用自己的事情。

这套哲学展示了 X，它不会规定窗口在屏幕上是什么样子，要如何移动鼠标，用什么来切身体（比如，Alt + Tab 按钮，在 Microsoft® Windows® 环境中的作用），这个窗口的工具条看起来像什么，他是否有什么按钮等等。

事实上，X 行使了一个叫做 "窗口管理器" 的应用程序的职责。有很多这样的程序可用：AfterStep, Blackbox, ctwm, Enlightenment, fvwm, Sawfish, twm, Window Maker, 等等。每一个窗口管理器都提供了不同的界面和感觉；其中一些支持 "虚拟桌面"；有一些允许可以定制一些用来管理桌面；一些有"初始"按钮，或者其他类似的；一些是 "可定制主题的(themeable)"，通常安装新的主题，可以完全改变外观。这些以及很多其他的窗口管理器，都可以在 Ports Collection 的 x11-wm 分类目里找到。

另外，KDE 和 GNOME 桌面环境都有他们自己的窗口管理器与桌面集成。

每个窗口管理器也有不同的配置机制；有些需要手工来写配置文件，而另外一些可以使用 GUI 工具来完成大部分的配置任务，例如而言，(Sawfish) 就使用 Lisp 语言写配置文件。

## 焦点策略

窗口管理器的一个特性是鼠标的 "focus policy"。每个窗口系都需要有一个窗口的方法接受鼠标的输入信息，以及当前窗口处于可用状态。

通常比熟悉的是一个叫做 "click-to-focus" 的焦点策略。它是 Microsoft® Windows® 使用的典型焦点策略，也就是在一个窗口上点击一下鼠标，该窗口就处于当前可用的状态。

X 不支持一些特殊的焦点策略。确切地讲，窗口管理器控制着在什么时候哪个窗口有焦点。不同的窗口管理器支持不同的焦点方案。它们都支持点击即得焦点，而且它们中的大多数都支持好几套方案。

最流行的焦点策略：

### focus-follows-mouse



鼠标指示器下面的窗口就是得焦点的窗口。该窗口不一定位于其他所有窗口之上。通常将鼠标移到一个窗口就可以改焦点，而不需要在它上面点击。

### sloppy-focus

该方式是 focus-follows-mouse 策略的一个小小扩展。对于 focus-follows-mouse，如果把鼠标移到了根窗口（或桌面背景）上，所有的其它窗口都会失去焦点，而相机的全部输入也会丢失。如果用了 sloppy-focus，只有当指针入新窗口，窗口焦点才会生成化，而当退出当前窗口是不会变化的。

### click-to-focus

当前窗口由鼠标点击来。窗口被"突出显示"，突出在所有其他窗口的前面。即使指针被移向了另一个窗口，所有的输入仍会被该窗口接收。

许多窗口管理器支持其他的策略，与这些相比又有些变化。可以看具体窗口管理器的文档。

## 6.2.4. 窗口部件

提供工具而非策略的 X 方法使得在 X 应用程序屏幕上看到的窗口部件得到了大大的扩展。

"Widget" 只是 X 窗口接口中所有列出的一个，它可以用某种方法来点击或操作；如按钮，文本框，复选按钮，列表框等等。Microsoft® Windows® 把它们叫做"控件"。

Microsoft® Windows® 和苹果公司的 Mac OS® 都有一个风格的窗口部件策略。应用程序开发者被建议保持他的应用程序共享一个普通的所谓即所得的用户界面。对于 X，它并不要求一个特殊的图形风格或一套相符合的窗口部件集。

好的效果是不能期望 X 应用程序只有一个普通的所谓即所得的界面。有很多的流行的窗口部件集，包括来自于 MIT 的 Athena，Motif® (模仿 Microsoft® Windows® 的窗口风格，所有部件都具有斜面和3D灰色度)，OpenLook，等等。

如今，大多数比新的 X 应用程序采用一种新式的窗口，包括 KDE 所使用的 Qt，以及 GNOME 所使用的 GTK+。在 X 窗口系下，UNIX® 界面的一些所谓即所得特性作了一些收敛，以使初学者感到更容易一些。

## 6.3. 安装 X11

Xorg 是 FreeBSD 上的默认 X11 实现。Xorg 是由 X.Org 基金会发行的开放源代码 X Window 系统中的 X 服务。Xorg 基于 XFree86™ 4.4RC2 和 X11R6.6 的代码。从 FreeBSD Ports 套件可以安装 Xorg 的 7.7 版本。

如果需要从 Ports Collection 安装和安装 Xorg：

```
# cd /usr/ports/x11/xorg
# make install clean
```



要完整地安装 Xorg 需要至少 4 GB 的剩余磁盘空间。

另外 X11 也可以直接从 package 来安装。我们提供了可以与 `pkg_add(1)` 工具配合使用的 X11 安装包。如果从程序下安装，在使用 `pkg_add(1)` 时不要指定版本号。`pkg_add(1)` 会自动地下载最新版本的安装包。

想要从 package 安装 Xorg，请输入下面的命令：

```
# pkg_add -r xorg
```



上面的例子介绍了如何安装完整的 X11 软件包，包括服务器端，客户端，字体等等。此外，也有一些单独的 X11 的 ports 和 packages。

另外，如果需要最小化的 X11 软件，也可以安装 [x11/xorg-minimal](#)。

一章余下的部分将会讲解如何配置 X11，以及如何配置一个高效的桌面环境。

## 6.4. 配置 X11

### 6.4.1. 开始之前

在配置 X11 之前，需要了解所安装的系统的相关信息：

- 显示器规格
- 显示器的芯片型号
- 显示器的内存容量

显示器的规格被 X11 用来决定显示的分辨率和刷新率。有些规格通常可以从显示器所附的文档中，以及制造商的网站得到。需要知道两个数字：垂直刷新率和水平刷新率。

显示器的芯片型号将决定 X11 使用什么模式来驱动硬件。尽管系统能自动输出大多数的硬件，但事先了解在自动输出的时候是很有用的。

显示器的内存大小决定了系统支持的分辨率和色色深度。了解这些限制非常重要。

## 6.4.2. 配置 X11

对于 Xorg 7.3 这个版本，可以不需要任何的配置文件就能运行，在提示符下输入如下命令：

```
% startx
```

从 Xorg 7.4 开始，可以使用 HAL 自动检测和鼠键。Ports [sysutils/hal](#) 和 [devel/dbus](#) 将被作为 [x11/xorg](#) 所依赖的包安装系统。并且需要在 `/etc/rc.conf` 文件中启用：

```
hald_enable="YES"
dbus_enable="YES"
```

在更深入的配置 Xorg 以前，需要进行一些服务（手工输入或者重启机器）。

自动配置对于某些硬件可能不起作用或者无法做到期望的配置。在这种情况下就有必要做一些手工配置。



例如 GNOME，KDE 或 Xfce 之类的桌面环境，大多都提供了一些允许使用非常易用的工具，来设置像分辨率之类的显示参数。所以如果得到默认的配置并不合适，而且打算安装一个桌面环境，那就完成桌面环境的安装，并使用合适的显示设置工具。

配置 X11 需要一些步骤。第一步是以超级用户的身分建立初始的配置文件：

```
# Xorg -configure
```

它会在 `/root` 中生成一个叫做 `xorg.conf.new` 的配置文件（无论使用 [su\(1\)](#) 或直接登录，都会改变默认的 `$HOME` 目录量）。X11 程序将探测系统中的图形硬件，并将探测到的硬件信息写入配置文件，以便加载正确的驱动程序。

下一步是保存配置文件，以便 Xorg 能同系统中的图形正常工作。对于 Xorg 7.3 或者之前的版本，输入：

```
# Xorg -config xorg.conf.new
```

从 Xorg 7.4 和更高的版本开始，它会将显示出一个灰色的屏幕，用于判断 X11 是否能正常工作会造成一些困难。可以通过 `retro` 使用旧的模式：

```
# Xorg -config xorg.conf.new -retro
```

如果看到灰色的格子以及 X 型鼠标指针，就表示配置成功了。要退出时，需要同按下 `Ctrl + Alt + Fn` 来切换到用于 X 的虚拟控制台（`F1` 表示第一个虚拟控制台）之后按 `Ctrl + C`。

在Xorg 7.3 以及更早期的版本中，使用 `Ctrl + Alt + Backspace` 组合来制退出 Xorg。如果需要在 7.4 和之后的版本中使用这个组合，可以在任意 X 终端模拟器中输入下面的命令：

```
% setxkbmap -option terminate:ctrl_alt_bksp
```

或者使用 `hald` 创建一个叫作 `x11-input.fdi` 的配置文件并保存至 `/usr/local/etc/hal/fdi/policy` 目录。这个文件需包含以下信息：



```
<?xml version="1.0" encoding="utf-8"?>
<deviceinfo version="0.2">
  <device>
    <match key="info.capabilities" contains="input.keyboard">
      <merge key="input.x11_options.XkbOptions"
type="string">terminate:ctrl_alt_bksp</merge>
    </match>
  </device>
</deviceinfo>
```

您可能需要重启您的机器来使得 `hald` 重新读取这个文件。

此外，需要在 `xorg.conf.new` 中的 `ServerLayout` 或 `ServerFlags` 小节中添加：

```
Option "DontZap" "off"
```

如果鼠标无法正常工作，在深入之前需要先配置它。参看 FreeBSD 安装一章中的 [配置鼠标](#)。另外，从 7.4 版本开始，`xorg.conf` 中的 `InputDevice` 部分将被忽略，这有助于自配置硬件。可以在这个文件中的 `ServerLayout` 或者 `ServerFlags` 加入以下使用旧的模式：

```
Option "AutoAddDevices" "false"
```

输入相同其他需要的信息（比如，布局切换）就可以像在之前的版本中的那样配置了。

正如前面所提到的，自版本 7.4 开始 hald 守护进程默认自启动的。可能输出不同的布局或型号有差异，在不同的环境中，比如 GNOME，KDE 或者 Xfce 提供了工具来配置它。一方面，也可在 [setxkbmap\(1\)](#) 工具的帮助下或者通过 hald 的配置文件来直接设置它的属性。

例如，如果某人想要使用一个 PC 102 键法布局的，我们就需要 hald 建立一个配置文件，叫作 x11-input.fdi 并保存入 /usr/local/etc/hal/fdi/policy 目录。这个文件需要包含如下一些：



```
<?xml version="1.0" encoding="utf-8"?>
<deviceinfo version="0.2">
  <device>
    <match key="info.capabilities" contains="input.keyboard">
      <merge key="input.x11_options.XkbModel"
type="string">pc102</merge>
      <merge key="input.x11_options.XkbLayout" type="string">fr</merge>
    </match>
  </device>
</deviceinfo>
```

如果这个文件已存在，只要把配置相关的部分拷入即可。

需要重启的机器使 hald 载入此文件。

也可以在 X 模式端或一个脚本中使用以下的命令得到相同的效果：

```
% setxkbmap -model pc102 -layout fr
```

/usr/local/shared/X11/xkb/rules/base.lst 列出了各种不同的，布局 and 可用的。

接下来是调整 xorg.conf.new 配置文件并作。用文本编辑器如 [emacs\(1\)](#) 或 [ee\(1\)](#) 打开这个文件。要做的第一件事是当前系统的显示器设置刷新率。这些包括垂直和水平的同步率。把它加入到 xorg.conf.new 的 "Monitor" 小节中：

```
Section "Monitor"
  Identifier      "Monitor0"
  VendorName      "Monitor Vendor"
  ModelName       "Monitor Model"
  HorizSync       30-107
  VertRefresh     48-120
EndSection
```

在配置文件中也有可能没有 HorizSync 和 VertRefresh。如果是的话，就只能手动添加，并在 HorizSync 和 VertRefresh 后面设置合适的数字了。在上面的例子中，给出了相关的显示器的参数。

X 能使用显示器所支持的 DPMS (能源之星) 功能。xset(1) 程序可以控制超睡眠，并制待机、挂起或关机。如果希望用显示器的 DPMS 功能，需要把下面的设置添加到 monitor 中：



|        |        |
|--------|--------|
| Option | "DPMS" |
|--------|--------|

在 `xorg.conf.new` 之前，默认分辨率和色深是在 `"Screen"` 小节中定义的：

```
Section "Screen"
    Identifier "Screen0"
    Device      "Card0"
    Monitor     "Monitor0"
    DefaultDepth 24
    SubSection "Display"
        Viewport 0 0
        Depth    24
        Modes    "1024x768"
    EndSubSection
EndSection
```

`DefaultDepth` 选项描述了要运行的默认色深。可以通过 [Xorg\(1\)](#) 的 `-depth` 命令行选项来替代配置文件中的设置。`Modes` 选项描述了给定色深度下屏幕的分辨率。需要说明的是，目前系列的图形硬件只支持由 VESA 定义的准模式。前面的例子中，默认色深是使用 24 位色。在采用这个色深，允许的分辨率是 1024x768。

最后就是将配置文件存盘，并使用前面介绍的启动模式启动一下。



在启动并解决问题的过程中，包含了与 X11 服务器相关的各个选项的信息的 X11 日志文件会帮助分析和排除故障。Xorg 日志的文件名是 `/var/log/Xorg.0.log` 的格式。实际的日志文件名可能是 `Xorg.0.log` 到 `Xorg.8.log` 等等。

如果一切准备妥当，就可以把配置文件放到公共的目录中了。可以在 [Xorg\(1\)](#) 里面找到具体位置。这个位置通常是 `/etc/X11/xorg.conf` 或 `/usr/local/etc/X11/xorg.conf`。

```
# cp xorg.conf.new /etc/X11/xorg.conf
```

现在已完成了 X11 的配置全过程。Xorg 可以通过 [startx\(1\)](#) 工具来启动。除此之外，X11 服务器也可以用 [xdm\(1\)](#) 来启动。

## 6.4.3. 高级配置主题

### 6.4.3.1. 配置 Intel® i810 显示芯片

配置 Intel i810 芯片的显示需要有一个 X11 的驱动用来驱动显示卡的 `agpgart` AGP 程序接口。请参考 [agp\(4\)](#) 程序的联机手册了解更多。

它也可用于其他的图形硬件配置。注意如果系统没有将 [agp\(4\)](#) 程序加载到内核，那么用 [kldload\(8\)](#) 加载模块是无效的。这个程序必须加载到内核或者使用 `/boot/loader.conf` 在启动时加入内核。

### 6.4.3.2. 添加屏平板显示器

—假定了解一些于高配置的知识。如果使用前面的准配置工具不能生可用的配置，在日志文件中提供的信息足以修正配置使其正常工作。如果需要的，使用一个文本器来完成工作。

目前的屏（WSXGA、WSXGA+、WUXGA、WXGA、WXGA+，等等）支持 16:10 和 10:9 或一些支持不大好的示比例。常的一些 16:10 比例的分辨率包括：

- 2560x1600
- 1920x1200
- 1680x1050
- 1440x900
- 1280x800

有，也可以地把些分辨率作 [Section "Screen"](#) 中的 [Mode](#) 来行配置，似下面：

```
Section "Screen"
Identifier "Screen0"
Device      "Card0"
Monitor     "Monitor0"
DefaultDepth 24
SubSection "Display"
    Viewport 0 0
    Depth    24
    Modes     "1680x1050"
EndSubSection
EndSection
```

Xorg 能自地通 I2C/DDC 信息来自取屏显示器的分辨率信息，并理显示器支持的率和分辨率。

如果程序没有的 [ModeLines](#)，就需要 Xorg 一些提示了。使用 `/var/log/Xorg.0.log` 能提取足的信息，就可以写一个可用的 [ModeLine](#) 了。信息如下所示：

```
(II) MGA(0): Supported additional Video Mode:
(II) MGA(0): clock: 146.2 MHz   Image Size:  433 x 271 mm
(II) MGA(0): h_active: 1680  h_sync: 1784  h_sync_end 1960 h_blank_end 2240 h_border:
0
(II) MGA(0): v_active: 1050  v_sync: 1053  v_sync_end 1059 v_blanking: 1089 v_border:
0
(II) MGA(0): Ranges: V min: 48  V max: 85 Hz, H min: 30  H max: 94 kHz, PixClock max
170 MHz
```

些信息称做 EDID 信息。从中建立 [ModeLine](#) 只是把些数据重新排列序而已：

```
ModeLine <name> <clock> <4 horiz. timings> <4 vert. timings>
```

如此，本例中的 Section "Monitor" 中的 Modeline 类似下面的形式：

```
Section "Monitor"
Identifier      "Monitor1"
VendorName     "Bigname"
ModelName      "BestModel"
Modeline       "1680x1050" 146.2 1680 1784 1960 2240 1050 1053 1059 1089
Option         "DPMS"
EndSection
```

配置之后，X 就可以在您的显示器上运行了。

## 6.5. 在 X11 中使用字体

### 6.5.1. Type1 字体

X11 使用的默认字体不是很理想。大型的字体显得参差不齐，看起来很不美，并且，在 [getenv\(3\)](#) 中，小字体简直无法看清。有好几类免版税、高质量的字体可以很方便地用在 X11 中。例如，URW 字体集合 ([x11-fonts/urwfonts](#)) 就包括了高质量的 标准 type1 字体 (Times Roman™, Helvetica™、Palatino™ 和其他一些)。在 Freefont 集合中 ([x11-fonts/freefonts](#)) 也包括更多的字体，但它中的大部分使用在图形软件中，如 Gimp，在屏幕字体中使用并不完美。另外，只要花很少的功夫，可以将 XFree86™ 配置成能使用 TrueType® 字体：参见后面的 [TrueType® 字体一章](#)。

如果希望使用 Ports Collection 来安装上面的 Type1 字体，只需运行下面的命令：

```
# cd /usr/ports/x11-fonts/urwfonts
# make install clean
```

freefont 或其他字体和上面所讲的大体类似。为了 X 服务器能识别到这些字体，需要在 X 服务器的配置文件 (/etc/X11/xorg.conf) 中添加下面的配置：

```
FontPath "/usr/local/lib/X11/fonts/URW/"
```

或者，也可以在命令行运行：

```
% xset fp+ /usr/local/lib/X11/fonts/URW
% xset fp rehash
```

这会起作用，但是当 X 会话结束后就会丢失，除非它被添加到配置文件 (~/.xinitrc 中，或者一个常用的 [startx](#) 会，或者当通过一个类似 XDM 的图形登录管理器登录添加到 ~/.xsession 中)。第三种方法是使用新的 /usr/local/etc/fonts/local.conf 文件：参见 [anti-aliasing](#) 章。

## 6.5.2. TrueType® 字体

Xorg 已内建了 TrueType® 字体的支持。有不同的模块能用不同的功能。在例子中使用 freetype 模块，因为它与其他的字体描后端是兼容的。要用 freetype 模块，只需要将下面行添加到 /etc/X11/xorg.conf 文件的 "Module" 部分。

```
Load "freetype"
```

在， TrueType® 字体建一个目录（比如， /usr/local/lib/X11/fonts/TrueType）然后把所有的 TrueType® 字体复制到该目录。你不能直接从 Macintosh® 计算机中提取 TrueType® 字体；能被 X11 使用的必须是 UNIX®/MS-DOS®/Windows® 格式的。一旦已将这些文件复制到了该目录，就可以用 ttmkfdir 来建 fonts.dir 文件，以便 X 字体引擎知道已安装了新文件。ttmkfdir 可以在 FreeBSD Ports 套件中的 [x11-fonts/ttmkfdir](#) 中找到。

```
# cd /usr/local/lib/X11/fonts/TrueType
# ttmkfdir -o fonts.dir
```

在把 TrueType® 字体目录添加到字体路径中。和上面 Type1 字体的是一回事，那就是，使用

```
% xset fp+ /usr/local/lib/X11/fonts/TrueType
% xset fp rehash
```

或者把 FontPath 行加到 xorg.conf 文件中。

就是。在 [getenv\(3\)](#)，Gimp，StarOffice™ 和其他所有的 X 应用程序 可以出安装的 TrueType® 字体。一些很小的字体（如在 Web 页面上高分辨率显示的文本）和一些很大的字体（在 StarOffice™ 下）在看起来已很好了。

## 6.5.3. Anti-Aliased 字体

于所有支持 Xft 的应用程序，所有放到 X11 /usr/local/lib/X11/fonts/ 和 ~/.fonts/ 中的字体都自动地被加入反锯齿支持。大多数新的程序都提供了 Xft 支持，包括 KDE、GNOME 以及 Firefox。

要控制哪些字体是 anti-aliased，或者配置 anti-aliased 特性，建（或者，如果文件已存在的）文件 /usr/local/etc/fonts/local.conf。Xft 字体系统的几个高级特性都可以使用一个文件来；一部分只描述几种最坏的情况。要了解更多的，看看 [fonts-conf\(5\)](#)。

个文件一定是 XML 格式的。注意确保所有的都完全的掉。个文件以一个很普通的 XML 开始，后跟一个 DOCTYPE 定义，接下来是 <fontconfig> ：

```
<?xml version="1.0"?>
<!DOCTYPE fontconfig SYSTEM "fonts.dtd">
<fontconfig>
```

像前面所做的那样，在 /usr/local/lib/X11/fonts/ 和 ~/.fonts/ 目录下的所有字体都可以被支持 Xft 的

应用程序使用了。如果你想添加一个目以外的其他路径，  
/usr/local/etc/fonts/local.conf文件中：

的添加下面行到

```
<dir>/path/to/my/fonts</dir>
```

添加了新的字体，尤其是添加了新的字体目后， 行下面的命令重建字体缓存：

```
# fc-cache -f
```

Anti-aliasing 会字体有些模糊，加了非常小的文本的可性， 并从大文本字体中除 " "。但如果使用普通的文本， 可能引起眼疲劳。要禁止 14磅 以下字体的反走， 需要加如下配置：

```
<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>
```

用 anti-aliasing 来隔一些等字体也是不当的。似乎是 KDE 的一个。要修这个需要保个字体之的距保持在100。加入下面些行：

```

<match target="pattern" name="family">
  <test qual="any" name="family">
    <string>fixed</string>
  </test>
  <edit name="family" mode="assign">
    <string>mono</string>
  </edit>
</match>
<match target="pattern" name="family">
  <test qual="any" name="family">
    <string>console</string>
  </test>
  <edit name="family" mode="assign">
    <string>mono</string>
  </edit>
</match>

```

(这里把其他普通的修切的字体作 "mono"), 然后加入：

```

<match target="pattern" name="family">
  <test qual="any" name="family">
    <string>mono</string>
  </test>
  <edit name="spacing" mode="assign">
    <int>100</int>
  </edit>
</match>

```

某些字体，比如 Helvetica，当 anti-aliased 的时候可能存在。通常的表字体本身似乎被垂直的切成一半。糟糕的时候，可能致用程序崩溃。为了避免的现象，考虑添加下面几行到 local.conf 文件里面：

```

<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>

```

一旦完成 local.conf 文件的，确保使用了 `</fontconfig>` 来束文件。不这样做将会致的更改被忽略。

最后，用可以通他个人的 .fonts.conf 文件来添加自己的定。要完成此工作，用只需地建 ~/.fonts.conf 并添加相配置。此文件也必须是 XML 格式的。

最后：于LCD屏幕，可能希望使用子像素的取。而言，是通分控制（水平方向分的） 、 、

像素，来改善水平分辨率；做的效果一般会非常明。要用它，只需在 local.conf 文件的某个地方加入：

```
<match target="font">
  <test qual="all" name="rgba">
    <const>unknown</const>
  </test>
  <edit name="rgba" mode="assign">
    <const>rgb</const>
  </edit>
</match>
```



随显示器的不同，可能需要把 `rgb` 改 `bgr`、`vrgb` 或 `vbgr`：看一下看看那个更好。

## 6.6. X 显示管理器

### 6.6.1. 概要

X 显示管理器(XDM) 是一个X系统用于运行会话管理的可。它可以用于多种情况下，包括小 "X Terminals"，面，大网显示服务器。既然 X 系统不受网和的限制，那于通网接起来的行 X 客户端和服务器的不同机器，就会有多的可配置。XDM 提供了一个要接到个显示服务器的形接口，只要入如登录用名和密码的息。

也可以把 XDM 想象成与 `getty(8)` 工具一(see [配置](#) for details)。用提供了同功能。它可以完成系的登录任，然后用行一个会话管理器(通常是一个 X 管理器)。接下来 XDM 就等待个程序退出，出信号用已登录完成，当退出屏幕。当，XDM 就可以下一个登录用显示和可屏幕。

### 6.6.2. 使用 XDM

如果希望使用 XDM 来，首先需要安装 `x11/xdm` port (在新版本的 Xorg 中它并不是默安装的)。XDM 服程序位于 `/usr/local/bin/xdm`。任何时候都可以 `root` 用的身来行它，以令其管理本地系的 X 示。如果希望 XDM 在系次程中自行，比方便的做法是把它写到 `/etc/ttys` 的配置中。有个文件的具体格式和使用方法参 [添加一个到/etc/ttys](#)。在默的 `/etc/ttys` 文件中已包含了在虚端上行 XDM 服的示配置：

```
tttyv8  "/usr/local/bin/xdm -nodaemon" xterm  off secure
```

默情况下，个是，要用它，需要把第5部分的 `off` 改 `on` 然后按照 [重新取/etc/ttys来制init](#) 的指 重新 `init(8)`。第一部分，个程序将管理的端名称是 `tttyv8`。意味着 XDM 将行在第9个虚端上。

### 6.6.3. 配置 XDM

XDM 的配置目是在 `/usr/local/lib/X11/xdm`中。在个目中，会看到几个用来改 XDM 行和外文件。会到些文件：

| 文件         | 描述                  |
|------------|---------------------|
| Xaccess    | 客户端授权。              |
| Xresources | 默认的X源。              |
| Xservers   | 进程和本地显示管理列表。        |
| Xsession   | 用于登录的默认会话脚本。        |
| Xsetup_*   | 登录之前用于加载应用程序的脚本。    |
| xdm-config | 运行在每台机器上的所有显示的全局配置。 |
| xdm-errors | 服务器程序产生的。           |
| xdm-pid    | 当前运行的 XDM 的进程 ID。   |

当 XDM 运行，在个目中有几个脚本和程序可以用来置面。 些文件中的个的用法都将被要地描述。 些文件的更的法和用法在 [xdm\(1\)](#) 中将有描述。

默认的配置是一个矩形的登录口，上面有机器的名称， "Login:" 和 "Password:"。如果想自己个性化的 XDM 屏幕， 是一个很好的起点。

### 6.6.3.1. Xaccess

用以接由 XDM 所控制的显示， 叫做 X 示管理器接口 (XDMCP)。 个文件是一用以控制来自程算机的 XDMCP 接的。 除非修改 xdm-config 使其接受程接， 否其内容将被忽略。 默情况下， 它不允来自任何客户端的接。

### 6.6.3.2. Xresources

是一个默的用来示和登录屏幕的程文件。 可以在个文件中登程序的外行定制。 其格式与 X11 文中描述的默程文件是一的。

### 6.6.3.3. Xservers

是一个者当提供的作可的程示列表。

### 6.6.3.4. Xsession

是一个用登录后 XDM 的默会话脚本。通常，在 ~/.xsession 中个用将有一个可定制的会话脚本。

### 6.6.3.5. Xsetup\_\*

在示者或登接口之前， 些将被自行。 是一个个示都要用到的脚本，叫做 Xsetup\_，后面会跟一个本地示的数字(比如 Xsetup\_0)。典型的， 些脚本将在后台 (如 [xconsole](#))行一个或个程序。

### 6.6.3.6. xdm-config

此文件以程默的形式， 提供了在安装所使用的普的示置。

### 6.6.3.7. xdm-errors

个文件包含了 XDM 正法行的的 X 服务器 的出。 如果 XDM 正法行的示由于某原因被挂起， 那



它是一个存储信息的好地方。有些信息会在一个会话的基础上被写到用它的 `~/.xsession-errors` 文件中。

#### 6.6.4. 运行一个网络显示服务器

不同于其他客户端，如果希望它能连接到显示服务器，就必须通过控制它，并让它接受连接。默认情况下，它都比它保守的。要让 XDM 能监听连接，首先要在 `xdm-config` 文件中注释掉一行：

```
! SECURITY: do not listen for XDMCP or Chooser requests ! Comment out this line if you
want to manage X terminals with xdm
DisplayManager.requestPort:      0
```

然后重新启动 XDM。记住默认程序文件的注释以 "!" 字母开始，不是 "#"。需要设置严格的控制 - 看看在 `Xaccess` 文件中的示例，并参考 [xdm\(1\)](#) 的联机手册，以了解一切的细节。

#### 6.6.5. 替换 XDM

有几个替换默认 XDM 程序的方案。其中之一是上一节已描述的 `kdm` (与 KDE 放在一起)。`kdm` 提供了更多的定制和局部的修改，同时也能用在可能令他喜欢的窗口管理器。

### 6.7. 桌面环境

本节描述了 FreeBSD 上用于 X 的不同桌面环境。"桌面环境" 可能是一个窗口管理器，也可能是一个像 KDE 或者 GNOME 的完整桌面应用程序套件。

#### 6.7.1. GNOME

##### 6.7.1.1. 有 GNOME

GNOME 是一个用户界面友好的桌面环境，能使用户很容易地使用和配置他的计算机。GNOME 包括一个面板 (用来运行程序和显示状态)，一个桌面 (存放数据和应用程序的地方)，一套标准的桌面工具和应用程序，和一套与其他人相互协同工作的集合。其他操作系统的用户在使用 GNOME 提供的强大的图形桌面环境时会得很好。更多的关于 FreeBSD 上 GNOME 的信息可以在 [FreeBSD GNOME Project](#) 的网站上得到。此外，该网站也提供了相当详尽的关于安装、配置和管理 GNOME 的常见问题解答 (FAQ)。

##### 6.7.1.2. 安装 GNOME

该软件可以很容易地通过软件包或 Ports 套件来安装：

要从网络安装 GNOME，只要输入：

```
# pkg_add -r gnome2
```

从源代码安装 GNOME，可以使用 ports：

```
# cd /usr/ports/x11/gnome2
# make install clean
```

GNOME 需要挂 `/proc` 文件系统才能正常工作。添加如下

```
proc          /proc          procfs  rw  0  0
```

到 `/etc/fstab` 以便在系统启动时挂上 `procfs(5)`。

一旦装好了 GNOME，就必须告诉 X server 用 GNOME 而不是默认的窗口管理器。

最简单的用 GNOME 的方法是使用 GDM，GNOME 显示管理器。随 GNOME 界面一同安装的 GDM 尽管默认是禁用的。可以在 `/etc/rc.conf` 中加入以下行启用：

```
gdm_enable="YES"
```

在重启机器的时候，GDM 将自行运行。

通常我希望在 GDM 启动，同时启用所有的 GNOME 服务，可以将如下行加入 `/etc/rc.conf`：

```
gnome_enable="YES"
```

GNOME 也可以通过本地配置名 `.xinitrc` 的文件来启动。如果已经有了自定义的 `.xinitrc`，将当前窗口管理器的那一行改写成 `/usr/local/bin/gnome-session` 就可以了。如果没有，那只需简单地：

```
% echo "/usr/local/bin/gnome-session" > ~/.xinitrc
```

接下来输入 `startx`，GNOME 界面环境就启动了。



如果之前使用了一些旧式的显示管理器，例如 XDM，这样做是没用的。此时建立一个可行的 `.xsession` 文件，其中包含同样的命令。要完成这项工作，需要用 `/usr/local/bin/gnome-session` 取代原有的窗口管理器：

```
% echo "#!/bin/sh" > ~/.xsession
% echo "/usr/local/bin/gnome-session" >> ~/.xsession
% chmod +x ~/.xsession
```

有一种做法，是配置显示管理器，以便在登录时提示用户选择窗口管理器；在 KDE 界面中介入了用于如何启动 kdm（KDE 的显示管理器）运行的配置。

## 6.7.2. KDE

### 6.7.2.1. 有 KDE

KDE 是一个容易使用的现代桌面环境。KDE 有很多很好的特性：

- 一个美丽的现代的面。

- 一个集合了完美网络环境的界面。
- 一个集成的帮助系统，能更方便、高效地帮助用户使用 KDE 界面和它的应用程序。
- 所有的 KDE 应用程序具有一致的所谓即所得界面。
- 标准的菜单和工具栏，窗口布局，颜色配置等。
- 国际化：KDE 可以使用超过 40 种语言。
- 集中化、统一的窗口框窗口的界面配置
- 许多有用的 KDE 应用程序。

KDE 附了一个名为 Konqueror 的 web 浏览器，它是其他流行于 UNIX® 系上的 web 浏览器的一个强大的竞争对手。要了解关于 KDE 的更多详情，可以看看 [KDE 网站](#)。与 FreeBSD 相关的 KDE 信息和资源，可以在 [FreeBSD 上的 KDE](#) 的网站找到。

FreeBSD 上提供了两个版本的 KDE。版本 3 已经推出了很稳定，十分成熟。而版本 4，也就是下一代版本，也可以通过 Ports 套件来安装。两个版本甚至能共存。

#### 6.7.2.2. 安装 KDE

与 GNOME 和其他桌面环境类似，这个软件可以很容易地通过软件包或 Ports 套件来安装：

要从网络安装 KDE3 只需要：

```
# pkg_add -r kde
```

要从网络安装 KDE4 需要：

```
# pkg_add -r kde4
```

[pkg\\_add\(1\)](#) 就会自动的下取最新版本的程序。

要从源代码安装 KDE3，可以使用 ports ：

```
# cd /usr/ports/x11/kde3
# make install clean
```

而从 ports 提供的源代码安装 KDE4， 同样的操作是：

```
# cd /usr/ports/x11/kde4
# make install clean
```

安装好 KDE 之后， 需要告诉 X server 哪个程序来代替默认的窗口管理器。 可以通过编辑 .xinitrc 文件来完成：

对于 KDE3：

```
% echo "exec startkde" > ~/.xinitrc
```

对于 KDE4 :

```
% echo "exec /usr/local/kde4/bin/startkde" > ~/.xinitrc
```

在, 无论什么时候用 `startx` 进入 X 系统, KDE 就将成为你的桌面环境。

如果使用一个像 XDM 的显示管理器, 那配置文件可能有点不同。需要一个 `.xsession` 文件, 有 `kdm` 的用法会在本章的后面介绍。

### 6.7.3. 有 KDE 的更多

在 KDE 已被安装在系统中了。通过帮助面或点多个菜单可以知道很多东西。Windows® 或 Mac® 用户会有回到家的感觉。

有 KDE 的最好的参考资料是它的在线文档。KDE 有它自己的 web 浏览器 Konqueror, 也有很多其他的程序和富文本。文档的余下部分将提供一些很实用走弯路的方法来学习的技术。

#### 6.7.3.1. KDE 显示管理器

如果在同一系统上有多用户, 管理通常会希望使用图形化的登录界面。前面已提到, 使用 `XDM` 可以完成工作。不过, KDE 本身也提供了一个, 即 `kdm`, 它的外观更富吸引力, 而且提供了更多的登录。值得一提的是, 用户能通过菜单很容易地希望使用的桌面环境 (KDE、GNOME 或其它)。

要用 `kdm`, 需要根据 KDE 的版本修改不同的配置文件。

对于 KDE3, `/etc/ttys` 中的 `ttv8` 需被改写成如下的形式:

```
ttv8 "/usr/local/bin/kdm -nodaemon" xterm on secure
```

对于 KDE4, 需要将如下行加入 `/etc/rc.conf`:

```
local_startup="${local_startup} /usr/local/kde4/etc/rc.d"
kdm4_enable="YES"
```

### 6.7.4. Xfce

#### 6.7.4.1. 有 Xfce

Xfce 是以被 GNOME 使用的 GTK+ 工具包为基础的桌面环境, 但是更加小巧, 适合于那些需要一个易于使用和配置并且更而高效的桌面的人。看起来, 它非常像使用在商业 UNIX 系统上的 CDE 环境。Xfce 的主要特性有下面一些:

- 一个, 易于使用的桌面。

- 完全通过鼠标的点击和按键来控制等。
- 与CDE 相似的主面板，菜单，applets和桌面launchers。
- 集成的窗口管理器，文件管理器，声音管理器，GNOME 桌面模式等等。
- 可配置界面的主题。(因为它使用GTK+)
- 快速，轻便，高效：对于比古老的/旧的机器或内存很少的机器仍然很理想。

更多有关Xfce 的信息可以参考[Xfce 网站](#)。

#### 6.7.4.2. 安装Xfce

有一个二进制的Xfce 软件包存在(在写作的時候)。要安装的话，运行下面的命令：

```
# pkg_add -r xfce4
```

另外，也可以使用 Ports Collection 从源代码编译：

```
# cd /usr/ports/x11-wm/xfce4
# make install clean
```

现在，要告诉X服务器在下次X启动时运行 Xfce。只要运行下面的命令：

```
% echo "/usr/local/bin/startxfce4" > ~/.xinitrc
```

接下来就是启动 X，Xfce将成为你的桌面。与以前一样，如果使用像 XDM 这样的显示管理器，需要创建一个.xsession文件，就像有[GNOME](#) 的那部分描述的，使用/usr/local/bin/startxfce4 命令，或者，配置显示管理器允许在启动时启动一个桌面，就像有[kdm](#)的那部分描述的。

## Part II: 常⌘的任⌘

前面已⌘介⌘了必要的基⌘知⌘，手册的⌘一部分将⌘⌘ FreeBSD 的一些最常用的功能。⌘些章⌘包括：

- 向⌘介⌘流行和⌘用的⌘面⌘用程序：⌘⌘器、⌘品工具、文⌘察看程序，等等。
- 向⌘介⌘一系列可以在 FreeBSD 上使用的多媒体工具。
- 介⌘⌘⌘定制的 FreeBSD 内核以⌘用附加功能的方法。
- ⌘⌘介⌘包括⌘面和网⌘打印机在内的打印系⌘⌘置。
- 向⌘展示如何在 FreeBSD 上⌘行 Linux ⌘用程序。

某些章⌘希望⌘首先⌘⌘⌘其他部分，在⌘些章的⌘⌘部分也会⌘出⌘⌘似的提示。

# Chapter 7. 桌面应用

## 7.1. 概述

FreeBSD 可以运行非常多的桌面应用程序，包括像浏览器和字处理器的软件。大多数这样的程序都可以通过 package 来安装，或者从 Ports Collection 自在地构建。许多新用户希望能在它的系统中得到所有的应用程序。这一章将向您展示如何轻松地使用 package 或者 Ports Collection 中安装所有的软件。

需要注意的是从 ports 安装意味着要源码。根据软件的大小和编译速度的不同，您可能需要花相当多的时间。若是觉得源码太耗时的话，大多数 ports 也有编译好的版本可供安装。

因为 FreeBSD 提供的二进制兼容 Linux 的特性，许多原本在 Linux 上的程序都可以直接用在 FreeBSD 的桌面。在安装任何的 Linux 应用程序之前，强烈的推荐您阅读 [Linux® 二进制兼容模式](#)。当您在特定的 ports 中，可以使用 `whereis(1)`。一般来，许多利用 Linux 二进制兼容特性的 ports 都以"linux-"开头。在下面的介绍中，都假设安装 Linux 应用程序前已启用了 Linux 二进制兼容功能。

本章涵盖以下应用程序：

- 浏览器 (例如 Firefox、Opera、Konqueror)
- 办公、图像处理 (例如 KOffice、AbiWord、GIMP、OpenOffice.org、LibreOffice)
- 文档查看 (例如 Acrobat Reader®、gv、Xpdf、GQview)
- 财务 (例如 GnuCash、Gnumeric、Abacus)

本章之前，您：

- 知道如何安装外部的第三方软件([安装应用程序. Packages 和 Ports](#))。
- 知道如何安装 Linux 软件([Linux® 二进制兼容模式](#))。

想要得到更多的有关多媒体环境的信息，请参 [多媒体](#)。如果您想要建立和使用自己的软件，请参考[自己的软件](#)。

## 7.2. 浏览器

FreeBSD并没有预先安装特定的浏览器。然而，在 ports 的目录 `www` 有许多浏览器可以安装。如果您没有一一测试它们 (有些时候您可能需要花相当多的时间) 大部分都有 package 可用。

KDE 和 GNOME 已提供 HTML 浏览器。请参考[桌面环境](#)得到更多完整的有关特定桌面环境的信息。

如果您要小型的浏览器，可以看看 [www/dillo2](#)、[www/links](#) 或 [www/w3m](#)。

以下是一些如下程序：

| 程序名称    | 源码需求 | 安装时间 | 主要依赖 |
|---------|------|------|------|
| Firefox | 中等   | 短    | Gtk+ |

| 程序名称      | 源需求 | 安装 | 主要依   |
|-----------|-----|----|---|
| Opera     | 少   | 松  | 同有可用的 FreeBSD 和 Linux 版本。Linux 版本需要使用 Linux 二进制兼容模式和 linux-openmotif。 |
| Firefox   | 中等  |    | Gtk+  |
| Konqueror | 中等  |    | 需要 KDE  |

### 7.2.1. Firefox

Firefox 是一个代，自由，放源代定的器，并完全移植到了 FreeBSD 上：它的特性包括有一个非常准的 HTML 指引，式，出口阻止，展件，改的安全性，等等。Firefox 是基于 Mozilla 的代。

可以通入下面的命令来安装包的：

```
# pkg_add -r firefox
```

将会安装 Firefox 7.0，如果希望行 Firefox 3.6，使用下面的命令：

```
# pkg_add -r firefox36
```

如果希望从源代的，可以通 Ports Collection 安装：

```
# cd /usr/ports/www/firefox
# make install clean
```

于 Firefox 3.6，的命令中的 **firefox** 改 **firefox36**。

### 7.2.2. Firefox 与 Java™ 件



在—和接下来的中，我均假定已安装了 Firefox。

通 Ports 套件来安装 OpenJDK 6，入下面的命令：

```
# cd /usr/ports/java/openjdk6
# make install clean
```

接下来安装 [java/icedtea-web](#) port：



```
# cd /usr/ports/java/icedtea-web
# make install clean
```

在 上述 port 使用的是系统的配置。

器并在地址中入 `about:plugins` 然后按 `Enter`。器将会呈现一个列出所有已安装件的页面；Java™ 件在其中出。

如果器不到件，用可能必行下面的命令，并重器：

```
% ln -s /usr/local/lib/IcedTeaPlugin.so \
$HOME/.mozilla/plugins/
```

### 7.2.3. Firefox 与 Adobe® Flash™ 件

Adobe® Flash™ 件并没有直接提供其 FreeBSD 版本。不，我有一个件（wrapper）可以用来行 Linux 版本的件。个 wrapper 也支持 Adobe® Acrobat®、RealPlayer 和很多其他件。

根据 FreeBSD 版本的不同相的安装：

#### 1. FreeBSD 7.X

安装 [www/nspluginwrapper](#) port。个 port 需要安装一个大的 [emulators/linux\\_base-fc4](#) port。

下一是安装 [www/linux-flashplugin9](#) port。将会安装 Flash™ 9.X，此版本目前能在 FreeBSD 7.X 上正常行。



在比 FreeBSD 7.1-RELEASE 更旧版本的系上，必安装 [www/linux-flashplugin7](#) 并跳以下 [linprocfs\(5\)](#) 的部。

#### 2. FreeBSD 8.X

安装 [www/nspluginwrapper](#) port。个 port 需要安装一个大的 [emulators/linux\\_base-f10](#) port。

下一是安装 [www/linux-f10-flashplugin10](#) port。将会安装 Flash™ 10.X，此版本目前能在 FreeBSD 8.X 上正常行。

个版本需要建一个符号接：

```
# ln -s /usr/local/lib/npapi/linux-f10-flashplugin/libflashplayer.so \
/usr/local/lib/browser_plugins/
```

如果系中没有 `/usr/local/lib/browser_plugins` 目，手工建它。

按照 FreeBSD 版本，在安装了正真的 Flash™ port 之后，组件必须由一个用 `nspluginwrapper` 安装：

```
% nspluginwrapper -v -a -i
```

如果希望播放 Flash™ 动画的，Linux® 的进程文件系统，[linprocfs\(5\)](#) 必须挂于 `/usr/compat/linux/proc`。可以通过以下的命令：

```
# mount -t linprocfs linproc /usr/compat/linux/proc
```

也可以在机器上自己挂，把以下行加入 `/etc/fstab`：

```
linproc /usr/compat/linux/proc linprocfs rw 0 0
```

然后就可以打开浏览器，并在地址栏中输入 `about:plugins` 然后按下 `Enter`。它将显示目前可用的组件列表。

## 7.2.4. Firefox and Swfdec Flash™ Plugin

Swfdec 是一个用以解码和渲染 Flash™ 动画的。Swfdec-Mozilla 是一个使用了 Swfdec 的 Firefox 能播放 SWF 文件的组件。它目前仍处于测试状态。

如果不能或者不想安装，可以通过网安装二进制包：

```
# pkg_add -r swfdec-plugin
```

如果二进制包不可用，可以通过 Ports Collection 安装：

```
# cd /usr/ports/www/swfdec-plugin
# make install clean
```

然后重启的浏览器使得这个组件生效。

## 7.2.5. Opera

Opera 是一个功能齐全，并符合标准的浏览器。它提供了内建的组件和新浏览器、IRC 客户端，RSS/Atom feed 阅读器以及更多功能。除此之外，Opera 是一个轻量级的浏览器，其速度很快。它提供了不同的版本：“native” FreeBSD 版本，以及通过 Linux 模拟运行的版本。

要使用 Opera 的 FreeBSD 版本来上网，安装以下的 package：

```
# pkg_add -r opera
```

有些 FTP 站点没有所有版本的 package，但仍然可以通过 Ports 套件来安装 Opera：

```
# cd /usr/ports/www/opera
# make install clean
```

要安装 Linux 版本的 Opera，将上面例子中的 `opera` 改为 `linux-opera` 即可。

Adobe® Flash™ 插件目前并没有提供 FreeBSD 可用的版本。不过，可以使用其 Linux® 版本的插件。要安装这个版本，需要安装 [www/linux-f10-flashplugin10](#) port，以及 [www/opera-linuxplugins](#)：

```
# cd /usr/ports/www/linux-f10-flashplugin10
# make install clean
# cd /usr/ports/www/opera-linuxplugins
# make install clean
```

然后可以看看插件是否可用了：在地址栏中输入 `opera:plugins` 然后按 `Enter`。浏览器将列出可用的插件列表。

添加 Java™ 插件的方法，与 [Firefox 添加插件](#) 的方法相同。

## 7.2.6. Konqueror

Konqueror 是 KDE 的一部分，不过也可以通安装 [x11/kdebase3](#) 在非 KDE 环境下使用。Konqueror 不止是一个浏览器，也是一个文件管理器和多媒体播放器。

也有丰富的插件能配合 Konqueror 一起使用，可以通过 [misc/konq-plugins](#) 来安装它们。

Konqueror 也支持 Flash™；关于如何获得用于 Konqueror 的 Flash™ 支持的 "How To" 文档可以在 <http://freebsd.kde.org/howtos/konqueror-flash.php> 找到。

## 7.3. 办公、图像处理

当需要运行办公或者图像处理，新用户通常都会有一些好用的办公套件或者图像处理件。尽管目前有一些 [桌面环境](#)，如 KDE 已提供了办公套件，但目前尚没有一定之规。无论使用那桌面环境，FreeBSD 都能提供需要的件。

涉及如下程序：

| 件名称            | 源需求 | 安装量 | 主要依赖                     |
|----------------|-----|-----|--------------------------|
| KOffice        | 少   | 多   | KDE                      |
| AbiWord        | 少   | 少   | Gtk+ 或 GNOME             |
| The Gimp       | 少   | 中   | Gtk+                     |
| OpenOffice.org | 多   | 中   | JDK™、Mozilla             |
| LibreOffice    | 很重  | 巨大  | Gtk+ 或 KDE/ GNOME 或 JDK™ |

### 7.3.1. KOffice

KDE 社区提供了一套办公套件，它能用在桌面环境。它包含四个标准的组件，有些组件可以在其它办公套件中找到。KWord 是字处理程序、KSpread 是电子表格程序、KPresenter 是演示文稿制作管理程序、Kontour 是矢量图形件。

安装最新的 KOffice 之前，先确定是否安装了最新版的 KDE。

使用 package 来安装 KOffice，安装如下：

```
# pkg_add -r koffice
```

如果没有可用的 package，可以使用 Ports Collection 安装。安装 KDE3 的 KOffice 版本，如下：

```
# cd /usr/ports/editors/koffice-kde3
# make install clean
```

### 7.3.2. AbiWord

AbiWord 是一个免费的字处理程序，它看起来和 Microsoft® Word 的感觉很相似。它组合用来打印文件、信函、广告、备忘录等等，它非常快且包含多特性，并且非常容易使用。

AbiWord 可以输入或输出很多文件格式，包括一些象 Microsoft® .doc 等有格式的文件。

AbiWord 也有 package 的安装方式。可以用以下方法安装：

```
# pkg_add -r abiword
```

如果没有可用的 package，它也可以从 Ports Collection 中。ports collection 中是最新的。它的安装方式如下：

```
# cd /usr/ports/editors/abiword
# make install clean
```

### 7.3.3. GIMP

图像的编辑或者加工，GIMP 是一个非常精通图像处理的组件。它可以被用来当作图像的编辑程序或者一个编辑的照片处理套件。它支持大量的组件和具有脚本界面的特性。GIMP 可以编写多的文件格式，支持描图和手写板。

可以用下列命令安装：

```
# pkg_add -r gimp
```

如果在 FTP 站点没有找到这个 package，也可以使用 Ports Collection 的方法安装。ports 的 [graphics](#) 目录也包含有 Gimp 手册。以下是安装它的方法：

```
# cd /usr/ports/graphics/gimp
# make install clean
# cd /usr/ports/graphics/gimp-manual-pdf
# make install clean
```



Ports 中的 [graphics](#) 目录也有其中的 GIMP 版本 [graphics/gimp-devel](#)。HTML 版本的 Gimp 手册 可以在 [graphics/gimp-manual-html](#) 得到。

### 7.3.4. OpenOffice.org

OpenOffice.org 包括一套完整的办公套件：文字处理程序、电子表格程序、演示文稿管理程序和数据库程序。它和其它的办公套件的特征非常相似，它可以输入输出不同的流行的文件格式。它支持多语言 - 国际化已渗透到了其界面、编写和字典等各个方面。

OpenOffice.org 的文字处理程序使用 XML 文件格式使它增加了可移植性和灵活性。电子表格程序支持宏语言和使用外来的数据界面。OpenOffice.org 已经可以平滑的运行在 Windows®、Solaris™、Linux、FreeBSD 和 Mac OS® X 等各操作系统下。更多的有关 OpenOffice.org 的信息可以在 [OpenOffice.org 网](#) 得到。对于特定的 FreeBSD 版本的信息，可以在直接在 [FreeBSD OpenOffice 移植](#) 的页面下。

安装 OpenOffice.org 方法如下：

```
# pkg_add -r openoffice.org
```



如果你正在使用 FreeBSD 的 -RELEASE 版本，一般来都是没问题的。如果不是，你就可能需要看一看 FreeBSD OpenOffice.org 移植小组的网站，并使用 [pkg\\_add\(1\)](#) 从那里下载并安装合适的软件包。最新的发布版本和测试版本都可以在那里得到。

装好 package 之后，你只需输入下面的命令就能运行 OpenOffice.org 了：

```
% openoffice.org
```



在第一次运行，将一些文件，并在你的主目录中建立一个 .openoffice.org 目录。

如果没有可用的 OpenOffice.org package，你仍旧可以编译 port。然而，你必须记住它的要求以及大量的磁盘空间和相当多的时间。

```
# cd /usr/ports/editors/openoffice.org-3
# make install clean
```

如果希望安装一套可运行本地化的版本，将前述命令行改：



```
# make LOCALIZED_LANG=your_language install clean
```

需要将 `your_language` 改成正确的 ISO-代码。所支持的代码可以在 `files/Makefile.localized` 文件中找到，每个文件位于 `port` 的目录。

一旦完成上述操作，就可以通过下面的命令来运行 OpenOffice.org 了：

```
% openoffice.org
```

### 7.3.5. LibreOffice

LibreOffice 是由 [The Document Foundation](#) 开发的自由软件办公套件，它与其他平台上的主流办公系统兼容。它是 OpenOffice.org 的一个分支版本，包含了完整办公效率套件中必需的应用：文字处理、电子表格、幻灯演示、绘图工具、数据库管理程序，以及用于创建和编辑数学公式的程序。它提供了多种不同语言的支持 - 国际化支持除了界面之外，还包括了编写器和字典。

LibreOffice 的文字处理程序使用了内建的 XML 文件格式，以期获得更好的可移植性和活性。电子表格程序提供了一组可以与外部数据交互的宏语言支持。LibreOffice 目前已可以稳定运行于 Windows®、Linux、FreeBSD 和 Mac OS® X。关于 LibreOffice 的更多信息可以在 [LibreOffice 网站](#) 找到。

如果希望通过二进制的安装包安装 LibreOffice，运行：

```
# pkg_add -r libreoffice
```



如果运行的是 FreeBSD 的 -RELEASE 版本，这个命令不会遇到任何问题。

装好软件包之后，需要用下面的命令来安装 LibreOffice：

```
% libreoffice
```



在首次运行时，系统会安装一系列文件，并在当前用户的主目录中创建 `.libreoffice` 目录。

如果 LibreOffice 软件包不可用，还是可以通过 `port` 安装。不过，请注意它需要相当多的磁盘空间和内存。

```
# cd /usr/ports/editors/libreoffice
# make install clean
```

如果希望本地化的版本，把前面的命令改成：



```
# make LOCALIZED_LANG=your_language install clean
```

需要把 *your\_language* 改成正确的语言 ISO 代码。可用的代码可以在 port 的 Makefile 中的 *pre-fetch* target 中找到。

完成编译和安装之后，就可以用下面的命令运行 LibreOffice 了：

```
% libreoffice
```

## 7.4. 文档查看器

UNIX® 系出现以来，一些新的文档格式开始流行起来；它们所需要的标准查看器可能不一定在系内。本文中，我将了解如何安装它们。

它们涵如下用程序：

| 组件名称            | 源需求 | 安装 | 主要依赖         |
|-----------------|-----|----|--------------|
| Acrobat Reader® | 少   | 少  | Linux 二进制兼容  |
| gv              | 少   | 少  | Xaw3d        |
| Xpdf            | 少   | 少  | FreeType     |
| GQview          | 少   | 少  | Gtk+ 或 GNOME |

### 7.4.1. Acrobat Reader®

在许多文档都用 PDF 格式，根据“便携式文档格式”的定义。一个被建使用的查看器是 Acrobat Reader®，由 Adobe 所发行的 Linux 版本。因 FreeBSD 能运行 Linux 二进制文件，所以它也可以用在 FreeBSD 中。

要从 Ports collection 安装 Acrobat Reader® 8，只需：

```
# cd /usr/ports/print/acroread8
# make install clean
```

由于授权的限制，我不提供自己的版本。

### 7.4.2. gv

gv 是 PostScript® 和 PDF 文件格式查看器。它源自 ghostview 因使用 Xaw3d 函数使它看起来更美。它很快而且界面很干。gv 有很多特性比如象大小、刻度或者抗。大部分操作都可以只用或鼠标完成。

安装 gv package，如下：

```
# pkg_add -r gv
```

如果无法取包的，可以使用 Ports Collection：

```
# cd /usr/ports/print/gv
# make install clean
```

### 7.4.3. Xpdf

如果想要一个小型的 FreeBSD PDF 查看器，Xpdf 是一个小巧并且高效的查看器。它只需要很少的源而且非常稳定。它使用标准的 X 字体并且不需要 Motif® 或者其它的 X 工具包。

安装 Xpdf package，使用如下命令：

```
# pkg_add -r xpdf
```

如果 package 不可用或者使用 Ports Collection，如下：

```
# cd /usr/ports/graphics/xpdf
# make install clean
```

一旦安装完成，就可以使用 Xpdf 并且使用鼠标右键来使用菜单。

### 7.4.4. GQview

GQview 是一个图片管理器。它可以鼠标来看一个文件、一个外部设备、使用更多和更多的功能。它也有幻灯片播放模式和一些基本的文件操作。它可以管理采集的图片并且很容易到重的。GQview 可以全屏幕看并且支持国际化。

如果想要安装 GQview package，如下：

```
# pkg_add -r gqview
```

如果没有可用的 package 或者使用 Ports Collection，如下：

```
# cd /usr/ports/graphics/gqview
# make install clean
```

## 7.5. 其它

假如，基于任何的理由，想要在 FreeBSD Desktop 管理个人的文档，有一些大并且易于使用的软件可以被安装。它其中的一些与流行的文件格式兼容象 Quicken 和 Excel 文件。



本涵如下程序：

| 件名称      | 源需求 | 安装 | 主要依    |
|----------|-----|----|--------|
| GnuCash  | 少   |    | GNOME  |
| Gnumeric | 少   |    | GNOME  |
| Abacus   | 少   | 少  | Tcl/Tk |
| KMyMoney | 少   |    | KDE    |

### 7.5.1. GnuCash

GnuCash 是 GNOME 的一部分，GNOME 致力于提供最易用、友好且功能大的件。使用 GnuCash，可以注的收入和支、的行， 或者的股票。它的界面特性看起来非常的。

GnuCash 提供一个智能化的注册、分系、很多快捷方式和自完成方式。它能分一个个的理到几个的部分。 GnuCash 能入和合并 Quicken QIF 文件格式。 它也支持大部分的国日期和流行的格式。

在的系中安装 GnuCash 所需的命令如下：

```
# pkg_add -r gnuCash
```

如果 package 不可用，可以使用 Ports Collection 安装：

```
# cd /usr/ports/finance/gnuCash
# make install clean
```

### 7.5.2. Gnumeric

Gnumeric 是一个子表格程序， GNOME 面境的一部分。 它以通元素格式和多片断的自填充系来方便的自"猜"用而入而著称。 它能入一些流行的文件格式，比如象 Excel、 Lotus 1-2-3 或 Quattro Pro。 Gnumeric 凭借 [math/guppi](#) 支持表。 它有大量的嵌入函数和允所有通常比如象、数字、、日期、等等的一些元格式。

以 package 方式安装 Gnumeric 的方法如下：

```
# pkg_add -r gnumeric
```

如果 package 不可用，可以使用 Ports Collection 安装：

```
# cd /usr/ports/math/gnumeric
# make install clean
```

### 7.5.3. Abacus

Abacus 是一个小巧易用的电子表格程序。它包含许多嵌入函数在一些领域如科学、工程和数学方面很有帮助。它能输入和输出 Excel 文件格式。Abacus 可以生成 PostScript® 输出。

以 package 的方式安装 Abacus 的方法如下：

```
# pkg_add -r abacus
```

如果 package 不可用，你可以使用 Ports Collection 安装：

```
# cd /usr/ports/deskutils/abacus
# make install clean
```

### 7.5.4. KMyMoney

KMyMoney 是一个 KDE 环境下的个人财务管理软件。KMyMoney 旨在提供并融合各商业财务管理软件所有的重要特性。它也同时注重易用性和特有的会计功能。KMyMoney 能从标准的 Quicken Interchange Format (QIF) 文件输入数据，追踪投资，管理多币种并能提供一个报告。它有可用的插件支持输入 OFX 格式的数据。

以 package 的方式安装 KMyMoney 的方法如下：

```
# pkg_add -r kmy money2
```

如果 package 不可用，你可以使用 Ports Collection 安装：

```
# cd /usr/ports/finance/kmy money2
# make install clean
```

## 7.6. 网络

尽管 FreeBSD 由于其高性能和可靠性而赢得了许多 ISP 的信赖，但它也完全可以用于桌面环境。有数以千计的 [packages](#) 和 [ports](#) 能帮助你迅速建立完美的桌面环境。

下面是本章涉及到的所有的软件的重要回归：

| 软件名称    | Package 名称              | Ports 名称                             |
|---------|-------------------------|--------------------------------------|
| Opera   | <a href="#">opera</a>   | <a href="#">www/opera</a>            |
| Firefox | <a href="#">firefox</a> | <a href="#">www/firefox</a>          |
| KOffice | <a href="#">koffice</a> | <a href="#">editors/koffice-kde3</a> |
| AbiWord | <a href="#">abiword</a> | <a href="#">editors/abiword</a>      |

| □件名称            | Package 名称                  | Ports 名称                                 |
|-----------------|-----------------------------|--|
| The GIMP        | <a href="#">gimp</a>        | <a href="#">graphics/gimp</a>            |
| OpenOffice.org  | <a href="#">openoffice</a>  | <a href="#">editors/openoffice.org-3</a> |
| LibreOffice     | <a href="#">libreoffice</a> | <a href="#">editors/libreoffice</a>      |
| Acrobat Reader® | <a href="#">acroread</a>    | <a href="#">print/acroread8</a>          |
| gv              | <a href="#">gv</a>          | <a href="#">print/gv</a>                 |
| Xpdf            | <a href="#">xpdf</a>        | <a href="#">graphics/xpdf</a>            |
| GQview          | <a href="#">gqview</a>      | <a href="#">graphics/gqview</a>          |
| GnuCash         | <a href="#">gnucash</a>     | <a href="#">finance/gnucash</a>          |
| Gnumeric        | <a href="#">gnumeric</a>    | <a href="#">math/gnumeric</a>            |
| Abacus          | <a href="#">abacus</a>      | <a href="#">deskutils/abacus</a>         |
| KMyMoney        | <a href="#">kymoney2</a>    | <a href="#">finance/kymoney2</a>         |

# Chapter 8. 多媒体

## 8.1. 概述

FreeBSD 广泛地支持各种声音，你可以从容地享受来自你的计算机的高保真输出。它包括了录制和播放 MPEG Audio Layer 3 (MP3)、WAV、以及 Ogg Vorbis 等多种格式声音的能力。FreeBSD 同时也包括了许多的应用程序，你可以录音、加声音效果以及控制附加的MIDI。

要是对于新手，FreeBSD 也能支持播放一般的音频文件和 DVD。各种媒体发行、录制和播放的应用程序比起处理声音的应用程序略少一些。例如，在撰写本章时，FreeBSD Ports Collection 中并没有像 [audio/sox](#) 那样好的重采样工具能用来在不同的格式之间。不过，这个领域的软件研展是很快的。

本章将介绍配置声音的必要步骤。X11 的安装和配置 ([X Window 系统](#)) 里已经考虑到了所有的步骤，但要想有更好的播放效果，仍需要调整一些东西。

读了本章后，你将知道：

- 如何配置系统声音。
- 声音是否正常工作的方法。
- 如何排除声音安装中的问题。
- 如何播放和录制MP3以及其它格式的音频。
- X 服务器如何支持声音。
- 一些好的音频播放/"ports"。
- 如何播放 DVD、.mpg 以及 .avi 文件。
- 如何从 CD 和 DVD 中提取文件。
- 配置设备。
- 如何配置图像输入。

在本章之前，你需要：

- 知道如何配置、安装一个新的内核 ([配置FreeBSD的内核](#))



用[mount\(8\)](#) 命令去装CD光盘，至少会产生一个问题，更糟的情况下会产生 *kernel panic*。音频媒体所用的设备与通常的ISO文件系统是不同的。

## 8.2. 安装声音

### 8.2.1. 配置系统

在开始之前，请弄清楚声音类型、所用的芯片以及它是 PCI 还是 ISA 卡。FreeBSD 支持非常多的 PCI 和 ISA 卡。[硬件兼容说明](#) 中支持的音频列表看看是否支持你的声音，硬件兼容说明也会说明支持声音的是哪个程序。

要使用声音，就安装正确的程序。完成的方式有：  
最常用的是使用命令 [kldload\(8\)](#) 来安装一个内核模块，在命令行输入

```
# kldload snd_emu10k1
```

或者在文件 `/boot/loader.conf` 里加入一行，内容如下

```
snd_emu10k1_load="YES"
```

上例用于 Creative SoundBlaster® Live! 声卡。其它可装的模块列在文件 `/boot/defaults/loader.conf` 里。如果不知道使用哪个，可以添加 `snd_driver` module:

```
# kldload snd_driver
```

是个 meta 项，一次加载了最常的驱动。会提高搜索正项的速度。也可以通过 `/boot/loader.conf` 工具来加载所有的声卡。

如果希望在加载了 `snd_driver` meta 项之后了解到底用了什么声卡，可以通过使用 `cat /dev/sndstat` 来读 `/dev/sndstat` 文件。

另外，也可以把支持声卡的代码静悄悄地放到内核里去。下一节就采用这种方式支持硬件输出提示。至于重新编译内核，参考 [配置FreeBSD的内核](#)。

#### 8.2.1.1. 定制内核使其支持声卡

要做的第一件事情就是添加通用音频框架 `sound(4)` 到内核中，需要添加下面一行到内核配置文件中：

```
device sound
```

接下来就是加入我所用声卡的支持了。首先需要确定我的声卡需要使用哪一个。可以参考 [硬件兼容列表](#) 所列出的音频卡，以确定声卡的型号。例如，Creative SoundBlaster® Live! 声卡由 `snd_emu10k1(4)` 来支持。要添加它，需要在内核配置文件中加入下面一行：

```
device snd_emu10k1
```

一定要看看的卡手册了解如何使用它。至于内核配置文件中声卡的具体写法，也可以在 `/usr/src/sys/conf/NOTES` 文件中找到。

非即插即用的 ISA 卡可能需要内核提供一些关于声卡配置的信息（IRQ、I/O 端口，等等），这一点与其他不支持即插即用的 ISA 卡类似。这项工作可以通过 `/boot/device.hints` 文件来完成。系统启动时，[loader\(8\)](#) 将读取这个文件，并将其中的配置放入内核。例如，旧式的 Creative SoundBlaster® 16 ISA 非即插即用卡需要使用 `snd_sbc(4)` 并配合 `snd_sb16(4)`。可以在内核配置文件中添加如下配置：

```
device snd_sbc
device snd_sb16
```

有下面些到 `/boot/device.hints` 中：

```
hint.sbc.0.at="isa"
hint.sbc.0.port="0x220"
hint.sbc.0.irq="5"
hint.sbc.0.drq="1"
hint.sbc.0.flags="0x15"
```

，声使用 `0x220` I/O 端口和 IRQ 5。

在 `/boot/device.hints` 文件中所使用的法，在 [sound\(4\)](#) 机手册中以及所用的具体声的机手册中，会行一的解。

上面所展示的是默的配置。有候，可能需要更改 IRQ 或其他配置，以声的情况。看 [snd\\_sbc\(4\)](#) 机手册了解更多信息。

### 8.2.2. 声

用修改的内核重起，或者加了需要的模之后，声将会出在的系消息存中 ([dmesg\(8\)](#))，就像：

```
pcm0: <Intel ICH3 (82801CA)> port 0xdc80-0xdcbf,0xd800-0xd8ff irq 5 at device 31.5 on
pci0
pcm0: [GIANT-LOCKED]
pcm0: <Cirrus Logic CS4205 AC97 Codec>
```

声的状态可以通 `/dev/sndstat` 文件来：

```
# cat /dev/sndstat
FreeBSD Audio Driver (newpcm)
Installed devices:
pcm0: <Intel ICH3 (82801CA)> at io 0xd800, 0xdc80 irq 5 bufsz 16384
kld snd_ich (1p/2r/0v channels duplex default)
```

系的出可能与此不同。如果没有看到 `pcm` ，回并一下前面做的。重新的内核配置文件并保了正的。常列在 [常](#) 一。

如果一切正常，在有一个多功能声了。如果的 CD-ROM 或者 DVD-ROM 器的音出已与声在一起，可以把 CD 放入器并用 [cdcontrol\(1\)](#) 来播放：

```
% cdcontrol -f /dev/acd0 play 1
```

多用程序，比如 [audio/workman](#) 可以提供一个友好的界面。可能想要安装一个用程序比如 [audio/mpg123](#) 来听 MP3 音文件。

一快速声的方法，是将数据送到 `/dev/dsp`，像做：

```
% cat filename > /dev/dsp
```

这里 filename 可以是任意文件。该行命令会产生一些噪音，证明声音果真在工作。



点 `/dev/dsp*` 会在需要的时候自动生成。如果没有使用它，它不会出现在 `ls(1)` 的输出中。

声音混音可以通过 `mixer(8)` 命令更改。更多可以在 `mixer(8)` 手册中找到。

8.2.2.1. 常见问题

| 信息  | 解决方法                        |
|---|-----------------------------|
| <code>sb_dspwr(XX) timed out</code>                   | I/O端口没有重置。                  |
| <code>bad irq XX</code>                               | IRQ设置不正确。指定的IRQ和声音的IRQ是一样的。 |
| <code>xxx: gus pcm not attached, out of memory</code> | 没有足够的内存空间供设置使用。             |
| <code>xxx: can't open /dev/dsp!</code>                | 使用命令 <code>`fstat</code>    |

一个问题是多新式的设备本身包含它自己的声音，用以配合 HDMI 设备的使用。这个声音会在真正的声音之前被探测到，从而成默认的回放，而使真正的声音无法发声。要解决这种情况，运行 `dmesg` 并观察 `pcm`。其输出似下面：

```
...
hdac0: HDA Driver Revision: 20100226_0142
hdac1: HDA Driver Revision: 20100226_0142
hdac0: HDA Codec #0: NVidia (Unknown)
hdac0: HDA Codec #1: NVidia (Unknown)
hdac0: HDA Codec #2: NVidia (Unknown)
hdac0: HDA Codec #3: NVidia (Unknown)
pcm0: <HDA NVidia (Unknown) PCM #0 DisplayPort> at cad 0 nid 1 on hdac0
pcm1: <HDA NVidia (Unknown) PCM #0 DisplayPort> at cad 1 nid 1 on hdac0
pcm2: <HDA NVidia (Unknown) PCM #0 DisplayPort> at cad 2 nid 1 on hdac0
pcm3: <HDA NVidia (Unknown) PCM #0 DisplayPort> at cad 3 nid 1 on hdac0
hdac1: HDA Codec #2: Realtek ALC889
pcm4: <HDA Realtek ALC889 PCM #0 Analog> at cad 2 nid 1 on hdac1
pcm5: <HDA Realtek ALC889 PCM #1 Analog> at cad 2 nid 1 on hdac1
pcm6: <HDA Realtek ALC889 PCM #2 Digital> at cad 2 nid 1 on hdac1
pcm7: <HDA Realtek ALC889 PCM #3 Digital> at cad 2 nid 1 on hdac1
...
```

此输出 (NVidia) 先于真正的声音 (Realtek ALC889) 被探测到。要使用声音作默认的回放，将 `hw.snd.default_unit` 改成的值号：

```
# sysctl hw.snd.default_unit=n
```

里的 `n` 是希望使用的声音通道号，在例子中是 `4`。可以在 `/etc/sysctl.conf` 中写上这个配置来令其永久性生效：

```
hw.snd.default_unit=4
```

### 8.2.3. 利用多个声源

通常而言，会希望多个音源能同时播放，例如，`esound` 或者 `artsd` 就可能不支持与其它程序共享音源。

FreeBSD 可以通过 虚声道(Virtual Sound Channels) 来达到类似的效果，它可以用 [sysctl\(8\)](#) 来用。虚声道可以能在内核里混合声音来混合声卡里播放的声道。

使用三条sysctl命令来设置虚声道的数目。如果你是 `root` 用，进行下面的操作：

```
# sysctl dev.pcm.0.play.vchans=4
# sysctl dev.pcm.0.rec.vchans=4
# sysctl hw.snd.maxautovchans=4
```

上面的例子定了4个虚声道，也是卡上所使用的数目。`dev.pcm.0.play.vchans=4` 和 `dev.pcm.0.rec.vchans=4` 是 `pcm0` 用来播放与录音的虚声道数，一当接上一个卡它就可配置了。`hw.snd.maxautovchans` 是分配新的音源的虚声道数，此卡要用 [kldload\(8\)](#) 来接。因 `pcm` 模可以独立装多硬件程序，因此 `hw.snd.maxautovchans` 也就可以存分配以后接到的卡的虚声道数。可参 [pcm\(4\)](#) 手册获取更多。



不能在使用某个卡的时候改其虚通道数。首先需要所有使用该卡的程序，如音播放器或声音服。

当用程序求 `/dev/dsp0` 时，系会自其分配正的 `pcm` 卡。

### 8.2.4. 如何设置混音器通道

不同的混音通道的默认音量是硬 [pcm\(4\)](#) 程序的。同时，也有很多用或服程序提供了允许用直接置并住些的功能。但这并不是一个很好的解决方案，可能希望在同一有一个可以置的默认。可以通过在 `/boot/device.hints` 定适当的来。例如：

```
hint.pcm.0.vol="50"
```

将在 [pcm\(4\)](#) 模加载，将通道音量置默认的 50。

## 8.3. MP3音

MP3 (MPEG Layer 3 Audio)达到CD音的效果，FreeBSD工作站没理由会缺少的好。



### 8.3.1. MP3播放器

目前为止，最流行的 X11 MP3 播放器是 XMMS (X 多媒体系统)。Winamp 的界面可以直接用于 XMMS，因为它的 GUI 几乎和 Nullsoft 的 Winamp 完全一样。另外，XMMS 也提供了内建的硬件支持。

XMMS 可以通过 [multimedia/xmms](#) port 或 package 来安装。

XMMS 的界面很直观，它提供了播放列表、图形化均衡器等等。如果你对 Winamp 很熟悉，就会感觉 XMMS 很容易使用。

[audio/mpg123](#) port 提供了一个命令行界面的 MP3 播放器。

mpg123 可以在一行通过命令行指定声音设备和要播放的 MP3 文件，假设声音设备是 /dev/dsp1.0 并且你想要播放的 MP3 文件是 *Foobar-GreatestHits.mp3* 可以输入以下的命令：

```
# mpg123 -a /dev/dsp1.0 Foobar-GreatestHits.mp3
High Performance MPEG 1.0/2.0/2.5 Audio Player for Layer 1, 2 and 3.
Version 0.59r (1999/Jun/15). Written and copyrights by Michael Hipp.
Uses code from various people. See 'README' for more!
THIS SOFTWARE COMES WITH ABSOLUTELY NO WARRANTY! USE AT YOUR OWN RISK!

Playing MPEG stream from Foobar-GreatestHits.mp3 ...
MPEG 1.0 layer III, 128 kbit/s, 44100 Hz joint-stereo
```

### 8.3.2. 提取CD音

在将CD或CD音轨转换成MP3之前，CD上的音频数据必须先读到硬盘里。这个可以通过控制原始的CDDA(CD数字音频)数据成波形(WAV)文件。

工具 [cdda2wav](#) 是 [sysutils/cdrtools](#) 套件的一部分，可用来从CD中提取音频及其相关信息。

把CD放到光驱里，下面的命令可以完成(作为`root`用)把整个CD分割成几个(几个音轨)的WAV文件：

```
# cdda2wav -D 0,1,0 -B
```

cdda2wav 支持 ATAPI (IDE)光驱。从IDE光驱中提取音频，需要用名称代替SCSI的元号。例如，想从 IDE 光驱中提取第7道音轨：

```
# cdda2wav -D /dev/acd0 -t 7
```

参数 **-D 0,1,0** 表示 SCSI 设备 0,1,0，与命令 **cdrecord -scanbus** 的输出相同。

提取时，要使用 **-t**，如下所示：

```
# cdda2wav -D 0,1,0 -t 7
```

这个例子用于提取第七个音轨。要提取一定范围的音轨，如从1到7：

```
# cdda2wav -D 0,1,0 -t 1+7
```

利用`dd(1)`也可以从ATAPI光盘中取音，从 [制音 CD](#) 可以了解更多。

### 8.3.3. MP3 编码

如今，可用的MP3编码器是 `lame`。Lame 可以从ports里的 [audio/lame](#) 得到。

利用取得的WAV文件，下面的命令就可以把 `audio01.wav` 编码成 `audio01.mp3`：

```
# lame -h -b 128 \  
--tt "Foo Song Title" \  
--ta "FooBar Artist" \  
--tl "FooBar Album" \  
--ty "2001" \  
--tc "Ripped and encoded by Foo" \  
--tg "Genre" \  
audio01.wav audio01.mp3
```

128 kbits 是标准的MP3位率(bitrate)。多人可能喜欢更高的品质例如 160 或 192。更高的位率，会使 MP3 占用更多的磁盘空间——但音质会更高。选项 `-h` 控制 "高品质但低速度 (higher quality but a little slower)" 模式的编码。选项 `--t` 表示把 ID3 标签——通常包含了歌曲的信息，植入到MP3文件里。其它的选项可以参见 `lame` 的联机手册。

### 8.3.4. MP3 解码

要把MP3歌曲刻成音盘CD，就需要把它转换成非编码的波形(WAV)格式。XMMS 和 `mpg123` 都支持把MP3输出成非编码格式文件。

在 XMMS 中输出到磁盘：

1. 启动 XMMS。
2. 在窗口里右击鼠标，调出 XMMS 菜单。
3. 在 **选项(Options)** 里设置 **偏好(Preference)**。
4. 改输出设备成 "写磁盘插件(Disk Writer Plugin)"。
5. 按 **配置(Configure)**。
6. 输入或创建一个目录用于存放解码的文件。
7. 象平常一样，把MP3文件装入到 XMMS 里，把音量调到100%并且关掉EQ设定。
8. 按一下 **播放(Play)** - XMMS 如同在播放mp3一样，只是听不到声音。实际上是在播放mp3到一个文件里。
9. 要想再听MP3歌曲，得把默认的输设备回原来的。

用 `mpg123` 进行标准输出：

```
1. 运行 `mpg123 -s audio01.mp3 > audio01.pcm`
```

XMMS 输出的文件是波形(WAV)格式，而 mpg123 把MP3转换成无压缩的PCM 音频数据。两种格式都支持用 cdrecord 刻成声音CD。使用 [burncd\(8\)](#) 就必须使用无压缩的PCM。如果用波形格式，就要注意在声道开始的一小点静音，一段声音是波形文件的头部。可以使用工具 SoX 来轻松去除。SoX 可从 [audio/sox](#) port 或包(package)中安装得到：

```
% sox -t wav -r 44100 -s -w -c 2 track.wav track.raw
```

有关 [构建和使用光学介质\(CD\)](#) 部分可以了解到更多在 FreeBSD 里刻盘的信息。

## 8.4. 回放

回放是个很新并且迅速发展中的应用领域。一定要有耐心，因为不是所有的事情都象听音乐那顺利。

在开始之前，要了解驱动的类型以及它所用的芯片的类型。尽管 Xorg 支持大量的驱动，但能得到好的回放效果的却寥寥无几。在X11运行，可以使用命令 [xdpyinfo\(1\)](#) 得到使用的驱动的X服务器所支持的驱动列表。

为了评估各播放器和配置，需要有一小段用作测试的MPEG文件。由于一些DVD播放器会默认地在 /dev/dvd 里去读DVD文件，因此，会建立符号链接到恰当的驱动会很有用：

```
# ln -sf /dev/acd0 /dev/dvd
# ln -sf /dev/acd0 /dev/rdvd
```

注意：由于 [devfs\(5\)](#) 本身的原因，像手工建立的链接在重启后将不会存在。想要无论什么时候系统都能自动建立符号链接，那就把下面行加到 /etc/devfs.conf 里：

```
link acd0 dvd
link acd0 rdvd
```

另外，DVD解密要求用用的DVD-ROM函数，要求把驱动可定到DVD驱动里。

为了改善 X11 界面使用共享内存的能力，建议提高一些 [sysctl\(8\)](#) 量的：

```
kern.ipc.shmmax=67108864
kern.ipc.shmall=32768
```

### 8.4.1. 驱动性能

在X11下有几驱动可以显示图像的方式。到底哪个能工作很大程度上依赖于硬件。首先，下面描述的每一种方法在不同的硬件上都会有不同的品质。其次，在X11里的图像显示近来引起普遍的关注，随着 Xorg 的一个版本，都会有很大的突破。

常⌘像接口列表：

1. X11: 一般性的使用共享内存的X11⌘出。
2. XVideo: 一⌘X11接口⌘展，支持任何X11⌘像的可⌘拉。
3. SDL: ⌘⌘直接媒体⌘。
4. DGA: 直接⌘片存取。
5. SVGAlib: 低⌘次掌控⌘片⌘。

#### 8.4.1.1. XVideo

Xorg 有⌘⌘展叫做 *XVideo* (或称Xvideo, Xv, xv)，它可以通⌘一个特殊的加速器直接把⌘像⌘示在可⌘拉的⌘象里。即使在低端的⌘算机 (例如我的PIII 400 Mhz膝上⌘⌘)，⌘个⌘展也提供了很好的播放⌘量。

要了解⌘一⌘展是否在正常工作，使用 `xvinfo` 命令：

```
% xvinfo
```

如果⌘示⌘果如下，那⌘的⌘⌘就支持XVideo：

```
X-Video Extension version 2.2
screen #0
  Adaptor #0: "Savage Streams Engine"
    number of ports: 1
    port base: 43
    operations supported: PutImage
    supported visuals:
      depth 16, visualID 0x22
      depth 16, visualID 0x23
    number of attributes: 5
      "XV_COLORKEY" (range 0 to 16777215)
        client settable attribute
        client gettable attribute (current value is 2110)
      "XV_BRIGHTNESS" (range -128 to 127)
        client settable attribute
        client gettable attribute (current value is 0)
      "XV_CONTRAST" (range 0 to 255)
        client settable attribute
        client gettable attribute (current value is 128)
      "XV_SATURATION" (range 0 to 255)
        client settable attribute
        client gettable attribute (current value is 128)
      "XV_HUE" (range -180 to 180)
        client settable attribute
        client gettable attribute (current value is 0)
    maximum XvImage size: 1024 x 1024
    Number of image formats: 7
    id: 0x32595559 (YUY2)
    guid: 59555932-0000-0010-8000-00aa00389b71
```

```

bits per pixel: 16
number of planes: 1
type: YUV (packed)
id: 0x32315659 (YV12)
guid: 59563132-0000-0010-8000-00aa00389b71
bits per pixel: 12
number of planes: 3
type: YUV (planar)
id: 0x30323449 (I420)
guid: 49343230-0000-0010-8000-00aa00389b71
bits per pixel: 12
number of planes: 3
type: YUV (planar)
id: 0x36315652 (RV16)
guid: 52563135-0000-0000-0000-000000000000
bits per pixel: 16
number of planes: 1
type: RGB (packed)
depth: 0
red, green, blue masks: 0x1f, 0x3e0, 0x7c00
id: 0x35315652 (RV15)
guid: 52563136-0000-0000-0000-000000000000
bits per pixel: 16
number of planes: 1
type: RGB (packed)
depth: 0
red, green, blue masks: 0x1f, 0x7e0, 0xf800
id: 0x31313259 (Y211)
guid: 59323131-0000-0010-8000-00aa00389b71
bits per pixel: 6
number of planes: 3
type: YUV (packed)
id: 0x0
guid: 00000000-0000-0000-0000-000000000000
bits per pixel: 0
number of planes: 0
type: RGB (packed)
depth: 1
red, green, blue masks: 0x0, 0x0, 0x0

```

同␣注意：列出来的格式(YUV2, YUV12, 等等) 并不␣是随着 XVideo的␣一次␣行而存在。没有它␣可能会迷惑某些人。

如果␣果看起来是␣␣：

```

X-Video Extension version 2.2
screen #0
no adaptors present

```

那□□的□□可以就不支持XVideo功能。

如果□的□不支持XVideo, □只是□明□的□示器在□足刷新□像的□算要求上存在更大的困□。 □尽管□□和□理器很重要, □仍然会有个不□的□示效果。此外, □也可以参考我□提供的文献, 在 [□—□了解](#) 中有所介□。

#### 8.4.1.2. □□直接媒体□

□□直接媒体□(SDL), 原意是做□ Microsoft® Windows®, BeOS 以及 UNIX® 之□的端口□, 允□跨平台□用□展, 更高效地利用声□和□形□。SDL □可以在低□□□硬件, 有□□□做就比 X11 接口□更□高效。

□于 SDL, 可以参考 [devel/sdl12](#)。

#### 8.4.1.3. 直接□形存取

直接□形存取 (Direct Graphics Access) 是一□ X11 □展, 通□它, □用程序能□□□ X 服□, 并直接修改画面□存 (framebuffer)。由于它依□一□底□的内存映射来□□其功能, 因此使用它的程序必□以 **root** 身□来□行。

DGA □展可以通□ [dga\(1\)](#) 来完成□□和性能□量。□行 **dga** □, 它将随按□改□□□的□色。按 **q** 退出□个程序。

### 8.4.2. Ports 和 包(Packages) □□□的解决

□部□主要□□在 FreeBSD Ports 集中提供的可用于□□回放的□件。 □□回放在□件□展中是个很活□的□域, 并且各□不同程序的功能可能与□里的描述不尽相同。

首先要弄清楚的重要一点是在 FreeBSD 上使用的□□程序其□展与在 Linux 里使用的是一□的。大部□程序都□□在β□段。使用 FreeBSD 的包可能面□的□□:

1. 一个□用程序不能播放其它程序制作的文件。
2. 一个□用程序不能播放其自己制作的文件。
3. 不同机上的同□的程序, 各自重新建立(rebuild)了一次, 播放同一个文件□果也会有不同。
4. 一个看起来没什□的□□器, 如□像尺寸的□整, 也有可能因□一个□整例程的□□□□得很不象□。
5. □用程序□繁地留下□□(dumps core)。
6. 没有随 port 一起安装的文□可以在网上或者 port 的 work 目□中□到。

□些程序中□多也体□了 "Linux主□"。即, □有些□□来自于(程序)使用的□准□存在于Linux的□行版中, 或者有些是 Linux 内核的功能, 而□程序的作者事先所假定的是 Linux内核。□些□□并不□是被 port □□人□注意到或□理□, □也就可能□致如下□□:

1. 使用/proc/cpuinfo去□□□理器的特性。
2. □用□程可能□致一个程序□挂完成, 而不是完全中止。
3. □件□不属于FreeBSD Ports集, 而又与其它程序□常地一起使用。

□在, □些程序的□□人□也已同 port 的□□人□□行了□合, 以□少制作port□出□。

#### 8.4.2.1. MPlayer

MPlayer 是近来□□的同□也正迅速□展着的一个□□播放器。MPlayer □□的目□是在 Linux 和其它 UNIX 系□中的速度和机□性能。在□□的□始人□在受不了当□可用的播放器的性能□, □个□□就□始了。有人也□会

图形接口已成为新型系统的牺牲品。但是一旦拥有了命令行和按钮控制方式，它就能表现得很好。

#### 8.4.2.1.1. 构建MPlayer

MPlayer 可以从 [multimedia/mplayer](#) 得到。MPlayer 在构建过程中会进行许多硬件测试，而得到的可执行文件因此将无法移植到其他系统中使用。因此，从 ports 完成测试而不是安装测试的包就很重要。另外，在 `make` 命令行中可以指定许多选项，在 Makefile 中有所描述，接下来我们开始：

```
# cd /usr/ports/multimedia/mplayer
# make
N - O - T - E

Take a careful look into the Makefile in order
to learn how to tune mplayer towards you personal preferences!
For example,
make WITH_GTK1
builds MPlayer with GTK1-GUI support.
If you want to use the GUI, you can either install
/usr/ports/multimedia/mplayer-skins
or download official skin collections from
http://www.mplayerhq.hu/homepage/dload.html
```

默认的 port 适用于大多数用户来使用。不过，如果需要 XviD 解码器，必须指定 `WITH_XVID` 这个命令行选项。默认的 DVD 选项也可以用 `WITH_DVD_DEVICE` 选项来定义，其默认是 `/dev/acd0`。

撰写一章的时候，MPlayer port 的构建过程包括了 HTML 文档和两个可执行文件，`mplayer` 和 `mencoder`，后者是一个再编码工具。

MPlayer 的 HTML 文档提供了丰富的内容。如果读者本章中缺少关于硬件的一些信息，MPlayer 的文档将是十分详尽的补充。如果正在寻找 UNIX® 中的支持的资料，请花一些时间来读 MPlayer 的文档。

#### 8.4.2.1.2. 使用MPlayer

任何 MPlayer 用户必须在其用户主目录下建立一个叫 `.mplayer` 的子目录。输入下的内容来建立一个必需的子目录：

```
% cd /usr/ports/multimedia/mplayer
% make install-user
```

在 `mplayer` 的手册里列出了它的命令。HTML 文档里有更详细的信息。这部分里，我只描述了很少的常用。

要播放一个文件，如 `testfile.avi`，可以通过各接口当中的某一个去设置 `-vo` 选项：

```
% mplayer -vo xv testfile.avi
```

```
% mplayer -vo sdl testfile.avi
```

```
% mplayer -vo x11 testfile.avi
```

```
# mplayer -vo dga testfile.avi
```

```
# mplayer -vo 'sdl:dga' testfile.avi
```

所有这些都是一致的，因为它性能依赖于很多因素，并且都与硬件密切相关。

要播放 DVD，需要把 testfile.avi 改为 `dvd://N -dvd-device DEVICE`。这里 `N` 是要播放的目录号，而 `DEVICE` 是 DVD-ROM 的接口点。例如，要播放 `/dev/dvd` 的第三个目录：

```
# mplayer -vo xv dvd://3 -dvd-device /dev/dvd
```



可以在 MPlayer 中，通过 `WITH_DVD_DEVICE` 来指定默认的 DVD 接口。系统内定的默认接口是 `/dev/acd0`。更多信息，请参考 `port` 的 Makefile。

要停止、暂停、前等等，可以参考指定的按钮——一些可以通过 `mplayer -h` 得到或看手册。

另外，回放的重要选项是：用于全屏模式的 `-fs -zoom` 和起辅助完成作用的 `-framedrop``。

除了 `mplayer` 的命令行不太方便，使用者可以通过建立一个文件 `.mplayer/config` 来设定如下默认值：

```
vo=xv
fs=yes
zoom=yes
```

最后，`mplayer` 可以把 DVD 目录(title)提取成 `.vob` 文件。为了从 DVD 中取出第二个目录，输入：

```
# mplayer -dumpstream -dumpfile out.vob dvd://2 -dvd-device /dev/dvd
```

输出文件 `out.vob` 将是 MPEG 并且可以被部分描述的其它“包”利用。

#### 8.4.2.1.3. mencoder

在使用 `mencoder` 之前，首先熟悉其 HTML 文档中所介绍的选项是一个不错的主意。它提供了联机手册，但如果没有 HTML 文档帮助不大。有无数种方法来提高品质、降低比特率、修改格式，而有些技巧可能会影响性能。下面是几个例子，第一个是原地制：

```
% mencoder input.avi -oac copy -ovc copy -o output.avi
```

不正确的命令组合可能使生成的文件不能被 `mplayer` 播放。因此，如果你只是想提取文件，一定在 `mplayer` 里使用 `“-dumpfile”`。



input.avi 成有MPEG3音 (要求 [audio/lame](#)) 的MPEG4 :

```
% mencoder input.avi -oac mp3lame -lameopts br=192 \
    -ovc lavc -lavcopts vcodec=mpeg4:vhq -o output.avi
```

就生了可被 `mplayer` 和 `xine` 播放的出。

input.avi 可以成 `dvd://1 -dvd-device /dev/dvd` 并以 `root` 的身来行, 以重新 DVD 目行。由于第一次做的工作很可能不会果不太意, 建首先把目制成文件, 然后它行操作。

#### 8.4.2.2. xine 播放器

xine 播放器是一个注很广的目, 它不看准多合一的解决, 而且出品了一个可再用的基本和一个可展件的可行模。行有 "包" 和port版本-- [multimedia/xine](#)。

xine 播放器仍然很粗, 但很然与好无。上 xine 要求有快速的 CPU 和快速的来行, 或者需要支持 XVideo 展。形界面(GUI)可以使用, 但很勉。

到写章, 没有可用于播放CSS的DVD文件的入模随同 xine 一起行。第三方的建造(builds)里内建有模, 但都不属于FreeBSD Ports 集。

与MPlayer 相比, xine 用考得更多, 但同, 用来也少了很多有条理的控制方式。xine 播放器在XVideo接口上做得不。

默情况下, 播放器 xine 的候会使用形界面。那就可以使用菜打指定的文件:

```
% xine
```

外, 没有形界面也可以使用如下命令立即打播放文件:

```
% xine -g -p mymovie.avi
```

#### 8.4.2.3. 使用transcode

transcode 个件并不是播放器, 而是一系列用于和音文件行重新的工具。通使用 transcode, 就可以有使用 stdin/stdout 接口的命令行工具来合并文件, 以及修坏文件的能力。

在 [multimedia/transcode](#) port 可以指定大量, 我建使用下面的命令行来建 transcode:

```
# make WITH_OPTIMIZED_CFLAGS=yes WITH_LIBA52=yes WITH_LAME=yes WITH_OGG=yes \
WITH_MJPEG=yes -DWITH_XVID=yes
```

于多数用而言, 前述配置已足了。

了明 transcode 的功能, 下面的例子展示了如何将 DivX 的 PAL MPEG-1 文件 (PAL VCD):

```
% transcode -i input.avi -V --export_prof vcd-pal -o output_vcd
% mplex -f 1 -o output_vcd.mpg output_vcd.m1v output_vcd.mpa
```

生成的 MPEG 文件, `output_vcd.mpg`, 可以通过 MPlayer 来播放。甚至可以直接将这个文件刻到 CD-R 介上来建 Video CD, 如果希望做的, 需要安装 [multimedia/vcdimager](#) 和 [sysutils/cdrdao](#) 这个程序。

`transcode` 提供了机手册, 但仍参考 [transcode wiki](#) 以了解更多信息和例子。

### 8.4.3. 一了解

FreeBSD里不同的部件包正迅速展中。很可能在不久的将来, 里所到的都将得到解决。同, 有些人想超越FreeBSD的音/像(A/V)能力, 那他就不不得不从一些FAQ和指南里学知, 并使用一些不同的用程序。里就些者指出一些充信息。

[MPlayer 文](#) 是很技性的。些文可以那些希望得于UNIX®高技的人提供参考。MPlayer 件列表很不喜没耐心文的人, 如果什想告他, 首先RTFM。

[xine HOWTO](#) 里有一章是关于提高性能的, 所有的播放器都很。

最后是一些很有前途的程序, 者可以一下:

- [Avifile](#), 它就是 [multimedia/avifile](#) port。
- [Ogle](#) 它就是 [multimedia/ogle](#) port。
- [Xtheater](#)
- [multimedia/dvdauthor](#), 一个制作 DVD 目的源放包。

## 8.5. 安装

### 8.5.1. 介

可以到的计算机里看到无或有。多是通RCA或S-video入接收合, 而且有些有广播接收器。

FreeBSD 通**`bktr(4)`**程序, 提供了基于PCI的支持, 要求些使用的是Brooktree Bt848/849/878/879 或 Conexant CN-878/Fusion 878a采集芯片。要保个板上有的有被支持的器, 参考**`bktr(4)`**手册看所支持的器列表。

### 8.5.2. 加程序

要使用的, 就要装**`bktr(4)`**程序。个可以通往 `/boot/loader.conf` 里添加下一行来。象:

```
bktr_load="YES"
```

外, 也可以把个内核, 要是的, 就把下几行加到内核配置里去:

```
device bktr
device iicbus
device iicbb
device smbus
```

一些附加的驱动程序是必需的，因为它的各部分是能跟一根I2C线相互连接在一起的。  
然后建立安装新的内核。

一旦这个支持被加到了你的系统里，你要重启系统。在过程中，你的系统显示up(0)，象这样：

```
bktr0: <BrookTree 848A> mem 0xd7000000-0xd7000fff irq 10 at device 10.0 on pci0
iicbb0: <I2C bit-banging driver> on bti2c0
iicbus0: <Philips I2C bus> on iicbb0 master-only
iicbus1: <Philips I2C bus> on iicbb0 master-only
smbus0: <System Management Bus> on bti2c0
bktr0: Pinnacle/Miro TV, Philips SECAM tuner.
```

当然，一些信息可能因你的硬件不同而有所区别。但是你能知道那个控制器是否被正确识别到了，  
可能要忽略一些识别到的同[sysctl\(8\)](#) MIB（管理系统）和内核配置文件一起的参数。例如，如果你想  
制使用Philips(飞利浦) SECAM制式的调谐器，你就把下列行加到内核配置文件里：

```
options OVERRIDE_TUNER=6
```

或者，你直接使用[sysctl\(8\)](#)：

```
# sysctl hw.bt848.tuner=6
```

参考 [bktr\(4\)](#) 手册和 /usr/src/sys/conf/NOTES 文件，以了解更多关于可用调谐器的资料。

### 8.5.3. 有用的应用程序

要使用你的设备，你需要安装下列应用程序之一：

- [multimedia/fxtv](#) 提供 "窗口电视(TV-in-a-window)" 功能和图像/声音/图像采集功能。
- [multimedia/xawtv](#) 也是一款应用程序，功能同 fxtv 一样。
- [misc/alevt](#) 解码和显示 Videotext/Teletext。
- [audio/xmradio](#)，一款用于一些调谐器的电台调谐器的程序。
- [audio/wmtune](#)，一款用于电台调谐器的便捷的图形程序。

更多的程序在FreeBSD Ports Collection(Ports 集)里。

### 8.5.4. 问题解决

如果你的设备遇到了什么问题，你首先检查一下你的图像采集芯片和调谐器是不是真正的被[bktr\(4\)](#)

程序支持，并且是不是使用了正确的配置。

想得到更多支持和关于的的各，

。

可以接触和使用[FreeBSD 多媒体设备列表](#) 设备列表的包。

## 8.6. 设备扫描

### 8.6.1. 介绍

在 FreeBSD 中，设备扫描的能力，是通过 SANE (Scanner Access Now Easy) API 提供的。SANE 也会使用一些 FreeBSD 设备来扫描硬件。

FreeBSD 支持 SCSI 和 USB 扫描。在做任何配置之前确保的设备被 SANE 支持。SANE 有一个 [支持的设备列表](#)，可以提供有设备的支持情况和状态的信息。在 FreeBSD 8.X 之前版本的系统中，[usanner\(4\)](#) 手册也提供了系统支持的 USB 扫描列表。

### 8.6.2. 内核配置

上面提到 SCSI 和 USB 接口都是支持的。取决于设备的扫描接口，需要不同的驱动程序。

#### 8.6.2.1. USB 接口

默认的 GENERIC 内核包含了支持 USB 扫描需要的设备。如果决定使用一个定制的内核，确保下面在系统的内核配置文件中存在下面这些行：

```
device usb
device uhci
device ohci
device ehci
```

在 FreeBSD 8.X 之前的版本中，需要下面两行配置：

```
device usanner
```

在某些 FreeBSD 版本中，是通过驱动程序 [usanner\(4\)](#) 来提供 USB 扫描的支持的。从 FreeBSD 8.0 开始，这些支持直接由 [libusb\(3\)](#) 函数提供。

使用正确的内核重新引导系统之后，插入 USB 扫描。系统消息缓冲区 (使用 [dmesg\(8\)](#) 查看) 中会出现下面的信息，表示到达了扫描：

```
ugen0.2: <EPSON> at usb0
```

或者，对于 FreeBSD 7.X 系统而言：

```
usanner0: EPSON EPSON Scanner, rev 1.10/3.02, addr 2
```

随 FreeBSD 版本不同，这些信息表示扫描设备位于哪个点 /dev/ugen0.2 或 /dev/usanner0。在例子中，

我使用的是 EPSON Perfection® 1650 USB 扫描仪。

### 8.6.2.2. SCSI 接口

如果你的扫描仪是 SCSI 接口的，重要的是要知道使用哪个 SCSI 控制器。取决于所使用的 SCSI 芯片，你可能需要调整内核配置文件。GENERIC 的内核支持最常用的 SCSI 控制器。看看 NOTES 文件并在你的内核配置文件中添加正确的行。除了 SCSI 适配器之外，你可能需要在内核配置文件中添加下述配置：

```
device scbus
device pass
```

在正确地编译并安装了内核之后，你就可以在系统中，从系统消息缓冲中看到一些：

```
pass2 at aic0 bus 0 target 2 lun 0
pass2: <AGFA SNAPSCAN 600 1.10> Fixed Scanner SCSI-2 device
pass2: 3.300MB/s transfers
```

如果你的扫描仪没有在系统中的时候添加，很可能你需要制手一下，用 [camcontrol\(8\)](#) 命令进行一次 SCSI 扫描：

```
# camcontrol rescan all
Re-scan of bus 0 was successful
Re-scan of bus 1 was successful
Re-scan of bus 2 was successful
Re-scan of bus 3 was successful
```

然后扫描仪就会出现在 SCSI 列表里：

```
# camcontrol devlist
<IBM DDRS-34560 S97B>          at scbus0 target 5 lun 0 (pass0,da0)
<IBM DDRS-34560 S97B>          at scbus0 target 6 lun 0 (pass1,da1)
<AGFA SNAPSCAN 600 1.10>      at scbus1 target 2 lun 0 (pass3)
<PHILIPS CDD3610 CD-R/RW 1.00> at scbus2 target 0 lun 0 (pass2,cd0)
```

有关 SCSI 的更多信息，可看 [scsi\(4\)](#) 和 [camcontrol\(8\)](#) 手册。

### 8.6.3. SANE 配置

SANE 系统分两部分：后端 ([graphics/sane-backends](#)) 和前端 ([graphics/sane-frontends](#))。后端部分提供到扫描仪自身的接口。SANE 的 [支持列表](#) 说明了哪个后端可以支持你的图像扫描仪。如果你想使用你的接口，就必须确定你的扫描仪绑定正确的后端。前端部分提供图形化的扫描界面 (xscanimage)。

要做的第一件事就是安装 [graphics/sane-backends](#) port 或者 package。然后，使用 [sane-find-scanner](#) 命令来让 SANE 系统做的扫描：

```
# sane-find-scanner -q
found SCSI scanner "AGFA SNAPSCAN 600 1.10" at /dev/pass3
```

输出显示了扫描器的接口类型和扫描器接到系统上的设备点。生产厂家和型号可能没有显示，不重要。



一些 USB 扫描器需要加载固件，后端的手册中有这方面的解释。你也可以看看 [sane-find-scanner\(1\)](#) 和 [linprocfs\(7\)](#) 手册。

在我需要知道扫描器是否可以被扫描前端时。默认情况下，SANE 后端自带一个叫做 [sane\(1\)](#) 的命令行工具。这个命令允许列出设备以及从命令行运行扫描。 `-L` 选项用来列出扫描器：

```
# scanimage -L
device `snapscan:/dev/pass3' is a AGFA SNAPSCAN 600 flatbed scanner
```

或者，如果使用的是 USB 接口中的 USB 扫描器：

```
# scanimage -L
device 'epson2:libusb:/dev/usb:/dev/ugen0.2' is a Epson GT-8200 flatbed scanner
```

上述输出来自于 FreeBSD 8.X 系统。 `'epson2:libusb:/dev/usb:/dev/ugen0.2'` 显示了扫描器所使用的后台名字 (`epson2`) 和设备点 (`/dev/ugen0.2`)。


如果没有输出任何信息，或提示没有找到扫描器，则表明 `sane(1)` 无法找到它。如果生成输出情况，则需要修改扫描器支持后端的配置文件，并定义所使用的扫描器。`/usr/local/etc/sane.d/` 目录中包含了所有的后端配置文件。通常会在某些 USB 扫描器上生成。

```
linkend="scanners-kernel-usb"> 中所使用的 USB 扫描器，`sane-find-scanner` 会输出下面的信息：
```

例如，对于在 **USB 接口**，在 FreeBSD 8.X 中，扫描器已被很好地识别并能正常工作了；而对于更早版本的 FreeBSD 而言（使用 `usscanner(4)` 程序）`sane-find-scanner` 会输出下面的信息：

```
# sane-find-scanner -q
found USB scanner (UNKNOWN vendor and product) at device /dev/usscanner0
```

扫描器被正确地探测到了，它使用 USB 接口，连接在 `/dev/usscanner0` 端点上。我现在可以看看扫描器是否被正确地识别了：

 `# scanimage -L`  
No scanners were identified. If you were expecting something different, check that the scanner is plugged in, turned on and detected by the sane-find-scanner tool (if appropriate). Please read the documentation which came with this software (README, FAQ, manpages).

由于扫描器没有识别成功，我则需要修改 `/usr/local/etc/sane.d/epson2.conf` 文件。所用的扫描器型号是 EPSON Perfection® 1650，而我则知道扫描器使用 `epson` 后端。确保后端配置文件中的注释。更改非常直接：注释掉导致扫描器使用错误接口的所有行（在我目前情况下，我将注释掉从 `scsi` 开始的所有行，因为我使用的扫描器使用 USB 接口），然后在文件的末尾添加指定的接口和所用的端点。目前情况下，添加下面两行：

```
usb /dev/usscanner0
```

确保后端配置文件提供的注释以及后端手册了解更多信息，并使用正确的方法。我现在可以看看扫描器是否被识别到了：

```
# scanimage -L
device `epson:/dev/usscanner0' is a Epson GT-8200 flatbed scanner
```

我的 USB 扫描器被识别到了。此扫描器如果商号和型号与扫描器的情况不符，并不会带来太大的麻烦。需要注意的是 ``epson:/dev/usscanner0'` 字段，它给了我正确地后端名称和正确的端点。

一旦 `scanimage -L` 命令可以看到扫描器，配置就完成了。现在在准备好等待扫描了。

[sane\(1\)](#) 允许我从命令行扫描图片，相比之下使用图形界面来扫描图片会更好。SANE 提供了一个但用的图形界面：[xscanimage \(graphics/sane-frontends\)](#)。

[Xsane \(graphics/xsane\)](#)是一个流行的图形扫描前端。这个前端提供了一些高级特性，比如多页扫描模式(photocopy, fax, 等。), 色彩校正, 批量扫描, 等等。这个程序都可以作为 GIMP 的插件使用。

#### 8.6.4. 授予其他用户 USB 访问权限

前面所有的操作都是用 `root` 权限来完成的。然而可能还需要其他的用户也可以访问 USB。用户需要有扫描器所用的设备节点的读和写权限。比如，我的 USB 扫描器使用设备节点 `/dev/ugen0.2` 上只是到设备节点 `/dev/usb/0.2.0` 的符号链接(可以通过查看 `/dev` 目录的内容来验证一点)。设备节点本身和这个符号链接分属于 `wheel` 和 `operator` 组。将用户 `joe` 添加到这些组中，就可以允许他使用扫描器了，不过，出于安全方面的原因，在将用户加到特定的用户组，特别是 `wheel` 组，无疑需三思而后行。更好的解决方法是建立一个专用于 USB 设备的组，并让这个组的成员能访问 USB 设备。

这里作为示例，我将使用名为 `usb` 的组。第一是借助 [pw\(8\)](#) 命令来创建它：

```
# pw groupadd usb
```

接下来，令 `/dev/ugen0.2` 符号链接和 `/dev/usb/0.2.0` 设备节点能以 `usb` 组的身分来访问，具体而言是配置正确的读写权限 (`0660` 或 `0664`)，因为默认情况下只有属主 (`root`) 才能写这些设备。这些配置是通过在 `/etc/devfs.rules` 文件中添加如下的配置来实现的：

```
[system=5]
add path ugen0.2 mode 0660 group usb
add path usb/0.2.0 mode 0666 group usb
```

FreeBSD 7.X 用户需要将上面的配置改使用与之对应的 `/dev/uscanner0`：

```
[system=5]
add path uscanner0 mode 660 group usb
```

随后需要在 `/etc/rc.conf` 中添加下面的内容并重新启动：

```
devfs_system_ruleset="system"
```

对于这些配置的唯一参考是手册 [devfs\(8\)](#)。

现在，只需将用户添加到 `usb` 组，就可以使用扫描器了：

```
# pw groupmod usb -m joe
```

更多详情，请参考手册 [pw\(8\)](#)。



# Chapter 9. 配置FreeBSD的内核

## 9.1. 概述

内核是 FreeBSD 操作系统的核心。它管理内存、执行安全控制、网络、磁盘等等。尽管 FreeBSD 可以修改的在已越来越多，但有的是需要重新配置和编译的内核。

读完本章，你将了解：

- 何时需要建立定制的内核。
- 如何编写内核配置文件，或修改已存在的配置文件。
- 如何使用内核配置文件构建和编译新的内核。
- 如何安装新内核。
- 如何清理旧的。

一章输出的命令必须以 `root` 身份运行，否则可能会不成功。

## 9.2. 何时需要建立定制的内核？

过去，FreeBSD 采用的是被人称作“切片式”的内核。内核本身是一个大的程序，它支持的不能随意地加以改变，而当希望改变内核的行为，就必须编译一个新的内核，并重新安装计算机才可以使用它。

如今，FreeBSD 正在迅速地移向一个新的模型，其特点是将大量内核功能放在可以添加和卸载的内核模块来提供。这使得内核能整合硬件的变更（例如笔记本电脑中的 PCMCIA 卡），以及内核引入新的功能，而无需在旧内核中就将其添加进去。这种做法称模块化内核。

尽管如此，仍然有一些功能需要静静地留在内核。有时，这是由于某些功能与内核的结合非常紧密而无法分离，也有一些情况是没有人将这些功能改写为可加载的模块。

定制的内核是成本高昂 BSD 用户所必需的一项。尽管它可能需要花费一些时间，但它能给你的 FreeBSD 系统带来一些好处。与必须支持大量硬件的 GENERIC 内核不同，定制的内核可以只包含于 PC 硬件的支持。它做有很多好处，例如：

- 更快地启动。因为内核只需要系统上的硬件，启动所花费的时间将大大缩短。
- 使用更少的内存。由于可以省去不需要的功能和代码，通常定制的内核会比 GENERIC 使用的内存更少。节省内核使用的内存之所以重要是因为内核必须常驻于物理内存中，从而使应用程序能用到更多的内存。正因如此，RAM 小的系统来定制内核就更重要了。
- 支持更多的硬件。定制的内核允许添加类似声卡的 GENERIC 内核没有提供内建支持的硬件。

## 9.3. 系统硬件

在配置内核以前，比较明智的做法是先得到机器硬件的清单。当 FreeBSD 并不是主操作系统，通常查看当前操作系统的配置可以很容易地建立一个机器硬件的配置清单。例如，Microsoft® 的设备管理器里通常含有关于已安装硬件的重要信息。设备管理器位于控制面板。



某些版本的 Microsoft® Windows® 有一个 系 会指明 管理器 的位置。

如果机器上并不存在其他的操作系， 系管理 只能手 些信息了。其中的一个方法是使用 [dmesg\(8\)](#) 工具以及 [man\(1\)](#) 命令。FreeBSD 上大多数的 程序都有一 手册（manual page）列出了所支持的硬件，在系 的候，被 的硬件也会被列出。 例来， 下面的 几行表示 psm 到了一个鼠：

```
psm0: <PS/2 Mouse> irq 12 on atkbdc0
psm0: [GIANT-LOCKED]
psm0: [ITHREAD]
psm0: model Generic PS/2 mouse, device ID 0
```

个 需要被包含在客 制定的内核配置文件里， 或着使用 [loader.conf\(5\)](#) 加。

有， [dmesg](#) 里只会 示来自系 消息的数据， 而不是系 的 信息。在 的情况下， 可以 看文件 `/var/run/dmesg.boot`。

一个 硬件信息的方法是使用 [pciconf\(8\)](#) 工具， 它能提供更 的 出， 比如：

```
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
```

个片断取自于 `pciconf -lv` 命令的 出， 示 ath 到了一个无 以太网 。 入命令 `man ath` 就能 有 [ath\(4\)](#) 的手册（manual page）了。

可以 用 [man\(1\)](#) 命令 `-k` ， 同 能 得有用的信息。例如：

```
# man -k Atheros
```

能得到一 包含特定 的手册（manual page）：

```
ath(4)          - Atheros IEEE 802.11 wireless network driver
ath_hal(4)      - Atheros Hardware Access Layer (HAL)
```

手 有一 硬件的配置清， 那 制定内核的 程就 得不那 困 了。

## 9.4. 内核 ， 子系 和模

在 一个制定的内核之前 三思一下 做的理由， 如果 是需要某个特定的硬件支持的， 那 很可能已 存在一个 成的模 了。

内核模 存放在目 `/boot/kernel` 中， 并能由 [kldload\(8\)](#) 命令加 入正在 行的内核。 基本上所有的内核

都有特定的模块和手册。比如，下面提到的 ath 无线以太网。在它的设备手册中有以下信息：

```
Alternatively, to load the driver as a module at boot time, place the
following line in man:loader.conf[5]:
```

```
if_ath_load="YES"
```

遵照示例，在 /boot/loader.conf 中加入 `if_ath_load="YES"` 能在机器启动的时候加载该模块。

某些情况下，没有相关的模块。通常是一些子系统非常重要的，比如，快速文件系统（FFS）就是一个内核必需的。同时的还有网络支持（INET）。不幸的是，分辨一个模块是否必需的唯一方法就是查看以下那个模块本身。



去除某个模块的支持或某个模块会非常容易得到一个坏掉的内核。例如，如果把 `ata(4)` 从内核配置文件中去掉，那一个使用 ATA 磁碟的系就可能就得不到引导，除非有在 loader.conf 中加。当无法确定的时候，删除一下那个模块并把它留在内核配置中。

## 9.5. 建立并安装一个定制的内核

首先内核构建做一个快速的。这里所提到的所有目录都在 /usr/src/sys 目录中；也可以通 /sys 来访问它。目录里的多子目录包含了内核的不同部分，但我所要完成的任何最重要的目录是 arch/conf，它将在里面定制的内核配置；以及 compile，编译过程中的文件将放置在这里。arch 表示 i386、amd64、ia64、powerpc、sparc64，或 pc98（在日本比较流行的一 PC 硬件分支）。在特定硬件架构目录中的文件只和特定的硬件有关；而其余代码是与机器无关的，所有已或将要移植并行 FreeBSD 的平台上都共享这些代码。文件目录是按照组织的，所支持的硬件、文件系统，以及可的部件通常都在它自己的目录中。

一章提供的例子假定使用 i386 架构的计算机。如果的情况不是，只需目录名作相应的调整即可。

如果的系中没有 /usr/src/sys 一个目录，说明没有安装内核源代码。安装它最的方法是通以 root 身份行 `sysinstall`，配置 Configure，然后是 Distributions、src，其中其中的 base 和 sys。如果不喜欢 sysinstall 并且有一“官方的”FreeBSD CDROM，也可以使用下列命令，从命令行来安装源代码：



```
# mount /cdrom
# mkdir -p /usr/src/sys
# ln -s /usr/src/sys /sys
# cat /cdrom/src/ssys.[a-d]* | tar -xzf -
# cat /cdrom/src/sbase.[a-d]* | tar -xzf -
```

接下来，进入 arch/conf 目录下面，定制 GENERIC 配置文件，并给这个文件起一个容易辨别的名称，它就是的内核名称。例如：

```
# cd /usr/src/sys/i386/conf
# cp GENERIC MYKERNEL
```

通常，`MYKERNEL` 名称是大写的，如果正在配置多台不同硬件的 FreeBSD 机器，  
用机器的域名来命名是非常好的主意。我把它命名 `MYKERNEL` 就是这个原因。

以



将内核配置文件直接保存在 `/usr/src` 可能不是一个好主意。如果遇到 `rm -rf /usr/src`，  
`/usr/src` 并重新初始化很可能是一个人的操作。一旦开始做这件事，可能几秒之后才会意识到  
会删除定制的内核配置文件。另外，也不要直接覆盖 `GENERIC`，因为下次更新代  
它会被覆盖，而你的修改也就随之丢失了。

也可以考虑把内核配置文件放到别的地方，然后再到 `i386` 目录中建立一个指向它的符号  
链接。

例如：

```
# cd /usr/src/sys/i386/conf
# mkdir /root/kernels
# cp GENERIC /root/kernels/MYKERNEL
# ln -s /root/kernels/MYKERNEL
```



必须以 `root` 身份运行这些和接下来命令，否则就会得到提示。

现在就可以用喜欢的文本编辑器来编辑 `MYKERNEL` 了。如果刚开始使用 FreeBSD，唯一可用的编辑器很可能是  
`vi`，它的使用比 `ed`，限于篇幅，这里不予介绍，可以在 [参考书目](#) 一章中看到很多相关书籍。不过，FreeBSD  
也提供了一个更好用的编辑器，它叫做 `ee`，对于新手来说，它很可能是一个不错的选择。  
可以修改配置文件中的注释以反映你的配置，或其他与 `GENERIC` 不同的地方。

如果你在 SunOS™ 或者其他 BSD 系下定制内核，那这个文件中的大部分将非常熟悉。如果你使用的是  
如 DOS 系的系统，那 `GENERIC` 配置文件看起来就非常困难，所以在下面的 [配置文件](#) 章节将慢慢地、仔细地  
进行介绍。



如果你和 FreeBSD project 进行了 [代码同步](#)，一定要在运行任何更新之前看  
`/usr/src/UPDATING`。这个文件中描述了更新你的代码中出现的重大问题或需要注意的地方。  
`/usr/src/UPDATING` 是和你的 FreeBSD 源代码同步的，因此能提供比手册更具  
时效性的新内容。

现在你已经有了内核的源代码了。

## Procedure: 编译内核

1. 进入 /usr/src 目录：

```
# cd /usr/src
```

2. 编译内核：

```
# make buildkernel KERNCONF=MYKERNEL
```

3. 安装新内核：

```
# make installkernel KERNCONF=MYKERNEL
```



使用这种方法编译内核，需要安装完整的 FreeBSD 源代码。

默认情况下，在源代码所定制的内核，全部 内核模块也会同时参与构建。如果你希望更快地升级内核，或者只希望源代码所需要的模块，可以在编译之前编辑 /etc/make.conf：

```
MODULES_OVERRIDE = linux acpi sound/sound sound/driver/ds1 ntfs
```



这个量的内容是所希望构建的模块列表。

```
WITHOUT_MODULES = linux acpi sound ntfs
```

这个量的内容是将不在源代码中编译的模块列表。 如果希望了解更多与构建内核有关的量，请参考 [make.conf\(5\)](#) 手册。

新内核将会被复制到 /boot/kernel 目录中成为 /boot/kernel/kernel 而旧的将被移到 /boot/kernel.old/kernel。在系统，然后用新的内核重启计算机。如果出现，后面的一些 [故障排除方法](#) 将帮助你脱困境。如果你的内核无法启动，请参考那一节。



其他与源代码相关的文件，如 [loader\(8\)](#) 及其配置，都放在 /boot。第三方或定制的模块也可以放在 /boot/kernel，但要注意保持模块和内核的同步是很重要的，否则会致不稳定和崩溃。

## 9.6. 配置文件

配置文件的格式是非常简单的。每一行都包括一个标识符，以及一个或多个参数。通常，大多数行都只包括一个参数。在 \# 之后的内容会被认为是注释而忽略掉。接下来几节，将以 GENERIC 中的序介绍所有标识符。如果需要与平台有关的标识符和值的列表，请参考与 GENERIC 文件在同一个目录中的那个

NOTES，而平台无的的，可以在 /usr/src/sys/conf/NOTES 到。

配置文件中可以使用 `include` 句。 个句能在内核配置文件中直接引用其他配置文件的内容，使得能使用小的、包含相对于存配置的而少所需的工作。 例如，如果只需 GENERIC 内核行少量定制，在其中添加几个程序和附加，只要相对于 GENERIC 的化就可以了：

```
include GENERIC
ident MYKERNEL

options      IPFIREWALL
options      DUMMYNET
options      IPFIREWALL_DEFAULT_TO_ACCEPT
options      IPDIVERT
```

多系管理会， 方法与先前从开始写配置文件的方法相比， 可以相当多的好：本地采用的配置文件只表与 GENERIC 内核的差， 在升的候往往就不需要做任何改，而新加入 GENERIC 的功能就会自加入到本地的内核，除非使用 `nooptions` 或 `nodevice` 句将其排除。 一章余下的部分将着重介典型的配置文件，以及内核和的作用。



如果需要一包含所有的文件，例如用于目的，以 `root` 身行下列命令：

```
# cd /usr/src/sys/i386/conf && make LINT
```

下面是一个 GENERIC 内核配置文件的例子， 它包括了一些需要解的注。 个例子和制的 /usr/src/sys/i386/conf/GENERIC 非常接近。

```
machine      i386
```

是机器的架，它只能是 `amd64`, `i386`, `ia64`, `pc98`, `powerpc`, 或 `sparc64` 中的一。

```
cpu          I486_CPU
cpu          I586_CPU
cpu          I686_CPU
```

上面的指定了系中所使用的 CPU 型。 可以使用多个 CPU 型 (例如，不指定是指定 `I586_CPU` 或 `I686_CPU`)。 然而于定制的内核， 最好能只指定使用的那 CPU。 如果于自己使用的 CPU 型没有把握，可以通看 /var/run/dmesg.boot 中的信息来了解。

```
ident        GENERIC
```

是内核的名字。 取一个自己的名字，例如取名叫 `MYKERNEL`，如果一直在按照前面的明做的。 放在 `ident` 后面的字符串在内核会示出来， 因此如果希望能容易区分常用的内核和定制的内核， 就采取不同的名字 (例如，想定制一个性的内核)。

```
#To statically compile in device wiring instead of /boot/device.hints
#hints          "GENERIC.hints"          # Default places to look for devices.
```

`device.hints(5)` 可以用来配置内核。在编译的时候 `loader(8)` 将会缺省位置 `/boot/devicehints`。使用 `hints` 就可以把一些 hints 静默到内核。就没有必要在 `/boot` 下建立 `devicehints`。

```
makeoptions      DEBUG=-g          # Build kernel with gdb(1) debug symbols
```

一般的 FreeBSD 编译，在编译的内核指定了 `-g` 选项，由于此选项将 `gcc(1)` 表示加入信息，因此会将符号也包含进来。

```
options          SCHED_ULE          # ULE scheduler
```

它是 FreeBSD 上使用的默认调度器。保留此选项。

```
options          PREEMPTION          # Enable kernel thread preemption
```

允许内核线程根据优先级的占用度。有助于改善交互性，并可以中断线程更早地运行，而无须等待。

```
options          INET                # InterNETworking
```

网络支持，即使你不打算联网，也保留它，大部分的程序至少需要回网络（就是和本机进行网络连接），所以强烈要求保留它。

```
options          INET6               # IPv6 communications protocols
```

将打开IPv6接口。

```
options          FFS                 # Berkeley Fast Filesystem
```

是最基本的硬盘文件系统，如果打算从本地硬盘，保留它。

```
options          SOFTUPDATES          # Enable FFS Soft Updates support
```

这个选项会启用内核中的 Soft Updates 支持，它会显著地提高磁盘的写入速度。尽管这个功能是由内核直接提供的，但仍然需要在磁盘上启用它。看看 `mount(8)` 的输出，以了解系统中的磁盘上是否已启用了 Soft Updates。如果没有看到 `soft-updates` 选项，需要使用 `tunefs(8)` (用于存系) 或 `newfs(8)` (用于新系) 命令来激活它。

```
options          UFS_ACL              # Support for access control lists
```

它会将用内核中的控制表的支持。它依赖于展属性以及 UFS2，以及在 [文件系统控制表](#) 中所介绍的那些特性。ACL 默认是用的，并且如果已在文件系统上使用了它特性，就不再去掉它，因为它会去掉文件的管理表，并以不可预期的方式改变受保护的文件的访问方式。

```
options          UFS_DIRHASH          # Improve performance on big directories
```

通常使用额外的内存，它可以加速在大目录上的磁盘操作。它在大型服务器和频繁使用的工作站上打它个，而在磁盘操作不是很重要的小型系统上它，比如防火墙。

```
options          MD_ROOT              # MD is a potential root device
```

它会将打以基于内存的虚拟磁盘作根的支持。

```
options          NFSCLIENT           # Network Filesystem Client
options          NFSSERVER            # Network Filesystem Server
options          NFS_ROOT             # NFS usable as /, requires NFSCLIENT
```

网络文件系统。如果不打算通过 TCP/IP 挂接 UNIX® 文件服务器的分区，就可以注释掉它。

```
options          MSDOSFS              # MSDOS Filesystem
```

MS-DOS® 文件系统。只要不打算在挂接由 DOS 格式化的硬盘分区，就可以把它注释掉。如前面所介绍的，在第一次挂接 DOS 分区，内核会自增加需要的模块。此外，[emulators/mtools](#) 软件提供了一个很方便的功能，通过它可以直接访问 DOS 而无需挂接或卸下它（而且也完全不需要 **MSDOSFS**）。

```
options          CD9660               # ISO 9660 Filesystem
```

用于 CDROM 的 ISO 9660 文件系统。如果没有 CDROM 驱动器或很少挂接光盘数据（因在首次使用数据 CD 时会自增加），就可以把它注释掉。音频 CD 并不需要它。

```
options          PROCFS               # Process filesystem (requires PSEUDofs)
```

进程文件系统。它是一个挂接在 /proc 的一个“假扮的”文件系统，其作用是允许像 **ps(1)** 的程序输出正在运行的进程的详细信息。多数情况下，并不需要使用 **PROCFS**，因为大多数管理和控制工具，已进行了一系列修改，使之不再依赖 **PROCFS**：默认安装的系统中并不会挂接任何文件系统。

```
options          PSEUDofs             # Pseudo-filesystem framework
```

如果希望使用 **PROCFS**，就必须加入 **PSEUDofs** 的支持。



```
options          GEOM_GPT          # GUID Partition Tables.
```

这个选项提供了在磁盘上使用大量的分区的能力。

```
options          COMPAT_43          # Compatible with BSD 4.3 [KEEP THIS!]
```

使系统兼容4.3BSD。不要去掉这一行，不然有些程序将无法正常运行。

```
options          COMPAT_FREEBSD4    # Compatible with FreeBSD4
```

如果希望支持在旧版 FreeBSD 上使用旧式接口的应用程序，就需要加入这一项。一般来说，推荐在所有的 i386™ 系统上使用这个选项，因为可能会用到一些旧的应用；到 5.X 才开始支持的平台，如 ia64 和 sparc64，不需要这个选项。

```
options          COMPAT_FREEBSD5    # Compatible with FreeBSD5
```

如果希望支持在 FreeBSD 5.X 版本上使用，且使用 FreeBSD 5.X 系统用接口的应用程序，加上这个选项。

```
options          COMPAT_FREEBSD6    # Compatible with FreeBSD6
```

如果希望支持在 FreeBSD 6.X 版本上使用，且使用 FreeBSD 6.X 系统用接口的应用程序，加上这个选项。

```
options          COMPAT_FREEBSD7    # Compatible with FreeBSD7
```

如果希望支持在 FreeBSD 8 以上版本的操作系统中运行在 FreeBSD 7.X 版本上使用，且使用 FreeBSD 7.X 系统用接口的应用程序，加上这个选项。

```
options          SCSI_DELAY=5000    # Delay (in ms) before probing SCSI
```

将内核在探测 SCSI 之前等待 5 秒。如果只有 IDE 硬盘器，就可以不管它，反之可能会希望降低这个数以加速进程。当然，如果做了之后 FreeBSD 在 SCSI 遇到，需要再把它改回去。

```
options          KTRACE              # ktrace(1) support
```

这个选项打开内核进程跟踪，在调试很有用。

```
options          SYSVSHM             # SYSV-style shared memory
```

提供 System V 共享内存(SHM)的支持，最常用到 SHM 的是 X Window 的 XSHM 延伸，不少程序相

程序会自行使用SHM来提供额外的速度。如果要使用X Window，最好加入一个。

```
options          SYSVMSG          # SYSV-style message queues
```

支持 System V 消息。只会在内核中加数百字的空间占用。

```
options          SYSVSEM          # SYSV-style semaphores
```

支持System V 信号量，不常用到，但只在kernel中占用几百个字的空间。



[ipcs\(1\)](#) 命令的 `-p` 可以显示出任何用到些 System V 机制的进程。

```
options          _KPOSIX_PRIORITY_SCHEDULING # POSIX P1003_1B real-time extensions
```

在 1993 年 POSIX® 添加的扩展。在 Ports Collection 中某些应用程序会用到些（比如StarOffice™）。

```
options          KBD_INSTALL_CDEV # install a CDEV entry in /dev
```

一个是在 /dev下建立点必需的。

```
options          ADAPTIVE_GIANT    # Giant mutex is adaptive.
```

内核全局（Giant）是一个互斥机制（休眠互斥体）的名字，它用于保护多内核源。在，它已成了一个无法接受的性能瓶颈，它已被越来越多地使用保护个源的代替。`ADAPTIVE_GIANT` 将使得内核全局作一个自旋。意味着，当有进程希望住内核全局互斥体，但互斥体已被一个 CPU 上的进程住的时候，它将自行，直到那个进程放止。一般情况下，一个进程将入休眠状态并等待下一次度。如果不是是否做的，一般打它。



注意在 FreeBSD 8.0-RELEASE 及以后的版本，所有的互斥体默认都是自旋的，除非在配置使用 `NO_ADAPTIVE_MUTEXES`，明确的指定非自旋。因此，内核全局（Giant）目前默认也是自旋的，而且 `ADAPTIVE_GIANT` 已从内核配置文件中移出。

```
device          apic              # I/O APIC
```

`apic` 将使用 I/O APIC 作中断送的能力。`apic` 可以被 UP 和 SMP 内核使用，但 SMP 内核必使用它。要支持多处理器，需要加上 `options SMP`。



只有在 i386 和 amd64 平台上才存在 `apic`，在其他硬件平台上不使用它。

```
device          eisa
```

如果系统的主机板上有EISA槽，加入一个槽。使用一个槽可以自动扫描并配置所有插在EISA槽上的槽。

```
device          pci
```

如果系统的主板有PCI槽，就加入一个槽。使用一个槽可以自动扫描PCI，并在PCI到ISA之间建立通路。

```
# Floppy drives
device          fdc
```

它是软盘控制器。

```
# ATA and ATAPI devices
device          ata
```

一个槽支持所有ATA和ATAPI槽。只要在内核中加入`device ata`，就可以让内核支持代计算机上的所有PCI ATA/ATAPI槽。

```
device          atadisk          # ATA disk drives
```

这个槽是使用ATAPI硬盘器时必须加入的槽。

```
device          ataraid          # ATA RAID drives
```

这个槽需要`device ata`，它用于ATA RAID槽。

```
device          atapicd          # ATAPI CDROM drives
```

这个是ATAPI CDROM槽器所必需的。

```
device          atapifd          # ATAPI floppy drives
```

这个是ATAPI槽器所必需的。

```
device          atapist          # ATAPI tape drives
```

这个是ATAPI磁机槽器所必需的。

```
options          ATA_STATIC_ID    # Static device numbering
```

指定控制器使用其静态的号；如果没有一个槽，会按地分配槽的号。

```
# SCSI Controllers
device      ahb      # EISA AHA1742 family
device      ahc      # AHA2940 and onboard AIC7xxx devices
options     AHC_REG_PRETTY_PRINT  # Print register bitfields in debug
                                           # output. Adds ~128k to driver.

device      ahd      # AHA39320/29320 and onboard AIC79xx devices
options     AHD_REG_PRETTY_PRINT  # Print register bitfields in debug
                                           # output. Adds ~215k to driver.

device      amd      # AMD 53C974 (Teckram DC-390(T))
device      isp      # Qlogic family
#device     ispfw     # Firmware for QLogic HBAs- normally a module
device      mpt      # LSI-Logic MPT-Fusion
#device     ncr      # NCR/Symbios Logic
device      sym      # NCR/Symbios Logic (newer chipsets + those of 'ncr')
device      trm      # Tekram DC395U/UW/F DC315U adapters

device      adv      # Advansys SCSI adapters
device      adw      # Advansys wide SCSI adapters
device      aha      # Adaptec 154x SCSI adapters
device      aic      # Adaptec 15[012]x SCSI adapters, AIC-6[23]60.
device      bt       # Buslogic/Mylex MultiMaster SCSI adapters

device      ncv      # NCR 53C500
device      nsp      # Workbit Ninja SCSI-3
device      stg      # TMC 18C30/18C50
```

SCSI控制器。可以注掉系中没有的。 如果只有IDE，可以把些一起掉。 \*\_REG\_PRETTY\_PRINT 的配置，是程序的。

```
# SCSI peripherals
device      scbus     # SCSI bus (required for SCSI)
device      ch        # SCSI media changers
device      da        # Direct Access (disks)
device      sa        # Sequential Access (tape etc)
device      cd        # CD
device      pass      # Passthrough device (direct SCSI access)
device      ses       # SCSI Environmental Services (and SAF-TE)
```

SCSI外。也可以像上面一操作。



目前系提供的 USB [umass\(4\)](#) 以及少量其它使用了 SCSI 子系， 尽管它并不是真的 SCSI 。因此，如果在内核配置使用了程序， 必不要除 SCSI 支持。

```
# RAID controllers interfaced to the SCSI subsystem
device      amr      # AMI MegaRAID
device      arcmsr   # Areca SATA II RAID
device      asr      # DPT SmartRAID V, VI and Adaptec SCSI RAID
device      ciss     # Compaq Smart RAID 5*
device      dpt      # DPT Smartcache III, IV - See NOTES for options
device      hptmv    # Highpoint RocketRAID 182x
device      rr232x   # Highpoint RocketRAID 232x
device      iir      # Intel Integrated RAID
device      ips      # IBM (Adaptec) ServeRAID
device      mly      # Mylex AcceleRAID/eXtremeRAID
device      twa      # 3ware 9000 series PATA/SATA RAID

# RAID controllers
device      aac      # Adaptec FSA RAID
device      aacp     # SCSI passthrough for aac (requires CAM)
device      ida      # Compaq Smart RAID
device      mfi      # LSI MegaRAID SAS
device      mlx      # Mylex DAC960 family
device      pst      # Promise Supertrak SX6000
device      twe      # 3ware ATA RAID
```

支持RAID控制器。如果没有这些，可以把它注释掉或是去掉。

```
# atkbd0 controls both the keyboard and the PS/2 mouse
device      atkbd    # AT keyboard controller
```

atkbd控制器（**atkbd**）提供AT输入以及PS/2指鼠的I/O服务。atkbd程序（**atkbd**）与PS/2鼠程序（**psm**）需要一个控制器，所以不要删除它。

```
device      atkbd    # AT keyboard
```

atkbd程序，与atkbd控制器一起使用，提供连接到AT控制器的AT 84与AT加型鼠的服务。

```
device      psm      # PS/2 mouse
```

如果鼠接到PS/2鼠端口，就使用一个psm程序。

```
device      kbdmux    # keyboard multiplexer
```

kbdmux多路器的基本支持。如果不打算使用多个鼠，可以放心地删除一行。

```
device      vga      # VGA video card driver
```

。。。。。

```
device          splash      # Splash screen and screen saver support
```

。。的 splash 画面！屏幕保护程序也需要。。。

```
# syscons is the default console driver, resembling an SCO console
device          sc
```

**sc** 是默认的控制台程序，类似 SCO 控制台。由于大部分全屏程序都通过类似 **termcap** 的终端数据函数控制控制台，因此无论使用哪个或与 **VT220** 兼容的 **vt** 都没有什么关系。如果在行控制台使用全屏程序生成，在登录后将 **TERM** 量置为 **scoansi**。

```
# Enable this for the pcvt (VT220 compatible) console driver
#device          vt
#options          XSERVER          # support for X server on a vt console
#options          FAT_CURSOR       # start with block cursor
```

是一个兼容 VT220 的控制台，它同时能向下兼容 VT100/102。在同 **sc** 硬件不兼容的一些基本上它能行的很好。当然，登录系把 **TERM** 量置为 **vt100** 或 **vt220**。此在接网上大量不同的机器也被证明非常有用，因此 **termcap** 或 **terminfo** 通常没有可用的 **sc** - 而 **vt100** 几乎平台都支持。

```
device          agp
```

如果的机器使用 AGP，把上面一行加入配置。将用 AGP，以及某些上的 AGP GART 支持。

```
# 源管理支持 (参 NOTES 了解更多)
#device          apm
```

高源管理支持。本有用，不在 **GENERIC** 里默认禁用。

```
# 加 i8254 的 挂起/恢复 支持。
device          pmtimer
```

用于源管理事件，例如 APM 和 ACPI 的。。。。。

```
# PCCARD (PCMCIA) support
# PCMCIA and cardbus bridge support
device          cbb              # cardbus (yenta) bridge
device          pccard           # PC Card (16-bit) bus
device          cardbus          # CardBus (32-bit) bus
```

PCMCIA支持。如果你使用膝上型计算机，你需要一个。

```
# Serial (COM) ports
device      sio          # 8250, 16[45]50 based serial ports
```

这些串口在 MS-DOS®/Windows® 的世界中称为 COM 口。



如果使用内置式的调制解调器，并占用 COM4 而还有一个串口在 COM2，你必须把调制解调器的 IRQ 改为 2（由于硬件的技术原因，IRQ2 = IRQ 9）才能它在 FreeBSD 中工作。如果有多口的串口，请参考 [sio\(4\)](#) 以了解需要在 /boot/device.hints 中进行的设置。某些（特别是基于 S3 芯片的）使用形如 0x\*2e8 的 IO 地址，而许多廉价的串口不能正确地址 16-位 IO 地址空间行解码，因此它们会产生冲突，并造成 COM4 无法使用。

每一个串口都需要有一个唯一的 IRQ（除非使用支持中断分享的串口），因此默认的 COM3 和 COM4 IRQ 是不能使用的。

```
# Parallel port
device      ppc
```

ISA-bus并行接口。

```
device      ppbus      # Parallel port bus (required)
```

提供并行口的支持。

```
device      lpt        # Printer
```

提供并口打印机的支持。



要使用并口打印机，就必须同时加入上面三行设置。

```
device      plip        # TCP/IP over parallel
```

它是并行网络接口的适配器。

```
device      ppi         # Parallel port interface device
```

普通用途的I/O ("geek port") + IEEE1284 I/O。

```
#device      vpo         # Requires scbus and da
```

它是Omega Zip适配器的。它要求scbus和da的支持。最好的运行效果是工作在EPP 1.9模式。

```
#device      puc
```

如果有由 [puc\(4\)](#) 支持的 "P" 串行或并行 PCI 卡，请去掉一行中的注释。

```
# PCI Ethernet NICs.
device      de      # DEC/Intel DC21x4x (Tulip)
device      em      # Intel PRO/1000 adapter Gigabit Ethernet Card
device      ixgb    # Intel PRO/10GbE Ethernet Card
device      txp     # 3Com 3cR990 (Typhoon)
device      vx      # 3Com 3c590, 3c595 (Vortex)
```

多个PCI网络适配器。注释或删除系中中没有的。

```
# PCI Ethernet NICs that use the common MII bus controller code.
# NOTE: Be sure to keep the 'device miibus' line in order to use these NICs!
device      miibus  # MII bus support
```

MIIS支持于一些PCI 10/100 Ethernet NIC来是必需的。

```
device      bce      # Broadcom BCM5706/BCM5708 Gigabit Ethernet
device      bfe      # Broadcom BCM440x 10/100 Ethernet
device      bge      # Broadcom BCM570xx Gigabit Ethernet
device      dc       # DEC/Intel 21143 and various workalikes
device      fxp      # Intel EtherExpress PRO/100B (82557, 82558)
device      lge      # Level 1 LXT1001 gigabit ethernet
device      msk      # Marvell/SysKonnect Yukon II Gigabit Ethernet
device      nge      # NatSemi DP83820 gigabit ethernet
device      nve      # nVidia nForce MCP on-board Ethernet Networking
device      pcn      # AMD Am79C97x PCI 10/100 (precedence over 'lnc')
device      re       # RealTek 8139C+/8169/8169S/8110S
device      rl       # RealTek 8129/8139
device      sf       # Adaptec AIC-6915 (Starfire)
device      sis      # Silicon Integrated Systems SiS 900/SiS 7016
device      sk       # SysKonnect SK-984x & SK-982x gigabit Ethernet
device      ste      # Sundance ST201 (D-Link DFE-550TX)
device      stge     # Sundance/Tamarack TC9021 gigabit Ethernet
device      ti       # Alteon Networks Tigon I/II gigabit Ethernet
device      tl       # Texas Instruments ThunderLAN
device      tx       # SMC EtherPower II (83c170 EPIC)
device      vge      # VIA VT612x gigabit ethernet
device      vr       # VIA Rhine, Rhine II
device      wb       # Winbond W89C840F
device      xl       # 3Com 3c90x (Boomerang, Cyclone)
```

使用MIIS控制器代卡的器。



```
# ISA Ethernet NICs.  pccard NICs included.
device      cs          # Crystal Semiconductor CS89x0 NIC
# 'device ed' requires 'device miibus'
device      ed          # NE[12]000, SMC Ultra, 3c503, DS8390 cards
device      ex          # Intel EtherExpress Pro/10 and Pro/10+
device      ep          # Etherlink III based cards
device      fe          # Fujitsu MB8696x based cards
device      ie          # EtherExpress 8/16, 3C507, StarLAN 10 etc.
device      lnc         # NE2100, NE32-VL Lance Ethernet cards
device      sn          # SMC's 9000 series of Ethernet chips
device      xe          # Xircom pccard Ethernet

# ISA devices that use the old ISA shims
#device      le
```

ISA 以太网。参 /usr/src/sys/i386/conf/NOTES 以了解于个程序能网的。

```
# Wireless NIC cards
device      wlan        # 802.11 support
```

通用 802.11 支持。行配置是无网所必需的。

```
device      wlan_wep    # 802.11 WEP support
device      wlan_ccmp   # 802.11 CCMP support
device      wlan_tkip   # 802.11 TKIP support
```

802.11 的加密支持。如果希望使用加密和 802.11i 安全，就需要些配置行。

```
device      an          # Aironet 4500/4800 802.11 wireless NICs.
device      ath         # Atheros pci/cardbus NIC's
device      ath_hal     # Atheros HAL (Hardware Access Layer)
device      ath_rate_sample # SampleRate tx rate control for ath
device      awi         # BayStack 660 and others
device      ral         # Ralink Technology RT2500 wireless NICs.
device      wi          # WaveLAN/Intersil/Symbol 802.11 wireless NICs.
#device     wl          # Older non 802.11 Wavelan wireless NIC.
```

用以支持多无网。

```
# Pseudo devices
device      loop        # Network loopback
```

是 TCP/IP 的通用回。如果 telnet 或 FTP 到 localhost (也就是 127.0.0.1) 将通个回到本机。个是必需的。

```
device random      # Entropy device
```

Cryptographically secure random number generator.

```
device ether        # Ethernet support
```

`ether` 只有在使用以太网时才需要。它包含了通用的以太网代码。

```
device sl           # Kernel SLIP
```

`sl` 用以提供 SLIP 支持。目前它几乎已完全被 PPP 取代了，因为后者更容易配置，而且更符合调制解调器之间的连接，并提供了更大的功能。

```
device ppp          # Kernel PPP
```

它用以提供内核的 PPP 支持，用于串行连接。也有以用户模式运行的 PPP 程序，使用 `tun` 并提供包括按需连接在内的更灵活的功能。

```
device tun          # Packet tunnel.
```

它会被用户模式的 PPP 程序用到。参考本节的 [PPP](#) 以了解更多的内容。

```
device pty          # Pseudo-ttys (telnet etc)
```

它是一个 "pseudo-terminal" 或模拟登录端口。它用来接收输入的 `telnet` 以及 `rlogin` 会话、`xterm`，以及一些其它程序如 Emacs 等。

```
device md           # Memory disks
```

内存设备。

```
device gif          # IPv6 and IPv4 tunneling
```

它支持了在 IPv4 上的 IPv6 隧道、IPv6 上的 IPv4 隧道、IPv4 上的 IPv4 隧道、以及 IPv6 上的 IPv6 隧道。`gif` 是 "自克隆" 的，它会根据需要自动建立接口点。

```
device faith        # IPv6-to-IPv4 relaying (translation)
```

这个设备能捕捉它的数据包，并把它通过 IPv4/IPv6 转换程序。

```
# The 'bpf' device enables the Berkeley Packet Filter.
# Be aware of the administrative consequences of enabling this!
# Note that 'bpf' is required for DHCP.
device    bpf                # Berkeley packet filter
```

它是 Berkeley 包过滤器。它允许网络接口被置于混杂模式，从而，截取广播网（例如，以太网）上的一个数据包。截取的数据可以保存到磁盘上，也可以使用 [tcpdump\(1\)](#) 程序来分析。



[bpf\(4\)](#) 它也被用于 [dhclient\(8\)](#) 来取默认路由器(网)的 IP 地址。如果使用 DHCP，就不要注释掉它。

```
# USB support
device    uhci                # UHCI PCI->USB interface
device    ohci                # OHCI PCI->USB interface
device    ehci                # EHCI PCI->USB interface (USB 2.0)
device    usb                 # USB Bus (required)
#device   udbp                # USB Double Bulk Pipe devices
device    ugen                # Generic
device    uhid                # Human Interface Devices
device    ukbd                # Keyboard
device    ulpt                # Printer
device    umass               # Disks/Mass storage - Requires scbus and da
device    ums                 # Mouse
device    ural                # Ralink Technology RT2500USB wireless NICs
device    urio                # Diamond Rio 500 MP3 player
device    uscanner            # Scanners
# USB Ethernet, requires mii
device    aue                 # ADMtek USB Ethernet
device    axe                 # ASIX Electronics USB Ethernet
device    cdce                # Generic USB over Ethernet
device    cue                 # CATC USB Ethernet
device    kue                 # Kawasaki LSI USB Ethernet
device    rue                 # RealTek RTL8150 USB Ethernet
```

支持各种 USB 设备。

```
# FireWire support
device    firewire            # FireWire bus code
device    sbp                 # SCSI over FireWire (Requires scbus and da)
device    fwe                 # Ethernet over FireWire (non-standard!)
```

支持各种 FireWire 设备。

要了解 FreeBSD 所支持的设备的其他情况，请参考 `/usr/src/sys/i386/conf/NOTES`。

### 9.6.1. 大内存支持(PAE)

大内存配置的机器需要超过 4 GB 的虚拟地址。 因为 4GB 的限制，Intel 在 Pentium® 及后续的 CPUs 上增加了 36 位物理地址的支持。

物理地址扩展 (PAE) 是 Intel® Pentium® Pro 和后续的 CPU 提供的一项允许将内存地址扩展到 64GB 的功能，FreeBSD 的所有最新版本均支持此功能，并通称 PAE 来引用这个能力。 因为 Intel 架构的限制，高于或低于 4GB 都没有什么区别，超过 4GB 的内存分配只是简单地添加到可用内存池中。

为了内核支持 PAE，只要添加下面一行到配置文件：

```
options             PAE
```



PAE 在 FreeBSD 里面只能在支持 Intel® IA-32 处理器。 同时，请注意，FreeBSD 的 PAE 支持没有很广泛的，和其他特定的特性相比只能当作是 beta 版。

PAE 在 FreeBSD 下有如下的一些限制：

- 程序不能接触大于 4GB 的 VM 空间。
- 没有使用 [bus\\_dma\(9\)](#) 接口的任何程序在打了 PAE 支持的内核中会导致数据损坏。 因为这个原因，PAE 内核配置文件会把所有在打了 PAE 的内核上不能工作的程序排除在外。
- 一些系统打了探测系统内存资源使用能力的功能，因此打了 PAE 支持，这些功能可能会被覆盖掉。 其中一个例子就是内核参数 `kern.maxvnodes`，它是控制内核能使用的最大 vnodes 数目的，建议重新调整它及其他类似参数到合适的值。
- 为了避免 KVA 的消耗，很有必要增加系统的内核虚拟地址，或者减少很耗系统资源的内核的总量（看上面）。`KVA_PAGES` 可以用来增加 KVA 空间。

为了稳定和性能，建议看 [tuning\(7\)](#) 手册。 [pae\(4\)](#) 手册包含 FreeBSD's PAE 支持的最新信息。

## 9.7. 如果出错

在定制一个内核时，可能会出错。它是：

**config** 失败：

如果 [config\(8\)](#) 在输出的内核描述失败，可能在某些地方引入了一小错误。 幸运的是，[config\(8\)](#) 会显示出它遇到错误的行号，你就能迅速地定位。 例如，如果你看到：

```
config: line 17: syntax error
```

可以通过与 GENERIC 或其他参考资料对比，来确定里面的错误是否写正。

**make** 失败：

如果 `make` 命令失败，它通常表示内核描述中产生了 [config\(8\)](#) 无法输出的错误。 同时，仔细检查配置，如果仍然不能解决，一封邮件到 [FreeBSD 一般邮件列表](#) 并附上你的内核配置，通常很快就能解决。

内核无法启动：

如果你的内核无法启动，或不兼容的内核，千万慌张！非常幸运的是，FreeBSD 有一个很好的机制帮助你从不兼容的内核恢复。在 FreeBSD 的加电器那里迅速地一下要启动的内核就可以了。当系统在引导菜单的 10 秒倒计时进入它，方法是输入 "Escape to a loader prompt" 输入，其符号是 `6`。输入 `unload kernel`，然后输入 `boot /boot/kernel.old/kernel`，或者其他任何一个可以正确引导的内核即可。当重新配置内核，保持一个已证明能正常工作的内核永远是一个好习惯。

当使用好的内核之后可以配置文件并重新启动它。比较有用的来源是 `/var/log/messages` 文件，它会记录每次成功启动所生成的所有内核消息。此外，`dmesg(8)` 命令也会显示每次启动生成的内核消息。



如果在启动内核时遇到麻烦，你必须保留一个 GENERIC 或已知可用的其他内核，并命名它的名字以免在下次启动时被覆盖。不要依赖 `kernel.old` 因为在安装新内核时，`kernel.old` 会被上次安装的那个可能不正常的内核覆盖掉。另外，尽快把可用的内核移动到 `/boot/kernel` 否则像 `ps(1)` 的命令可能无法正常工作。为了完成这一点，需要修改目录的名字：

```
# mv /boot/kernel /boot/kernel.bad
# mv /boot/kernel.good /boot/kernel
```

内核工作，但是 `ps(1)` 根本不工作

如果你安装了一个与系统中内建工具版本不同的内核，例如在 `-STABLE` 系统上安装了 `-CURRENT` 的内核，许多用于系统状态的工具如 `ps(1)` 和 `vmstat(8)` 都将无法正常使用。重新安装一个和内核版本一致的系。这也是为什么一般不鼓励使用与系统其他部分版本不同的内核的一个主要原因。

# Chapter 10. 打印

## 10.1. 概述

FreeBSD 可以支持多种的打印机，从最古老的点式打印机到最新的激光打印机以及它之间的所有型号的打印机，令所有的应用程序产生高质量的打印输出。

FreeBSD 也可以配置成网络打印服务器。它可以从包括 FreeBSD、Windows® 及 Mac OS® 在内的多种其他计算机上接收打印任务。FreeBSD 将保证打印任务之间不会相互干扰并一次性完成，而且能向机器或用户提交打印任务的情况并行并到其中用量最多的人，以及生成用于每个打印任务属于该用户的“账单”等等。

在看完本章后，你将知道：

- 配置 FreeBSD 后台打印。
- 安装打印服务器来特殊的打印任务做特殊的处理，包括把来的文本转换成打印机能理解的格式。
- 在打印输出上垂直或者横幅功能。
- 打印到连接在其他计算机上的打印机。
- 打印到直接连接在网络上的打印机。
- 控制打印机的限制，包括限制打印任务的大小和阻止某些用户打印。
- 查看打印机列表和使用情况。
- 解决打印故障。

在第二章之前， ：

- 知道如何配置并安装新内核 ([配置FreeBSD的内核](#))。

## 10.2. 介绍

为了在 FreeBSD 中使用打印机，需要首先配置好伯克利行式打印机后台打印系统即 LPD。它是 FreeBSD 的标准打印控制系统。本章介绍 LPD 后台打印系统，在接下来将介绍称 LPD，并且将指导完成其配置。

如果你已经熟悉了 LPD 或者其他后台打印系统，可以跳到 [配置后台打印系统](#) 部分。

LPD 完全控制一台计算机上的打印机。它做的事情：

- 它控制本地和连接在网络上的其他计算机上打印机的。
- 它允许用户提交要打印的文件；这些通常被称为任务。
- 它给打印机一个队列来防止多个用户在同一时刻用一台打印机。
- 它可以打印页眉(也叫做**banner**或者 **burst**)使用户可以轻松地从一堆打印输出中找到它打印的任务。
- 它来设置连接在串口上的打印机的通信参数。
- 它能通过网络将任务送到外一台计算机的 LPD 后台打印队列中。
- 它可以根据不同型号的打印机语言和打印机的性能进行特殊的过滤器来格式化任务。
- 它跟踪打印机的使用情况。

通配置文件 (/etc/printcap)和提供的特殊程序，可以使LPD系在多的打印机硬件上完成上面全部的或者一些子集的功能。

### 10.2.1. 为什么要用后台打印

如果是系唯一的用，可能会奇怪什么要在不需要控制，或者打印机使用后台打印心。它可以置成允直接打印机，但是是使用后台打印，因：

- LPD在后台打印任；不用被迫等待数据被完全副本到打印机的。
- LPD可以方便的通器任加上日期/ 的眉或者把一特殊的文件格式（比如TeX DVI 文件）成一打印机可以理解的格式。不必去手做些。
- 多提供打印功能的免和商程序想要和计算机上的的后台打印系通。通置后台打印系，将更松的支持其他以后要添加的或者有的件。

## 10.3. 基本置



从 FreeBSD 8.0 起，串口名的由 /dev/ttydN 或 /dev/ttyuN。FreeBSD 7.X 用将篇文的示例中的名改原先的子。

要想在 LPD后台打印系上使用打印机，需要置打印机硬件和 LPD件。个文描述了置：

- 参[打印机置](#)来了解接一个打印机，告 LPD与它通，并且打印文本到打印机。
- 参[高打印机置](#)来了解打印多特殊格式的文件，打印，通网打印，控制打印机的限，并且学会打印作。

### 10.3.1. 打印机置

部分解配置打印机硬件和 LPD使之与打印机配合。解的基知有：

- [硬件置](#)部分将解把一台打印机接到计算机的一个端口上。
- [件置](#)部分将解配置 LPD后台打印的配置 文件 (/etc/printcap)。

如果正在置一台通网接收数据来打印而不是通串口或者并口的打印机，参[使用网数据流界面的打印机](#)。

尽管部分叫“打印机置”，但是是相当。使打印机配合 LPD 后台打印系在计算机上正常是最的部分。一旦的打印机可以正常工作后，那些高，比如文和，是相当。

#### 10.3.1.1. 硬件置

部分述了打印机接到计算机的多途径。主要了多接口和接，有允 FreeBSD 与打印机通所需的内核配置。

如果已接好了的打印机而且已用它在外一个操作系下成功的打印，或可以跳到个部分[件置](#)。

##### 10.3.1.1.1. 端口和接

在所出售的在 PC 上使用的打印机通常至少有 以下三接口中的一个：

- 串口，也叫 RS-232 或者 COM 口，使用计算机上的串口来送数据到打印机。串口在计算机上已非常普遍，而且也非常容易到且容易制作。串口有`需要特殊的`，而且可能需要去配置`微有点儿`的通`。大多数 PC 的串口的最高速度只有 115200 bps，使得打印很大的像需要的很。`
- 并口 使用计算机上的并口来送数据到打印机。并口在计算机上也已非常普遍，而且速度高于 RS-232 串口。`非常容易到，但很手工制作。并口通常没有通，使得配置它相当。`

并口按打印机上的接`来命名也叫做 "Centronics"接口。`

- USB 接口，即通用串行`，可以到比并口和串口高很多的速度。其既又便宜。USB 用来打印比串口和并口更有，但 UNIX® 系不能很好的支持它。避免个的方法就是台像大多数打印机一的既有 USB 接口又有并口的打印机。`

一般来`并口只提供向通 (计算机到打印机)，而串口和 USB 可以提供双向通。新的并口 (EPP 和 ECP) 及打印机在使用了 IEEE-1284 准的之后，可以在 FreeBSD 下双向通。`

与打印机通`并口双向通通常由方法中的一来完成。第一个方法是使用 FreeBSD 写的可以通打印机使用的言与打印机通的程序。通常用在墨打印机上，且可以用来告剩余墨水多少和其他状信息。第二方法使用在支持 PostScript® 的打印机上。`

PostScript® 任`事上由程序送打印机；但它并不行打印而是直接将果返回计算机。PostScript® 也采取双向通来将打印中的告计算机，比如 PostScript® 程序中的或者打印机。些信息于用来也是非常有价。此外，最好的在支持 PostScript® 的打印机上方法需要双向通：打印机打印数 (打印机从出厂一共打印多少)，然后送用的任，之后再次打印数。将打印前后得到的个相就可以得到用要付多少。`

#### 10.3.1.1.2. 并口

用并口`接打印机需要用 Centronics 把打印机与计算机接起来。具体明指在打印机，计算机的明上，或者干脆个上面都有。`

住用的计算机上的个并口。第一个并口在 FreeBSD 上叫 `/dev/ppc0`；第二个叫 `/dev/ppc1`，依此推。打印机也用同的方法命名：`/dev/lpt0` 是接在第一个并口上的打印机，依此推。

#### 10.3.1.1.3. 串口

用串口`接打印机需要用 合的串口把打印机与计算机接起来。具体明指在打印机，计算机的明上有，或者同干脆个上面都有。`

如果`不定什儿的才是 " 合的串口 "，可以以下几不同的：`

- 制解器 一端的 一根引脚都直接接到一端 相的引脚上。`也叫做 "DTE-to-DCE" 。`
- 非制解器上 一端的有些引脚 是与一端相引脚直接接的，而有一些是交叉接的 (比如，`送数据引脚接到 接收数据引脚 )，有一些引脚直接在接儿内 短接。也叫做 "DTE-to-DTE" 。`
- 一些特殊的打印机需要的串口打印机 `，是一和非制解器似的，只是一些信号是送到了 一端，而不是直接在接儿内短路。`

当然，`得打印机置通参数。一般是通打印机面板上的按或者 DIP 行置。在计算机和打印机上都它所支持的最高 波特 (秒多少比特，有也叫 波特率) 的速率。7或者`



8个数据位； 0不校验， 偶校验或者奇校验； 001个或2个停止位。 0要0流量控制00： 无， XON/XOFF (也叫做 "in-band" 或 "0件") 流量控制。 0住0的0件配置中的参数也要0成上面的数0。

### 10.3.1.2. 0件0置

0部分描述了要使用 FreeBSD 系0中的 LPD 后台打印系0行打印所需的0件0置。

包括0几个00：

1. 在需要的0候配置内核来允00接 打印机的端口； [配置内核](#) 部分会告00 需要做什0。
2. 如果0使用并口， 0需要0置一下 并口的通0模式；[0置 并口通0模式](#) 部分会告00具体的 00。
3. 00操作系0是否能00送数据到打印机。 [00打印机 0机状况](#) 部分会告00要00 做。
4. 0 LPD 0置与打印机匹配的参数0 通0修改 /etc/printcap 0个文件来完成。 0章后面 的部分将 0解如何来完成0置。

#### 10.3.1.2.1. 配置内核

操作系0的内核0了使某些特殊00工作需要重新 00。 打印机所用的串口、 并口就属于那些特殊00。 因此， 可能需要 添加0串口或并口的支持， 如果内核并没有配置它0的0。

要想知道00在使用的内核是否支持串口， 0入：

```
# grep sioN /var/run/dmesg.boot
```

其中  $N$  是串口的 0号， 从00始。 如果0看到 0似下面的0出：

```
sio2 at port 0x3e8-0x3ef irq 5 on isa
sio2: type 16550A
```

00明00在使用的内核支持串口。

要想知道00在使用的内核是否支持并口， 0入：

```
# grep ppcN /var/run/dmesg.boot
```

其中  $N$  是并口的 0号， 同0从00始。 如果得到0似 下面的0出：

```
ppc0: <Parallel port> at port 0x378-0x37f irq 7 on isa0
ppc0: SMC-like chipset (ECP/EPP/PS2/NIBBLE) in COMPATIBLE mode
ppc0: FIFO with 16/16/8 bytes threshold
```

那000在使用的内核支持并口。

0可能必00了使操作系0支持0打印机需要的串口或 并口而 重新配置内核。

要增加串口的支持，参看 内核配置 部分。要增加并口的支持，除了参看 上面提到的那部分之外，还要参看下面的部分。

### 10.3.1.3. 设置并口的通信模式

在使用并口时，你可以选择 FreeBSD 用中断方式或是轮询方式来与打印机通信。在 FreeBSD 上，通用的打印机驱动 (`lpt(4)`) 使用 `ppbus(4)` 系统，它利用 `ppc(4)` 来控制端口芯片。

- 中断方式是 GENERIC 核心的默认方式。在这种方式下，操作系统占用一条中断请求来询问打印机是否已做好接收数据的准备。
- 轮询方式是操作系统反复不断的询问打印机是否做好接收数据的准备。当它返回就绪时，核心开始发送下面要发送的数据。

中断方式速度通常会快一些，但却占用了一条宝贵的中断请求。一些新出的 HP 打印机不能正常的工作在中断模式下，是由于一些定时的（没正确的理解）造成的。有些打印机需要使用轮询方式。你可以使用任何一种方式，只要它能正常工作就行。一些打印机即使在轮询模式下都可以工作，但在中断模式下会慢的要命。

你可以用以下两种方法设定通信模式：通过配置内核或者使用 `lptcontrol(8)` 个程序。

要通过配置内核的方法设置通信模式：

1. 修改内核配置文件。找到一个叫 `ppc0` 的条目。如果你想要设置的是第二个并口，那用 `ppc1` 代替。使用第三个并口的時候用 `ppc2` 代替，依此类推。

- 如果你希望使用中断通信模式，看看下面的配置：

```
hint.ppc.0.irq="N"
```

它在 `/boot/device.hints` 个文件中，其中 `N` 用正确的中断 号代替。同时，核心配置文件也必须包括 `ppc(4)` 的条目：

```
device ppc
```

- 如果你想要使用轮询方式，只需要把 `/boot/device.hints` 个文件中的下面一行删除掉：

```
hint.ppc.0.irq="N"
```

在 FreeBSD 下，有上面的方法并不能使并口工作在轮询方式。大多数情况是由于 `acpi(4)` 造成的，它可以自动检测到并口并将其挂接到系统上，但也因此，它控制着打印机端口的通信模式。你需要 `acpi(4)` 的配置来解决这个问题。

2. 保存文件。然后配置，建立，并安装你配置的内核，最后重新引导。参看 内核配置 一章来了解更多。

使用 `lptcontrol(8)` 设置通信模式：

1. 输入：

```
# lptcontrol -i -d /dev/lptN
```

将 `lptN` 置成中断方式。

2. 输入：

```
# lptcontrol -p -d /dev/lptN
```

将 `lptN` 置成并行方式。

可以把这些命令加入到 `/etc/rc.local` 个文件中，每次系统启动都会置成想要的方式。参 [lptcontrol\(8\)](#) 来得更多信息。

#### 10.3.1.4. 打印机通信

在置后台打印系之前，确保的计算机可以把数据送到打印机上。分独立打印机的通信和后台打印系会更。

我了一打印机，将送一些文本它。一个叫 [lptest\(1\)](#) 的程序能任工作，它可以打印机立即打印出程序它的字符：它在行打出可以打印的 96 个 ASCII 字符。

当我使用的是一台 PostScript®（或者以其他言基的）打印机，那需要更仔细的。一段小小的 PostScript® 程序足以完成的任，比如下面段程序：

```
%!PS
100 100 moveto 300 300 lineto stroke
310 310 moveto /Helvetica findfont 12 scalefont setfont
(Is this thing working?) show
showpage
```

可以把上面段 PostScript® 代写一个文件里，并且像下面部分的例子里那使用。



上面的小程序是 PostScript® 而不是惠普的 PCL 写的。由于 PCL 有其他打印机没有的大功能，比如它支持在打印文本的同特殊的命令，而 PostScript® 不能直接打印文本，所以需要打印机言行特殊的理。

##### 10.3.1.4.1. 并口打印机

部分内容将指 FreeBSD 是否可以与一台已接在并口上的打印机通。

要并口上的打印机：

1. 用 `su(1)` 命令到 `root` 用。

2. 送数据到打印机。

- 如果打印机可以直接打印文本，可以用 `lp(1)`。入：

```
# lp test > /dev/lptN
```

其中 *N* 是并口的号，从0始。

- 如果打印机支持 PostScript® 或其他打印机言，可以送一段小程序到打印机。入：

```
# cat > /dev/lptN
```

然后，一行一行地入入段程序。因在按下 行 或者 回 之后，一行就不能再修改了。当入完段程序之后，按 `CONTROL+D`，或者其他表示文件束的。

外一法，可以把段程序写在一个文件里，并入：

```
# cat file > /dev/lptN
```

其中 *file* 是包含要打印机程序的文件名。

之后，看到打印出了一些西。如果打印出的西看起来并不正，不要着急；我将在后面指如何解决。

#### 10.3.1.4.2. 串口打印机

部分将告如何 FreeBSD 是否可以与接在串口上的打印机通。

要接在串口上的打印机：

1. 通过 `su(1)` 命令以 `root` 用户。
2. 修改 `/etc/remote` 文件。添加下面内容：

```
printer:dv=/dev/port:br#bps-rate:pa=parity
```

其中 `port` 是串口的设备点 (`ttyu0`、`ttyu1`，等等)，`bps-rate` 是与打印机通信使用的速率，而 `parity` 是通信打印机要求的校验方法 (是 `even`、`odd`、`none`，或 `zero` 之一)。

这儿有一个串口打印机的例子，它接在第三个串口上，速度 19200 波特，不校验：

```
printer:dv=/dev/ttyu2:br#19200:pa=none
```

3. 用 `tip(1)` 连接打印机。输入：

```
# tip printer
```

如果没能成功，要再次修改 `/etc/remote` 文件，并且用 `/dev/cuaaN` 代替 `/dev/ttydN`。

4. 送数据到打印机。

- 如果打印机可以直接打印文本，用 `lptest(1)`。输入：

```
% $lptest
```

- 如果打印机支持 PostScript® 或者其他打印机语言，送一段小程序到打印机。一行一行的输入程序，必须非常仔细，因为像退格或者其他控制字符也可能会让打印机来有它的意义。同时，也需要按一个特殊的文件结束符，让打印机知道它已接收了整个程序。对于 PostScript® 打印机，按 `CONTROL+D`。

或者，同时也可以把程序存到一个文件里并输入：

```
% >file
```

其中 `file` 是包含要送程序的文件名。在 `tip(1)` 送完文件之后，按代表文件结束的键。

看到打印出了一些东西。如果它看起来并不正确也不要着急；我将在后面的章节中介紹如何解决。

### 10.3.1.5. 用后台打印：文件 `/etc/printcap`

目前，你的打印机已经准备好了，系统内核也与打印机联机而重新配置好 (如果需要的)，而且你已经可以送一些数据到打印机。现在，我要配置 LPD 来使其控制你的打印机。

配置 LPD 要修改 `/etc/printcap` 文件。由于 LPD 后台打印系统在每次使用后台打印的时候，都会取

一个文件，因此一个文件的修改会立即生效。

`printcap(5)` 一个文件的格式很怪。 可以用最喜欢的文本编辑器来修改 `/etc/printcap` 一个文件。 格式和其他的像 `/usr/shared/misc/termcap` 和 `/etc/remote` 文件是一样的。 要得到关于格式的更多信息， 参看手册 `cgetent(3)`。

后台打印配置包括下面的几项：

1. 打印机起一个名字（和使用的名字）， 然后把它写入文件 `/etc/printcap`； 参看 [如何打印机命名](#) 一章来得到更多的关于起名的帮助。
2. 通加 `sh` 掉（它默认是用的）； 参看 [如何禁用](#) 部分来得到更多信息。
3. 建立一个后台打印队列的目录， 并且通 `sd` 目录指定它的位置； 可参看 [建后台打印队列](#) 一章了解更多信息。
4. 在 `/dev` 下置打印机点， 并且在写在 `/etc/printcap` 文件中 `lp` 目录里； 参看 [打印机](#) 部分可以得到更多信息。 此外， 如果打印机接在串口上， 通参数的位置需要写在 `ms#` 中。 些参数在 [配置后台打印通参数](#) 在前面已。
5. 安装文本器； 情参看 [安装文本器](#) 小。
6. 用 `lpr(1)` 命令来置。 想得到更多信息可以参看 和 [故障排除](#) 部分。



使用打印机言的打印机， 如 PostScript® 打印机， 通常是不能直接打印文本的。 前面提到， 并且将在后面行介绍的置方法， 均假定正在安装只能打印它能的文件格式的打印机。

通常会希望直接在系提供的打印机上打印文本。 采用 LPD 接口的程序也通常是。 如果正在安装一台打印机， 并且希望它不能打印使用它支持的打印机言的任何 而且 能打印文本的任何的， 那烈建在上面提到的置的上加一： 安装从自文本到 PostScript®（或者其他打印机言）的程序。 更多的， 参看 [在 PostScript® 打印机上打印文本](#)。

#### 10.3.1.5.1. 打印机的命名

第一（） 就是打印机起一个名字。 是按功能起名字是干脆起个古怪的名字都没有系， 因可以打印机置多的名。

在 `/etc/printcap` 里至少有一个打印机必指定， 名是 `lp`。 是默认的打印机名。 如果用既没有 `PRINTER` 境量， 也没有在任何 LPD 命令的命令行中指定打印机名， `lp` 将是默认要使用的打印机。

有， 我通常把最后一个名置成能完全描述打印机的名字， 包括厂家和型号。

一旦好了名字或者一些名， 把它放文件 `/etc/printcap` 里。 打印机的名字从最左的一列写起。 用杠来隔个名， 并且在最后一个名后面加上一个冒号。

在下面的例子中， 我从一个基本的 `/etc/printcap` 始， 它只定了台打印机（一台 Diablo 630 行式打印机和一台 Panasonic KX-P4455 PostScript® 激光打印机）：

```
#
# /etc/printcap for host rose
#
rattan|line|diablo|lp|Diablo 630 Line Printer:

bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:
```

在这个例子中，第一台打印机被命名 `rattan` 并且配置了 `line`, `diablo`, `lp`, 和 `Diablo 630 Line Printer` 几个名。因为它被配置了 `lp` 一个名，所以它是默认打印机。第二台被命名 `bamboo`，并且配置了 `ps`, `PS`, `S`, `panasonic`, 和 `Panasonic KX-P4455 PostScript v51.4` 几个名。

#### 10.3.1.5.2. 不打印

LPD 后台打印系统默认会打印任何打印的东西。这些东西包含了送给任何用的，送给任何的计算，任何的名字，并用大写字母打出。但不幸的是，所有这些外的文本，都会因在打印机进行最初的配置排除故障来困，所以我将先不打印。

要停打印，打印机的加 `sh`，在 `/etc/printcap` 文件中。这儿有一个 `/etc/printcap` 文件中使用 `sh` 的例子：

```
#
# /etc/printcap for host rose - no header pages anywhere
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
    :sh:

bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
    :sh:
```

注意我的正格式：第一行从最左一列开始，而后的每一行用 `TAB` 一次。一行写不下需要换行，在行前打一个反斜杠。

#### 10.3.1.5.3. 建立后台打印目录

下一配置就是要建立一个后台打印目录，也就是在打印任务最完成之前用于存放某些任务的目录，这个目录中也会存放后台打印系统用到的其他一些文件。

由于后台打印目录的容量本，通常把这些目录安排在 `/var/spool` 下。也没有必要去后台打印目录里的内容。重新建立它只要使用 `mkdir(1)` 命令。

通常，我将目录名起成和打印机一样的名字，像下面：

```
# mkdir /var/spool/printer-name
```

然而，如果有很多网络打印机，可能想要把些后台打印的目录放在一个独立的目录使用 LPD 打印而准的目录里。我将用我的两台打印机 `rattan` 和 `bamboo` 作例子：

```
# mkdir /var/spool/lpd
# mkdir /var/spool/lpd/rattan
# mkdir /var/spool/lpd/bamboo
```

如果担心用任何人的保密性，可能会希望保护相关的后台打印目录，使之不能被其他用户访问。后台打印的目录的属主是 `daemon` 用户，而 `daemon` 用户和 `daemon` 没有写入和搜索的权限，但其他用户没有。接下来用我们的后台打印机作例子：



```
# chown daemon:daemon /var/spool/lpd/rattan
# chown daemon:daemon /var/spool/lpd/bamboo
# chmod 770 /var/spool/lpd/rattan
# chmod 770 /var/spool/lpd/bamboo
```

最后，需要通 `/etc/printcap` 文件告诉 LPD 一些目录。可以用 `sd` 来指定后台打印目录的路径：

```
#
# /etc/printcap for host rose - added spooling directories
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
      :sh:sd=/var/spool/lpd/rattan:

bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
      :sh:sd=/var/spool/lpd/bamboo:
```

注意打印机的名字要从第 1 列开始，其他行都要用 TAB 缩进一次，写不需要行在最后加上反斜杠。

如果没用 `sd` 指定后台打印目录，后台打印系统会将 `/var/spool/lpd` 目录作为默认目录。

#### 10.3.1.5.4. 打印机

在 [Hardware Setup](#) 一章中，我明白了 FreeBSD 与打印机通信将使用哪个端口和 `/dev` 目录下的点。我要告诉 LPD 一些信息。当后台打印系统有任何需要打印，它将启动程序（发送数据到打印机）打印指定的点。

用 `lp` 在 `/etc/printcap` 里列出 `/dev` 下的点。

在我们的例子中，假设打印机 `rattan` 在第一个并口上，打印机 `bamboo` 在第六个串口上；下面是要加到 `/etc/printcap` 文件里的内容：



```
#
# /etc/printcap for host rose - identified what devices to use
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
    :sh:sd=/var/spool/lpd/rattan:\
    :lp=/dev/lpt0:

bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
    :sh:sd=/var/spool/lpd/bamboo:\
    :lp=/dev/ttyu5:
```

如果没在的 `/etc/printcap` 文件中用 `lp` 指定点，LPD 将默使用 `/dev/lp`。 `/dev/lp` 目前在 FreeBSD 中不存在。

如果正在安装的打印机是接在 并口上的，跳到 [安装文本 器](#) 章。如果不是的，是最好按下面介的做。

#### 10.3.1.5.5. 配置后台打印通参数

于在串口上的打印机，LPD 可以送数据到打印机的程序置好波特率，校，和其他串口通参数。是有利的，因：

- 它可以只需的修改 `/etc/printcap` 就能不同的通参数；并不需要去重新器程序。
- 它使得后台打印系可以在多台有不同串口通置的打印机上使用相同的器程序。

下面个 `/etc/printcap` 中用 `lp` 来控制列出个的串口通参数：

#### `br#bps-rate`

置的通速度 `bps-rate`，里 `bps-rate` 可以 50, 75, 110, 134, 150, 200, 300, 600, 1200, 1800, 2400, 4800, 9600, 19200, 38400, 57600, or 115200 比特秒。

#### `ms#stty-mode`

置已打的中端的。 `stty(1)` 将述可用的。

当 LPD 打用 `lp` 指定的，它会 将的特性置成在 `ms#` 后指定的那。特是 `parenb`, `parodd`, `cs5`, `cs6`, `cs7`, `cs8`, `cstopb`, `crtsets`, 和 `ixon` 些模式，它在 `stty(1)` 手册中有明。

我个例子来添加我在第6个串口上的 打印机。我将波特38400。至于模式，我将用 `-parenb` 置成不校，用 `cs8` 置成8位字符，用 `clocal` 置成不要制解器控制，用 `crtsets` 置成硬件流量控制：

```
bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
    :sh:sd=/var/spool/lpd/bamboo:\
    :lp=/dev/ttyu5:ms#-parenb cs8 clocal crtsets:
```

#### 10.3.1.5.6. 安装文本器

我在准告 LPD 使用什文本器 打印机送任。文本器，也叫 入器，是一个 在 LPD 有一个任要 打印机行的程序。当 LPD 打印机行文本器，它置器的 准入要打印机的任

，而标准输出用 `lp` 指定的打印机。过滤器先从标准输入取任，打印机行一些，并将结果写到标准输出，些结果 将被打印。想得到更多关于文本过滤器的信息，过滤器。

于的打印机置，文本过滤器可以是一段行 `/bin/cat` 的 shell 脚本来送任到打印机。FreeBSD 提供了一个叫做 `lpf` 的器，它可以理退格和下来使那些可能不能很好理字符流的打印机正常工作。而且，当然，可以用任何其他的想用的程序。`lpf` 器在 [lpf: 一个文本器](#) 将有描述。

首先，我来写一段叫做 `/usr/local/libexec/if-simple` 的 shell 脚本作文本器。用熟悉的文本器将下面的内容放个文件：

```
#!/bin/sh
#
# if-simple - Simple text input filter for lpd
# Installed in /usr/local/libexec/if-simple
#
# Simply copies stdin to stdout. Ignores all filter arguments.

/bin/cat && exit 0
exit 2
```

使个文件可以被行：

```
# chmod 555 /usr/local/libexec/if-simple
```

然后用 `if` 在 `/etc/printcap` 里告 LPD 使用个脚本。我将仍然一直作例子的台打印机在 `/etc/printcap` 里加个：

```
#
# /etc/printcap for host rose - added text filter
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
    :sh:sd=/var/spool/lpd/rattan:\ :lp=/dev/lpt0:\
    :if=/usr/local/libexec/if-simple:

bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
    :sh:sd=/var/spool/lpd/bamboo:\
    :lp=/dev/ttyu5:ms#-parenb cs8 clocal crtscs:\
    :if=/usr/local/libexec/if-simple:
```



`if-simple` 脚本的副本可以在 `/usr/shared/examples/printing` 目中到。

#### 10.3.1.5.7. 器 LPD

`lpd(8)` 在 `/etc/rc` 中被行，它是否被行由 `lpd_enable` 个量控制。个量默是 `NO`。如果没有修改，那加行：

```
lpd_enable="YES"
```

到 `/etc/rc.conf` 文件当中，然后既可以重启的机器，也可以直接运行 `lpd(8)`。

```
# lpd
```

#### 10.3.1.5.8. 测试

现在您已经基本完成了 LPD 的基本配置。但不幸的是，这不是庆祝的时候，因为我需要配置并且修正所有的配置。要配置，请打印一些东西。要用 LPD 系统打印，您可以使用 `lpr(1)` 命令，它可以提交一个任意的文件来打印。

您可以结合使用 `lpr(1)` 和 `lpctest(1)` 程序，在您的打印机上测试并生成一些测试文本。

要测试 LPD 配置：

输入：

```
# lpctest 20 5 | lpr -Pprinter-name
```

其中 `printer-name` 是在 `/etc/printcap` 中指定的打印机的一个名字（或者一个别名）。要默认打印机，输入 `lpr(1)` 不带任何 `-P` 选项。同时，如果您正在使用一台使用 PostScript® 的打印机，请发送一个 PostScript® 程序到打印机而不是使用 `lpctest(1)`。您可以把程序放在一个文件里，然后输入：`lpr file`。

在一台 PostScript® 打印机上，您将得到那段程序的输出。而如果您使用的 `lpctest(1)`，您将得到的输出看起来像下面这样：

```
!"#$%&'()*+,-./01234
"# $%&'()*+,-./012345
#$%&'()*+,-./0123456
$%&'()*+,-./01234567
%&'()*+,-./012345678
```

要更改您的打印机，请下载一些大的程序（基于特定语言的打印机）或者运行 `lpctest(1)` 并使用不同的参数。比如，`lpctest 80 60` 将生成 60 行每行 80 个字符。

如果打印机不能工作，参考 [故障排除](#)。

## 10.4. 高级配置



从 FreeBSD 8.0 起，串口设备的名称由 `/dev/ttydN` 和 `/dev/ttyuN`。FreeBSD 7.X 用户将本文档的示例中的名称改回原先的样子。

这部分将描述用来打印特殊格式文件，页眉，通过网络打印，以及打印机使用限制和选项。

### 10.4.1. 过滤器

尽管 LPD 管理网络，任排， 控制， 和打印的其他方面， 但大部分 工作是由 过滤器。过滤器是一 与打印机通并且理依和特殊需要的 程序。 在打印机置里， 我安装了一个文本过滤器 - 一个 可以用在大多数打印机上的的过滤器 (安装文本过滤器)。

然而，了行格式， 打印， 特殊的打印机， 等等， 需要明白过滤器是工作的。在根本上过滤器理些方面。但坏消息是大多数候 必自己提供过滤器。好消息是很多过滤器通常都已有了；当没有的候， 它通常也是很好写的。

FreeBSD 也提供了一个过滤器， /usr/libexec/lpr/lpf， 可以大多数可以打印文本的打印机工作。（它理文件里的退格和跳格，并且行， 但基本就是它所有能做的了。） 里有几个过滤器和过滤器件在 FreeBSD Ports Collection 里。

是在里将到的内容：

- 在 [过滤器是如何工作的](#) 小中将介在打印程中过滤器的作用。如果希望了解在 LPD 使用过滤器，在 "幕后" 生的事情，便一小。了解些知能助在打印机安装过滤器更快地排可能会遇到的各。
- LPD 假定任何打印机在默状下均能打印文本内容。于不能直接打印文本的 PostScript® 打印机 (以及其他基于打印言的打印机) 而言会来。在 [在 PostScript® 打印机上使用文本任](#) 中将会介如何解决个的方法。如果使用 PostScript® 打印机，就内容。
- PostScript® 于多程序来都是一个非常受欢迎的出格式。一些人甚至直接写 PostScript® 代。但不幸的是，PostScript® 打印机非常昂。模 [PostScript® 在非 PostScript® 打印机上](#) 将告 一修改打印机的文本过滤器，使得一台非 PostScript® 打印机接受并打印 PostScript® 数据。如果没有 PostScript® 打印机，那一个小。
- [过滤器](#) 描述了一个自把指定格式文件，比如像或排版数据，成打印机可以理解的格式的方法。在了之后，就配置打印机，用可以用 `lpr -t` 来打印 troff 数据、用 `lpr -d` 来打印 TeX DVI 数据，或用 `lpr -v` 来打印光像数据等工作了。建。
- [出 过滤器](#) 描述了个不是常使用的 LPD: 的功能 - 出过滤器。除非要打印眉 ( [眉](#) ), 或可以完全跳。
- `lpf`: [一个文本过滤器](#) 描述了 `lpf`， 一个 FreeBSD 自的相当完整而又的文本过滤器，可以使用在行式打印机(和那些担当行式打印机功能的激光打印机)上。如果需要一个快速的方法来打印机打印文本的工作量，或者有一台遇到退格字符就冒的打印机，考 `lpf`。



可以在 /usr/shared/examples/printing 目中到下面将提到的那些脚本的副本。

#### 10.4.1.1. 过滤器是工作的

前面，过滤器是一个被 LPD ，用来理与打印机通程中依的部分的可行程序。

当 LPD 想要打印一个任中的文件，它一个过滤器 程序。它把要打印的文件置成过滤器的准入， 准出 置成打印机， 并且把信息定向到 日志文件 (在 `lf` 里指定，默在 /etc/printcap， 或者 /dev/console 文件里)。

过滤器被 LPD ，并且过滤器的参数依于 /etc/printcap 文件中所列出的和用任用 `lpr(1)` 命令所指定的。例如，如果用入 `lpr -t`，LPD 会 troff 过滤器，即在目打印机的 `tf` 里所列出的过滤器。如果用想要打印文本，它将会 `if` 过滤器 (是通常的情况：参 [出过滤器](#) 来得到)。

在 `/etc/printcap` 文件中，`□` 可以指定三样东西：

- The 文本过滤器，在 LPD 文档中也叫做 输入过滤器，处理 常例的文本打印。可以把它想象成默认过滤器。LPD 假定一台打印机默认情况下都可以打印文本，而文本过滤器的任务就是来规定退格、跳格，或者其他在某台打印机上容易出现的特殊字符。如果它所在的系统打印机的使用情况不同，那文本过滤器也必须打印的页数不同，通常是根据打印的行数和打印机在页上能打印的行数不同算得出。文本过滤器的命令：

```
filter-name [-c] -w width -l length -i indent -n login -h host acct-file
```

这里

**-c**

当使用 `lpr -l` 命令提交输出

**width**

这里取在 `/etc/printcap` 文件中指定的 **pw** (页宽) 的值，默认为 132。

**length**

这里取的 **pl** (页长) 的值，默认为 66

**indent**

这里是来自 `lpr -i` 命令的页量，默认为 0

**login**

这里是在打印文件的用户名

**host**

这里是提交打印任务的主机名

**acct-file**

这里是来自 **af** 变量中指定的用于页的文件名。

- 过滤器的功能是，将特定格式的文件转换成打印机能识别并打印的格式。例如，ditroff 格式的排版数据就是无法直接打印的，但可以安装一个过滤器来将 ditroff 文件转换成一台打印机可以识别和打印的形式。参看 过滤器 一章来了解更多。如果需要打印不同行，那过滤器也必须完成页工作。过滤器的命令：

```
filter-name -x pixel-width -y pixel-height -n login -h host acct-file
```

其中 **pixel-width** 的值来自 **px** 页 (默认为 0)，而 **pixel-height** 的值来自 **py** 页 (默认为 0)。

- 输出过滤器在没有文本过滤器，或者不被打印使用。就我的而言，输出过滤器是很少用到的。在 输出过滤器 中会介绍它。输出过滤器的命令行只有 3 个参数：**filter-name** -w width -l length

它的作用与文本过滤器的 **-w** 和 **-l** 参数是一样的。

过滤器也会在退出时输出下面的几行退出状态：

**exit 0**

过滤器已成功打印了文件。

## exit 1

过滤器打印失败了，但希望 LPD 试着再打印一次。如果过滤器返回了 0 个状态，LPD 将重新配置过滤器。

## exit 2

过滤器打印失败并且不希望 LPD 重试。这种情况下 LPD 会放一个文件。

文本过滤器随 FreeBSD 一起发布，文件名 `/usr/libexec/lpr/lpf`，它利用 `lp` 和 `lpd` 参数来决定何时发送指令，并提供位打印的方法。它使用登录名、主机名，和文件参数来生成输出。

如果你想写过滤器，要注意它是否是和 LPD 兼容。如果兼容的，它必须支持前面提到的那些参数。如果你打算写普通的过滤器程序，它同样需要使之支持前面那些参数和退出状态。

### 10.4.1.2. 在 PostScript® 打印机上打印文本

如果你是一台计算机和 PostScript®（或其他语言的）打印机的唯一用户，而且你不打算发送文本到打印机，并因此不打算从应用程序直接将文本到打印机的，就完全不需要再关心它的内容了。

但是，如果打印机同样需要接收 PostScript® 和文本的任务，就需要对打印机进行配置了。要完成这项工作，我需要一个文本过滤器来过滤任务是文本的还是 PostScript® 格式的。所有 PostScript® 的任务必须以 `%!`（其他打印机语言参照打印机的文档）开始。如果任务的第一个字符是 `%`，就代表它是 PostScript® 格式的，并且可以直接忽略任务剩余的部分。如果任务的第一个字符不是 `%`，那过滤器将把文本转换成 PostScript® 并打印结果。

我该如何去做？

如果你有一台串口打印机，一个好办法就是安装 `lprps`。`lprps` 是一个可以与打印机进行双向通信 PostScript® 打印机过滤器。它用打印机传来的信息来更新打印机的状态文件，所以用户和管理员可以准确地看到打印机在什么状态（比如 **缺墨** 或者 **满**）。但更重要的是，它包含了一个叫做 `psif` 的程序，它可以接收到的文件是否是文本的，并且将使用 `textps` 命令（也是由 `lprps` 提供的程序）将文本到 PostScript®。然后它用 `lprps` 将任务送到打印机。

`lprps` 可以在 FreeBSD Ports Collection（参见 [The Ports Collection](#)）中找到。你可以根据桌面的尺寸安装 `print/lprps-a4` 和 `print/lprps-letter`。在安装了 `lprps` 之后，只需指定 `psif` 这个程序的路径，它也是包含在 `lprps` 中的一个程序。如果你已用 `ports` 安装好了 `lprps`，将下面的内容添加到 `/etc/printcap` 文件中 PostScript® 打印机的部分中：

```
:if=/usr/local/libexec/psif:
```

同样需要指定 `rw` 来告诉 LPD 使用读写模式打印打印机。

如果你有一台并口的 PostScript® 打印机（因此不能与打印机进行 `lprps` 需要的双向通信），可以使用下面一段 shell 脚本来充当文本过滤器：

```
#!/bin/sh
#
# psif - Print PostScript or plain text on a PostScript printer
# Script version; NOT the version that comes with lprps
# Installed in /usr/local/libexec/psif
#

IFS="" read -r first_line
first_two_chars=`expr "$first_line" : '\(..\)'`

if [ "$first_two_chars" = "%!" ]; then
    #
    # PostScript job, print it.
    #
    echo "$first_line" && cat && printf "\004" && exit 0
    exit 2
else
    #
    # Plain text, convert it, then print it.
    #
    ( echo "$first_line"; cat ) | /usr/local/bin/textps && printf "\004" && exit 0
    exit 2
fi
```

在上面的脚本中，`textps` 命令是一个独立安装的程序用来将文本转换成 PostScript®。你可以使用任何喜欢的文本到 PostScript® 的程序。FreeBSD Ports Collection ([Ports Collection](#)) 中包含了一个功能非常完整的文本到 PostScript® 的程序，它叫做 `a2ps`。

#### 10.4.1.3. 模拟 PostScript® 在非 PostScript® 打印机上

PostScript® 是高质量排版和打印事实上的标准。而 PostScript® 也是一个昂贵的标准。幸好，Aladdin 开发了一个和 PostScript® 类似的叫做 Ghostscript 的程序可以用在 FreeBSD 上。Ghostscript 可以取大多数 PostScript® 的文件并处理其中的界面交多，包括多品牌的非 PostScript® 打印机。通过安装 Ghostscript 并使用一个特殊的文本处理器，可以使一台非 PostScript® 打印机用起来就像真的 PostScript® 打印机一样。

Ghostscript 被收录在 FreeBSD Ports Collection 中，有多可用的版本，比常用的版本是 [print/ghostscript-gpl](#)。

要模拟 PostScript®，文本处理器要决定是否要打印一个 PostScript® 文件。如果不是，那处理器将直接将文件送到打印机；否则，它会用 Ghostscript 先将文件转换成打印机可以理解的格式。

这里有一个例子：下面的脚本是一个 Hewlett Packard DeskJet 500 打印机的文本处理器。对于其他打印机，替换 `gs` (Ghostscript) 命令中的 `-sDEVICE` 参数就可以了。（输入 `gs -h` 来得当前安装的 Ghostscript 所支持的列表。）

```
#!/bin/sh
#
# ifhp - Print Ghostscript-simulated PostScript on a DeskJet 500
# Installed in /usr/local/libexec/ifhp
#
# Treat LF as CR+LF (to avoid the "staircase effect" on HP/PCL
# printers):
#
printf "\033&k2G" || exit 2

#
# Read first two characters of the file
#
IFS="" read -r first_line
first_two_chars=`expr "$first_line" : '\(..\)'`

if [ "$first_two_chars" = "%!" ]; then
    #
    # It is PostScript; use Ghostscript to scan-convert and print it.
    #
    /usr/local/bin/gs -dSAFER -dNOPAUSE -q -sDEVICE=djet500 \
        -sOutputFile=- - && exit 0
else
    #
    # Plain text or HP/PCL, so just print it directly; print a form feed
    # at the end to eject the last page.
    #
    echo "$first_line" && cat && printf "\033&l0H" &&
exit 0
fi

exit 2
```

最后，需要告知 LPD 所使用的过滤器，通过 `if` 完成：

```
:if=/usr/local/libexec/ifhp:
```

可以输入 `lpr plain.text` 和 `lpr whatever.ps`，它都可以成功打印。

#### 10.4.1.4. 过滤器

在完成了 [打印机配置](#) 中所描述的内容之后，一件事 恐怕就是喜欢的格式的文件安装过滤器了（除了 ASCII 文本）。

##### 10.4.1.4.1. 什么安装过滤器？

过滤器使打印多格式的文件很容易。比如，假如我大量使用 TeX 排版系，并且有一台 PostScript® 打印机。次从 TeX 生成一个 DVI 文件，我都不能直接打印它直到我将 DVI 文件成 PostScript®。



的命令是下面的子:

```
% dvips seaweed-analysis.dvi
% lpr seaweed-analysis.ps
```

通安装 DVI 文件的器, 我可以跳次手一, 而 LPD 来完成个。在, 次要打印 DVI 文件, 我只需要一就可以打印它:

```
% lpr -d seaweed-analysis.dvi
```

我要 LPD DVI 文件是通指定 -d 完成的。格式和 列出了所有的。

于想要打印机支持的, 首先要安装 然后在 /etc/printcap 中指定它的路径。在打印置中, 器似于文本器 (安装文本器) 不同的是它不是用来打印文本, 而是将一个文件成打印机能理解的格式。

10.4.1.4.2. 我安装个器?

安装希望使用的器。如果要打印很多 DVI 数据, 就需要 DVI 器; 如果有大量的 troff 数据, 就安装 troff 器。

下面的表格了可以与 LPD配合 工作的器, 以及它在 /etc/printcap文件中的量名, 有如何在 lpr命令中用它:

| 文件型           | 在/etc/printcap文件中的量名 | 在lpr命令中使用的参数    |
|---------------|----------------------|-----------------|
| cifplot       | cf                   | -c              |
| DVI           | df                   | -d              |
| plot          | gf                   | -g              |
| ditroff       | nf                   | -n              |
| FORTTRAN text | rf                   | -f              |
| troff         | tf                   | -f              |
| raster        | vf                   | -v              |
| plain text    | if                   | none, -p, or -l |

在例子中, lpr -d就是指 打印机需要在/etc/printcap文件中 df量所指的器。

不管人, 像 FORTRAN 的文本 和 plot 些格式已基本不用了。所以在的机器上, 就可以安装其他的器来替些参数原有的意。例如, 假想要能直接打印 Printerleaf 文件 (由 Interleaf desktop publishing 程序生成), 而且不打算打印 plot 文件, 就可以安装一个 Printerleaf 器并且用 gf 量指定它。然后就可以告诉的用使用 lpr -g 就可以 "打印 Printerleaf 文件。"

10.4.1.4.3. 安装器

以安装的器不是 FreeBSD 基本系的一部分, 所以它可能是在 /usr/local 目下。通常目 /usr/local/libexec 是保存它的地方, 因它通常是通 LPD 行的; 普通用并不需要直接行它。

要用一个过滤器，只需要在 `/etc/printcap` 文件中目打印机中含的变量上过滤器所在的路径。

在接下来的例子当中，我将一台叫做 **bamboo** 的打印机添加一个过滤器。下面是个例子的 `/etc/printcap` 文件，其中使用新变量 **df** 来打印机 **bamboo** 置过滤器：

```
#
# /etc/printcap for host rose - added df filter for bamboo
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
    :sh:sd=/var/spool/lpd/rattan:\
    :lp=/dev/lpt0:\
    :if=/usr/local/libexec/if-simple:

bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
    :sh:sd=/var/spool/lpd/bamboo:\
    :lp=/dev/ttyu5:ms#-parenb cs8 clocal crtsets:rw:\
    :if=/usr/local/libexec/psif:\
    :df=/usr/local/libexec/psdf:
```

里的 DVI 过滤器是一段 shell 脚本，名字叫做 `/usr/local/libexec/psdf`。下面是它的代码：

```
#!/bin/sh
#
# psdf - DVI to PostScript printer filter
# Installed in /usr/local/libexec/psdf
#
# Invoked by lpd when user runs lpr -d
#
exec /usr/local/bin/dvips -f | /usr/local/libexec/lprps "$@"
```

这段脚本以过滤器模式运行 **dvips** (参数 **-f**) 并从标准输入取要打印的任。然后运行 PostScript® 文本过滤器 **lprps** (在 **PostScript® 打印机上打印文本任** 一)，并且着 LPD 脚本的全部参数。 **lprps** 工具将利用些参数来打印行。

#### 10.4.1.4.4. 更多过滤器用例

因安装过滤器的并不是固定的，所以介了一些可行的例子。在以后的安装配置程中可以以些例子参考。甚至如果合的，可以完全照搬去。

段例子中的脚本是一个 Hewlett Packard LaserJet III-Si 打印机的光格式数据 (上也就是 GIF 文件)：

```
#!/bin/sh
#
# hpvf - Convert GIF files into HP/PCL, then print
# Installed in /usr/local/libexec/hpvf

PATH=/usr/X11R6/bin:$PATH; export PATH
giftopnm | pmtopgm | pgmtopbm | pbmtolj -resolution 300 \
    && exit 0 \
    || exit 2
```

它的工作原理就是将 GIF 文件转换成 portable anymap，再转换成 portable graymap，然后再转换成 portable bitmap，最后再转换成 LaserJet/PCL- 兼容的数据。

下面是打印机配置上上述设备的 /etc/printcap 文件：

```
#
# /etc/printcap for host orchid
#
teak|hp|laserjet|Hewlett Packard LaserJet 3Si:\
    :lp=/dev/lpt0:sh:sd=/var/spool/lpd/teak:mx#0:\
    :if=/usr/local/libexec/hpif:\
    :vf=/usr/local/libexec/hpvf:
```

下面的脚本是一个在名叫 **bamboo** 的平台 PostScript® 打印机上打印用 groff 排版软件生成的 troff 数据的打印设备：

```
#!/bin/sh
#
# pstf - Convert groff's troff data into PS, then print.
# Installed in /usr/local/libexec/pstf
#
exec grops | /usr/local/libexec/lprps "$@"
```

上面这段脚本是用 **lprps** 来与打印机通信。如果打印机是接在并口上的，那就使用下面的脚本：

```
#!/bin/sh
#
# pstf - Convert groff's troff data into PS, then print.
# Installed in /usr/local/libexec/pstf
#
exec grops
```

这里是我要用设备需要在 /etc/printcap 里添加的内容：

```
:tf=/usr/local/libexec/pstf:
```

下面的例子也许会多 FORTRAN 老手羞愧。它是一个 FORTRAN- 文本 的过滤器，能在任意一台 可以打印 文本的打印机上使用。我将打印机 **teak** 安装一个过滤器：

```
#!/bin/sh
#
# hprf - FORTRAN text filter for LaserJet 3si:
# Installed in /usr/local/libexec/hprf
#

printf "\033&k2G" && fpr && printf "\033&l0H" &&
exit 0
exit 2
```

然后我需要在 `/etc/printcap` 中打印机能 **teak** 用一个过滤器添加下面的内容：

```
:rf=/usr/local/libexec/hprf:
```

最后，再出一个有些过滤的例子。我将以前介绍的 **teak** 激光打印机添加一个 DVI 过滤器。首先，最容易的部分：更新 `/etc/printcap` 加入 DVI 过滤器的路径：

```
:df=/usr/local/libexec/hpdf:
```

在，困难的部分了：写过滤器。为了过滤器，我需要一个 DVI-到-LaserJet/PCL 程序。FreeBSD Ports Collection (即 [Ports Collection](#) 一) 中有一个：[print/dvi2xx](#)。安装一个 port 就会得到我需要的程序，**dvilj2p**，它可以将 DVI 数据转换成 LaserJet IIP, LaserJet III, 和 LaserJet 2000 兼容的数据。

**dvilj2p** 工具使得过滤器 **hpdf** 得十分简单，因为 **dvilj2p** 不能读取标准输入。它需要从文件中读取数据。更糟糕的是，这个文件的名字必须以 `.dvi` 结尾。所以使用 `/dev/fd/0` 作标准输入是有问题的。我可以通过链接 (符号链接) 来解决这个问题。链接一个过滤的文件名 (一个以 `.dvi` 结尾的文件名) 到 `/dev/fd/0`，从而制止 **dvilj2p** 从标准输入读取。

在迎面而来的是另外一个问题，我不能使用 `/tmp` 存放链接。符号链接是被用 `root` 和 `bin` 所有的。而过滤器是以 **daemon** 用运行的。并且 `/tmp` 目录置了 `sticky` 位。所以过滤器只能建立符号链接，但它不能在用完之后清除掉这些链接。因为它属于不同的用户。

所以过滤器将在当前工作目录下建立符号链接，即后台打印队列目录 (用量 **sd** 在 `/etc/printcap` 中指定)。这是一个非常好的过滤器完成它工作的地方，特别是因为 (有) 一个目录比起 `/tmp` 来有更多的可用磁盘空间。

最后，出过滤器的代码：

```
#!/bin/sh
#
# hpdf - Print DVI data on HP/PCL printer
# Installed in /usr/local/libexec/hpdf

PATH=/usr/local/bin:$PATH; export PATH
```

```

#
# Define a function to clean up our temporary files. These exist
# in the current directory, which will be the spooling directory
# for the printer.
#
cleanup() {
    rm -f hpdf$$ .dvi
}

#
# Define a function to handle fatal errors: print the given message
# and exit 2. Exiting with 2 tells LPD to do not try to reprint the
# job.
#
fatal() {
    echo "$@" 1>&2
    cleanup
    exit 2
}

#
# If user removes the job, LPD will send SIGINT, so trap SIGINT
# (and a few other signals) to clean up after ourselves.
#
trap cleanup 1 2 15

#
# Make sure we are not colliding with any existing files.
#
cleanup

#
# Link the DVI input file to standard input (the file to print).
#
ln -s /dev/fd/0 hpdf$$ .dvi || fatal "Cannot symlink /dev/fd/0"

#
# Make LF = CR+LF
#
printf "\033&k2G" || fatal "Cannot initialize printer"

#
# Convert and print. Return value from dvi2p does not seem to be
# reliable, so we ignore it.
#
dvi2p -M1 -q -e- dfhp$$ .dvi

#
# Clean up and exit
#
cleanup

```

#### 10.4.1.4.5. 自印：一种替代打印器的方法

以上这些打印机基本上建成了自己的打印环境，但也有不足就是必须由用户来指定（在 `lpr(1)` 命令行中）要使用哪一个打印机。如果用户的用不是计算机很在行，那用打印机将是一件麻烦的事情。更糟的是，当打印机指定的不正，打印机被用在了不它类型的文件上，打印机也会出上百。

比只安装打印机更好的方法，就是文本打印机（因它是默认的打印机）来需要打印文件的类型，然后自行修正的打印机。像 `file` 的工具可以给我一定的帮助。当然，要区分有些文件的类型是有困难的 - 但是，当然，它可以提供打印机。

FreeBSD 的 Ports 套件提供了一个可以自行运行的文本打印机，名字叫做 `apsfilter` ([print/apsfilter](#))。它可以文本、PostScript®、DVI 以及几乎任何格式的文件，并在运行相关的之后完成打印工作。

#### 10.4.1.5. 输出器

LPD 后台打印系统支持一种我没有的打印机：输出器。输出器只是用来打印文本的，类似于文本打印机，但化了多地方。如果正在使用输出器而不是文本打印机，那：

- LPD 整个任务一个输出器，而不是任务中的个文件都一次。
- LPD 不会提供任务中文件开始和结束的信息输出器。
- LPD 不会提供用户名或者主机名打印机，所以它是无法做打印的。事实上它只有个参数：

`打印机-名字 -w 宽度 -l 长度`

宽度来自于 `pw` 量，而 `length` 来自于 `pl` 量，些都是系统中打印机置的。

不要输出器的化所耽。如果想要输出器完成任务中的个文件都重新始一打印是不可能。使用文本打印机（也叫入打印机）；安装文本打印机。此外，上，输出器更，它要它的字流中是否有特殊的志字符，并且自己送信号来代替 LPD 的。

可是，如果打算要或者需要送控制字符或者其他初始化字符串来完成打印，那输出器是必需的。（但是它也是无用的如果打算打印的用，因 LPD 不会输出器任何用或者主机的信息。）

在一台个的打印机上，LPD 同允输出器、文本打印机和其他的打印机。在某些情况下，LPD 将会输出器来打印（即）。然后 LPD 会要求输出器自己停止行，它送输出器个字：ASCII 031 跟着一个 ASCII 001。当输出器看个字（031, 001），它通送 `SIGSTOP` 信号来停止自己的行。当 LPD 已行好了其他的打印机，它会通输出器送 `SIGCONT` 信号来输出器重新行。

如果有一个输出器而没有文本打印机，并且 LPD 正在理一个文本任务，LPD 会使用输出器来完成个任务。像以前行一，输出器会按序打印任务中的文件，而不会入送或其他的命令，但它也不是想要的果。在大多数情况下，是需要一个文本打印机。

`lpf` 个我前面介绍的文本打印机程序，也可以用来做输出器。如果需要使用快速且混乱的输出器，但又不想写字和信号代，那 `lpf`。 `lpf` 也可以包含在一个 `shell` 脚本中来理任何打印机可能需要的初始化代。

#### 10.4.1.6. lpf：一个文本过滤器

`/usr/libexec/lpr/lpf` 这个程序包含在 FreeBSD 的二进制程序中，它是一个文本过滤器（输入过滤器）。它可以排序输出（用 `lpr -i` 命令提交的任何），可以打印控制字符禁止断用 `lpr -l` 提交的任何，可以调整输出中退格和制表符打印的位置，它可以打印任何行。它同样可以像过滤器一样工作。

`lpf` 用于很多打印环境。尽管它本身没有向打印机发送初始化代码的功能，但写一个 shell 脚本来完成所需的初始化并运行 `lpf` 是很容易的。

除了 `lpf` 可以正确的行打印，那需要 `/etc/printcap` 中的 `pw` 和 `pl` 变量都填入正确的。它用这些来定义能打印多少文本，并算出任有多少。想得到更多关于打印的信息，参看[打印机使用行](#)。

#### 10.4.2. 过滤器

如果有很多用，他正在使用各式各样的打印机，那或许要考一下把过滤器当作无可避免之了。

过滤器，也叫 *banner* 或者 *burst*，可以用来辨别打印出的文件是谁打印的。它通常用大号的粗体字母打印出来，也可能用装过滤器四周，所以在一堆打印出的文件中，突出的显示了哪个文件属于哪个用的哪个任务。它可以用快速的到他任务的。而过滤器一个明显的缺点就是，在哪个任务中都要有一或者几行作过滤器印出来，可是它的有用的地方只几分的作用，最后它会被放回回收站或者过滤器堆。（注意过滤器只是一个任务一个，而不是任务中的哪个文件都有一个，所以可能过滤器不算很浪费。）

LPD 系可以自过滤器的打印提供，如果过滤器的打印机可以直接打印文本。如果过滤器的打印机是一台 PostScript® 打印机，它将需要一个外部的程序来生成过滤器；参看[在 PostScript® 打印机上打印过滤器](#)。

##### 10.4.2.1. 打印过滤器

在[打印配置](#)中，我通常在 `/etc/printcap` 文件中指定 `sh`（“禁止过滤器”）来把过滤器功能掉了。要重新打印机过滤器功能，只需要除掉 `sh`。

听起来很容易，不是吗？

是的。可能不得不出过滤器来打印机发送初始化字符串。下面是一个用在 Hewlett Packard PCL-兼容打印机上的输出过滤器的例子：

```
#!/bin/sh
#
# hpof - Output filter for Hewlett Packard PCL-compatible printers
# Installed in /usr/local/libexec/hpof

printf "\033&k2G" || exit 2
exec /usr/libexec/lpr/lpf
```

用 `of` 变量指定输出过滤器的路径。参看[输出过滤器](#)一章来得到更多信息。

下面是一个我以前介绍的叫做 `teak` 的打印机配置的 `/etc/printcap` 文件；在配置当中我加了过滤器并且加入了上述的打印过滤器：

```
#
# /etc/printcap for host orchid
#
teak|hp|laserjet|Hewlett Packard LaserJet 3Si:\
    :lp=/dev/lpt0:sd=/var/spool/lpd/teak:mx#0:\
    :if=/usr/local/libexec/hpif:\
    :vf=/usr/local/libexec/hpvf:\
    :of=/usr/local/libexec/hpof:
```

在，当用再任一打印机 **teak** 的时候，个任都会有一个。如果用想要花来他  
自己打印的文件，那他可以通过 **lpr -h** 命令来提交任；参考 来得到更多关于 **lpr(1)** 的。



LPD 在之后出一个字符。如果的打印机使用一个不同的字符或者字符串当作退  
指令，在 `/etc/printcap` 中用 **ff** 量指定即可。

#### 10.4.2.2. 控制

通用，LPD 将生成出一个，一整的大字母，着用，主机和任名。下面是一个例子 (**kelly**  
从主机 **rose** 打印了一个叫做 "outline" 的任)：



```

k          ll      ll
k          l       l
k          l       l
k  k      eeee    l       l       y       y
k  k      e   e    l       l       y       y
k  k      eeeee   l       l       y       y
kk k      e        l       l       y       y
k  k      e   e    l       l       y      yy
k  k      eeee    lll      lll      yyy y
                               y
                               y  y
                               yyyy

                               ll
                               t       l       i
                               t       l
o o o o  u   u  t t t t  l       i i   n n n n   eeee
o   o   u   u   t       l       i   n n   n   e   e
o   o   u   u   t       l       i   n   n   eeeee
o   o   u   u   t       l       i   n   n   e
o   o   u   uu   t  t    l       i   n   n   e   e
o o o o   u u u u   t t    lll      i i i   n   n   eeee

r r r r   o o o o   s s s s   eeee
r r   r   o   o   s   s   e   e
r       o   o   s s       eeeee
r       o   o       s s   e
r       o   o   s   s   e   e
r       o o o o   s s s s   eeee

```

Job: outline

Date: Sun Sep 17 11:04:58 1995

LPD 会附加一个换行符在每段文本之后，所以任何都会在新的一行上开始（除非设置了 **sf**（禁止换行）在 `/etc/printcap` 文件里目录打印机的配置中）。

如果喜欢，LPD 可以生成一个短行；指定 **sb**（短 banner）在文件 `/etc/printcap` 中。配置就会看起来像下面：

```
rose:kelly Job: outline Date: Sun Sep 17 11:07:51 1995
```

同是默认的，LPD 也是先打印配置，然后才是任何。要想反过来，在 `/etc/printcap` 中指定 **hl**（最后配置）。

### 10.4.2.3. 配置文件的任何配置

使用 LPD 内置的配置会在每行打印配置的时候产生一种特殊情况：配置肯定是免空的。

什么？

因出器是有的一个在打印能行的外部程序， 但却没有提供它任何 用或者主机 的信息或者文件， 所以它无法知道打印机的使用付。 如果是 "加一数" 文本器或者其他器（它有用和主机的信息）是不的， 因用可以用 `lpr -h` 命令跳。 他是需要自己并没有打印的付。基本上， `lpr -h` 是明知用的首， 但也不能制人使用它。

个器生成自己的（因此可以它）是 仍然不的。 如果用想要用 `lpr -h` 命令禁止， 它将仍然印出并且它付。 因 LPD 不会把 `-h` 个参数任何器。

， ？

可以：

- 可 LPD 的个， 并且免提供打印。
- 安装一个替代 LPD 的件， 比如 LPRng。 参考 [替准的后台打印件](#) 来得到更多于可以替代 LPD 的件的信息。
- 写一个 明的 出器。 通常， 出器不去完成除了初始化打印机或者行一些字符以外的任何事情。 它合完成和文本任（当没有文本（入）器）。 但是， 如果有文本器文本任服， 那 LPD 将打印出器。 而且， 个出器可以理解里 LPD 生成的信息， 然后决定位用和主机付。 方法有的的是出器仍然不知道使用什文件（`af` 量的内容并没有被来）， 但是如果有一个所周知的文件， 就可以直接把文件名写出器。 了化解的， 我定 `sh` (短的) 量在 `/etc/printcap` 文件中。 但些是太麻烦了， 而且用也更喜他免打印的慷慨的系管理。

#### 10.4.2.4. 在 PostScript® 打印机上打印

像上面描述的那， LPD 可以生成一个文本的来多打印机。 当然， PostScript® 不能直接打印文本， 所以 LPD 没什用-或者大候是。

一个而易的方法来得到就是个个器和文本器都来生成。 些器用用名和主机的参数来生成一个相的。 方法的缺点就是用是打印出， 无他是否用 `lpr -h` 命令来提交的任。

我我来深入深入的研究一下这个方法。 下面的脚本入三个参数（用登名， 主机名， 和任名）然后生成一个的 PostScript® ：

```
#!/bin/sh
#
# make-ps-header - make a PostScript header page on stdout
# Installed in /usr/local/libexec/make-ps-header
#
#
# These are PostScript units (72 to the inch). Modify for A4 or
# whatever size paper you are using:
#
page_width=612
page_height=792
border=72
```

```

#
# Check arguments
#
if [ $# -ne 3 ]; then
    echo "Usage: `basename $0` <user> <host> <job>" 1>&2
    exit 1
fi

#
# Save these, mostly for readability in the PostScript, below.
#
user=$1
host=$2
job=$3
date=`date`

#
# Send the PostScript code to stdout.
#
exec cat <<EOF
%!PS

%
% Make sure we do not interfere with user's job that will follow
%
save

%
% Make a thick, unpleasant border around the edge of the paper.
%
$border $border moveto
$page_width $border 2 mul sub 0 rlineto
0 $page_height $border 2 mul sub rlineto
currentscreen 3 -1 roll pop 100 3 1 roll setscreen
$border 2 mul $page_width sub 0 rlineto closepath
0.8 setgray 10 setlinewidth stroke 0 setgray

%
% Display user's login name, nice and large and prominent
%
/Helvetica-Bold findfont 64 scalefont setfont
$page_width ($user) stringwidth pop sub 2 div $page_height 200 sub moveto
($user) show

%
% Now show the boring particulars
%
/Helvetica findfont 14 scalefont setfont
/y 200 def
[ (Job:) (Host:) (Date:) ] {
    200 y moveto show /y y 18 sub def }

```

```
forall

/Helvetica-Bold findfont 14 scalefont setfont
/y 200 def
[ ($job) ($host) ($date) ] {
    270 y moveto show /y y 18 sub def
} forall

%
% That is it
%
restore
showpage
EOF
```

在， 一个过滤器和文本过滤器都能用一段脚本来生成， 然后打印用的任何。 下面是我早些时候在个文中提到的 DVI 过滤器， 被修改之后来生成一个：

```
#!/bin/sh
#
# psdf - DVI to PostScript printer filter
# Installed in /usr/local/libexec/psdf
#
# Invoked by lpd when user runs lpr -d
#

orig_args="$@"

fail() {
    echo "$@" 1>&2
    exit 2
}

while getopts "x:y:n:h:" option; do
    case $option in
        x|y) ;; # Ignore
        n)    login=$OPTARG ;;
        h)    host=$OPTARG ;;
        *)    echo "LPD started `basename $0` wrong." 1>&2
              exit 2
              ;;
    esac
done

[ "$login" ] || fail "No login name"
[ "$host" ] || fail "No host name"

( /usr/local/libexec/make-ps-header $login $host "DVI File"
  /usr/local/bin/dvips -f ) | eval /usr/local/libexec/lprps $orig_args
```

器是解参数列表来决定用名和主机名的。解的方法于其他器来也是一的。尽管文本器需要入的参数有些小的不同，(参器是工作的)。

像我以前提到的那，上面的配置，尽管相当，掉了"禁止"的(-h)在lpr中。如果用想要保木(或者是几便士，如果打印收的)，它不能完成件事情，因个器都要个任打印一个。

要允用于个任都可以，需要使用在 中介的那技巧：写一个出器来解 LPD-生成的并且生成一个 PostScript® 的版本。如果用用lpr -h命令提交任，那LPD将不会生成，并且出器也不会生成。否，出器将从LPD取文本，然后送当的PostScript®打印机。

如果有的是一台在串口上的 PostScript® 打印机，可以使用lprps里的一个出器，psof，它可以完成上述任。但注意psof不。

### 10.4.3. 网打印

FreeBSD 支持网打印：送任程打印机。网打印通常指不同的方式：

- 一台接在程主机上的打印机。在一台主机上安装一台常的串口或并口打印机。然后，置LPD来通网其他主机上的打印机。具体安装在程主机上的打印机。
- 一台直接接在网上的打印机。打印机有一个网接口(或者替代常的串口或者并口)。的打印机可能像下面工作：
  - 它或可以理解LPD的，并且甚至可以接收程主机来的任排列。它，它就像一个普通的主机行着LPD一。做在安装在程主机上的打印机里介的，可以置好的打印机。
  - 它或支持网数据流。它，把打印机"接"在一台网上的主机上，由台主机安排任并送任到打印机。参网数据流接口的打印机来得到更多安装打印机的建。

#### 10.4.3.1. 安装在程主机上的打印机

LPD 后台打印系内建了其他也行着LPD(或者是与LPD兼容的)的主机送任的功能。个功能使可以在一台主机上安装打印机，并它可以在其他主机上。个功能同用在那些有网接口并且可以理解LPD的打印机上。

要程打印的功能，首先在一台主机上安装打印机，就是打印服器，可以使用在打印机置中置的方法。高的置可以参考高时打印机置中需要的部分。一定要一下打印机，看看它是不是所有的LPD的功能都正常工作。此外需要本地主机允使用程主机上的LPD服(参限制程主打印任)。

如果正在使用一台网接口并与LPD兼容的打印机，那我那下面中的打印服器就是打印机本身，而打印机名就是打印机配置的名字。参考随打印机和/或者打印机-网接口供的文。



如果正使用惠普的 Laserjet，打印机名text将自地完成LF到CRLF的，因而也就不需要hpif脚本了。

然后，在外一台想要打印机的主机上的/etc/printcap文件中加入它的，像下面：

1. 可以随意个起名字。起，可以打印服器使用相同的名字或者名。

2. 保留 `lp` 量空, (`:lp=:`)。
3. 建立一个后台打印列目, 并用 `sd` 量指明其位置。LPD 将把任何提交时打印服务器之前, 会把一些任何保存在里。
4. 在 `rm` 量中放入打印服务器的名字。
5. 在 `rp` 中放入打印服务器上打印机的名字。

就是。不需要列出器, 面大小, 或者其他的一些西在 `/etc/printcap` 文件中。

有一个例子。主机 `rose` 有台打印机, `bamboo` 和 `rattan`。我要主机 `orchid` 的用户可以使用台打印机。下面是 `/etc/printcap` 文件, 用在主机 `orchid` (即 `localhost`) 上的。文件中已经有了打印机 `teak` 的; 我在主机 `rose` 上加了台打印机:

```
#
# /etc/printcap for host orchid - added (remote) printers on rose
#

#
# teak is local; it is connected directly to orchid:
#
teak|hp|laserjet|Hewlett Packard LaserJet 3Si:\
    :lp=/dev/lpt0:sd=/var/spool/lpd/teak:mx#0:\
    :if=/usr/local/libexec/ifhp:\
    :vf=/usr/local/libexec/vfhp:\
    :of=/usr/local/libexec/ofhp:

#
# rattan is connected to rose; send jobs for rattan to rose:
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
    :lp=:rm=rose:rp=rattan:sd=/var/spool/lpd/rattan:

#
# bamboo is connected to rose as well:
#
bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
    :lp=:rm=rose:rp=bamboo:sd=/var/spool/lpd/bamboo:
```

然后, 我只需要在主机 `orchid` 上建立一个后台打印列目:

```
# mkdir -p /var/spool/lpd/rattan /var/spool/lpd/bamboo
# chmod 770 /var/spool/lpd/rattan /var/spool/lpd/bamboo
# chown daemon:daemon /var/spool/lpd/rattan /var/spool/lpd/bamboo
```

在, 主机 `orchid` 上的用户可以打印到 `rattan` 和 `bamboo` 了。如果, 比如, 一个用户在主机 `orchid` 上入了:

```
% lpr -P bamboo -d sushi-review.dvi
```

LPD 系在主机 **orchid** 上会制个任到后台打印列目 `/var/spool/lpd/bamboo` 并且下是一个 DVI 任。当主机 **rose** 上的打印机 **bamboo** 的后台打印列目有空的时候，这个 LPD 系将会把文件到主机 **rose** 上。文件将排在主机 **rose** 的列中直到最后被打出来。它将被从 DVI 转成 PostScript® (因为 **bamboo** 是一台 PostScript® 打印机) 在主机 **rose**。

#### 10.4.3.2. 有网数据流接口的打印机

通常，当打印机连了一网，可以得到两个版本：一个是模拟后台打印 (一些的版本)，或者一个只送数据打印机就像在使用串口或者并口一样 (便宜一些的版本)。描述如何使用一个便宜一些的版本。要得到一些版本的更多信息，参前面章 [安装在远程主机上的打印机](#)。

`/etc/printcap` 文件的格式指定使用一个串口或并口，并且要指定 (如果正在使用串口)，使用多快的波特，是否使用流量控制，制表符延迟，行数，等等。但是没有一种方法指定一个连接到一台正在听 TCP/IP 的或者其他网接口的打印机。

要送数据到网打印机，就需要一个通程序，它可以被文本或者解释器用。下面是一些例子：脚本 **netprint** 将输入的所有数据送到一个在网上的打印机。我将打印机的名字作第一个参数，端口号跟在后面作第二个参数，即 **netprint**。注意它只支持向通 (FreeBSD 到打印机)；很多网打印机支持双向通，并且是可能利用到的 (得到打印机状态，行打印，等等的候。 )。

```
#!/usr/bin/perl
#
# netprint - Text filter for printer attached to network
# Installed in /usr/local/libexec/netprint
#
$#ARGV eq 1 || die "Usage: $0 <printer-hostname> <port-number>";

$printer_host = $ARGV[0];
$printer_port = $ARGV[1];

require 'sys/socket.ph';

($ignore, $ignore, $protocol) = getprotobyname('tcp');
($ignore, $ignore, $ignore, $ignore, $address)
    = gethostbyname($printer_host);

$sockaddr = pack('S n a4 x8', &AF_INET, $printer_port, $address);

socket(PRINTER, &PF_INET, &SOCK_STREAM, $protocol)
    || die "Can't create TCP/IP stream socket: $!";
connect(PRINTER, $sockaddr) || die "Can't contact $printer_host: $!";
while (<STDIN>) { print PRINTER; }
exit 0;
```

然后我就可以在多器里使用这个脚本了。加入我有一台 Diablo 750-N 行式打印机在网上。打印机在 5100 端口上接收要打印的数据。打印机的主机名是 **scrivener**。里是打印机写的文本器：

```
#!/bin/sh
#
# diablo-if-net - Text filter for Diablo printer 'scrivener' listening
# on port 5100. Installed in /usr/local/libexec/diablo-if-net
#
exec /usr/libexec/lpr/lpf "$@" | /usr/local/libexec/netprint scrivener 5100
```

#### 10.4.4. 限制打印机的使用

我们将描述用于限制打印机使用的。LPD 系统可以控制可以打印的打印机，无论本地或是远程的，是否他打印多副本，任可以有多大，以及打印队列的尺寸等。

##### 10.4.4.1. 限制多副本

LPD 系统能简化在打印多副本的工作。用可以用 `lpr -#5` (例子) 来提交打印任务，会将任务中每个文件都打印五副本。是不是一件很棒的事情。

如果感觉多副本会对打印机造成不必要的磨损和消耗，可以屏蔽掉 `lpr(1)` 的 `-#` 选项，可以通过在 `/etc/printcap` 文件中加 `sc` 选项来完成。当用用 `-#` 提交任务，他将看到：

```
lpr: multiple copies are not allowed
```

注意当一台远程打印机行配置 (参看 [安装在远程主机上的打印机](#) 一节) 需要同在远程主机的 `/etc/printcap` 文件中加 `sc` 选项，否则用是可以从其他主机上提交使用多副本的任务。

下面是一个例子。这个是 `/etc/printcap` 文件在主机 `rose` 上。打印机 `rattan` 非常慢，所以我允许多副本，但是激光打印机 `bamboo` 有些忙，所以我禁止多副本，通过加 `sc` 选项：

```
#
# /etc/printcap for host rose - restrict multiple copies on bamboo
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
    :sh:sd=/var/spool/lpd/rattan:\
    :lp=/dev/lpt0:\
    :if=/usr/local/libexec/if-simple:

bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
    :sh:sd=/var/spool/lpd/bamboo:sc:\
    :lp=/dev/tty5:ms#-parenb cs8 clocal crtscts:rw:\
    :if=/usr/local/libexec/psif:\
    :df=/usr/local/libexec/psdf:
```

在，我可能需要机 `sc` 选项在主机 `orchid` 的 `/etc/printcap` 文件中 (即便我也禁止打印机 `teak` 多打印)：



```
#
# /etc/printcap for host orchid - no multiple copies for local
# printer teak or remote printer bamboo
teak|hp|laserjet|Hewlett Packard LaserJet 3Si:\
    :lp=/dev/lpt0:sd=/var/spool/lpd/teak:mx#0:sc:\
    :if=/usr/local/libexec/ifhp:\
    :vf=/usr/local/libexec/vfhp:\
    :of=/usr/local/libexec/ofhp:

rattan|line|diablo|lp|Diablo 630 Line Printer:\
    :lp=:rm=rose:rp=rattan:sd=/var/spool/lpd/rattan:

bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
    :lp=:rm=rose:rp=bamboo:sd=/var/spool/lpd/bamboo:sc:
```

通过 `sc` 变量，我阻止了 `lpr -#` 命令的使用，但仍然没有禁止用多次行 `lpr(1)`，或者多次提交任何中同的文件，像下面：

```
% lpr forsale.sign forsale.sign forsale.sign forsale.sign forsale.sign
```

里有很多方法可以阻止行（包括忽略它），并且是免的。

#### 10.4.4.2. 限制打印机的

可以控制可以打印到台打印机通 UNIX® 的机制和文件 `/etc/printcap` 中的 `rg` 变量。只要把可以打印机的用放适当的，然后在 `rg` 变量中写上名字。

如果以外的用（包括 `root`）打印到被限制的打印机，将会得到提示：

```
lpr: Not a member of the restricted group
```

像使用 `sc`（禁止多副本）变量一，需要指定 `rg` 在程同打印机有限制的主机上，如果感合的（参考 [安装在程主机上的打印机](#) 一）。

比如，我将任何人都可以打印机 `rattan`，但只有在 `artists` 中的人可以使用打印机 `bamboo`。里是似的主机 `rose` 上的 `/etc/printcap` 文件：

```
#
# /etc/printcap for host rose - restricted group for bamboo
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
    :sh:sd=/var/spool/lpd/rattan:\
    :lp=/dev/lpt0:\
    :if=/usr/local/libexec/if-simple:

bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
    :sh:sd=/var/spool/lpd/bamboo:sc:rg=artists:\
    :lp=/dev/tty5:ms#-parenb cs8 clocal crtsets:rw:\
    :if=/usr/local/libexec/psif:\
    :df=/usr/local/libexec/psdf:
```

Let us leave the other example `/etc/printcap` file (for the host `orchid`) alone. Of course, anyone on `orchid` can print to `bamboo`. It might be the case that we only allow certain logins on `orchid` anyway, and want them to have access to the printer. Or not.



这里可能有一个限制的。

#### 10.4.4.3. 控制提交的任大小

如果有很多用打印机，可能需要用可以提交的文件尺寸置一个上限。毕竟，文件系统后台打印队列的空是有限的，需要保证里有空来存放其他用的任。

LPD 允许通使用 `mx` 量来限制任中文件的最大字数，方法是指定位的 `BUFSIZ` 数，表示 1024 字。如果在个量的是 0，表示不限制；不，如果不指定 `mx` 量的，会使用默认 1000。



个限制是于任中文件的，而不是任共的大小。

LPD 不会拒比限制大小大的文件。但它是将限制大小以内的部分排入列，并且打印出来的只有些。剩下的部分将被。个行是否正需。

我来例子打印机 `rattan` 和 `bamboo` 加限制。由于那些 `artists` 的 PostScript® 文件可能会很大，我将限制大小 5 兆字。我将不文本行式打印机做限制：

```
#
# /etc/printcap for host rose
#

#
# No limit on job size:
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
    :sh:mx#0:sd=/var/spool/lpd/rattan:\
    :lp=/dev/lpt0:\
    :if=/usr/local/libexec/if-simple:

#
# Limit of five megabytes:
#
bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
    :sh:sd=/var/spool/lpd/bamboo:sc:rg=artists:mx#5000:\
    :lp=/dev/tty5:ms#-parenb cs8 clocal crtsets:rw:\
    :if=/usr/local/libexec/psif:\
    :df=/usr/local/libexec/psdf:
```

同，限制只本地用起作用。如果置了允程序用使用的打印机，程序将不会受到些限制。也需要指定 `mx` 量在程主机的 `/etc/printcap` 文件中。参 [安装在程主机上的打印机](#) 来得到更多有程打印的信息。

除此之外，有来限制程任大小的方法；参 [限制程主机打印任](#)。

#### 10.4.4.4. 限制程主机打印任

LPD 后台打印系提供了多方法来限制从程主机提交的任：

##### 主机限制

可以控制本地 LPD 接收台程主机来的求，通 `/etc/hosts.equiv` 文件和 `/etc/hosts.lpd` 文件。LPD 看是否到来的任求来自被个文件中列出的主机。如果没有，LPD 会拒个求。

些文件的格式非常：行一个主机名。注意 `/etc/hosts.equiv` 文件也被 [ruserok\(3\)](#) 使用，并影响着 [rsh\(1\)](#) and [rcp\(1\)](#) 等程序，所以要小心。

个例子，下面是 `/etc/hosts.lpd` 文件在主机 `rose` 上：

```
orchid
violet
madrigal.fishbaum.de
```

意思是主机 `rose` 将接收来自 `orchid`，`violet`，和 `madrigal.fishbaum.de` 的求。如果任何其他的主机到主机 `rose` 的 LPD，任将被拒。

## 大小限制

可以控制后台打印队列需要保留多少空间。 建立一个叫做 `minfree` 的文件在后台打印队列下本地打印机。 在这个文件中输入一个数字来代表多少磁盘数 (512 字节) 的剩余空间来接收进程。

可以保持进程不会填满的文件系统。 也可以用它来本地用一个先： 他可以在磁盘剩余空间低于 `minfree` 文件中的指定后仍然可以提交任务。

比如， 我添加一个 `minfree` 文件打印机 `bamboo`。 我通过 `/etc/printcap` 文件来得到这个打印机的后台打印队列； 它是打印机 `bamboo` 的：

```
bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
:sh:sd=/var/spool/lpd/bamboo:sc:rg=artists:mx#5000:\
:lp=/dev/ttyu5:ms#-parenb cs8 clocal crtsets:rw:mx#5000:\
:if=/usr/local/libexec/psif:\
:df=/usr/local/libexec/psdf:
```

后台打印队列在 `sd` 空间中输出。 我设置 3 兆字节 (6144 磁盘) 的文件系统上必须存在的公共剩余空间， 让 LPD 可以接受进程：

```
# echo 6144 > /var/spool/lpd/bamboo/minfree
```

## 用限制

可以控制哪些进程可以打印到本地打印机， 通过指定 `rs` 空间在 `/etc/printcap` 文件中。 当 `rs` 输出在一个本地打印机的空间中， LPD 将接收来自进程主机 并在本地有同名登录名的用户提交的任何。 否则， LPD 会拒绝任何。

这个功能在一个（比如）有多部门共享一个网络的场景中特别有用， 并且有些用户可以越过部门的边界。 通过他在自己的系统上建立账号， 他可以他自己的部门的系统里使用他的打印机。 如果只允许他的打印机， 而不是他的计算机资源， 可以给他“象征”空间， 不主目并且设置一个没用的 shell， 比如 `/usr/bin/false`。

### 10.4.5. 打印机使用

当然， 需要打印付款。 什么不？ 纸和墨水都需要花钱的。 并且里面有纸的用途 - 打印机是由很多部件装成的， 并且零件会坏掉。 可以看看的打印机， 使用形式， 和用来得出（或者尺寸， 厘米， 或者什么）的用途。 在， 看看打印？

好了， 坏消息是 LPD 后台打印系统在这个部分没有提供很多帮助。 它是一个使用的打印机的， 打印的格式， 和 的 在打印机的使用的需求依赖性很高的。

要看看， 必须更改打印机的文本处理器（文本任务）和处理器（其他格式的文件）， 要看看数或者打印了多少的。 不可以通过使用的输出处理器来逃脱， 因为它不能运行。 参看 处理器。

通常， 有方法来进行：

- 定期 是更常用的方法， 可能因为它更简单。 无论合何人打印一个任务， 处理器都将用用户名， 主机名， 和打印的页数到一个文件。 个月， 学期， 年， 或者任何想定的时段， 收集一些不同打印机上的文件， 按用打印的页数行计算， 并使用行付款。 然后去掉所有文件， 开始一个新的周期。

- 不太常用，可能因为它比。 方法器用的打印行的。 像磁配， 是。 可以用打印当他超的时候， 并且可能提供一方法用并整他的 "打印配。" 但个方法需要一些数据代来跟踪用和他配。

LPD 后台打印系方法都支持且很： 所以需要提器（大多数候）， 要提供代。 但好的方面是： 可以有非常活的方法。 比如， 可以使用段是。 可以些信息： 用名， 主机名， 任型， 打印数， 使用了多少平方尺的， 任打印了多， 等等。 可以通修改器来存些信息。

#### 10.4.5.1. 快速并且混乱的打印

FreeBSD 包含个可以立刻可以建立起段的程序。 它是文本器 `lpf`， 在 `lpf`： 一个文本器 中描述， 和 `pac(8)`， 一个收集并打印机文件中程序。

像在前面章提到的器一（器）， LPD 文本或者器并在器命令行里上文件的名字。 器可以使用个参数知道往写。 个文件的名字来自于 `af` 量在 `/etc/printcap` 文件里， 并且如果没有指定路径， 默认是相于后台打印列目的。

LPD 的 `lpf` 着和的参数（通 `pw` 和 `pl` 量）。 `lpf` 使用些参数来判定将使用多少。 在文件送到打印机之后， 它就会在文件中写入。 像下面个：

```
2.00 rose:andy
3.00 rose:kelly
3.00 orchid:mary
5.00 orchid:mary
2.00 orchid:zhang
```

个打印机都使用一个独立的文件， 像 `lpf` 就没有内建文件， 个 `lpf` 可能会生彼此混合的情况， 如果它同要在同一个文件写入内容的时候。 一个最的保个打印机都使用一个独立的文件的方法就是将 `af=acct` 写在 `/etc/printcap` 文件中。 然后， 个打印机的文件都会在台打印机的后台打印列目中， 文件的名字叫做 `acct`。

当准用的打印行收， 行 `pac(8)` 程序。 只要到要收集信息的台打印机的后台打印列目， 然后入 `pac`。 将会得到一个美元摘要像下面：

| Login        | pages/feet | runs | price   |
|--------------|------------|------|---------|
| orchid:kelly | 5.00       | 1    | \$ 0.10 |
| orchid:mary  | 31.00      | 3    | \$ 0.62 |
| orchid:zhang | 9.00       | 1    | \$ 0.18 |
| rose:andy    | 2.00       | 1    | \$ 0.04 |
| rose:kelly   | 177.00     | 104  | \$ 3.54 |
| rose:mary    | 87.00      | 32   | \$ 1.74 |
| rose:root    | 26.00      | 12   | \$ 0.52 |
| total        | 337.00     | 154  | \$ 6.74 |

些是 `pac(8)` 需要的参数：

## -P 打印机

每台打印机要一个。这个选项在用 `af` 选项在 `/etc/printcap` 文件中指定了路径的情况下起作用。

## -c

以金文来排序出来代替以用名字字母排序。

## -m

忽略文件中的主机名。上一个选项，用 `smith` 在主机 `alpha` 上与同选项的用 `smith` 在主机 `gamma` 上不同。不同选项的，他们是不同的用。

## -p 价

使用 `price` 作选项或每尺美元的价来替代 `pc` 选项指定的价在 `/etc/printcap` 文件中，或者分（默认）。`price` 可以用一个浮点数来指定。

## -r

反向排序。

## -s

建立一个摘要文件，并且截短文件。

名字 ...

只打印指定名字用的信息。

在 `pac(8)` 默认生的摘要中，可以看到在不同主机上的每个用户打印了多少。如果在里，主机不考（因用可以使用任何主机），行 `pac -m`，来得到下面的摘要：

| Login | pages/feet | runs | price   |
|-------|------------|------|---------|
| andy  | 2.00       | 1    | \$ 0.04 |
| kelly | 182.00     | 105  | \$ 3.64 |
| mary  | 118.00     | 35   | \$ 2.36 |
| root  | 26.00      | 12   | \$ 0.52 |
| zhang | 9.00       | 1    | \$ 0.18 |
| total | 337.00     | 154  | \$ 6.74 |

要以美元算付数，`pac(8)` 指定 `pc` 选项在 `/etc/printcap` 文件中（默认是 200，或者 2 分）。这个参数的单位是百分之一分，在这个量中指定或者每尺的价格。可以覆盖这个当行 `pac(8)` 着参数 `-p` 的时候。参数 `-p` 的单位是美元，而不是百分之一分。例如，

```
# pac -p1.50
```

定的是价格是 1 美元 5 美分。可以通过这个来得到目利。

最后，行 `pac -s` 将存些信息在一个文件里，文件名和打印机名字相同，但是着 `_sum` 的后。然后截短文件。当再次行 `pac(8)` 的时候，它再次取文件来得到初始的，然后在文件中加信息。

#### 10.4.5.2. 打印的行数？

打印行程的精确度，需要判断一个任务将会消耗多少。是打印多少行。

对于文本任务，这个问题不是太好解决：任务中的行数行数然后与打印机支持的行数行比。别忘了也打印的行，或者很窄的行上的一行但在打印机上会折成两行的行。

文本处理器 `lpf` (在 [lpf：一个文本处理器](#) 中介绍) 会在考虑些。如果正在写一个可以行行的文本处理器，可能需要看 `lpf` 的源代码。

处理其他格式的文件？

好，对于 DVI- 到 -LaserJet 或者 DVI- 到 -PostScript® 的，可以出的器出断信息，对于 `dvilj` 或者 `dvips` 命令，并且看到多少被了。也可以对于其他型的文件和程序行似操作。

但是些方法的弱点就是事实上打印机并不是打印了所有的。比如，，缺墨，或者炸掉了 - 但用是要没有打印的部分付。

做？

只有一条肯定的方法来行精确的。一台可以告它使用了多少的打印机，并且将它接到串口或者网上。几乎所有 PostScript® 打印机都支持个小功能。其他制造厂或其他型号也可以有个功能 (比如 Imagen 激光网打印机)。些打印机更改器使它在打印完个任务之后接收用量，并基于个行。不需要算行数，也不需要容易出的文件。

当然，也是可以大方的使打印免。

## 10.5. 使用打印机

将述如何使用在 FreeBSD 下置好的打印机。下面是一个用命令的：

### `lpr(1)`

打印任务

### `lpq(1)`

打印队列

### `lprm(1)`

从打印机的队列中移除任务

有一个管理命令，`lpc(8)`，在 [管理打印机](#) 一章中有所介绍，它可以用于控制打印机及其列。

`lpr(1)`, `lprm(1)`, and `lpq(1)` 三个命令都接受 `-P printer-name` 来指定个打印机 / 列行操作，在 `/etc/printcap` 文件中列出的打印机。允许提交，删除，并任务在多个打印机上。如果不使用 `-P`，那些命令会使用在环境变量 `PRINTER` 中指定的打印机。最，如果也没有 `PRINTER` 个环境变量，些命令的默认是叫做 `lp` 的台打印机。

从此以后，默认打印机就是指 `PRINTER` 环境变量中指定的台，或者叫做 `lp` 的台当没有环境变量 `PRINTER` 的时候。

### 10.5.1. 打印任何

要打印文件，输入：

```
% lpr filename ...
```

这个命令会打印所有列出的文件到默认打印机。如果没有列出文件，[lpr\(1\)](#) 会从标准输入读取打印数据。比如，这个命令打印一些重要的系统文件：

```
% lpr /etc/host.conf /etc/hosts.equiv
```

要到一个指定的打印机，输入：

```
% lpr -P printer-name filename ...
```

这个例子打印一个当前目录的文件的列表到叫做 [rattan](#) 的打印机：

```
% ls -l | lpr -P rattan
```

因为没有 [lpr\(1\)](#) 命令列出文件，[lpr](#) 从标准输入数据，在这里是 [ls -l](#) 命令的输出。

[lpr\(1\)](#) 命令同样可以接受多控制格式的选项，如文件选项，生成多副本，等等。要得到更多信息，参考[打印选项](#)。

### 10.5.2. 打印任务

当使用 [lpr\(1\)](#) 打印时，希望打印的所有数据被放在一起打包成了一个 "打印任务"，它被送到 LPD 后台打印系统。每台打印机都有一个任务队列，并且任务在队列中等待其他用户的其他任务打印。打印机按照先来先印的原则打印某些任务。

要显示默认打印机的队列，输入 [lpq\(1\)](#)。要指定打印机，使用 [-P](#) 选项。例如，命令

```
% lpq -P bamboo
```

会显示打印机 [bamboo](#) 的队列。下面是命令 [lpq](#) 出的一个例子：

```
bamboo is ready and printing
Rank  Owner   Job  Files                                Total Size
active kelly    9    /etc/host.conf, /etc/hosts.equiv    88 bytes
2nd    kelly    10    (standard input)                    1635 bytes
3rd    mary     11    ...                                  78519 bytes
```

这里显示了队列中有三个任务在 [bamboo](#) 中。第一个任务，用 [kelly](#) 提交的，任务 "任务号" 9。每个要打印的任务都会得到一个不同的任务号。大多数时候可以忽略任务号，但在需要取消任务时会用到任务号；参考



移除任␣ 得到更多信息。

␣号 9 的任␣包含了␣个文件； 在 `lpr(1)` 命令行中指定的多个文件被看作是一个␣个的任␣。它是当前激活的任␣ (注意␣个␣ **激活** 在 "Rank" ␣列下面)，意思是打印机当前正在打印那个任␣。第二个任␣包含了␣准␣入␣ `lpr(1)` 命令的数据。第三个任␣来自用␣ `mary`；，它是一个比␣大的任␣。␣要打印的文件的路径名太␣了，所以 `lpq(1)` 命令只␣示了三个点。

`lpq(1)` ␣出的␣一行也很有用：它告␣我␣打印机正在做什␣ (或者至少是 LPD ␣打印机␣正在做的)。

`lpq(1)` 命令同␣支持 `-l` ␣来生成一个␣的␣列表。下面是一个 `lpq -l` 命令的例子：

```
waiting for bamboo to become ready (offline ?)
kelly: 1st                [job 009rose]
      /etc/host.conf      73 bytes
      /etc/hosts.equiv    15 bytes

kelly: 2nd                [job 010rose]
      (standard input)    1635 bytes

mary: 3rd                 [job 011rose]
      /home/orchid/mary/research/venus/alpha-regio/mapping 78519 bytes
```

### 10.5.3. 移除任␣

如果␣一个打印任␣改␣了主意，可以用 `lprm(1)` 将任␣从␣列中␣除。通常，␣甚至可以用 `lprm(1)` 命令来移除一个当前激活的任␣，但是任␣的一部分或者所有␣是可能打印出来。

要从默␣打印机中移除一个任␣，首先使用 `lpq(1)` ␣到任␣号。然后␣入：

```
% lprm job-number
```

要从指定打印机中␣除任␣，␣加 `-P` ␣。下面的命令会␣除␣号 10 的任␣从 `bamboo` ␣台打印机：

```
% lprm -P bamboo 10
```

`lprm(1)` 命令有一些快捷方式：

#### `lprm -`

␣除所有属于␣的任␣ (默␣打印机的)。

#### `lprm user`

␣除所有属于用␣ `user` 的任␣ (默␣打印机的)。超␣用␣可以␣除用␣的任␣；␣只可以␣除自己的任␣。

#### `lprm`

命令行中不␣任␣号，任␣名，或者 `-` ␣，`lprm(1)` 会␣除默␣打印机上当前激活的任␣，如果它属于␣。超␣用␣可以␣除任␣激活的任␣。

使用参数 `-P` 和上面的快捷方式来指定打印机替代默认打印机。例如，下面的命令会删除当前用 `rattan` 列中的所有任务：

```
% lprm -P rattan -
```

如果你正在工作在一个网络环境中，[lprm\(1\)](#) 将只允许在提交任务的主机上删除任务，甚至是同一台打印机也可以在其他主机上使用。下面的命令说明了这一点：



```
% lpr -P rattan myfile
% rlogin orchid
% lpq -P rattan
Rank  Owner      Job  Files      Total Size
active seeyan    12   ...      49123 bytes
2nd   kelly      13  myfile     12 bytes

% lprm -P rattan 13
rose: Permission denied

% logout
% lprm -P rattan 13
dfA013rose dequeued
cfA013rose dequeued
```

#### 10.5.4. 超越文本：打印

[lpr\(1\)](#) 支持许多控制文本格式的参数，包括图形和其他格式文件，生成多副本，管理任务，等等。下面将描述一些。

##### 10.5.4.1. 格式与设备

下面的 [lpr\(1\)](#) 参数控制任务中文件的格式。使用这些参数，如果任务不含文本，或者你想文本通过 [pr\(1\)](#) 格式化。

例如，下面的命令打印一个 DVI 文件（来自 TeX 排版系统）文件名 `fish-report.dvi` 到打印 `bamboo`：

```
% lpr -P bamboo -d fish-report.dvi
```

有些设备用到任务中的多个文件，所以不能混合（即）DVI 和 ditroff 文件在同一个任务中。替代的方法是，用独立的任务提交这些文件，使用不同的设备或不同的任务。



所有这些除了 `-p` 和 `-T` 都需要设备安装目标打印机。例如，`-d` 需要 DVI 设备。参考 [设备](#) 得到更多。

**-c**

打印 cifplot 文件。

**-d**

打印 DVI 文件。

**-f**

打印 FORTRAN 文本文件。

**-g**

打印 plot 数据。

**-i number**

□□ *number* 列；如果没有指定 *number*，□□□ 8 列。□个□□□可以工作在某些□□器上。



不要在□□ **-i** 和数字之□加入空格。

**-l**

打印文字数据，包括控制字符。

**-n**

打印 ditroff (无□□依□ troff) 数据。

**-p**

打印之前用 [pr\(1\)](#) 格式化□文本。参考 [pr\(1\)](#) 得到更多信息。

**-T title**

使用 *title* 在 [pr\(1\)](#) 上来替代文件名。□个□□□在使用 **-p** □□□起作用。

**-t**

打印 troff 数据。

**-v**

打印 raster 数据。

下面是一个例子：□个命令打印了一个很好的 [ls\(1\)](#) □机手册到默认□打印机：

```
% zcat /usr/shared/man/man1/ls.1.gz | troff -t -man | lpr -t
```

[zcat\(1\)](#) 命令解□□ [ls\(1\)](#) 的手册并且将内容□□ [troff\(1\)](#) 命令，它将格式化□些内容并且生成 GNU troff □出□ [lpr\(1\)](#)，它提交任□到 LPD 后台打印。因□使用了 **-t** □□□ [lpr\(1\)](#)，后台打印将会□□ GNU troff □出到默认□打印机可以理解的格式当任□被打印□。

#### 10.5.4.2. 任□理□□

下面的 [lpr\(1\)](#) □□告□ LPD □任□特殊□理：

**-# copies**

生成 *copies* 个副本□任□中的□个文件，替代□个文件一□副本。管理□可以禁止□个□□来□少打印机的浪费和鼓励□印机的使用。参考 [限制多□副本](#)。

一个例子打印三个副本的文件 `parser.c` 跟着三个副本的文件 `parser.h` 到默认打印机：

```
% lpr -#3 parser.c parser.h
```

#### **-m**

打印完成后回信。使用一个回信，LPD 系统将会发送邮件到回信的，当它完成了处理回信的任后。在信中，它将会告诉回信是否成功完成或者出了回信，并且（通常）指明是什回信。

#### **-s**

不要复制文件到后台打印队列，要使用符号链接。

如果正在打印一个很大的回信，回信可能需要一个回信。它节省后台打印队列的空（回信的任可能使后台打印队列所在的文件系统剩余空超出）。它同时也节省了回信，因为 LPD 将不会副本回信的每个字到后台打印队列。

回信也有一个缺点：因为 LPD 将直接指向源文件，回信不能修改或者删除回信直到它被打印出来。



如果打印到一台远程打印机，LPD 将最坏将文件从本地主机副本到远程主机上，所以回信 `-s` 只能节省本地后台打印队列的空，而不是远程的。尽管如此，但它还是很有用。

#### **-r**

移除回信中的文件在它被复制到后台打印队列之后，或者在用 `-s` 回信打印它之后。谨慎使用一个回信！

### 10.5.4.3. 回信回信

回信 `lpr(1)` 的回信整理了通常出现在回信回信上的文本。如果回信被跳过了在目标打印机上，回信回信将不会起作用。参考回信得到更多关于回信回信的信息。

#### **-C text**

替回信回信上的主机名 `text`。主机名通常都是提交回信的主机名称。

#### **-J text**

替回信回信上的回信名 `text`。回信名通常是回信中一个文件的名称，或者 `stdin` 如果正在打印回信回信。

#### **-h**

不打印任何回信。



在某些地点，回信回信可能无效，与回信的回信方法有回信。参考回信得到回信信息。

### 10.5.5. 管理打印机

作为一个打印机的管理者，回信回信要安装，回信回信，并且回信回信。使用 `lpc(8)` 命令，回信回信可以与打印机以更多的方式交流。用 `lpc(8)`，回信回信可以

- 回信回信或停止打印机
- 回信回信用或禁止回信回信的回信回信
- 重新安排回信回信回信回信中的回信回信。

首先，一个易于理解的解释：如果一个打印机被停止了，它将不会打印它队列中的任何东西。但用lpd是可以提交任何作业的，它会在队列中等待直到打印机被lpd或者队列被清空。

如果一个队列被禁止，没有用户（除了root）可以提交任何作业到打印机。一个用户使用的队列允许任何作业被提交。一个打印机可以被lpd但它的队列被禁止，在这种情况下打印机将打印队列中的任何作业，直到队列清空。

通常，只有root用户来使用lpc(8)命令。普通用户可以使用lpc(8)命令来得到打印机状态并且重新挂起的打印机。

这里是一个关于lpc(8)命令的摘要。大部分命令需要一个printer-name参数来知道要操作哪台打印机。可以用all填在printer-name的位置来代表所有在/etc/printcap文件中列出的打印机。

### abort printer-name

取消当前任何作业并停止打印机。用户仍然可以提交任何作业，如果队列是启用的。

### clean printer-name

从打印机的后台打印队列中移除旧的文件。有作业，或任何作业的文件没有被LPD正确删除，特别是在打印中出错或者管理活动比作业多的时候。这个命令将不属于后台打印队列中的文件并删除它们。

### disable printer-name

禁止新作业输入。如果打印机正在工作，它将会打印队列中剩余的作业。超级用户(root)是可以提交任何作业，甚至提交到一个禁止的队列。

这个命令在添加一台新打印机或者安装新设备非常有用：禁止队列并提交以root提交任何作业。其他用户将不能提交任何作业直到完成了禁用并用命令enable重新启用了队列的时候。

### down printer-name message

打印机下线。等于disable命令后跟一个stop命令。message将作为打印机状态，当用户使用lpq(1)或者lpc status命令查看打印机队列状态的时候显示出来。

### enable printer-name

打印机重新上线。用户可以提交任何作业到打印机但是在打印机重新之前不会打印出任何东西。

### help command-name

打印关于command-name命令的帮助。不带command-name，打印可用命令的摘要。

### restart printer-name

重启打印机。普通用户可以使用这个命令，当一些特殊的环境导致LPD死亡，但他不能禁用一台使用stop或者down命令停用的打印机。restart命令等同于abort后跟着一个start。

### start printer-name

启用打印机。打印机将开始打印队列中的任何作业。

### stop printer-name

停止打印机。打印机将完成当前任何作业并且将不再打印队列中的任何作业。尽管打印机被停用，用户仍然可以提交任何作业到一个禁止的队列。

### topq printer-name job-or-username

重新以printer-name安排队列，将列出的job编号或者指定的所属username的任何作业放在

列的最前面。于一个命令，不可以使用 `all` 当作 `printer-name`。

### up printer-name

打印机上；相当于 `down` 命令。等同于 `start` 后跟着一个 `enable` 命令。

`lpc(8)` 的命令行接受上面的命令。如果不入任何命令，`lpc(8)` 入一个交互模式，在里可以入命令直到入 `exit`, `quit`, 或者文件束符。

## 10.6. 替准后台打印

如果已通了个手册，那到在已了解了于 FreeBSD 包含的后台打印系 LPD 的一切。可能了它很多的缺点，它很自然的提出： "里有什么后台打印系 (并且可以工作在 FreeBSD 上) ?"

### LPRng

LPRng, 它的意思是 "LPR: 下一代", 是一个完全重写的 PLP。Patrick Powell 和 Justin Mason (PLP 的主要人) 合作完成了 LPRng。LPRng 的主站是 <http://www.lprng.org/>。

### CUPS

CUPS, 通用 UNIX 打印系, 提供了一个便的打印 UNIX®-基的操作系。它是由 Easy Software Products 的, 并且成了 UNIX® 供商和用的准打印解决方案。

CUPS 使用 Internet 打印 (IPP) 作管理打印任和列的基。行式打印机守程序 (LPD) 服器消息 (SMB), 和 AppSocket (a.k.a. JetDirect) 的部分功能也被支持。CUPS 加了基于网打印机和 PostScript 打印机描述 (PPD) 的打印来支持 UNIX® 下的真打印。

CUPS 的主站是 <http://www.cups.org/>。

### HPLIP

HPLIP, HP Linux® 成像及打印系 (Imaging and Printing system), 是一套由 HP 的用于支持 HP 的打印、描和真的工具。套程序利用 CUPS 打印系作后端来提供一些打印方面的功能。

HPLIP 的主位于 <http://hplipopensource.com/hplip-web/index.html>。

## 10.7. 疑

在使用 `lpctest(1)` 行之后，可能得到了下面的果，而不是正的果：

了一会儿，它工作了；或者，它没有退出一整。

打印机行了打印，但在之前它呆了一段而且什么都没做。事上，可能需要按一下打印机上的打印剩余 或者 送 按来果出。

如果是所在，打印机可能在等待，看看在打印之前，的任是否有更多的数据。要修正个，可以文本器送一个送字符 (或者其他需要的) 到打印机。通常足打印机立即打印出内部存内剩余的文本。它同可以用来保个任的尾都占用一整，下一个任才不会在前一个任最后一的中始。

接下来的 shell 脚本 `/usr/local/libexec/if-simple` 的脚本打印了一个送符在它送任到打印机之后：

```
#!/bin/sh
#
# if-simple - Simple text input filter for lpd
# Installed in /usr/local/libexec/if-simple
#
# Simply copies stdin to stdout. Ignores all filter arguments.
# Writes a form feed character (\f) after printing job.

/bin/cat && printf "\f" && exit 0
exit 2
```

它的输出生了 "楼梯效果"。

可能在上面得到下面这些：

```
!"#$%&'()*+,-./01234
      "$%&'()*+,-./012345
            #$%&'()*+,-./0123456
```

也成就了 "楼梯效果" 的受害者，是由新行的标志字符的解不一致造成的。UNIX® 规格的操作系使用一个字符：ASCII 10，即行 (LF)。MS-DOS®, OS/2®, 和其他的系使用一个儿字符，ASCII 10 和 ASCII 13 (回 CR)。多打印机使用 MS-DOS® 的来代表新行。

当在 FreeBSD 上打印，的文本用了行字符。打印机，打印机看到行字符后，走一行，但光位置是在上要打印的下一个字符。就是回的作用：将下一个要打印的字符的位置移到的左。

里是 FreeBSD 想要打印机做的：

|          |               |
|----------|---------------|
| 打印机收到 CR | 打印机打印 CR      |
| 打印机收到 LF | 打印机打印 CR + LF |

下面有几完成个的法：

- 使用打印机的配置或者控制面板来更改它些字符的解。看打印机的手册来到更改。



如果引到的系到其他除了 FreeBSD 之外的操作系，可能不得不重新配置打印机使用个操作系 CR 和 LF 字符的解。可能更喜下面一解决方案。

- FreeBSD 的串口自 LF 到 CR+LF。当然，工作在串口打印机上。要个功能，定 `ms#` 量并置 `onlcr` 模式在 `/etc/printcap` 文件中相打印机。
- 送一个 到打印机来它 LF 字符做不同的理。参考的打印机手册来了解的打印机支持些。当到合的，修改文本器其先送个，然后再送打印任。

里是一个得 Hewlett-Packard PCL 的打印机写的文本器。个器使得打印机将 LF 作一个 LF 和一个 CR 来待；然后它送任；最后送一个送符出任的最后一。它可以在几乎所有 Hewlett Packard 打印机上工作。

```
#!/bin/sh
#
# hpif - Simple text input filter for lpd for HP-PCL based printers
# Installed in /usr/local/libexec/hpif
#
# Simply copies stdin to stdout. Ignores all filter arguments.
# Tells printer to treat LF as CR+LF. Ejects the page when done.

printf "\033&k2G" && cat && printf "\033&l0H" && exit 0
exit 2
```

下面是一个 `/etc/printcap` 文件的例子在叫做 `orchid` 的主机上。它只有一台打印机直接接在第一个并口上，一台 Hewlett Packard LaserJet 3Si 名字叫做 `teak`。它使用上面那段脚本作为文本过滤器：

```
#
# /etc/printcap for host orchid
#
teak|hp|laserjet|Hewlett Packard LaserJet 3Si:\
    :lp=/dev/lpt0:sh:sd=/var/spool/lpd/teak:mx#0:\
    :if=/usr/local/libexec/hpif:
```

行行覆。

打印机从来不换行。所有的文本都打印在一行文本的上面。

这个就是“相反”于楼梯效果，像上面描述的那样，并且更少。一些地方，LF 这个 FreeBSD 用来结束一行的字符被作为 CR 这个将打印位置返回到行的左边的字符等待。而没有向下走一行。

使用打印机的配置或者控制面板来控制 LF 和 CR 行下面的：

| 打印机收到 | 打印机打印   |
|-------|---------|
| CR    | CR      |
| LF    | CR + LF |

打印掉字符。

当打印，一行里打印机都掉一些字符没有打。这个可能随着打印的行越重，掉越来越多的字符。

这个是由打印机跟不上计算机通过串口送数据的速度造成的（这个不会发生在并口打印机上）。有方法能克服这个：

- 如果打印机支持 XON/XOFF 流量控制，那就 FreeBSD 使用它，通过加入 `ixon` 模式在 `ms#` 量里。
- 如果打印机支持请求/清除硬件握手信号（通常 `RTS/CTS`），指定 `crtsets` 模式在 `ms#` 量里。并且要确定连接打印机和计算机的是支持硬件流量控制的。

它打印出。

打印机打印出的西看起来是一些随机的字符，而不是想要打印的西。



通常意味着——串口打印机通参数置不正的。 `br` 量中定的波特， 和 `ms#` 中的校置；  
定打印机也在使用和 `/etc/printcap` 文件中相同的置。

没有反。

如果没有反， 就可能出在 FreeBSD 而不是硬件上了。 加日志文件 (`lf`) 量到 `/etc/printcap` 文件里出打印机的。比如，下面是打印机 `rattan` 的，使用了 `lf` 量：

```
rattan|line|diablo|lp|Diablo 630 Line Printer:\
:sh:sd=/var/spool/lpd/rattan:\
:lp=/dev/lpt0:\
:if=/usr/local/libexec/if-simple:\
:lf=/var/log/rattan.log
```

然后， 再次打印。 日志文件（在我例子当中， 是 `/var/log/rattan.log` 个文件）来看是否有信息出。根据出的信息， 着来修正。

如果没有指定 `lf` 量， LPD 会使用 `/dev/console` 作默。

# Chapter 11. Linux® 二进制兼容模式

## 11.1. 概述

FreeBSD 提供了与 Linux® 32-bit 二进制兼容，允许用户在 FreeBSD 系统上安装和运行大多数的 32-bit Linux® 二进制程序而无需做任何修改。据在某些情况下，FreeBSD 上运行的 32-bit Linux® 二进制程序能有更好的表现。

然而，仍然有一些 Linux® 操作系统特有的功能在 FreeBSD 上并不被支持。例如，要是 Linux® 程序过度地使用了诸如用虚 8086 模式 i386™ 特有的调用，就无法在 FreeBSD 上运行。此外，目前不支持 64-bit 的 Linux® 二进制程序。

读完本章，你将了解到：

- 如何在 FreeBSD 系统中使用 Linux® 二进制兼容模式。
- 如何安装额外的 Linux® 共享库。
- 如何在 FreeBSD 上安装 Linux® 应用程序。
- FreeBSD 上 Linux® 兼容模式的限制。

在开始本章之前，你应该知道：

- 知道如何安装 [额外的第三方软件](#)。

## 11.2. 配置 Linux® 二进制兼容模式

默认情况下，Linux® 并没有被安装而且 Linux® 二进制兼容模式也没有被启用。Linux® 可以通过手动安装或者使用 FreeBSD 的 Ports Collection。

安装 [emulators/linux-base-f10](#) 包或者 port 是最容易在 FreeBSD 系统上获得一套基本的 Linux® 的方法。使用如下方法安装 port：

```
# cd /usr/ports/emulators/linux_base-f10
# make install distclean
```

安装完成以后，加载 **linux** 模块启用 Linux® 二进制兼容模式：

```
# kldload linux
```

查看模块是否已被加载：

```
% kldstat
Id Refs Address      Size      Name
  1     2 0xc0100000 16bdb8    kernel
  7     1 0xc24db000 d000      linux.ko
```

在 `/etc/rc.conf` 中加入以下行后 Linux® 兼容模式便会在系统启动时：

```
linux_enable="YES"
```

想要在自制内核中静默链接 Linux® 二进制兼容支持的库可以在自定义的内核配置文件中加入 `options COMPAT_LINUX`。然后按照 [配置 FreeBSD 的内核](#) 中所描述的方法并安装新内核。

### 11.2.1. 手动安装库外的库

在配置了 Linux® 兼容模式之后，如果某个 Linux® 库程序依然提示找不到共享库，需先找出此 Linux® 二进制程序需要的共享库再手动安装。

在 Linux® 系统上使用 `ldd` 找出库程序所需的共享库文件。比如，在安装有 Doom 的 Linux® 系统上运行如下的命令列出 `linuxdoom` 所需用到的共享库文件：

```
% ldd linuxdoom
libXt.so.3 (DLL Jump 3.1) => /usr/X11/lib/libXt.so.3.1.0
libX11.so.3 (DLL Jump 3.1) => /usr/X11/lib/libX11.so.3.1.0
libc.so.4 (DLL Jump 4.5p126) => /lib/libc.so.4.6.29
```

然后把上面输出中最后一列中的所有文件从 Linux® 系统复制到 FreeBSD 上的 `/compat/linux`。复制完成之后，建立指向第一列中文件名的符号链接。在 FreeBSD 系统上将会有如下的文件：

```
/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
```

如果已经有了一个与 `ldd` 输出中第一列的主修版本号相同的 Linux® 共享库文件，就不再需要复制最后那列文件，已有的共享库可以正常使用。如果是更新版本的共享库通常建议复制。只要有符号链接指向新的版本，那就可以删除旧版的了。

比如，FreeBSD 系统中有些共享库文件：

```
/compat/linux/lib/libc.so.4.6.27
/compat/linux/lib/libc.so.4 -> libc.so.4.6.27
```

并且 `ldd` 指出某个二进制程序需要之后版本：

```
libc.so.4 (DLL Jump 4.5p126) -> libc.so.4.6.29
```

既然有文件最后的版本号只相差一到两个版本，程序仍然可以正常使用旧些的版本。不管如何，使用新版本替换有 `libc.so` 都是安全的。

```
/compat/linux/lib/libc.so.4.6.29
/compat/linux/lib/libc.so.4 -> libc.so.4.6.29
```

通常最初几次在 FreeBSD 上安装 Linux® 程序需要 Linux® 二进制程序所依赖的共享文件。在此之后，系统里便会有足够的 Linux® 共享文件来运行新安装的 Linux® 二进制程序而无需额外操作。

### 11.2.2. 安装 Linux® ELF 二进制程序

ELF 二进制程序有依赖的。当未被依赖的 ELF 二进制程序被运行的时候，会生成如下的信息：

```
% ./my-linux-elf-binary
ELF binary type not known
Abort
```

借助 FreeBSD 内核分辨 FreeBSD ELF 二进制程序和 Linux® 二进制程序，使用 [brandelf\(1\)](#)：

```
% brandelf -t Linux my-linux-elf-binary
```

由于在的 GNU 工具能自动把依赖的信息写入 ELF 二进制程序中，这个通常不是必须做的。

### 11.2.3. 安装基于 Linux® RPM 的应用程序

安装基于 Linux® RPM 的应用程序，首先需要安装 [archivers/rpm](#) 包或者 port。安装好之后 root 用户就能使用此命令安装 .rpm 了：

```
# cd /compat/linux
# rpm2cpio < /path/to/linux.archive.rpm | cpio -id
```

如有必要的使用 [brandelf](#) 安装好的 ELF 二进制程序。注意此安装将无法干卸。

### 11.2.4. 配置主机名解析器

如果 DNS 不能正常工作或是出以下的信息：

```
resolv+: "bind" is an invalid keyword resolv+:
"hosts" is an invalid keyword
```

参照此方法配置 /compat/linux/etc/host.conf：

```
order hosts, bind
multi on
```

里指定了先 /etc/hosts 再 DNS。如果 /compat/linux/etc/host.conf 不存在的，Linux® 程序便会

取 `/etc/host.conf` 并提示与 FreeBSD 的语法不兼容。如果没有在 `/etc/resolv.conf` 文件中配置域名服务器，可以删除 `bind`。

## 11.3. 高主

本章将描述是 Linux® 二进制兼容如何工作的，内容基于 Terry Lambert [tlambert@primenet.com](mailto:tlambert@primenet.com) (Message ID: <199906020108.SAA07001@usr09.primenet.com>) 发表在 FreeBSD 邮件列表的邮件。

FreeBSD 有一个叫 "execution class loader" 的抽象。它被嵌入到了 `execve(2)` 系调用。

历史上 UNIX® 加载器会依靠看魔数（通常是文件的 4 至 8 个字）来判断是否是系已知的的二制程序，如果是的，就会调用二制程序加载器。

如果它不是二制型的程序，`execve(2)` 调用会返回一个，shell 会把它当作 shell 命令行。"不当前是单一 shell" 都会默认做出此假设。

随后，`sh(1)` 会读取的个字符，如果它是 `:\n`，那就用 `csh(1)`。

FreeBSD 有一加载器列表而不是一个一的加载器，并能回退到 `#!` 加载器来行 shell 解释器或者 shell 脚本。

了支持 Linux® ABI，FreeBSD 看到了二制 ELF 程序的魔数。ELF 加载器会一个用的，那是在 ELF 像中的一个注部分，此区域在 SVR4/Solaris™ ELF 二制中并不存在。

要行 Linux® 二制程序，必须先使用 `brandelf(1)` 命令 Linux 型：

```
# brandelf -t Linux file
```

当 ELF 加载器看到了 Linux 型，便会替 `proc` 型中的一个指。所有的系调用都通此指来索引。除此以外，程被以便 `signal trampoline` 代的陷向量做特殊理，有一些其他由 Linux® 内核模来理的（微）修。

Linux® 系调用向量包含一个 `sysent[]` 型的列表，它的地址位于内核模之中。

当一个系调用被 Linux® 二制程序调用，陷代会把系调用函数指从 `proc` 解引用至 Linux® 而不是 FreeBSD 的系调用入口。

Linux® 模式会地 `reroots`。与 `union` 文件系统是等效的。首先会地在 `/compat/linux/original-path` 目文件。如果失了，就会在 `/original-path` 目下。使得需要其它程序的程序得以行。例如，Linux® 工具都可以在 Linux® ABI 的支持下行。也就是 Linux® 二制程序可以加并行 FreeBSD 二制程序，如果当前没有相的 Linux® 二制程序，可以在 `/compat/linux` 目中放置一个 `uname(1)` 命令，使 Linux® 程序不易察它并没有行在 Linux® 系上。

事实上，在 FreeBSD 内核中有一个 Linux® 内核。所有由内核提供的服务的各底功能在 FreeBSD 系调用表的和 Linux® 系调用表的是一的：文件系统操作，虚内存操作，信号送，和 System V IPC。唯一不同的是 FreeBSD 会得到 FreeBSD 的 `glue` 功能，而 Linux® 程序会得到 Linux® 的 `glue` 功能。FreeBSD 的 `glue` 功能是静接入内核的，而 Linux® 的 `glue` 功能可以静接，或者通内核模。

格来其并没有真正的模，是一 ABI 的。有被称 "Linux® 模" 是因在的时候没有其他合的

用来描述。要 FreeBSD 运行 Linux® 二进制程序并不切，因当代并没有被去。

# Part III: 系管理

FreeBSD 手册中其余章的内容都是关于系管理。每一章都从描述本章将要介绍的内容开始，由浅入深地相关内容进行介绍。

有些章在撰写时，已成了许多相互独立的部分，如果你需要了解某部分内容，直接阅读部分内容即可，而无需按照顺序，也不必在开始使用 FreeBSD 之前完整地阅读它。

# Chapter 12. 配置和调整

## 12.1. 概述

使用 FreeBSD 的一个重要部分是系统配置。正确地配置系统能充分地减少以后维护和升级系统所需的工作量。本章将解释一些 FreeBSD 的配置过程，包括一些可以调整 FreeBSD 系统的一些参数。

读完本章，你将了解：

- 如何有效地利用文件系统 and 交叉分区。
- rc.conf 的基本配置以及 /usr/local/etc/rc.d 脚本体系。
- 如何配置和联网。
- 如何在公网的服务器上配置虚拟机。
- 如何使用 /etc 下的各配置文件。
- 如何通过 `sysctl` 变量来调整 FreeBSD 系统运行。
- 调整磁盘性能和修改内核限制。

在开始本章之前，你应该了解：

- 了解 UNIX® 和 FreeBSD 的基础知识 ([UNIX 基础](#))。
- 熟悉内核配置的基础知识 ([配置 FreeBSD 的内核](#))。

## 12.2. 初始配置

### 12.2.1. 分区

#### 12.2.1.1. 基本分区

当使用 `bsdlabel(8)` 或者 `sysinstall(8)` 来分割新的文件系统的时候，要记住硬盘驱动器外磁道数据要比从内磁道数据快。因此应将小的和常用的文件系统放在驱动器外的位置，一些大的分区比如 /usr 应放在磁盘内部的位置。以相似的顺序建立分区是一个不错的主意：root, swap, /var, /usr。

/var 分区的大小能反映你的机器使用情况。/var 文件系统用来存文件，日志文件和打印队列等，特别是邮箱和日志文件可能会达到无法预料的大小，这主要取决于在你的系统上有多少用你的日志文件可以保存多久。大多数用户很少需要 /var 有 1GB 以上的可用空间。



有时候 /var/tmp 需要很多的磁盘空间。在使用 `pkg_add(1)` 安装新的软件包，包管理工具会在 /var/tmp 中解包出一堆文件。大的软件包，像 Firefox, OpenOffice 或者 LibreOffice 在安装时如果 /var/tmp 中没有足够的空间就可能需要一些技巧了。

/usr 分区存很多用来系统运行所需要的文件例如 `ports(7)` (建库) 和源代码 (可编译的)。ports 和基本系统的源代码在安装时都是可编译的，但我建议每个分区至少保留 2GB 的可用空间。

当分区大小的时候，记住保留一些空间。用完了一个分区的空间而在另一个分区上还有很多，可能会导致出一些问题。





一些用会 `sysinstall(8)` 的 `Auto-defaults` 自分区有会分配 `/var` 和 `/` 小的分区空。分区精一些并且大一些。

### 12.2.1.2. 交分区

一般来，交分区大是系内存 (RAM) 的倍。例如，如果机器有 128M 内存，交文件是 256M。小内存的系可以通多一点地交分区来提升性能。不建小于 256 兆的交分区，并且充的内存被考一下。当交分区最少是主内存的倍的候，内核的 VM (虚内存) 面度算法可以将性能整到最好。如果机器添加更多内存，配置太小的交分区会致 VM 面描的代效率低下。

在使用多 SCSI 磁(或者不同控制器上的 IDE 磁)的大系上，建在个器上建立交分区(直到四个器)。交分区大小一大小。内核可以使用任意大小，但内部数据是最大交分区的 4 倍。保持交分区同的大小，可以允内核最佳地度交空来磁。即使不太使用，分配大的交分区也是好的，在被迫重之前它可以更容易的从一个失的程序中恢来。

### 12.2.1.3. 什要分区？

一些用个独的大分区将会很好，但是有很多原因会明什是个坏主意。首先，个分区有不同的分区特性，因此分可以文件系整它。例如，根系和 `/usr` 一般只是取，写入很少。很多写繁的被放在 `/var` 和 `/var/tmp` 中。

当的分一个系，在其中使用小的分区，，那些以写主的分区将不会比以主分区付出更高的代价。将以写主的分区放在近磁的，例如放在的大硬的前面代替放在分区表的后面，将会提高需要的分区的 I/O 性能。在可能也需要在比大的分区上有很好的 I/O 性能，把他移到磁外不会来多大的性能提升，反而把 `/var` 移到外面会有很好的效果。最后及到安全。一个主要是只的小的、整的根分区可以提高从一个重的系崩中恢来的机会。

## 12.3. 核心配置

系的配置信息主要位于 `/etc/rc.conf`。个文件包含了配置信息很大的一部分，主要在系的时候来配置系，这个名字直接明了点；它也是 `rc*` 文件的配置信息。

系管理在 `rc.conf` 文件中建立来覆 `/etc/defaults/rc.conf` 中的默置。个默文件不被逐字的制到 `/etc` —— 它包含的是默而不是一个例子。所有特定的改在 `rc.conf` 中。

在集群用中，了降低管理成本，可以采用多策略把及全站的置从特定于系的置中分出来。推的方法是把系的配置放到 `/etc/rc.conf.local` 文件中。例如：

- `/etc/rc.conf`:

```
sshd_enable="YES"
keyrate="fast"
defaultrouter="10.1.1.254"
```

- `/etc/rc.conf.local`:

```
hostname="node1.example.org"
ifconfig_fxp0="inet 10.1.1.1/8"
```

rc.conf 文件可以通过 `rsync` 或类似的程序来分发到所有的机器上，而各自的 rc.conf.local 文件保持不同。

使用 `sysinstall(8)` 或者 `make world` 来升级系统不会覆盖 rc.conf 文件，所以系统配置信息不会丢失。



配置文件 `/etc/rc.conf` 是通过 `sh(1)` 解析的。这使得系统管理可以在其中添加一些内容，从而构建能够非常复杂的场景的配置。参见手册 `rc.conf(5)` 来了解关于它的一些信息。

## 12.4. 应用程序配置

典型的，被安装的应用程序有他自己的配置文件、方法等等。从基本系统中分给他是很重要的以至于他可以容易被 package 管理工具定位和管理

一般来，这些文件被安装在 `/usr/local/etc`。个例子中，一个应用程序有很多配置文件并且构建了一个子目录来存放他们。

通常，当一个 port 或者 package 被安装的时候，配置文件示例也同时被安装了。它通常用 `.default` 的后缀来命名。如果不存在这个应用程序的配置文件，它会自动复制 `.default` 文件来构建。

例如，看一下这个目录下的内容 `/usr/local/etc/apache`：

```
-rw-r--r--  1 root  wheel   2184 May 20  1998 access.conf
-rw-r--r--  1 root  wheel   2184 May 20  1998 access.conf.default
-rw-r--r--  1 root  wheel   9555 May 20  1998 httpd.conf
-rw-r--r--  1 root  wheel   9555 May 20  1998 httpd.conf.default
-rw-r--r--  1 root  wheel  12205 May 20  1998 magic
-rw-r--r--  1 root  wheel  12205 May 20  1998 magic.default
-rw-r--r--  1 root  wheel   2700 May 20  1998 mime.types
-rw-r--r--  1 root  wheel   2700 May 20  1998 mime.types.default
-rw-r--r--  1 root  wheel   7980 May 20  1998 srm.conf
-rw-r--r--  1 root  wheel   7933 May 20  1998 srm.conf.default
```

文件大小显示了只有 `srm.conf` 改变了。以后 Apache 的升级就不会改变这个文件。

## 12.5. 服务

多用户会使用 Ports Collection 来在 FreeBSD 上安装第三方软件。很多情况下可能需要进行一些配置以便这些软件能在系统初始化的过程中运行。服务，例如 [mail/postfix](#) 或 [www/apache13](#) 就是那些需要在系统初始化时安装的软件包中的个典型代表。了解了解第三方软件所需要的。

FreeBSD 包含的大多数服务，例如 `cron(8)`，就是通过系统脚本实现的。这些脚本也会有些不同，取决于 FreeBSD 版本。但是不管如何，需要考虑的一个重要方面是他们的配置文件要能被基本脚本捕获。

### 12.5.1. 扩展用程序配置

在 FreeBSD 提供了 `rc.d`，使得用件的行配置得更加方便，并提供了更多的其他功能。例如，使用在 [rc.d](#) 一中所介绍的，用程序就可以置在某些其他服务，例如 DNS 之后；除此之外，可以通过 `rc.conf` 来指定一些外的参数，而不再需要将它硬到脚本中。基本的脚本如下所示：

```
#!/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"
```

这个脚本将保证 `utility` 能在 **DAEMON** 服务之后。它同时也提供了置和跟踪 PID，也就是进程 ID 文件的方法。

可以在 `/etc/rc.conf` 中加入：

```
utility_enable="YES"
```

这个方法也使得命令行参数、包含 `/etc/rc.subr` 中所提供的功能，兼容 [rcorder\(8\)](#) 工具并提供更的通 `rc.conf` 文件来配置的方法。

### 12.5.2. 用服务来管理服务

其他服务，例如 POP3 服务器，IMAP，等等，也可以通过 [inetd\(8\)](#) 来。一程包括从 Ports Collection 安装相应的用程序，并把配置加入到 `/etc/inetd.conf` 文件，或去掉当前配置中的某些注。如何使用和配置 `inetd` 在 [inetd](#) 一中行了更深入的述。

一些情况下，通 [cron\(8\)](#) 来系服务也是一可行的。方法有很多好，因 **cron** 会以 `crontab` 的文件属主身行那些程。使得普通用也能行他的用。

`cron` 工具提供了一个独有的功能，以 `@reboot` 来指定。它的配置将在 `cron(8)` 中描述，通常也是系统初始化的时候。

## 12.6. 配置 `cron`

FreeBSD 最有用的软件包(utilities)中的一个就是 `cron(8)`。`cron` 软件在后台运行并且常驻 `/etc/crontab` 文件。`cron` 软件也在 `/var/cron/tabs` 目录，搜索新的 `crontab` 文件。这些 `crontab` 文件存有一些 `cron` 在特定时间运行的信息。

`cron` 程序使用两种不同类型的配置文件，即系统 `crontab` 和用户 `crontabs`。两种格式的唯一区别是第六个字段。在系统 `crontab` 中，第六个字段是用于运行命令的用户名。它赋予了系统 `crontab` 以任意用户身份运行命令的能力。在用户 `crontab` 中，第六个字段是要运行的命令，所有的命令都会以用户自己的身份运行；这是一项重要的安全功能。



同其他用户一样，`root` 用户也可以有自己的 `crontab`。它不同于 `/etc/crontab` (也就是系统 `crontab`)。由于有系统 `crontab` 的存在，通常并不需要 `root` 建立自己的用户 `crontab`。

我们来看一下 `/etc/crontab` 文件：

```
# /etc/crontab - root's crontab for FreeBSD
#
# $FreeBSD: src/etc/crontab,v 1.32 2002/11/22 16:13:39 tom Exp $
#①
#
SHELL=/bin/sh
PATH=/etc:/bin:/sbin:/usr/bin:/usr/sbin ②
HOME=/var/log
#
#
#minute hour    mday    month    wday     who command ③
#
#
*/5 * * * * root    /usr/libexec/atrun ④
```

- ① 像大多数 FreeBSD 配置文件一样，`#` 字符是注释。因此，就可以用注释来写明要执行什么操作，以及所做的原因。需要注意的是，注释必须起一行，而不能跟命令放在同一行上，否则它会被看成命令的一部分。这个文件中的空行会被忽略。
- ② 首先确定环境变量。等号(=)字符用来定义任何环境变量，像这个例子用到了 `SHELL`、`PATH` 和 `HOME` 变量。如果 `shell` 行被忽略掉，`cron` 将会用默认 `sh`。如果 `PATH` 变量被忽略，那就意味着没有默认并且需要指定文件的位置。如果 `HOME` 被忽略，`cron` 将用运行者的 `home` 目录。
- ③ 下一行定义了七个字段。它是 `minute`、`hour`、`mday`、`month`、`wday`、`who` 和 `command`。它们差不多已表明了各自的用途。Minute 是命令要运行的分钟，Hour 跟 minute 差不多，只是用小写字母来表示。Mday 是个月的天。Month 跟 hour 跟 minute 都差不多，用月来表示。wday 字段表示星期几。所有这些字段的值必须是数字并且用24小时制来表示。"who" 字段是特殊的，并且只在 `/etc/crontab` 文件中存在。这个字段指定了命令将以哪个用户的身份来运行。当一个用户添加了他的 `crontab` 文件的时候，他就没有这个字段。最后，是 `command` 字段。它是最后的一个字段，所以自然就是它指定要运行的程序。
- ④ 最后一行定义了上面所定义的。注意这里我有一个 `/5` 列表，它跟着是一些字符。`*` 字符代表"从开始到最后"，

也可以被解成 0 次。所以，根据 0 行， 0 然表明了无 0 在何 0 隔 5 分 0 以 root 身 0 来 0 行 atrun 命令。 0 看 atrun(8) 手册 0 以 0 得 atrun 的更多信息。命令可以有任意多个 0 0 0 它的 0 志。无 0 0 0， 0 展到多行的命令 0 0 用反斜 0 ("") 来 0 行。

0 是 0 个 crontab 文件的基本 0 置， 0 然它 0 有一个不同。第六行我 0 指定的用 0 名只存在于系 0 /etc/crontab 文件。 0 个字段在普通用 0 的 crontab 文件中 0 被忽略。

### 12.6.1. 安装 Crontab



0 0 不要用 0 方法来 0 0 /安装系 0 crontab。 0 需要做的只是使用自己喜 0 的 0 0 器： cron 程序会注意到文件 0 生了 0 化， 并立即 0 始使用新的版本。参 0 0 个 FAQ 0 目 以了解 0 一 0 的情况。

要安装 0 写好的用 0 crontab， 首先使用最 0 的 0 0 器来 0 建一个符合要求格式的文件，然后用 cronstab 程序来完成。最常 0 的用法是：

```
% crontab crontab-file
```

在前面的例子中， crontab-file 是一个事先写好的 crontab。

0 有一个 0 0 用来列出安装的 crontab 文件： 只要 0 0 -l 0 0 0 crontab 然后看一下 0 出。

用 0 想不用模板(已 0 存在的文件)而直接安装他的 crontab 文件，用 cronstab -e 0 0 也是可以的。 它将会 0 0 一个 0 0 器并且 0 建一个新文件，当 0 个文件被保存的 0 候， 它会自 0 的用 cronstab 来安装 0 个文件。

如果 0 0 后想要 0 底 0 除自己的用 0 crontab 可以使用 cronstab 的 -r 0 0。

## 12.7. 在 FreeBSD 中使用 rc

在 2002 年， FreeBSD 整合了来自 NetBSD 的 rc.d 系 0， 并通 0 它来完成系 0 的初始化工作。用 0 要注意在 /etc/rc.d 目 0 下的文件。 0 里面的 0 多文件是用来管理基 0 服 0 的， 它 0 可以通 0 start、 stop， 以及 restart 0 0 来控制。 0 例来 0， sshd(8) 可以通 0 下面的命令来重 0：

```
# /etc/rc.d/sshd restart
```

0 其它服 0 的操作与此 0 似。 当然， 0 些服 0 通常是在 0 0 0 根据 rc.conf(5) 自 0 0 0 的。 例如， 要配置使系 0 0 0 0 0 网 0 地址 0 0 服 0， 可以 0 0 地通 0 在 /etc/rc.conf 中加入如下 0 置来完成：

```
natd_enable="YES"
```

如果 natd\_enable="NO" 行已 0 存在， 只要 0 0 的把 NO 改成 YES 即可。 rc 脚本在下次重新 0 0 的 0 候会自 0 的装 0 所需要的服 0， 像下面所描述的那 0。

由于 rc.d 系 0 在系 0 0 0 0 0 0 首先 0 0 /停止服 0， 如果 0 置了 0 当的 /etc/rc.conf 0 量， 0 准的 start、 stop 和 restart 0 0 将会 0 行他 0 的 0 作。例如 sshd restart 命令只在 /etc/rc.conf 中的 sshd\_enable 0 置成 YES 的 0 候工作。不管是否在 /etc/rc.conf 中 0 置了， 要 start、 stop 或者 restart 一个服 0

, 命令前可以加上一个"one"前。例如要不当前 /etc/rc.conf 的置重新 sshd, 行下面的命令：

```
# /etc/rc.d/sshd onerestart
```

用 rcvar 可以来的 /etc/rc.conf 中用当的 rc.d 脚本的服是否被用。从而管理可以行的程序来 sshd 是否真的在 /etc/rc.conf 中被了：

```
# /etc/rc.d/sshd rcvar
# sshd
$sshd_enable=YES
```



第二行 (# sshd) 是从 sshd 命令中出的, 而不是 root 控制台。

了定一个服是否真的在行, 可以用 status 。例如 sshd 是否真的了：

```
# /etc/rc.d/sshd status
sshd is running as pid 433.
```

有些候也可以 reload 服。一操作上是向服送一个信号, 来制其重新加配置。多数情况下, 服的是 SIGHUP 信号。并非所有服都支持一功能。

rc.d 系不用于网服, 它也系初始化中的多数程提供支持。比如 bgfsck 文件, 当它被行, 将会出下述信息：

```
Starting background file system checks in 60 seconds.
```

个文件用做后台文件系, 系初始化的候完成。

很多系服依其他服提供的相功能。例如, NIS 和其他基于 RPC 的服可能在 rpcbind 服之前失。要解决个, 依系信息和其他信息当作注被包含在个脚本文件的前面。程序在系初始化分析些注以决定用其他系服来足依系。

下面的字句必被包含在所有的脚本文件里, (他都是 rc.subr(8) 用来 "enable" 脚本必需的)：

- **PROVIDE:** 指定此文件所提供的服的名字。

以下的字句可以被包含在文件的部。格来他不是必需的, 但作于 rcorder(8) 有一定的提示作用：

- **REQUIRE:** 列出此服之前所需要的其他服。此脚本提供的服会在指定的那些服之后 。
- **BEFORE:** 列出依此服的其他服。此脚本提供的服将在指定的那些服之前 。

通在脚本中仔定些字, 系管理可以很有条理的控制脚本的序, 而避免使用像其他 UNIX® 操作系那混乱的 "runlevels"。

更多关于 rc.d 系的信息, 可以在 rc(8) 和 rc.subr(8) 机手册中到。如果有意撰写自己的 rc.d 脚本, 或有的脚本行一些改, 也可以参考 篇文章。

## 12.8. 置网

在我不可想象一台计算机没有网接的情况。添加和配置一网是任何 FreeBSD 系管理的一基本任。

### 12.8.1. 正的程序

在始之前，知道网的型，它用的芯片和它是 PCI 是 ISA 网。FreeBSD 支持很多 PCI 和 ISA 网。可以看的版本硬件兼容性列表以定网的被支持。

系能支持的网之后，需要它合的程序。 /usr/src/sys/conf/NOTES 和 /usr/src/sys/arch/conf/NOTES 将提供所支持的一些网和芯片的信息。如果疑程序是否使所要的那个，参考程序的机手册。机手册将提供于所支持的硬件更的信息，甚至包括可能生的。

如果的网很常的，大多数候不需要浪精力。常用的网在 GENERIC 内核中已支持了，所以的网在就会示出来，像是：

```
dc0: <82c169 PNIC 10/100BaseTX> port 0xa000-0xa0ff mem 0xd3800000-0xd38000ff irq 15 at device 11.0 on pci0
miibus0: <MII bus> on dc0
bmtphy0: <BCM5201 10/100baseTX PHY> PHY 1 on miibus0
bmtphy0: 10baseT, 10baseT-FDX, 100baseTX, 100baseTX-FDX, auto
dc0: Ethernet address: 00:a0:cc:da:da:da
dc0: [ITHREAD]
dc1: <82c169 PNIC 10/100BaseTX> port 0x9800-0x98ff mem 0xd3000000-0xd30000ff irq 11 at device 12.0 on pci0
miibus1: <MII bus> on dc1
bmtphy1: <BCM5201 10/100baseTX PHY> PHY 1 on miibus1
bmtphy1: 10baseT, 10baseT-FDX, 100baseTX, 100baseTX-FDX, auto
dc1: Ethernet address: 00:a0:cc:da:da:db
dc1: [ITHREAD]
```

在这个例子中，我看到有使用 [dc\(4\)](#) 的网在系中。

如果的网没有出在 GENERIC 中，需要手工加合的程序。要完成工作可以使用下面方法之一：

- 最的方法是[用 kldload\(8\)](#) 加网的内核模。除此之外，通在 /boot/loader.conf 文件中加入当的置，也可以系在引自加些模。不，并不是所有的网都能通方法提供支持；ISA 网是比典型的例子。
- 外，也可以将网的支持静内核。察看 /usr/src/sys/conf/NOTES, /usr/src/sys/arch/conf/NOTES 以及程序的机手册以了解需要在的内核配置文件中加一些什。要了解于重新内核的一，参[配置FreeBSD的内核](#)。如果的在引可以被内核 (GENERIC)，不需要新的内核。

#### 12.8.1.1. 使用 Windows® NDIS 程序

不幸的是，多厂商由于程序会及多敏感的商机密，至今仍不意将把程序作放源代形式布列入他的表。因此，FreeBSD 和其他操作系的者就只剩下了：要的痛苦程来



进行逆向工程，要使用存在的 Microsoft® Windows® 平台提供的版本的程序。包括参与 FreeBSD 的许多人，都用了后一方法。

得益于 Bill Paul (wpaul) 的工作，已可以 "直接地" 支持 网接口标准 (NDIS, Network Driver Interface Specification) 了。FreeBSD NDISulator (也被称 Project Evil) 可以支持二进制形式的 Windows® 程序，并它相信正在行的是 Windows®。由于 [ndis\(4\)](#) 使用的是用于 Windows® 的二进制形式的，因此它只能在 i386™ 和 amd64 系上使用。



[ndis\(4\)](#) 在主要提供了 PCI、CardBus 和 PCMCIA 的支持，而 USB 目前还没有提供支持。

要使用 NDISulator，需要三件西：

1. 内核的源代码
2. 二进制形式的 Windows® XP 程序 (扩展名 .SYS)
3. Windows® XP 程序配置文件 (扩展名 .INF)

需要到用于的些文件。一般而言，些文件可以在随附送的 CD 或制造商的网站上到。在下面的例子中，我用 W32DRIVER.SYS 和 W32DRIVER.INF 来表示些文件。



不能在 FreeBSD/amd64 上使用 Windows®/i386 程序。必使用 Windows®/amd64 才能在其上正常工作。

接下来的是将二进制形式的程序装成内核模。要完成一任，需要以 **root** 用的身行 [ndisgen\(8\)](#)：

```
# ndisgen /path/to/W32DRIVER.INF /path/to/W32DRIVER.SYS
```

[ndisgen\(8\)](#) 是一个交互式的程序，它会提示入所需的一些其他的外信息；些工作完成之后，它会在当前目生成一个内核模文件，个文件可以通过下述命令来加：

```
# kldload ./W32DRIVER_SYS.ko
```

除了生成的内核模之外，必加 `ndis.ko` 和 `if_ndis.ko` 个内核模，在加需要 [ndis\(4\)](#) 的模，通常系会自完成一操作。如果希望手工加它，可以使用下列命令：

```
# kldload ndis
# kldload if_ndis
```

第一个命令会加 NDIS 袖珍端口封装模，而第二条命令加的网接口。

在看看 [dmesg\(8\)](#) 来了解是否生了。如果一切正常，会看到似下面的出：



```
ndis0: <Wireless-G PCI Adapter> mem 0xf4100000-0xf4101fff irq 3 at device 8.0 on pci1
ndis0: NDIS API version: 5.0
ndis0: Ethernet address: 0a:b1:2c:d3:4e:f5
ndis0: 11b rates: 1Mbps 2Mbps 5.5Mbps 11Mbps
ndis0: 11g rates: 6Mbps 9Mbps 12Mbps 18Mbps 36Mbps 48Mbps 54Mbps
```

之后，就可以像使用其它网口接口（例如 dc0）一样来使用 ndis0 了。

与任何其它模块一样，它也可以配置系统，令其在启动时加载 NDIS 模块。首先，将生成的模块 W32DRIVER\_SYS.ko 复制到 /boot/modules 目录中。接下来，在 /boot/loader.conf 中加入：

```
W32DRIVER_SYS_load="YES"
```

## 12.8.2. 配置网口

在正确的网口驱动程序已安装，那就配置它了。跟其他配置一样，网口可以在安装时用 sysinstall 来配置。

要显示系统上的网口接口的配置，输入下列命令：

```
% ifconfig
dc0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
    options=80008<VLAN_MTU,LINKSTATE>
    ether 00:a0:cc:da:da:da
    inet 192.168.1.3 netmask 0xffffffff broadcast 192.168.1.255
    media: Ethernet autoselect (100baseTX <full-duplex>)
    status: active
dc1: flags=8802<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
    options=80008<VLAN_MTU,LINKSTATE>
    ether 00:a0:cc:da:da:db
    inet 10.0.0.1 netmask 0xffffffff broadcast 10.0.0.255
    media: Ethernet 10baseT/UTP
    status: no carrier
plip0: flags=8810<POINTOPOINT,SIMPLEX,MULTICAST> metric 0 mtu 1500
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> metric 0 mtu 16384
    options=3<RXCSUM,TXCSUM>
    inet6 fe80::1%lo0 prefixlen 64 scopeid 0x4
    inet6 ::1 prefixlen 128
    inet 127.0.0.1 netmask 0xff000000
    nd6 options=3<PERFORMNUD,ACCEPT_RTADV>
```

在这个例子中，显示了下列内容：

- dc0: 第一个以太网接口
- dc1: 第二个以太网接口
- plip0：并口（如果系统中有并口的话）
- lo0: 回环口

FreeBSD 使用内核引出的网程序来命名网。例如 `sis2` 是系中使用 `sis(4)` 的第三网。

在个例子中，`dc0` 用了。主要表在：

1. `UP` 表示网已配置完成准工作。
2. 网有一个 Internet (`inet`) 地址 (个例子中是 `192.168.1.3`)。
3. 它有一个有效的子网掩 (`netmask` ; `0xffffffff` 等同于 `255.255.255.0`)。
4. 它有一个有效的广播地址 (个例子中是 `192.168.1.255`)。
5. 网的 MAC (`ether`) 地址是 `00:a0:cc:da:da:da`
6. 物理媒介模式于自状态 (`media: Ethernet autoselect (100baseTX <full-duplex>)`)。我看到 `dc1` 被配置成行在 `10baseT/UTP` 模式下。要了解媒介型的更多信息，看它的使用手册。
7. 状态 (`status`) 是 `active`，也就是接信号被到了。于 `dc1`，我看到 `status: no carrier`。通常是网没有好。

如果 `ifconfig(8)` 的输出示了似于：

```
dc0: flags=8843<BROADCAST,SIMPLEX,MULTICAST> metric 0 mtu 1500
      options=80008<VLAN_MTU,LINKSTATE>
      ether 00:a0:cc:da:da:da
      media: Ethernet autoselect (100baseTX <full-duplex>)
      status: active
```

的信息，那就是没有配置网。

要配置网，需要 `root` 限。网配置可以通过使用 `ifconfig(8)` 命令行方式来完成，但是每次都要做一遍。放置网配置信息的文件是 `/etc/rc.conf`。

用自己喜的器打 `/etc/rc.conf`。并且需要一系中存在的网添加一行，在我个例子中，添加如下几行：

```
ifconfig_dc0="inet 192.168.1.3 netmask 255.255.255.0"
ifconfig_dc1="inet 10.0.0.1 netmask 255.255.255.0 media 10baseT/UTP"
```

用自己正的名和地址来替例子中的 `dc0`, `dc1` 等内容。网和 `ifconfig(8)` 的手册来了解各，也要看一下 `rc.conf(5)` 助来了解 `/etc/rc.conf` 的法。

如果在安装的时候配置了网，于网的一些行可能已存在了。所以在添加新行前仔一下 `/etc/rc.conf`。

也可能需要 `/etc/hosts` 来添加局域网中不同的机器名称和 IP 地址，如果它不在那里的。看看机手册 `hosts(5)` 和 `/usr/shared/examples/etc/hosts` 以了解更多信息。

如果通过这台机器访问 Internet，可能需要手工配置默认网关和域名解析服务器：



```
# echo 'defaultrouter="your_default_router"' >> /etc/rc.conf
# echo 'nameserver your_DNS_server' >> /etc/resolv.conf
```

### 12.8.3. 重启和配置

`/etc/rc.conf` 做了必要的修改之后重启系统以应用接口的修改，并且系统重启后没有任何配置。另外也可以重启网络系统：

```
# /etc/rc.d/netif restart
```



如果在 `/etc/rc.conf` 中配置了默认网关，需要运行下面的命令：

```
# /etc/rc.d/routing restart
```

网络系统重启之后，刷新网络接口。

#### 12.8.3.1. 以太网

一旦网络被正确的配置了，在这里我要做两件事情。首先，ping 自己的网络接口，接着 ping 局域网内的其他机器。

首先本地接口：

```
% ping -c5 192.168.1.3
PING 192.168.1.3 (192.168.1.3): 56 data bytes
64 bytes from 192.168.1.3: icmp_seq=0 ttl=64 time=0.082 ms
64 bytes from 192.168.1.3: icmp_seq=1 ttl=64 time=0.074 ms
64 bytes from 192.168.1.3: icmp_seq=2 ttl=64 time=0.076 ms
64 bytes from 192.168.1.3: icmp_seq=3 ttl=64 time=0.108 ms
64 bytes from 192.168.1.3: icmp_seq=4 ttl=64 time=0.076 ms

--- 192.168.1.3 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.074/0.083/0.108/0.013 ms
```

在我这里 ping 局域网内的其他机器：

```
% ping -c5 192.168.1.2
PING 192.168.1.2 (192.168.1.2): 56 data bytes
64 bytes from 192.168.1.2: icmp_seq=0 ttl=64 time=0.726 ms
64 bytes from 192.168.1.2: icmp_seq=1 ttl=64 time=0.766 ms
64 bytes from 192.168.1.2: icmp_seq=2 ttl=64 time=0.700 ms
64 bytes from 192.168.1.2: icmp_seq=3 ttl=64 time=0.747 ms
64 bytes from 192.168.1.2: icmp_seq=4 ttl=64 time=0.704 ms

--- 192.168.1.2 ping statistics ---
5 packets transmitted, 5 packets received, 0 packet loss
round-trip min/avg/max/stddev = 0.700/0.729/0.766/0.025 ms
```

如果配置了 `/etc/hosts` 文件，也可以用机器名来替 `192.168.1.2`。

### 12.8.3.2. 网络

硬件和软件配置一直是一件痛苦的事情，从最开始的开始可以有一些痛苦。例如网络是否好了？是否配置好了网络服务？防火墙配置正确？是否使用了被 FreeBSD 支持的网卡？在发送警告之前，先看一下硬件说明，升级到 FreeBSD 到最新的 STABLE 版本，看一下设备列表或者在 Internet 上搜索一下。

如果网络工作了，但性能低下，好好看一下 [tuning\(7\)](#) 手册。也可以看一下网络配置，不正确的配置会导致慢速的网络连接。

一些用户可能会在一些网卡上遇到一次 `device timeouts`，这通常是正常现象。如果经常甚至引起麻烦，确定一下它跟其他设备没有冲突。仔细检查网络连接，或者换一个网卡。

有用用户会看到少量 `watchdog timeout` 消息。这种情况要做的第一件事就是检查连接。一些网卡需要支持固件控制的 PCI 插槽。在一些老的主板上，只有一个 PCI 插槽支持（一般是 slot 0）。网卡和主板说明来判定是不是这个问题。

`No route to host` 通常产生在如果系统不能送一个路由到目的主机的包的候。在没有指定默认路由或者网卡没有插上会产生。用 `netstat -rn` 的输出并看有一个有效的路由能到相关的主机。如果没有，看看 [高网络用](#)。

`ping: sendto: Permission denied` 信息通常由防火墙的配置引起。如果 `ipfw` 在内核中用了但是没有定义，那默认的就是拒绝所有通信，甚至 ping 请求！看看 [防火墙](#) 以了解更多信息。

有网卡性能低下或者低于平均水平，情况最好把媒介模式从 `autoselect` 改成正确的媒介模式。通常大多数硬件有用，但可能不会解决所有人的问题。接着，设置所有网络配置，并且看看 [tuning\(7\)](#) 手册。

## 12.9. 虚拟主机

FreeBSD 的一个很普通的用途是虚拟主机站点，一个服务器虚拟成很多服务器——提供网络服务。通常在一个接口上绑定多个网络地址来。

一个特定的网络接口有一个“真”的地址，也可能有一些“伪”地址。这些伪名通常用 `/etc/rc.conf` 中的配置来添加。

一个 `fxp0` 的伪名类似于：

```
ifconfig_fxp0_alias0="inet xxx.xxx.xxx.xxx netmask xxx.xxx.xxx.xxx"
```

每个接口名必须从 `alias0` 开始并且按顺序（例如 `_alias1`、`_alias2`）。配置程序将会停止在第一个缺少的数字的地方。

计算接口名的子网掩码是很重要的，幸运的是它很简单。对于一个接口来说，必须有一个描述子网掩码的地址。任何在一个网段下的地址必须有一个全是 1 的子网掩码（通常表示为 `255.255.255.255` 或 `0xffffffff`）。

例如，假如使用 `fxp0` 接口连接到网络，分配的是 `10.1.1.0`，其子网掩码为 `255.255.255.0`，以及 `202.0.75.16`，其子网掩码为 `255.255.255.240`。我希望从 `10.1.1.1` 到 `10.1.1.5` 以及从 `202.0.75.17` 到 `202.0.75.20` 的地址能互相通信。如前所述，只有在一个网段中的第一个地址（本例中，`10.0.1.1` 和 `202.0.75.17`）使用真的子网掩码；其余的（`10.1.1.2` 到 `10.1.1.5` 以及 `202.0.75.18` 到 `202.0.75.20`）必须配置使用 `255.255.255.255` 作为子网掩码。

下面是根据上述描述所行的 `/etc/rc.conf` 配置：

```
ifconfig_fxp0="inet 10.1.1.1 netmask 255.255.255.0"
ifconfig_fxp0_alias0="inet 10.1.1.2 netmask 255.255.255.255"
ifconfig_fxp0_alias1="inet 10.1.1.3 netmask 255.255.255.255"
ifconfig_fxp0_alias2="inet 10.1.1.4 netmask 255.255.255.255"
ifconfig_fxp0_alias3="inet 10.1.1.5 netmask 255.255.255.255"
ifconfig_fxp0_alias4="inet 202.0.75.17 netmask 255.255.255.240"
ifconfig_fxp0_alias5="inet 202.0.75.18 netmask 255.255.255.255"
ifconfig_fxp0_alias6="inet 202.0.75.19 netmask 255.255.255.255"
ifconfig_fxp0_alias7="inet 202.0.75.20 netmask 255.255.255.255"
```

# 12.10. 配置文件

## 12.10.1. /etc 布局

在配置信息中有很多的目录，一些包括：

|                                  |   |
|----------------------------------|---|
| <code>/etc</code>                | 一般的系统配置信息。其中的数据是与特定系统相关的。   |
| <code>/etc/defaults</code>       | 系统配置文件的默认版本。  |
| <code>/etc/mail</code>           | 除了 <a href="#">sendmail(8)</a> 配置信息，其他 MTA 配置文件。                        |
| <code>/etc/ppp</code>            | 用于用户和内核 <code>ppp</code> 程序的配置。   |
| <code>/etc/namedb</code>         | <a href="#">named(8)</a> 数据的默认位置。通常 <code>named.conf</code> 和区域文件存放在这里。 |
| <code>/usr/local/etc</code>      | 被安装的用户程序配置文件。可以参考每个用户程序的子目录。  |
| <code>/usr/local/etc/rc.d</code> | 被安装程序的启动/停止脚本。  |
| <code>/var/db</code>             | 特定系统自生成的数据文件，像 <code>package</code> 数据库，位置数据库等等。                        |

12.10.2. 主机名

12.10.2.1. /etc/resolv.conf

/etc/resolv.conf 指示了 FreeBSD 如何解析域名系统(DNS)。

resolv.conf 中最常见的选项是：

|            |                           |
|------------|---------------------------|
| nameserver | 按顺序要解析的名字服务器的 IP 地址，最多三个。 |
| search     | 搜索机器名的列表。通常由本地机器名的域决定。    |
| domain     | 本地域名。                     |

一个典型的 resolv.conf 文件：

```
search example.com
nameserver 147.11.1.11
nameserver 147.11.100.30
```



只能使用一个 search 和 domain 选项。

如果你在使用 DHCP, [dhclient\(8\)](#) 经常使用从 DHCP 服务器接受来的信息重写 resolv.conf。

12.10.2.2. /etc/hosts

/etc/hosts 是 Internet 早期使用的一个纯文本数据库。它符合 DNS 和 NIS 提供名字到 IP 地址的映射。通常局域网连接的机器可以用自己的命名方案来替代设置一个 [named\(8\)](#) 服务器。另外, /etc/hosts 也可以提供一个 Internet 名称的本地解析以减轻需要从外部解析来的负担。

```
# $FreeBSD$
#
#
# Host Database
#
# This file should contain the addresses and aliases for local hosts that
# share this file. Replace 'my.domain' below with the domainname of your
# machine.
#
# In the presence of the domain name service or NIS, this file may
# not be consulted at all; see /etc/nsswitch.conf for the resolution order.
#
#
::1          localhost localhost.my.domain
127.0.0.1     localhost localhost.my.domain
#
# Imaginary network.
#10.0.0.2     myname.my.domain myname
#10.0.0.3     myfriend.my.domain myfriend
#
# According to RFC 1918, you can use the following IP networks for
# private nets which will never be connected to the Internet:
#
# 10.0.0.0    -    10.255.255.255
# 172.16.0.0  -    172.31.255.255
# 192.168.0.0 -    192.168.255.255
#
# In case you want to be able to connect to the Internet, you need
# real official assigned numbers. Do not try to invent your own network
# numbers but instead get one from your network provider (if any) or
# from your regional registry (ARIN, APNIC, LACNIC, RIPE NCC, or AfriNIC.)
#
```

/etc/hosts 用□□的格式：

```
[Internet address] [official hostname] [alias1] [alias2] ...
```

例如：

```
10.0.0.1 myRealHostname.example.com myRealHostname foobar1 foobar2
```

参考 [hosts\(5\)](#) 以□得更多信息。

### 12.10.3. 日志文件配置

### 12.10.3.1. syslog.conf

syslog.conf 是 [syslogd\(8\)](#) 程序的配置文件。它指出了的 **syslog** 信息型被存储在特定的日志文件中。

```
# $FreeBSD$
#
#   Spaces ARE valid field separators in this file. However,
#   other *nix-like systems still insist on using tabs as field
#   separators. If you are sharing this file between systems, you
#   may want to use only tabs as field separators here.
#   Consult the syslog.conf(5) manual page.
*.err;kern.debug;auth.notice;mail.crit      /dev/console
*.notice;kern.debug;lpr.info;mail.crit;news.err /var/log/messages
security.*                                   /var/log/security
mail.info                                    /var/log/maillog
lpr.info                                     /var/log/lpd-errs
cron.*                                       /var/log/cron
*.err                                         root
*.notice;news.err                           root
*.alert                                       root
*.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
#*. *                                         /var/log/all.log
# uncomment this to enable logging to a remote log host 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
!startslip
*. *                                         /var/log/slip.log
!ppp
*. *                                         /var/log/ppp.log
```

参考 [syslog.conf\(5\)](#) 手册以得更多信息

### 12.10.3.2. newsyslog.conf

nnewsyslog.conf 是一个通常用 [cron\(8\)](#) 执行的 [nnewsyslog\(8\)](#) 程序的配置文件。 [nnewsyslog\(8\)](#) 指出了什么时候日志文件需要打包或者重新整理。比如 logfile 被移到 logfile.0, logfile.0 被移到 logfile.1 等等。另外, 日志文件可以用 [gzip\(1\)](#) 来, 它是的命名格式: logfile.0.gz, logfile.1.gz 等等。

nnewsyslog.conf 指出了个日志文件要被管理, 要保留多少和它什么时候被建。 日志文件可以在它到一定大小或者在特定的日期被重新整理。



```
# configuration file for newsyslog
# $FreeBSD$
#
# filename      [owner:group]  mode count size when [ZB] [/pid_file] [sig_num]
/var/log/cron           600  3    100  *    Z
/var/log/amd.log        644  7    100  *    Z
/var/log/kerberos.log   644  7    100  *    Z
/var/log/lpd-errs       644  7    100  *    Z
/var/log/maillog        644  7    *    @T00 Z
/var/log/sendmail.st    644 10    *    168  B
/var/log/messages       644  5    100  *    Z
/var/log/all.log        600  7    *    @T00 Z
/var/log/slip.log       600  3    100  *    Z
/var/log/ppp.log        600  3    100  *    Z
/var/log/security       600 10    100  *    Z
/var/log/wtmp           644  3    *    @01T05 B
/var/log/daily.log      640  7    *    @T00 Z
/var/log/weekly.log     640  5    1    $W6D0 Z
/var/log/monthly.log    640 12    *    $M1D0 Z
/var/log/console.log    640  5    100  *    Z
```

参考 [newsyslog\(8\)](#) 手册以得更多信息。

## 12.10.4. sysctl.conf

`sysctl.conf` 和 `rc.conf` 两个文件的格式很接近。其中的配置均以 `变量=值` 的形式。在这个文件中配置的，均会在系统进入多用模式之后进行值的修改操作。需要注意的是，并不是所有的变量都能在多用模式下修改。

如果希望进程收到致命的信号退出的进程行，并阻止普通用户看到其他用户的进程，可以在 `sysctl.conf` 中行下列配置：

```
# 不由于致命信号致的进程退出（例如信号 11，越界）
kern.logsigexit=0

# 阻止用户看到以其他用户 UID 身行的进程。
security.bsd.see_other_uids=0
```

## 12.11. 用 sysctl 行整

[sysctl\(8\)](#) 是一个允许改正在行中的 FreeBSD 系统的接口。它包含一些 TCP/IP 堆和虚内存系统的高，可以有管理提高引人注目的系统性能。用 [sysctl\(8\)](#) 可以取置超五百个系统量。

基于点，[sysctl\(8\)](#) 提供个功能：取和修改系统置。

看所有可量：

```
% sysctl -a
```

一个指定的变量，例如 `kern.maxproc`：

```
% sysctl kern.maxproc
kern.maxproc: 1044
```

要设置一个指定的变量，直接用 `variable=value` 的语法：

```
# sysctl kern.maxfiles=5000
kern.maxfiles: 2088 -> 5000
```

sysctl 变量的设置通常是字符串、数字或者布尔型。（布尔型用 `1` 来表示 'yes'，用 `0` 来表示 'no'）。

如果你想在下次机器启动时设置某些变量，可将它加入到文件 `/etc/sysctl.conf` 之中。更多信息，参手册 [sysctl.conf\(5\)](#) 及 [sysctl.conf](#)。

### 12.11.1. 只读的 `sysctl(8)`

有可能会需要修改某些只读的 `sysctl(8)` 的。尽管有不得不做，但只有通过（重新）才能达到的目的。

例如一些膝上型电脑的 `cardbus(4)` 不会探测内存，并且产生看似于的：

```
cbb0: Could not map register memory
device_probe_and_attach: cbb0 attach returned 12
```

像上面的通常需要修改一些只读的 `sysctl(8)` 默认设置。要重点，用可以在本地的 `/boot/loader.conf.local` 里面放一个 `sysctl(8)` "OIDs"。那些设置定位在 `/boot/defaults/loader.conf` 文件中。

修上面的用需要在才所的文件中置 `hw.pci.allow_unsupported_io_range=1`。在 `cardbus(4)` 就会正常的工作了。

## 12.12. 整磁

### 12.12.1. Sysctl 量

#### 12.12.1.1. `vfs.vmiodirenable`

`vfs.vmiodirenable` sysctl 量可以置成0(或)或者1(或)；默认是1。个量控制目是否被系存。大多数目是小的，在系中只使用个片断(典型的是1K)并且在存中使用的更小(典型的是512字)。当个量置(0)，存器存固定数量的目，即使有很大的内存。而将其置(置1)，允存器用 VM 面存来存些目，所有可用内存来存目。不利的是最小的用来存目的核心内存是大于 512 字的物理面大小(通常是 4k)。我建议如果在行任何操作大量文件的程序保持个打的默认。些服包括 web 存，大容量件系和新系。尽管可能会浪一些内存，但打个通常不会降低性能。但是试一下。

### 12.12.1.2. `vfs.write_behind`

`vfs.write_behind` `sysctl` 变量默认是 1 (打开)。它告诉文件系统在被收集的时候把内容写进，典型的是在写入大的文件。主要的想法是，如果可能 I/O 性能会生面影， 尽量避免缓冲被未同缓冲区充。然而它可能降低处理速度并且在某些情况下可能想要它。

### 12.12.1.3. `vfs.hirunningspace`

`vfs.hirunningspace` `sysctl` 变量决定了在任何特定情况下，有多少写 I/O 被排列以系磁控制器。默认一般是足够的，但是有很多磁的机器来可能需要把它置成 4M 或 5M。注意个置成很高的(超存储器的写限)会致坏的性能。不要盲目的把它置太高！高的数会致同生的操作的延。

`sysctl` 中有许多与 `buffer cache` 和 `VM` 面 `cache` 有关的，一般不推荐修改它。虚拟内存系已能很好地行自整了。

### 12.12.1.4. `vm.swap_idle_enabled`

`vm.swap_idle_enabled` `sysctl` 变量在有很多用入、系和有很多空程的大得多用系中很有用。些系注重在空的内存中产生力的理。通 `vm.swap_idle_threshold1` 和 `vm.swap_idle_threshold2` 打开个特性并且整交滞后 (在空)允降低内存中空程的先，从而比正常的出 (pageout) 算法更快。出守程来了助。除非需要否不要把个打开，因所衡的是更快地入内存，因而它会吃掉更多的交和磁。在小的系上它会有决定性的效果，但是在大的系上它已做了合的面度个允 VM 系容易的全部的程出内存。

### 12.12.1.5. `hw.ata.wc`

FreeBSD 4.3 中默认将 IDE 的写存掉了。会降低到 IDE 磁用于写入操作的，但我有助于避免硬厂商所引入的，可能引致重的数据不一致。上是由于 IDE 硬就写操作完成件事的不致。当用了 IDE 写入存， IDE 硬器不但不会按序将数据写到上，而且当磁承受重，它甚至会自作主地推某些的写操作。一来，在系生崩或掉，就会致重的文件系统坏。基于些考，我将 FreeBSD 的默认配置改成了更安全的禁用 IDE 写入存。然而不幸的是，做致了性能的大幅降低，因此在后来的行版中个配置又改默认用了。可以通过察 `hw.ata.wc` `sysctl` 变量，来系的系中所采用的默认。如果 IDE 写存被禁用，可以通过将内核量置 1 来用它。一操作必在通 boot loader 来完成。在内核之后做是没有任何作用的。

要了解更多的信息，参 [ata\(4\)](#)。

### 12.12.1.6. `SCSI_DELAY` (`kern.cam.scsi_delay`)

`SCSI_DELAY` 内核配置会短系。默认在系程中有 15 秒的延，是一个足多且可的。把它少到 5 通常也能工作(特是代的器)。可以在系引整引加器量 `kern.cam.scsi_delay` 来改它。需要注意的是，此使用的位是毫秒而不是秒。

## 12.12.2. Soft Updates

`tunefs(8)` 程序能用来很好的整文件系统。个程序有很多不同的，但是在只介 Soft Updates 的打和，做：

```
# tune2fs -n enable /filesystem
# tune2fs -n disable /filesystem
```

在文件系统被挂起之后不能用 `tune2fs(8)` 来修改。打补丁 Soft Updates 的最佳时机是在空闲模式下任何分区被挂起前。

Soft Updates 大大改善了元数据修改的性能，主要是文件创建和删除，通过内存缓存。我们建议在所有的文件系统上使用 Soft Updates。我们知道 Soft Updates 的缺点：首先，Soft Updates 保证了崩溃后的文件系统完整性，但是很可能有几秒（甚至一分钟！）之前的数据没有写到物理磁盘。如果文件系统崩溃了可能会丢失很多工作。第二，Soft Updates 推迟文件系统的数据释放。如果在文件系统（例如根文件系统）快速的情况下系统运行大模式的升级比如 `make installworld`，可能会引起磁盘空间不足而造成升级失败。

### 12.12.2.1. Soft Updates 的元数据

有四种的方法把文件系统的元数据（meta-data）写入磁盘。（Meta-data更新是更新类似 inodes 或者目录中一些没有内容的数据）

从前，默认方法是同步更新元数据(meta-data)。如果一个目录改变了，系统在真正写到磁盘之前一直等待。文件数据缓存(文件内容)在之后以非同形式写入。这样做有利的一点是操作安全。如果更新产生，元数据(meta-data)一直处于完整状态。文件要不就被完整的创建要不根本就不创建。如果崩溃得不到文件的数据，`fsck(8)` 可以读到并且依靠把文件大小置0来修复文件系统。另外，这样做既清楚又简单。缺点是元数据(meta-data)更新很慢。例如 `rm -r` 命令，依次触及目录下的所有文件，但是每个目录的改变(删除一个文件)都要同步写入磁盘。它包含它自己更新目录，inode 表和可能文件分散的数据的更新。同步输出大的文件操作上(比如 `tar -x`)。

第二种方法是非同步元数据更新。它是 Linux/ext2fs 和 \*BSD ufs 的 `mount -o async` 默认的方法。所有元数据更新也是通过缓存。也就是它会混合在文件内容数据更新中。这个方法的特点是它不需要等待元数据更新都写到磁盘上，所以所有引起元数据更新大的操作比同步方式更快。同时，这个方法也是清楚且简单的，所以代码中的漏洞很小。缺点是不能保证文件系统的一致性。如果更新大量元数据失败（例如掉电或者按了重启按钮），文件系统会在不可知的状态。系统再也没有机会知道文件系统的状态；inode 表更新的时可能文件的数据已经写入磁盘了但是相应的目录没有，却不能用 `fsck` 命令来清理(因为磁盘上没有所需要的信息)。如果文件系统修复后坏了，唯一的办法是使用 `newfs(8)` 并且从备份中恢复它。

一个通常的解决方法是使用 *dirty region logging* 或者 *journaling* 尽管它不是一般的被使用并且有它自己的用到其他的事情中更好。这个方法元数据更新依然同步写入，但是只写到磁盘的一个小区域。之后它将会被移到正确的位置。因为区域很小，磁盘上接近的区域磁盘不需要移动很远的距离，所以它比写同步快一些。另外这个方法的一致性有限，所以出问题的机会也很少。缺点是元数据要写多次（一次写到错误区域，一次写到正确的区域）。正常情况下，它的性能可能会产生。从一方面来说，崩溃的时候所有未产生的元数据操作可以很快的在系统重启之后从备份中恢复来。

Kirk McKusick, 伯克利 FFS 的作者，用 Soft Updates 解决了这个问题：元数据更新保存在内存中并且按照排列的顺序写入到磁盘（“有序的元数据更新”）。如果是，在繁重的元数据操作中，如果先前的更新在内存中没有被写入磁盘，后来的更新就会捕捉到。所以所有的目录操作在写磁盘的时候首先在内存中进行（数据按照它的位置来排列，所以它不会在元数据前被写入）。如果系统崩溃了将导致一个固定的“日志回溯”：所有不知如何写入磁盘的操作都像没有产生一样。文件系统的一致性保持在 30 到 60 秒之前。它保证了所有正在使用的资源被例如和 inodes。崩溃之后，唯一的资源分配是一个它是“空”的资源的资源被“使用”。`fsck(8)` 可以输出

情况并且放不再使用的源。它于忽略崩后用 `mount -f` 制挂上的文件系统状态是安全的。放了放可能没有使用的源, `fsck(8)` 需要在后的行。一个主意是用后台 `fsck`: 系的时候只有一个文件系的快照被下来。`fsck` 可以在后行。所有文件系统可以在"有"的时候被挂接, 所以系可以在多用模式下。接着, 后台 `fsck` 可以在所有文件系统需要的候来放可能没有使用的源。(尽管, 不用 Soft Updates 的文件系依然需要通常的 `fsck`。)

它的点是元数据操作几乎跟非同快(也就是比需要次元数据写操作的 `logging` 更快)。缺点是代的性(意味着于失用敏感数据有更多的)和高的内存使用量。外它有些特点需要知道。崩之后, 文件系统状态会"落后"一些。同的方法用 `fsck` 后在一些地方可能生一些零字的文件, 些文件在用 Soft Updates 文件系统之后不会存在, 因元数据和文件内容根本没有写磁(可能生在行 `rm` 之后)。可能在文件系统上安装大量数据候引, 没有足的剩余空来次存所有文件。

## 12.13. 整内核限制

### 12.13.1. 文件/程限制

#### 12.13.1.1. `kern.maxfiles`

`kern.maxfiles` 可以根据系的需要当。个量用于指定在系中允的文件描述符的最大数量。当文件描述符表的时候, `file: table is full` 会在系消息缓冲区中反出, 可以使用 `dmesg` 命令来察一象。

个打的文件、套接字和管道, 都会占用一个文件描述符。在大型生服务器上, 可能会易地用掉数千个文件描述符, 具体用量取决于服的型和并行的服数量。

在早期版本的 FreeBSD 中, `kern.maxfiles` 的默, 是根据内核配置文件中的 `maxusers` 算的。`kern.maxfiles` 个数, 会随 `maxusers` 成比例地。当定制的内核, 按照系的用途来修改是个好主意。个数字同决定内核的多限制。有, 尽管并不会真的有 256 个用同接一台生服务器, 但对于高的 web 服务器而言, 却可能需要与之似的源。

量 `kern.maxusers` 会在系, 根据可用内存的尺寸行算, 在内核始行之后, 可以通过只的 `kern.maxusers` `sysctl` 量来行察。有些情况下, 可能会希望使用更大或更小一些的 `kern.maxusers`, 它可以以加器量的形式行配置; 似 64、128 和 256 的都不。我不推使用超 256 的, 除非需要巨量的文件描述符; 根据 `kern.maxusers` 推算默的那些量, 一般都可以在引甚至行通 `/boot/loader.conf` (参 `loader.conf(5)` 机手册或 `/boot/defaults/loader.conf` 文件来得相的指) 或篇文的其余部分所介的方式来整。

在早的版本中, 如果明地将 `maxusers` 置 0, 系会自地根据硬件配置来定个。在 FreeBSD 5.X 和更高版本中, `maxusers` 如果不指定的, 就会取默 0。如果希望自行管理 `maxusers`, 配置一个不低于 4 的, 特是使用 X Window System 或件的候。做的原因是, `maxusers` 所决定的一个最重要的表的尺寸会影响最大程数, 个数将是  $20 + 16 * \text{maxusers}$ 。因此如果将 `maxusers` 置 1, 就只能同行 36 个程, 包括了 18 个左右的系引的程, 以及 15 个左右的, 在 X Window System 所引的程。即使是任何的, 如机手册, 也需要多至九个的程, 用以、解, 并示它。将 `maxusers` 64 将允同行最多 1044 个程, 几乎足以足任何需要了。不, 如果看其它程序, 或行用以支持大量用的服(例如 [ftp.FreeBSD.org](http://FreeBSD.org)) , 看到令人担的, 就提高一数, 并重新内核。





`maxusers` 并不能限制能够登录到系统上来的用户的数量。它的主要作用是根据可能支持的用数量来一系列系统数据表置合理的尺寸，以便提供支持他所需行的程序源。

### 12.13.1.2. `kern.ipc.somaxconn`

`kern.ipc.somaxconn` `sysctl` 量限制了接收新 TCP 连接队列的大小。于一个常理新连接的高 web 服务器境来，默的 128 太小了。大多数境个建加到 1024 或者更多。服务器会自己限制连接队列的大小(例如 `sendmail(8)` 或者 Apache)，常常在它配置文件中置队列大小的。大的连接队列防止拒绝服务攻击也会有所助。

### 12.13.2. 网限制

`NMBCLUSTERS` 内核配置指出了系可用的网 Mbuf 的数量。一个高流量的服务器使用一个小数目的网内存会影 FreeBSD 的性能。个 cluster 可能需要 2K 内存，所以一个 1024 的需要在内核中网内存保留 2M 内存。可以用算的方法算出来需要多少网内存。如果有一个同生 1000 个以上连接的 web 服务器，并且个连接用掉 16K 接收和送内存，就需要大概 32M 网内存来保 web 服务器的工作。一个好的算方法是乘以 2，所以  $2 \times 32\text{Mb} / 2\text{Kb} = 64\text{MB} / 2\text{kb} = 32768$ 。我建在有大量内存的机器上把个置在 4096 到 32768 之间。没有必要把它置成任意太高的，它会在引起崩。 `netstat(1)` 的 `-m` 可以用来察网 cluster 使用情况。

`kern.ipc.nmbclusters` 可以用来在刻个。在旧版本的 FreeBSD 需要使用 `NMBCLUSTERS config(8)`。

常使用 `sendfile(2)` 系用的繁忙的服务器，有必要通 `NSFBUFS` 内核或者在 `/boot/loader.conf` (看 `loader(8)` 以得更多) 中置它的来 `sendfile(2)` 内存数量。个参数需要的原因是在程中看到 `sfbufa` 状态。`sysctl kern.ipc.nsfbufs` 量在内核配置量中是只的。个参数是由 `kern.maxusers` 决定的，然而它可能有必要因此而整。



即使一个套接字被成非阻塞，在个非阻塞的套接字上呼叫 `sendfile(2)` 可能致 `sendfile(2)` 呼叫阻塞直到有足的 `struct sf_buf` 可用。

#### 12.13.2.1. `net.inet.ip.portrange.*`

`net.inet.ip.portrange.*` `sysctl` 量自的控制定在 TCP 和 UDP 套接字上的端口。里有三个：一个低端，一个默和一个高端。大多数网程序分使用由 `net.inet.ip.portrange.first` 和 `net.inet.ip.portrange.last` 控制的从 1024 到 5000 的默。端口用作外连接，并且某些情况可能用完系的端口，常生在行一个高荷 web 代理服务器的时候。个端口不是用来限制主要的例如 web 服务器入连接或者有固定端口例如件外连接的。有可能用完了端口，那就建当的加 `net.inet.ip.portrange.last`。10000、20000 或者 30000 可能是当的。更改端口的时候也要考到防火。一些防火会阻止端口的大部分 (通常是低端的端口) 并且用高端行外连接(-)。基于个建不要把 `net.inet.ip.portrange.first` 的太小。

#### 12.13.2.2. TCP 延迟(Bandwidth Delay Product)

限制 TCP 延迟和 NetBSD 的 TCP/Vegas 似。它可以通将 `sysctl` 量 `net.inet.tcp.inflight.enable` 置成 1 来用。系将算一个连接的延迟，并将排的数据量限制在恰好能保持最吞吐量的水平上。

一特性在的服务器同向使用普通制解器，千兆以太网，乃至更高速度的光与网连接 (或其他延迟

很大的接口) 的时候尤为重要, 特别是当同时使用滑窗放大, 或使用了大的发送窗口的时候。如果用了个, 把 `net.inet.tcp.inflight.debug` 置 0 (禁用), 于生境而言, 将 `net.inet.tcp.inflight.min` 置成至少 6144 会很有好处。然而, 需要注意的是, 个置大事上相当于禁用了接口延迟限制功能。个限制特性少了在路由和交包列的堵塞数据数量, 也少了在本地主机接口列阻塞的数据的数量。在少数的等候列中、交互式接口, 尤其是通慢速的调制解器, 也能用低的往返操作。但是, 注意只影响到数据送(上/服务端)。数据接收(下)没有效果。

整 `net.inet.tcp.inflight.stab` 是不推的。个参数的默认是 20, 表示把 2 个最大包加入到延迟窗口的计算中。外的窗口似的算法更稳定, 并改善于多网境的相应能力, 但也会致慢速接下的 ping (尽管是会比没有使用 `inflight` 算法低得多)。于些情形, 可能会希望把个参数少到 15, 10, 或 5; 并可能因此而不得不少 `net.inet.tcp.inflight.min` (比如, 3500) 来得到希望的效果。少些参数的, 只作最后不得已的手段来使用。

## 12.13.3. 虚内存

### 12.13.3.1. `kern.maxvnodes`

vnode 是文件或目的一内部表。因此, 加可以被操作系统利用的 vnode 数量将降低磁的 I/O。一般而言, 是由操作系统自行完成的, 也不需要加以修改。但在某些时候磁 I/O 会成瓶颈, 而系的 vnode 不足, 一配置被加。此需要考是非活和空内存的数量。

要看当前在用的 vnode 数量:

```
# sysctl vfs.numvnodes
vfs.numvnodes: 91349
```

要看最大可用的 vnode 数量:

```
# sysctl kern.maxvnodes
kern.maxvnodes: 100000
```

如果当前的 vnode 用量接近最大, 将 `kern.maxvnodes` 大 1,000 可能是个好主意。看看看 `vfs.numvnodes` 的数, 如果它再次攀升到接近最大的程度, 仍需提高 `kern.maxvnodes`。在 `top(1)` 中示的内存用量有显著化, 更多内存会于活(active)状态。

## 12.14. 添加交空

不管得如何好, 有时候系并不像所期待的那样行。如果需要更多的交空, 添加它很。有三方法加交空: 添加一新的硬器、通 NFS 使用交空和在一个有的分区上建一个交文件。

要了解于如何加密交区, 相应配置, 以及什么要, 参手册的 [交区行加密](#)。

### 12.14.1. 在新的硬器上使用交空

是添加交空最好的方法, 当然到了到个目的需要添加一新硬。竟是可以使用一磁。如果能做, 重新一下手册中于交空的 [初配置](#) 来了解如何最地安排交空。

## 12.14.2. 通过 NFS 交换

除非没有可以用作交换空间的本地硬盘，否则不推荐使用 NFS 来作交换空间使用。NFS 交换会受到可用网络限制并且增加 NFS 服务器的负担。

## 12.14.3. 交换文件

可以创建一个指定大小的文件用来当作交换文件。在我的例子中我将使用叫做 `/usr/swap0` 的 64MB 大小的文件。当然也可以使用任何所希望的名字。

例 9. 在 *FreeBSD* 中创建交换文件

1. 内核配置包含虚拟磁盘(Memory disk) (`md(4)`)。它在 GENERIC 内核中是默认的。

```
device    md    # Memory "disks"
```

2. 创建一个交换文件(`/usr/swap0`)：

```
# dd if=/dev/zero of=/usr/swap0 bs=1024k count=64
```

3. 予它(`/usr/swap0`)一个适当的权限：

```
# chmod 0600 /usr/swap0
```

4. 在 `/etc/rc.conf` 中使用交换文件：

```
swapfile="/usr/swap0"    # Set to name of swapfile if aux swapfile desired.
```

5. 通过重新引导机器或下面的命令使交换文件立刻生效：

```
# mdconfig -a -t vnode -f /usr/swap0 -u 0 && swapon /dev/md0
```

## 12.15. 电源和电源管理

BIOS 接口管理，例如可拔 BIOS (*PNPBIOS*)或者高级电源管理(*APM*) 等等。电源和电源管理是现代操作系统的组成部分。例如可能当系统温度高的时候操作系统能够到 (并且可能提醒)。

以有效的方式利用硬件电源是非常重要的。在引入 ACPI 之前，管理电源使用和系统散热操作系统是很困难的。硬件由 BIOS 进行管理，因而用电源管理配置的控制和查看都比困难。一些系统通过高级电源管理 (*APM*) 提供了有限的配置能力。电源和电源管理是现代操作系统的一个部件。例如，可能希望操作系统的一些限制，例如系统的温度是否超出了预期的速度 (并在需要时发出警告)。

在 *FreeBSD* 使用手册的一章中，我将提供 ACPI 全面的信息。参考资料会在末尾给出。



### 12.15.1. 什么是 ACPI？

高级配置和电源接口 (ACPI) 是一个业界标准的硬件电源和电源管理接口 (因此而得名)。它是操作系统控制的配置和电源管理(*Operating System-directed configuration and Power Management*), 也就是, 它操作系统(OS)提供了更多的控制和特性。在引入 ACPI 之前, 旧操作系统使得目前即使用接口的局限性更加 "凸" 出来。ACPI 是 APM(高级电源管理) 的直接继承者。

### 12.15.2. 高级电源管理 (APM) 的缺点

高级电源管理 (APM) 是一种基于系统目前的活动控制其电源使用的机制。APM BIOS 由 (系统的) 制造商提供, 并且是硬件平台专属的。在 OS 中的 APM 操作中介来 APM 软件接口, 从而管理电源使用的管理。在 2000 年或更早的时期生产的计算机系统, 仍需要使用 APM。

APM 有四个主要的缺点。首先, 电源管理是通用 (制造商专属) BIOS 做的, 而 OS 完全不了解其原理。例如, 用户在 APM BIOS 中设置了硬盘驱动器的空闲等待数, 当超过一空闲的限制, 它 (BIOS) 将会慢慢降低硬盘驱动器的速度, 而不会征求 OS 的同意。第二, APM 是嵌入 BIOS 的, 因此它是在 OS 的控制之外的。这意味着用户只能通过刷新他的 ROM 中的 APM BIOS 才能解决某些问题; 而这是一个很危险的操作, 因为它可能使系统进入一个无法恢复的状态。第三, APM 是一种制造商专属的技术, 也就是有很多第三方的 (重工业的工作) 以及 bugs, 如果在一个制造商的 BIOS 中有, 也未必会在其他的产品中解决。最后但不是最小的缺点, APM BIOS 没有电源策略提供足够的余地, 也无法能够非常符合具体机器的策略。

即使用 BIOS (PNPBIOS) 在很多时候都是不可靠的。PNPBIOS 是 16-位的技术, 因此 OS 不得不使用 16-位模型才能与 PNPBIOS 的方法 "接口"。

FreeBSD APM 在 [apm\(4\)](#) 手册中有描述。

### 12.15.3. 配置 ACPI

默认情况下, `acpi.ko` 会在系统引导由 [loader\(8\)](#) 加载, 而不直接加载内核。这样做的原因是模块操作起来更方便, 例如, 无需重新加载内核就可以切换到另一个 `acpi.ko` 版本。它可以做得更一些。一个原因是, 很多时候在加载之后再次 ACPI 可能会有些问题。如果遇到, 可以全面禁用 ACPI。这个选项不, 目前也无法卸载, 因为系统通过它与许多不同的硬件进行交互。ACPI 可以通过在 `/boot/loader.conf` 中配置或在 [loader\(8\)](#) 提示符配置 `hint.acpi.0.disabled="1"` 来禁用。



ACPI 和 APM 不能共存, 相反, 它们分开使用。后加载的模块如果系统中已经运行了其中的一个, 便会停止运行。

ACPI 可以用来将系统进入休眠模式, 方法是使用 [acpicnf\(8\)](#) 的 `-s` 参数, 加上一个 1-5 的数字。多数用户会希望使用 1 或 3 (挂起到 RAM)。而 5 会让系统运行与下列命令效果类似的机器:

```
# halt -p
```

除此之外, 有一些通过 [sysctl\(8\)](#) 提供的选项。参看手册 [acpi\(4\)](#) 和 [acpicnf\(8\)](#) 以得到更多信息。

## 12.16. 使用和 FreeBSD ACPI

ACPI 是一种全新的电源、管理电源使用、以及提供去由 BIOS 管理的不同硬件的标准化方法。ACPI 在各系统上都能正常使用的工作一直在进行, 但许多主板的 ACPI 机器语言 (AML) 字代码中的 bug, FreeBSD

的内核中子系的不完善，以及 Intel® ACPI-CA 解器中的 bug 仍然不会出。

本文期望能帮助你帮助 FreeBSD ACPI 的人来找到所观察到的问题的根源，并通到其解决方法。感谢本文，我也希望能解决的系上的。

### 12.16.1. 提交信息



在提交之前，已可在行最新的 BIOS 版本，此外，也包括嵌入式控制器的固件版本。

如果希望提交一个，保持将下述信息到 [freebsd-acpi@FreeBSD.org](mailto:freebsd-acpi@FreeBSD.org):

- 行的描述，包括系型、型号，以及任何触的相信息。外，注意尽可能准地描述一是否陌生的。
- 在 "boot -v" 之后得到的 `dmesg(8)` 出，以及任何在重 bug 出的信息。
- 在禁用了 ACPI 之后的 "boot -v" 的 `dmesg(8)` 出，如果禁用 ACPI 能助消除。
- 来自 `fsysctl hw.acpi` 的出。也是到的系所提供的功能的一好法。
- 能得到 *ACPI Source Language (ASL)* 的 URL。不要把 ASL 直接到件列表中，因它可能非常大。了得到 ASL 可以行个命令：

```
# acpidump -dt > name-system.asl
```

(把 *name* 改的登名，并把 *system* 改的硬件制造商及其型号。例如：njl-FooCo6000.asl)

多者也会 [FreeBSD-CURRENT 件列表](#) 但是到 [FreeBSD ACPI 件列表](#) 它会被更多人看到。耐心等待，因我都有全的其他工作。如果的 bug 不是而易的，我可能会要求通 `send-pr(1)` 来提交一个 PR。在入 PR 时，将同的信息包含去。将助我来追踪和解决。不要在 [FreeBSD ACPI 件列表](#) 写信之前送 PR 因我把它当作已知文体的忘而不是告机制。的很可能已被其他人告了。

### 12.16.2. 背景

ACPI 存在于采用 ia32 (x86)、ia64 (安)、以及 amd64 (AMD) 架的所有代计算机上。完整的准具有大量的各式功能，包括 CPU 性能管理、源控制、温度控、池系、嵌入式控制器以及枚。大多数系比完整准的功能要少一些。例如，面系通常只枚部分，而本通常支持降温 and 源管理功能。本通常提供休眠和醒支持，并提供与此的功能。

符合 ACPI 的系中有多件。BIOS 和芯片制造商提供一些固定的表 (例如，FADT) 在存器中，以提供以 APIC 映射 (用于 SMP)、配置寄存器、以及的配置等等。外，一个字代 (bytecode) 表 (系区描述表 DSDT) 提供了通状态命名空来指定及其功能的方法。

ACPI 必需要理固定表，字解器，并修改程序和内核，以接受来自 ACPI 子系的信息。于 FreeBSD，Intel® 提供了一个解器 (ACPI-CA)，它在 Linux 和 NetBSD 也可以使用。ACPI-CA 源代可以在 `src/sys/contrib/dev/acpica` 到。用于在 FreeBSD 中允 ACPI-CA 正的代在 `src/sys/dev/acpica/Osd`。最后，用于 ACPI 的可以在 `src/sys/dev/acpica` 到。

### 12.16.3. 常问题

要使 ACPI 正常工作，它的一部分都必须工作正常。下面是一些常见问题，按照出现的频率程度排序，并给出了一些问题或修正它们的方法。

#### 12.16.3.1. 鼠问题

某些时候，唤醒操作会导致鼠标不再正常工作。已知的一个问题—问题的方法，是在 `/boot/loader.conf` 文件中添加 `hint.psm.0.flags="0x3000"` 设置。如果这样做不能解决问题，请参考按前面介绍的方法提交问题报告。

#### 12.16.3.2. 休眠/唤醒

ACPI 提供了三种休眠到 RAM (STR) 的状态，`S1-S3`，以及一个休眠到磁碟的状态 (`STD`)，称作 `S4`。`S5` 是“关机”同时也是系统接好电源但没有机器的正常状态。`S4` 通常上可以用不同的方法来设置。`S4BIOS` 是一种由 BIOS 帮助的挂起到磁碟方法，而 `S4OS` 是完全由操作系统管理的。

可以使用 `sysctl hw.acpi` 来看与休眠有关的项目。下面是我的 Thinkpad 上得到的结果。

```
hw.acpi.supported_sleep_state: S3 S4 S5
hw.acpi.s4bios: 0
```

这表示我可以使用 `acpiconf -s` 来设置 `S3`，`S4OS`，以及 `S5`。如果 `s4bios` 是一 (1)，则可以使用 `S4BIOS` 来代替 `S4OS`。

当休眠/唤醒时，从 `S1` 开始，如果它被支持的。这个状态是最可能正常工作的状态，因为它不需要太多的硬件支持。没有人设置 `S2` 但如果它有它的支持，通常和 `S1` 类似。下一件值得做的是 `S3`。它是最深的 STR 状态，并需要一系列硬件的支持才能正常地重新初始化硬件。如果在唤醒系统遇到时，不要吝惜硬件 [FreeBSD ACPI 硬件列表](#) 硬件列表，尽管不要指望它一定会很快解决，因为它有多数程序/硬件需要运行更多的测试和更改。

休眠和唤醒操作最常的问题是某些系统程序不会保存、恢复或正确地重新初始化其固件、寄存器或内存。通常这些情况，首先可以：

```
# sysctl debug.bootverbose=1
# sysctl debug.acpi.suspend_bounce=1
# acpiconf -s 3
```

这个问题会模拟休眠和恢复过程而不真的进入 `S3` 状态。有时，它用这种方式很容易地挂住 (例如，它失固件状态、 watchdog 超时，以及一直重等)。注意系统不会真的进入 `S3` 状态，这意味着某些可能不会掉，而多数在完全不提供休眠和恢复方法仍可正常工作，而不像使用真的 `S3` 状态那样。

通常的情况需要更多的硬件，例如用于串口控制台的串口/IO，以及用于 [dcons\(4\)](#) 的火口/IO 和内核技能。

为了帮助隔离问题，可以在内核中去尽可能多的测试。如果这样做能解决问题，通常逐个加测试直到再次出问题。通常测试的程序如 `nvidia.ko`、`X11` 显示，以及 USB 的测试最多，而以太网通常工作的很好。如果可能，通常添加和卸载使系统正常工作，可以通过将适当的命令放到 `/etc/rc.suspend` 和 `/etc/rc.resume` 来将这个过程自动化。在下面的文件中有一个注释掉的卸载和加载程序的例子供参考。此外，可以将 `hw.acpi.reset_video` 设置为零 (0)，如果它在唤醒之后显得很混乱。此外，可以设置更短或更长的

`hw.acpi.sleep_delay` 看看是否有所助益。

一件得做的事情是使用一个比新的包含 ACPI 支持的 Linux 行版来看看他的 休眠/醒 功能是否在同的硬件上能正常工作。如果在 Linux 下正常，很可能是 FreeBSD 程序的，而隔并到存在有助于解决它。需要注意的是 ACPI 的人通常并不其他（例如 声音、ATA，等等）因此如果最好是到 [FreeBSD-CURRENT 件列表](#) 件列表并程序的者。如果喜欢冒，可以加一些 `printf(3)` 到有它的中，以到它的恢功能生到的位置。

最后，看看禁用 ACPI 并代之以用 APM。如果 休眠/醒 能在 APM 下正常工作，使用 APM 可能会更好，特别是于老的硬件（2000年以前）。硬件制造商需要一些来老硬件的 ACPI 工作正常，而 ACPI 的十之八九是 BIOS 中的毛病引的。

### 12.16.3.3. 系停止（或永久性地）

大多数系停止是由于未能及中断或生了中断暴致的。芯片有很多最会溯源到 BIOS 如何在引系之前配置中断，APIC (MADT) 表的正性，以及系控制中断 (SCI) 如何路由。

通察看 `vmstat -i` 的出中包括 `acpi0` 的那一行可以区分中断暴和未能及中断。如果秒数器的速度多于一，是遇到了中断暴。如果系停止了，可以停止内核并入 DDB (在控制台上按 `CTRL + ALT + ESC`) 并入 `show interrupts`。

理中断的救命稻草是禁用 APIC 支持，是通在 `loader.conf` 中加入 `hint.apic.0.disabled="1"` 完成的。

### 12.16.3.4. 崩

崩于 ACPI 是比的情况，如果，我将会非常重并很快修它。要做的第一件事是法隔出能重崩（如果可能的）的操作并取一用堆。用 `options DDB` 并置串行控制台（参 [通串口入DDB器](#)）或配置一个 `dump(8)` 分区。将在 DDB 中通 `tr` 得到用堆。如果只能用手抄的方法它，一定要下五 (5) 行和最后五 (5) 行。

然后，通在禁用 ACPI 来隔故障。如果做能正常工作，通置 `debug.acpi.disable` 的那数来隔具体是个 ACPI 子系的。参 [acpi\(4\)](#) 机手册中出的那些例子。

### 12.16.3.5. 系在休眠或机之后又了

首先在 `loader.conf(5)` 中置 `hw.acpi.disable_on_poweroff="0"`。将 ACPI 不再在机程中禁用一些事件。基于同的原因，一些系需要把个置 "1"（是默）。通常能修在休眠或机立即再次。

### 12.16.3.6. 其他

如果有 ACPI 的其他（同 docking station 同工作、无法，等等），把描述件列表；不，些也有可能和 ACPI 中尚未完成的部分有，它可能需要才能被。点耐心，并准我可能会的丁。

## 12.16.4. ASL、acpidump, 以及 IASL

最常的是 BIOS 制造商提供的不正（甚至完全的！）字代。通常会以似下面的内核消息示在控制台上：

```
ACPI-1287: *** Error: Method execution failed [\\_SB_.PCI0.LPC0.FIGD._STA] \\
(Node 0xc3f6d160), AE_NOT_FOUND
```

有时候，可以通过将 BIOS 升到最新版本来解决此问题。大多数控制台消息是无害的，但如果有其他问题例如电池工作不正常，从 AML 开始可能将是一条捷径。字代码，或常量的 AML，是从一个叫做 ASL 的语言写成的源代码行得到的结果。AML 一般存放在 DSDT 表中。要得到系统的 ASL，需要使用 [acpidump\(8\)](#)。需要同指定 `-t` (指示固定内容) 和 `-d` (将 AML 反成 ASL) 个选项。参看 [如何提交信息](#) 了解如何使用它。

最方便的初始步骤是重新 ASL 来看看是否有问题。通常可以忽略一过程中产生的警告，但它们一般就都是 bug，它们通常就是导致 ACPI 无法正常工作的原因。要重新 ASL，可以使用下面的命令：

```
# iasl your.asl
```

### 12.16.5. 修复 ASL

我们的期望是每个人都能在不需要任何干预的情况下使用 ACPI。然而，目前我们仍然在 BIOS 制造商常使用的方法。Microsoft® 解器 (acpi.sys 和 acpiec.sys) 并不会严格地是否遵守了规范，因此我们只在 Windows® 中 ACPI 的 BIOS 制造商很可能永远不会修正他们的 ASL。我们希望不断地出并用文说明 Microsoft® 的解器到底允许那些不规范的行，并在 FreeBSD 进行行的修改使它能正常工作而不需要用到修正 ASL。作一解器的方法，并助我们行，可以手工修正 ASL。如果能解决，把新旧 ASL 的 [diff\(1\)](#) 给我，我们就有可能 ACPI-CA 中的行，从而不再需要来手工修正。

下面是一些常见的信息，它的原因，以及如何修正。

#### 12.16.5.1. \_OS dependencies (\_OS 依赖)

某些 AML 假定世界是由不同版本的 Windows® 组成的。可以 FreeBSD 声称自己是任意 OS 来看一看是否能修正。比的方法是置 `hw.acpi.osname="Windows 2001"` 到 `/boot/loader.conf` 中，或使用在 ASL 中到的其他字符串。

#### 12.16.5.2. Missing Return statements (缺少返回句)

一些方法可能没按照规范要求的那式地返回。尽管 ACPI-CA 无法理它，但 FreeBSD 提供了一个它并允其隐含地返回的方法。也可以加一个式的 Return 句，如果知道那里需要返回一个的。要制 iasl ASL，需要使用 `-f` 标志。

#### 12.16.5.3. 替换默认的 AML

在定制 your.asl 之后，可以通过下面的命令它：

```
# iasl your.asl
```

可以使用 `-f` 标志来制建 AML，即使在过程中生了问题。注意某些 (例如，缺少 Return 句) 会自被解器忽略掉。

DSDT.aml 是 `iasl` 命令的默认输出文件名。可以加它来取代 BIOS 中存在它的副本 (它仍然存在于内存中), 其方法是按下面的说明修改 `/boot/loader.conf` :

```
acpi_dsdt_load="YES"
acpi_dsdt_name="/boot/DSDT.aml"
```

一定要把它的 DSDT.aml 复制到 `/boot` 目录中。

### 12.16.6. 从 ACPI 中获取输出信息

ACPI 驱动程序提供了非常灵活的调试机制。允许指定一子系, 以及所需要的输出信息。需要输出的子系可以按 "layers()" 来指定, 并分 ACPI-CA 组件 (ACPI\_ALL\_COMPONENTS) 和 ACPI 硬件支持 (ACPI\_ALL\_DRIVERS)。输出的程度可以通过 "level(程度)" 来指定, 其值是 ACPI\_LV\_ERROR (只告警) 到 ACPI\_LV\_VERBOSE (显示所有)。`"level"` 是一个位掩码因此可以一次设置多个, 中间用空格分隔。使用中通常考虑使用串行控制台来输出, 如果它太慢以至于冲掉了控制台消息缓冲的。不同的输出程度的完整列表可以在 [acpi\(4\)](#) 手册中看到。

输出默认并不开。要起用它, 需要在内核配置中添加 `options ACPI_DEBUG`, 如果内核中加入了 ACPI 的。可以在 `/etc/make.conf` 中加入 `ACPI_DEBUG=1` 来在全局起用它。如果它只是模块, 可以用下面的方法来重新编译 `acpi.ko` :

```
# cd /sys/modules/acpi/acpi
&& make clean &&
make ACPI_DEBUG=1
```

安装 `acpi.ko` 到 `/boot/kernel` and add your 并把所需的程度和在 `loader.conf` 中指定。个例子将启用所有 ACPI-CA 组件以及所有 ACPI 硬件 (CPU、LID, 等等) 的消息。只输出信息, 也就是最低的程度。

```
debug.acpi.layer="ACPI_ALL_COMPONENTS ACPI_ALL_DRIVERS"
debug.acpi.level="ACPI_LV_ERROR"
```

如果需要的信息是由某个特定的事件触发的 (比如, 休眠之后的唤醒), 可以不修改 `loader.conf` 而使用 `sysctl` 来在和那个事件挂钩之后再指定程度。些 `sysctl` 的名字和 `loader.conf` 中的一致。

### 12.16.7. 参考文献

关于 ACPI 的更多信息可以从下面这些地方得到 :

- The [FreeBSD ACPI 组件列表](#)
- ACPI 组件列表存 <http://lists.freebsd.org/pipermail/freebsd-acpi/>
- 旧的 ACPI 组件列表存 <http://home.jp.FreeBSD.org/mail-list/acpi-jp/>
- The ACPI 2.0 规范 <http://acpi.info/spec.htm>
- FreeBSD 手册: [acpi\(4\)](#), [acpi\\_thermal\(4\)](#), [acpidump\(8\)](#), [iasl\(8\)](#), [acpidb\(8\)](#)
- [DSDT 源码](#). (使用 Compaq 作例子但通常情况下都很有用。)



# Chapter 13. FreeBSD 引导程序

## 13.1. 概述

引导以及加载操作系统的程序被称为“引导程序”，或者称“引导”。FreeBSD 的引导程序用自定义提供了很大的伸缩性，它可以引导不同的操作系统，或者是同一系统的不同版本及内核。

本章将介绍能在 FreeBSD 引导程序中设置的配置。它包括了引导内核、探测并运行 `init(8)` 等等之前所发生的所有事情。有些事一般发生在文本由白到灰。

读完本章将会知道：

- FreeBSD 引导系统里的各组件，以及它们之间的交互方式。
- 在 FreeBSD 引导各组件配置以控制引导程序。
- `device.hints(5)` 的基本知识。



只用于x86

本章只描述了运行于 Intel x86 体系之上的 FreeBSD 的引导程序。

## 13.2. 引导

引导及加载和引导操作系统成了一个有趣的境地。按照定义在操作系统被加载之前计算机是无法完成任何任务的，包括运行磁盘上的程序。如果计算机在没有操作系统的情况下不能运行来自于磁盘上的程序而操作系统又是放在磁盘上的，那操作系统是如何启动的？

在 Munchausen男爵历险记 (The Adventures of Baron Munchausen) 剧本中有一个和这个过程类似的故事，一个人掉到了下水管道里，然后拉着自己的靴子 (bootstrap) 克服重重困难爬了出来。在早期文献中，多以 *bootstrap* 来指代操作系统的加载机制，如今它逐渐被写作 “booting”。

在 x86 硬件体系中，基本输入/输出系统 (BIOS) 加载操作系统，为了做到这一点，BIOS 在磁盘上寻找主引导记录 (MBR)，而 MBR 必在放置的磁盘的特定位置。BIOS 有足够的能力来读取和执行 MBR，且假使地 MBR 能完成加载操作系统的剩余任务，MBR 可能需要 BIOS 的帮助。

在 MBR 中的代码通常被提为引导管理器，尤其是与用户交互的那部分。单一引导器通常有更多代码位于磁盘第一道或在操作系统的文件系统中。（引导管理器有时也被称作 *boot loader*，但是 FreeBSD 后面的引导段才使用这个词。）流行的引导管理器包括 boot0 (亦称 Boot Easy，标准的 FreeBSD 引导管理器)、Grub、GAG，以及 LILO。（只有 boot0 能装得进 MBR。）

如果只安装了一个操作系统，那一个标准的 MBR 就足够了。这个 MBR 先在磁盘上搜索可引导的 (亦称“活动的”) 分区，然后执行分区上的代码以加载操作系统的其它部分。MBR 由 `fdisk(8)` 安装，是一个缺省的 MBR。相关文件 `/boot/mbr`。

如果在磁盘上安装了多个操作系统那就可以安装一个不同的引导管理器，它能显示一个操作系统的列表，你能从中选择一个。新的引导器将在下一小节中。

系统的剩余部分被分成三个阶段。第一阶段由 MBR 执行，它只是使计算机进入特定的状态然后执行第二阶段。第二阶段微干得多一些。第三阶段完成加载操作系统的任务。工作被分成三个阶段是因为 PC 标准第一阶段

行的程序的大小有所限制。把一些任务放在一起使得 FreeBSD 可以提供更大伸缩性的加载器 (loader)。

然后内核启动，它开始探测并初始化它。一旦内核引导程序完成任务，内核将控制权交给用程序 `init(8)`，它检查磁盘是否处于可用状态。`init(8)` 然后开始用磁盘源配置：加载文件系统、网络，及粗略地安装所有 FreeBSD 系统加载器常运行的程序。

## 13.3. 引导管理器和各引导阶段

### 13.3.1. The Boot Manager

在 MBR 或引导管理器中的代码有被提引导程序的阶段。一小便是前面提到引导器中的：boot0 和 LILO。

**boot0 引导管理器：**由 FreeBSD 的安装程序以及 `boot0cfg(8)` 所安装的 MBR，默认基于 `/boot/boot0`。(程序 boot0 非常小，由于在中的程序只能有 446 字节，分区表和 MBR 末端的 `0x55AA` 也要占一些空间。)如果已经安装 boot0 并且有多个操作系统在硬盘上，那如果安装了 FreeBSD MBR 而且安装了多个操作系统，会在系统看到似下面的提示：

例 10. boot0 截屏

```
F1 DOS
F2 FreeBSD
F3 Linux
F4 ??
F5 Drive 1

Default: F2
```

目前已知道一些其它操作系统，特别是 Windows®，会以自己的 MBR 覆盖有 MBR。如果生了什么事情，或者想用 FreeBSD 的 MBR 覆盖有的 MBR，可以使用以下的命令：

```
# fdisk -B -b /boot/boot0 device
```

*device* 是要写入 MBR 的名称，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，`da0` 代表第一个 SCSI 磁盘，等等。抑或，如果需要一个自行配置的 MBR，使用 `boot0cfg(8)`。

**The LILO Boot Manager:** 要想安装个引导管理器并也用来引导 FreeBSD，首先安装 Linux，并将以下加入到已有的配置文件 `/etc/lilo.conf`：

```
other=/dev/hdXY
table=/dev/hdX
loader=/boot/chain.b
label=FreeBSD
```

在上面的内容里，使用 Linux 的符号指定了 FreeBSD 的主分区和驱动器，将 X 替换 Linux 驱动器字母，将 Y 替换 Linux 主分区号。如果使用的是 SCSI 驱动器，需要将 `/dev/hd` 改成 `/dev/sd`，这里再次使用了 XY 的



法。如果安装的两个系在同一机器上，`loader=/boot/chain.b` 可以去掉。在可以行 `/sbin/lilo -v` 使修改生效；屏幕上的消息修改。

### 13.3.2. 第一段，`/boot/boot1`，和第二段，`/boot/boot2`

概念上，第一，第二段同属于一个程序，于磁的相同区域。但由于空间限制，它被分部分。可是它是会一起安装它。它由安装器或 `bsdlablel`(下文)制自被合而成的 `/boot/boot`。

它位于文件系统外，引导分区的第一道，从第一扇区始。在里 `boot0`，或者任何其它引导管理器，期望到一个程序行，引导程。所使用的扇区数可由 `/boot/boot` 的大小定。

`boot1` 非常，因它再多也只能有 512 字，只能存着分区信息的 `bsdlablel`，及行 `boot2`。

`boot2` 微有点加，能理解 FreeBSD 的文件系以便于里面的文件，能提供内核和加器的界面。

因 `loader` 有着更的功能，提供了一套易于使用的引导配置，`boot2` 一般都行 `loader`，但以前它的任是直接行内核。

例 11. `boot2` 的屏幕出

```
>> FreeBSD/i386 BOOT
Default: 0:ad(0,a)/boot/loader
boot:
```

如果要更改已安装的 `boot1` 和 `boot2`，使用命令 `bsdlablel(8)`。

```
# bsdlablel -B diskslice
```

`diskslice` 是用于引导的磁和分区，比如 `ad0s1` 代表第一个 IDE 磁上的第一个分区。



#### *dangerously dedicated*

如果在 `bsdlablel(8)` 命令中只使用了磁名，比如 `ad0`，就会破坏磁上的所有分区。当然不是所希望的，所以在按下 `回` 之前一定要命令行多次。

### 13.3.3. 第三段，`/boot/loader`

加器 (`loader`) 是三个段中的最后段，且是放置在文件系统之中的，一般是文件 `/boot/loader`。

`loader` 被作一友好的配置方式，使用了一内建且易用的命令集。些命令由一个大的多的解器支持建，其本身有得多的命令集。

#### 13.3.3.1. `Loader` 程序流程

初始，`loader` 会探控制台和磁，是从引导的。它会根据些信息置量，解器以接受通脚本或交互方式来的用命令。

`loader` 然后会取并行 `/boot/loader.rc`，默地取 `/boot/defaults/loader.conf` 以置可的默量，取

/boot/loader.conf 这些量作本地修改。loader.rc 依据这些量行作，加任何被的模和内核。

最后，默地，loader 会停留 10 秒等待按，若没有生中断，就始引内核。如果被中断，用会得到一个命令行提示符，在里用得更改量、卸所有模、加模、最后引或重新引。

### 13.3.3.2. Loader 内建的命令

些是最常用的 loader 命令。所有可用命令的解参 loader(8)。

#### **autobootseconds**

在定的内如果没有中断生就引内核。它示一个倒数，默的是 10 秒。

#### **boot [-options] [kernelname]**

立即按指定的指定名字的内核 (如果有指定的)。只有首先行 *unload* 命令之后指定的内核名字才会生效，否，的将是先前已加的内核。

#### **boot-conf**

基于量各模行自配置 (和引内核生的一)。只住要先使用 *unload* 命令，然后修改一些量，比如 *kernel*。

#### **help [topic]**

示从文件 /boot/loader.help 取的助信息。如果定的主是 *index*，那列出来的是所有可用的主。

#### **include filename ...**

通定的文件名理文件。文件被入，然后被一行一行地解。任何都会立即中止 include 命令。

#### **load [-t type] filename**

加内核、内核模，或者是定型的文件 (通定的文件名)。任何在文件名后面的参数都会被文件。

#### **ls [-l] [path]**

示定路径或者是根目 (如果路径没有指定) 下面的文件列表。如果指定了 *-l*，文件大小也会示。

#### **lsdev [-v]**

列出所有可以加模的。如果指定了 *-v*，会示出更多的。

#### **lsmod [-v]**

示已被加的模。如果指明了 *-v*，会示更多的。

#### **more filename**

示指定的文件，隔 *LINES* 停一次。

#### **reboot**

立即重系。

#### **set variable**

置 loader 的境量。

#### **unload**

移除所有已被加的模。

### 13.3.3.3. Loader 示例

这里有一些 loader 用法的示例

- 只是引导默认内核，不同的是输入模式：

```
boot -s
```

- 卸载默认内核和模块，然后加载旧的 (或者其它) 的内核：

```
unload  
  
load kernel.old
```

可以使用被称为通用内核的 `kernel.GENERIC`，或者以前安装的内核 `kernel.old` (当升级或配置了自己的内核等时候)。



使用以下命令加载常用的模块和一个内核：

```
unload  
set kernel="kernel.old"  
boot-conf
```

- 加载内核配置脚本：

```
load -t userconfig_script /boot/kernel.conf
```

### 13.3.3.4. 系统的 Splash 图像

在系统启动的 splash 图像比起原本的系统信息更加可靠。这个图像将被始终显示在屏幕上直到出现控制台的登录提示或者 X 显示管理器提供了登录画面。

在 FreeBSD 系统中有几个基本的环境。第一个是默认的控制台命令行环境。在系统启动之后，会在控制台上出现一个登录提示。第二个环境是 X11 图形环境。在安装好了 X11 和一个图形界面环境，比如 GNOME，KDE，或者 XFce，X11 图形可以用 `startx` 命令运行。

比起基于字符的登录提示，有些用户可能更喜欢 X11 图形化的登录界面。图形化的登录管理器像 Xorg 的 XDM，GNOME 的 gdm，KDE 的 kdm (以及其他 Port Collection 中的) 基本上都提供了一个图形化的登录界面代替控制台上的登录提示符。在成功登录之后，它展示给用户一个图形化的界面。

在命令行环境，splash 图像将在显示登录提示符之前隐藏所有系统的与任何系统的消息。在 X11 环境，用户将会得到一个视觉上更加清爽的界面，类似于某些像 (Microsoft® Windows® 或者非 UNIX® 类型的系统) 用户所希望得到的。

#### 13.3.3.4.1. Splash 图像功能

目前的 splash 图像的功能仅限于支持 256 色的位图 (.bmp) 或者 ZSoft PCX (.pcx) 文件。此外，splash 图像文件的分辨率必须是 320x200 像素或者更少，才能够在标准 VGA 适配器上使用。

要使用尺寸更大的图像，达到最大分辨率 1024x768 像素，需要 FreeBSD 的 VESA 支持。可以通过在系统添加 VESA 模式完成，或者在内核配置文件中加入 VESA 选项并编译（参看 [配置FreeBSD的内核](#)）。VESA 支持给予了用图像覆盖整个显示器的画面能力。

在适当的时候 splash 图像就会被显示在屏幕上，它可以在任何时候都按任意键。

Splash 图像同样也会是 X11 之外默认的屏幕保护。在一段适当的位置后，屏幕便会周期性的显示 splash 图像，从明亮至暗淡，周而复始。默认的 splash 图像（屏幕保护）可由 /etc/rc.conf 中的 `saver=` 控制。`saver=` 有一些内置的屏幕保护可供选，完整的列表可以再 [splash\(4\)](#) 手册中找到。默认的屏幕保护被称为 "warp"。注意在 /etc/rc.conf 中所指定 `saver=` 选项限于用于虚拟控制台。对于 X11 图形化的登录管理器无效。

一些有引导器的信息，包括菜单和一个定时倒数提示符都会在适当显示，即是用了 splash 图像功能。

splash 图像文件本身可以从 <http://artwork.freebsdgr.org> 下载。安装了 `sysutils/bsd-splash-changer` port 之后，下次适当的时候便能从集合中随机选 splash 图像。

#### 13.3.3.4.2. 使用 Splash 图像功能

Splash 图像 (.bmp) 或者 (.pcx) 文件必须放置在 root 分区上，比如 /boot 目录。

对于默认的显示分辨率 (256 色，320x200 像素或更少) 在 /boot/loader.conf，添加如下的设置：

```
splash_bmp_load="YES"
bitmap_load="YES"
bitmap_name="/boot/splash.bmp"
```

对于更高的分辨率，最大至 1024x768 像素，在 /boot/loader.conf，添加如下的设置：

```
vesa_load="YES"
splash_bmp_load="YES"
bitmap_load="YES"
bitmap_name="/boot/splash.bmp"
```

以上这些设置假设 /boot/splash.bmp 需要被使用的 splash 图像。当需要使用 PCX 文件的时候，添加入下列设置，根据分辨率的高低添加 `vesa_load="YES"`。

```
splash_pcx_load="YES"
bitmap_load="YES"
bitmap_name="/boot/splash.pcx"
```

文件名并不限于以上例子中的 "splash"。它可以是任何名称，只要是 BMP 或者 PCX 类型的文件，比如 splash\_640x400.bmp 或者 blue\_wave.pcx。

一些有趣的 loader.conf 选项：

**beastie\_disable="YES"**

将显示菜单，但是倒数仍然会出。即使是在菜单被禁用的时候，在倒数段输入相应的仍然有效。

**loader\_logo="beastie"**

将替换菜单右侧默认显示的 "FreeBSD" 彩色的小魔鬼标志，就像以往的行版那样。

参见 [splash\(4\)](#)，[loader.conf\(5\)](#) 和 [vga\(4\)](#) 手册获取更多信息。

## 13.4. 内核在引导的交互

一旦内核被 [loader](#)（一般情况下）或者 [boot2](#)（越过 loader）加载，它将引导标志，如果有的话，就会进行必要的调整。

### 13.4.1. 内核引导标志

这里是一些常用的引导标志：

**-a**

在内核初始化时，作为根加载的选项。

**-C**

从 CDROM 引导。

**-c**

运行 UserConfig (引导的内核配置器)

**-S**

引导入单用户模式

**-v**

在内核引导过程中显示更多的信息



有更多的引导标志，参见 [boot\(8\)](#) 以获取有关它的信息。

## 13.5. Device Hints

在初始化系统时，[loader\(8\)](#) 会读取 [device.hints\(5\)](#) 文件。该文件以量的形式存储着内核引导信息，有被称为 "device hints"。系统程序用 "device hints" 进行配置。

Device hints 也可以在 [第三段的boot loader](#) 的命令行提示符中指定。量可以用 **set** 命令添加，**unset** 命令删除，**show** 命令查看。在文件 `/boot/device.hints` 设置的量亦可以在这里被覆盖。输入 boot loader 中的量不是永久性的，在下次重启就会被忘记。

一旦系统引导成功，[kenv\(1\)](#) 命令可以用来清楚所有的量。

文件 `/boot/device.hints` 的写法是一行一个变量，使用 `"#"` 作注释。该行是按照如下方式写的：

```
hint.driver.unit.keyword="value"
```

第三段 `boot loader` 的写法是：

```
set hint.driver.unit.keyword=value
```

`driver` 是程序名，`unit` 是程序单位名，`keyword` 是 `hint` 关键字。关键字可以由以下组成：

- `at`：指明所指定的
- `port`：指明所使用 I/O 的起始地址。
- `irq`：指明所使用的中断请求号。
- `drq`：指明 DMA channel 号。
- `maddr`：指明占用的物理内存地址。
- `flags`：设置各标志位。
- `disabled`：如果成 1，被禁用。

程序能接受更多的 hints，推荐参看它的机手册。参看 [device.hints\(5\)](#)、[kenv\(1\)](#)、[loader.conf\(5\)](#) 和 [loader\(8\)](#) 机手册以取更多的信息。

## 13.6. Init：进程控制及初始化

一旦内核完成引导，它就把控制交给了用程序 [init\(8\)](#)，它放置在 `/sbin/init`，或者 `init_path` 变量指定的程序路径中。这个变量是在 `loader` 里面设置的。

### 13.6.1. 自重启程序

自重启程序会检查系统中可用的文件系统于健康的状态。如果不是，而且使用 [fsck\(8\)](#) 也无法修复些，[init\(8\)](#) 会入 [单用户模式](#) 以便系统管理直接修正些。

### 13.6.2. 单用户模式

此模式可以通过 [自重启程序](#) 或者通过有 `-s` 选项的引导或通过 `loader` 里置 `boot_single` 变量等多种方式来得到。

也可以在多用户模式下无重启 (`-r`) 和停机 (`-h`) 的 [shutdown\(8\)](#) 命令来入单用户模式。

如果系统控制台在文件 `/etc/ttys` 中被置为不安全 (`insecure`)，在初始化单用户模式前会出要求入 `root` 密码的命令行提示符。

```
# 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
```



把控制台置成 **不安全 (insecure)** 使只知道 **root** 密码的人才能进入单用户模式，因为物理控制台在物理上是不安全的。因此如果考虑到安全性，应该将控制台置成 **不安全 (insecure)**，而非 **安全 (secure)**。

### 13.6.3. 多用户模式

如果 `init(8)` 管理的文件系统一切正常，又或者用户在单用户模式完成了工作，系统就会进入多用户模式，开始系统的源配置。

#### 13.6.3.1. 源配置 (rc)

源配置分从文件 `/etc/defaults/rc.conf`、`/etc/rc.conf` 中取默认配置和配置，然后加在文件 `/etc/fstab` 中提及的文件系统、网络服务、各个系统守护进程，最后本地安装包的脚本。

`rc(8)` 手册是于源配置的很好的参考。

## 13.7. 关机 (shutdown) 进程

由命令 `shutdown(8)` 的起的关机进程中，`init(8)` 会执行 `/etc/rc.shutdown` 脚本，所有进程送 **TERM** 信号，最后不按停止的进程送 **KILL** 信号。

在支持源管理的平台上如 FreeBSD 系统的源，只要地使用命令 `shutdown -p now` 即可。此外，可以用命令 `shutdown -r now` 来重启 FreeBSD。要行 `shutdown(8)` 必须是 **root** 用户或 **operator** 的成员。也可以使用 `halt(8)` 和 `reboot(8)` 命令来关机，参看它们的机手册以得更多的信息。



源管理需要支持，要求内核支持 `acpi(4)` 或以模形式加它。

# Chapter 14. 用户和基本的用户管理

## 14.1. 概述

FreeBSD允许多个用户同时使用计算机。当然,有些用户中不是很多人同时坐在同一台计算机前。 ,而是其他用户可以通过网络来使用同一台计算机以完成他的工作.要使用系统,每个人都必须有一个用户。

读完本章,你将了解到:

- 在一个FreeBSD系统上不同用户之间的区别。
- 如何添加用户。
- 如何删除用户。
- 如何改变用户,如用户的全名,或用户的shell。
- 如何在用户的基本上设置限制,来控制像内存, CPU使用率的资源。
- 如何使用系统来使管理更容易。

在开始本章之前,应当了解:

- 了解UNIX®和FreeBSD的基本知识 ([UNIX 基础](#))。

## 14.2. 介绍

所有系统级的用户都是通过系统完成的,所以用户和管理是FreeBSD系统不可或缺的重要部分。

每个FreeBSD系统的用户都有一些和它相关的信息去描述它。

用户名

用户名在 **login:** 提示符的后面输入。 用户名于一台计算机来是唯一的; 不可以使用两个相同的用户名来登录。 有很多用来建立正用户名的命令, 具体参考 [passwd\(5\)](#); 使用的用户名通常需要8个或更少的小写字母。

口令

每个用户都有一个口令与它。 口令可以是空的, 用户不需要口令就可以登录。 通常不是一个好主意; 每个用户都有口令。

用户 ID (UID)

UID是系统用来描述用户的数字, 范围上它是0到65536之间的, 用以唯一地描述用户。 FreeBSD在内部使用UID来描述用户 - 在工作以前。 任何允许指定一个用户名的 FreeBSD 命令都会把它转换成UID。 这意味着可以用不同的用户名使用多个用户, 但它的UID是一致的。 FreeBSD 会把某些用户固定是同一个用户。

组ID (GID)

GID是用来描述用户所在的组的, 范围上它在0到65536之间的数字。 它是一个基于用户GID而不是它的UID的用来控制用户资源分配的机制。 可以减少一些配置文件的大小。 一个用户也可以属于多个组。

登录

登录是系统机制的扩展,当把系统分配给不同用户时,它提供了额外的灵活性。



口令的定期更改

默认情况下，FreeBSD 并不强制用root去改其他用户的口令。root可以用root位强制要求一些或所有的用户定期改他们的口令。

root的到期root

默认情况下，FreeBSD 不会自动完成root到期操作。如果root正在建root，root知道一个root的有效使用期限。例如，root在学校里root会root个学生建立一个root，root可以指定它root何root期。root期后，root然root的目录和文件仍然存在，但root已root不能root使用了。

用户的全名

用户名可以唯一地rootFreeBSD的root，但它不会反映用户的全名。有些信息可能与root是相同的。

主目录

主目录是用root用来root的目录的完全路径。一个通常的root是把所有用户的主目录都放在 /home/username 下，或者 /usr/home/username 下。用户root把他root的个人文件放在自己的主目录下，他root可以在那里root建任何目录。

用户 shell

Shell提供了用户用来操作系统的默认root境。有很多不同的shell，有root的用户会根据他root的root来root自己喜好的shell。

有三种类型的root: 超root用root，系root用root，以及 普通root用root。超root用root通常叫做 root，可以没有限制地管理系统。系root用root行服务。最后，普通root用root那些登录root以及root附件的人使用。

## 14.3. 超root用root

超root用root，通常叫做 root，可以重新配置和管理系统，在收root件，系root或root程root的日常工作中，尽量不要使用rootroot限。

root是因root不象普通root用root，超root用root能root无限制地操作系统，超root用root的root用可能会引起无法想象的root。普通的root用root不会由于出root而破坏系统，所以要尽可能的使用普通root，除非root需要root外的特root。

在使用超root用root命令root要再三root，因root一个root外的空格或缺少某个字符的命令都可能会引起数据root失。

所以，在root完root章后root第一件要做的事就是，在平常root使用的root候，root建一个没有特root的root用root。无root使用的是root用root是多root系root的root都是相同的。在root章的后面，我root将root如何root建一个root外的root和如何在普通root用root和超root用root之root行切root。

## 14.4. 系root用root

系root用root是那些要使用root如DNS、root件，web等服务root的root用root。使用root的原因就是安全；如果所有的root用root都由超root用root来root行，那它root就可以不受root束地做任何事情。

典型的系root用root包括 daemon、operator、bind (供 域名服务root 使用)、news，以及 www。

nobody 是普通的没有特root的系root用root。然而，大多数与root用root系很密切的服务root是使用 nobody的，root的root点非常重要，root可能使用root的非常有特root。

# 14.5. 用□□□

用□□□是□真□的用□□□系□的主要方式， □些□□把用□和□境隔□， 能阻止用□□坏系□和其他用□， 在不影□其他用□的情况之下定制自己的□境。

任何人□□□的系□必□要有他□自己唯一的□□。 □可以□□□到□做了什□事， 并且阻止人□破坏其他用□的□置和□□其他人的□件等等。

□个用□能□□置他□自己的□境， 以利于他□通□改□shell， □□器， □□□定和□言等□□并且更好的使用□个系□。

# 14.6. 修改□□

在UNIX® 的□理用□□□的□境中有很多不同的命令可用. 最普通的命令如下， 接下来是□□使用它□的例子。

| 命令                         | 摘要                   |
|----------------------------|----------------------|
| <a href="#">adduser(8)</a> | 在命令行添加新用□.           |
| <a href="#">rmuser(8)</a>  | 在命令行□除用□.            |
| <a href="#">chpass(1)</a>  | 一个□活的用于修改用□数据□信息的工具. |
| <a href="#">passwd(1)</a>  | 一个用于修改用□口令的□□的命令行工具. |
| <a href="#">pw(8)</a>      | 一个□大□活修改用□□□的工具.     |

## 14.6.1. 添加用□

[adduser\(8\)](#) 是一个□□的添加新用□的命令. 它□用□□建 passwd 和 group 文件。 它也□新用□□建一个主目□， 之后， 它会□制一□默□的配置文件 ("dotfiles") 从 /usr/shared/skel □个目□， 然后□新用□□送一封□□迎信息的□件。

```
# adduser
Username: jru
Full name: J. Random User Uid (Leave empty for default):
Login group [jru]:
Login group is jru. Invite jru into other groups? []: wheel
Login class [default]:
Shell (sh csh tcsh zsh nologin) [sh]: zsh
Home directory [/home/jru]:
Home directory permissions (Leave empty for default):
Use password-based authentication? [yes]:
Use an empty password? (yes/no) [no]:
Use a random password? (yes/no) [no]:
Enter password:
Enter password again:
Lock out the account after creation? [no]:
Username   : jru
Password   : ****
Full Name  : J. Random User
Uid        : 1001
Class      :
Groups     : jru wheel
Home       : /home/jru
Shell      : /usr/local/bin/zsh
Locked     : no
OK? (yes/no): yes
adduser: INFO: Successfully added (jru) to the user database.
Add another user? (yes/no): no Goodbye!
#
```



输入的口令并不会回显到屏幕上，此外系统也不会显示星号。请务必保证没有回显口令。

### 14.6.2. 删除用户

可以使用 `rmuser(8)` 从系统中完全删除一个用户。 `rmuser(8)` 运行如下：

1. 删除用 `crontab(1)` 的 (如果有的)。
2. 删除属于用的 `at(1)` 工作。
3. 删除属于用的所有进程。
4. 删除本地口令文件中的用。
5. 删除用的主目录 (如果他有自己的主目录)。
6. 删除来自 `/var/mail` 属于用的文件。
7. 删除所有如 `/tmp` 的文件存区中的文件。
8. 最后, 删除 `/etc/group` 中所有属于用的用户名。



如果一个成空, 而名和用户名一, 将被删除。  
唯一的。

`adduser(8)` 命令建立一个用。

`rmuser(8)` 不能用来删除超用的, 因那做是系大的破坏。

默情况下, 使用交互模式, 能清楚的知道在做什。

例 14. 删除 交互模式下的删除

```
# rmuser jru Matching password entry:
jru:*:1001:1001::0:0:J. Random User:/home/jru:/usr/local/bin/zsh Is this the entry
you wish to remove? y Remove user's home directory (/home/jru)? y Updating
password file, updating databases, done.
Updating group file: trusted (removing group jru -- personal group is empty) done.
Removing user's incoming mail file /var/mail/jru: done.
Removing files belonging to jru from /tmp: done.
Removing files belonging to jru from /var/tmp: done.
Removing files belonging to jru from /var/tmp/vi.recover: done.
#
```

### 14.6.3. `chpass`

`chpass(1)` 可以改用的口令, shells, 和包括个人信息在内的数据信息。

只有系管理, 即超用, 才可以用 `chpass(1)` 改其他用口令和信息。

除了可的用户名, 不需要任何, `chpass(1)` 将示一个包含用信息的器。可以改用在数据中的信息。



如果不是超用的, 在退出状态之后, 系会口令。

#### 例 15. 以超用户交互行 `chpass` 命令

```
#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:
```

普通用户只能改他很少的一部分信息。

#### 例 16. 以普通用户交互行 `chpass` 命令

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



`chfn(1)` 和 `chsh(1)` 只是到 `chpass(1)` 的符号链接，类似地，`ypchpass(1)`, `ypchfn(1)` 以及 `ypchsh(1)` 也是。NIS 是自动支持的，不一定要在命令前指定 `yp`。如果有点不太明白，不必担心，NIS 将在介绍。

### 14.6.4. `passwd` 命令

`passwd(1)` 是改自己作一个普通用户口令或者作超用户口令常用的方法。



用改口令前必须输入原来的口令，防止用终端非授权的用输入改合法用户的口令。

### 例 17. 改root的口令

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

### 例 18. 改root其他用户的口令同超root用root的一

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



就象 [chpass\(1\)](#) 一样， [yppasswd\(1\)](#) 只是一个到 [passwd\(1\)](#) 的链接， 所以NIS用任何一个命令都可以正常工作。

## 14.6.5. pw命令

[pw\(8\)](#) 是一个用来创建、删除、修改、显示用户和组的命令行工具。 它有系统用户和文件服务器的功能。 [pw\(8\)](#) 有一个非常大的命令行选项设置， 但新用户可能会觉得它比里面的其它命令要复杂很多。

## 14.7. 限制用户使用系统资源

如果你有一些用户， 并想要对他所使用的系统资源加以限制， FreeBSD 提供了一些系统管理限制用户系统资源的方法。 这些限制通常被分三类: 磁盘配额， 以及其它资源限制。

磁盘配额限制用户对磁盘的使用， 而且它还提供一种快速对用户使用磁盘数量而不需要时刻计算的方法。 配额将在文件系统配额实现。

其它资源限制包括CPU、 内存以及用户可能会使用的其它资源。 这些是通过登录进程行分完成的， 下面将做。

登录的由 `/etc/login.conf` 文件定义。 比精确的描述超出了本章的范围， 但 [login.conf\(5\)](#) 手册会有比详细的描述。 可以每个用户都分配到一个登录（默认是 `default`）， 每个登录都有一套和它相关的功能。 登录功能是 `名字=` 后面的一串， 其中名字 是一个众所周知的符号， 是一个根据名字管理得到的任意字符串。 配置登录和功能相当， 在 [login.conf\(5\)](#) 手册会有比详细的描述。



系并不直接取 `/etc/login.conf` 中的配置，而是采用数据文件 `/etc/login.conf.db` 以提供更快的能力。要从 `/etc/login.conf` 文件生成 `/etc/login.conf.db`，使用下面的命令：

```
# cap_mkdb /etc/login.conf
```

源限制与普通登录限制是有区别的。首先，对于限制，有限制（比常）和硬限制之分。一个限制可能被用整，但不会超硬限制。越往后可能越低，但不会升高。其次，大多数源限制会分配特定用的个程，而不是用的全部程。注意，些区别是源限制的特殊操作所定的，不是登录功能框架的完成（也就是，他上不是一个登录功能的特例）。

不再了，下面是大多数源限制的例子（可以在 [login.conf\(5\)](#) 到其它与登录功能相关的内容）。

### coredumpsize

很明，由程序生的核心文件大小的限制在磁使用上是属于其它限制的（例如，文件大小，磁配）。不，由于用自己无法生核心文件，而且通常并不除它，置个可以尽量避免由于一个大型用程序的崩所造成的大量磁空的浪。（例如，emacs）崩。

### cputime

是一个用程所能消耗的最 CPU 。反限制的程，将被内核掉。



是一个有 CPU 消耗的限制，不是 [top\(1\)](#) 和 [ps\(1\)](#) 命令屏幕上示的 CPU 消耗的百分比。在写此明，后者的限制是不太可能和没有价的：器 - 一个可能是合法的工作 - 可以在某一时刻易的用掉 100% 的 CPU。

### filesize

是用可以理一个文件的最大。不象磁配，个限制是个文件制行的，不是用自己的所有文件。

### maxproc

是一个用可以行的最大程数。包括前台和后台程。很明，不可能比系指定 [kern.maxproc sysctl\(8\)](#) 的限制要大。同也要注意，置的小会妨碍用的理能力：可能需要多次登录或行多个管道。一些任，例如一些大的程序，也可能会生很多程（例如，[make\(1\)](#)，[cc\(1\)](#) 以及其它一些理程序）。

### memorylocked

是一个程允到主存中的最大内存容量（参 [mlock\(2\)](#)）。大型程序，例如像 [amd\(8\)](#) 在遇到，它得到的巨大交量无法系行理。

### memoryuse

是在定内一个程可能消耗的最大内存数量。它包括核心内存和交内存。在限制内存消耗方面，不是一个完全的限制，但它是一个好的始。

### openfiles

是一个程可以打的最大文件数。在 FreeBSD 中，文件可以被表套接字和 IPC 通道；注意不要把个数置的太小。系的限制是由 [kern.maxfiles](#) 定的，情参 [sysctl\(8\)](#)。

## sbsize

它是网口内存数量的限制，它主要是通建立多套接字的老式 DoS 攻击的，但也可以用来限制网口通信。

## stacksize

它是一个程序堆可能到的最大。它不能单独的限制一个程序可能使用的内存数量；所以，需要与其他的限制手段配合使用。

在设置源限制，有一些其他的事需要注意。下面是一些通常的技巧、建议和注意事项。

- 系口的程序/etc/rc会被指派守护程序的登录。
- 当然 /etc/login.conf 文件是一个大多数限制做合理配置的源文件，但只有它也就是系口管理，才知道什么最适合的系口。设置的太高可能会因过于放而致系口被用，而设置低，可能降低效率。
- 使用 X Window 的用口可能要比其他用口使用更多的源。因X11本身就使用很多源，而且它鼓励用口同时行更多的程序。
- 必须注意，多限制措施是多个程序来实施的，它并不限制某一用口所能用到的量。例如，将 `openfiles` 设置 50 表示以口用身口的程序最多只能打 50 个文件。因而，用口可以打的文件数就是 `maxproc` 和 `openfiles` 的乘。内存用量的限与此似。

有源限制,登录的更深入信息可以看相关机手册: `cap.mkdb(1)`, `getrlimit(2)`, `login.conf(5)`。

## 14.8. 口

口的口就是一个用口列表。通口名和GID (口 ID) 来。在 FreeBSD (以及大多数其他 UNIX® 系) 中，内核用以决定一个程序是能完成一作的口因素是它所属的用口 ID 和口 ID。与用口 ID 不同，口个口程都有一个和它相关的口的列表。可能听口用口或口的 "口 ID"；大多数情况下，口表示列表中的第一个。

与口口的口名在/etc/group中。它是一个由冒号来界定的文本文件。第一部分是口名，第二部分是加密后的口令，第三部分是口ID，第四部分是以逗号相隔的成口列表。它可以用手工方式行 (当然，如果能保不出法的口!)。于更完整的法描述，参 `group(5)` 机手册。

如果不想手工 /etc/group，也可以使用 `pw(8)` 添加和。例如，要添加一个叫 `teamtwo` 的口，定它存在：

例 19. 使用 `pw(8)` 添加一个口

```
# pw groupadd teamtwo
# pw groupshow teamtwo
teamtwo:*:1100:
```

上面的数字 1100 是 `teamtwo` 的口 ID。目前，`teamtwo` 没有成，因此也就没有多大用。接下来，把 `jru` 加入到 `teamtwo`。



## 例 20. 使用 `pw(8)` 设置组成员列表

```
# pw groupmod teamtwo -M jru
# pw groupshow teamtwo
teamtwo:*:1100:jru
```

`-M` 所需的参数是一个用逗号分隔的将要成为成员的用列表。前面我已知道，口令文件中，一个用已指定了一个所属。之后用被自地添加到列表里；当我使用 `groupshow` 命令 `pw(8)` 用列表不被示出来。但当通过 `id(1)` 或者类似工具看，就会看到用列表。言之，`pw(8)` 命令只能取 `/etc/group` 文件；它从不从 `/etc/passwd` 文件取更多信息。

## 例 21. 使用 `pw(8)` 添加新的成员

```
# pw groupmod teamtwo -m db
# pw groupshow teamtwo
teamtwo:*:1100:jru,db
```

`-m` 的参数是一个由逗号分隔的即将被添加的列表。与先前那个例子的不同之处在于，个列表中的用将被添加而非取代中的有用。

## 例 22. 使用 `id(1)` 来决定成员

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

正如所看到的，`jru` 是 `jru` 和 `teamtwo` 的成员。

有 `pw(8)` 的更多信息，参看其它机手册。更多的关于 `/etc/group` 文件格式的信息，参考 `group(5)` 机手册。

# Chapter 15. 安全

## 15.1. 概述

这一章将介绍系统安全的基本概念。除此之外，还将介绍一些好的实践，以及 FreeBSD 下的一些更深入的实践。本章的许多内容对于一般的系统和 Internet 安全也有用。如今，Internet 已不再像以前那样是一个人人都愿意与作好邻居的“友善”的地方。系统更加安全，将保护的数据、智力财产、隐私，以及其他很多东西不至于被入侵者或心存恶意的人所窃取。

FreeBSD 提供了一系列工具和机制来保护系统和网络的完整及安全。

读完本章，你将了解：

- 基本的 FreeBSD 系统安全概念。
- FreeBSD 中许多可用的密码学措施，例如 DES 和 MD5。
- 如何设置一次性口令机制。
- 如何配置 TCP Wrappers 以便与 inetd 配合使用。
- 如何在 FreeBSD 上设置 Kerberos5。
- 如何配置 IPsec 并在 FreeBSD/Windows® 机器之间建立 VPN。
- 如何配置并使用 OpenSSH，以及 FreeBSD 的 SSH 运行方式。
- 系统 ACL 的概念，以及如何使用它。
- 如何使用 Portaudit 工具来核对从 Ports Collection 安装的第三方软件包的安全性。
- 如何从 FreeBSD 的安全公告中获得有用信息并采取相应措施。
- 对于编程功能的感性认识，并了解如何在 FreeBSD 中使用它。

在开始本章之前，你需要：

- 理解基本的 FreeBSD 和 Internet 概念。

其他安全方面的实践，贯穿本章的始终。例如，限制性访问控制 (MAC) 在[限制访问控制](#)中进行了介绍，而 Internet 防火墙在[防火墙](#)中进行了介绍。

## 15.2. 介绍

安全是系统管理自始至终的基本要求。由于所有的 BSD UNIX® 多用系统都提供了与生俱来的安全性，因此建立和维护额外的安全机制，以保护你的“隐私”可能也就是最需要系统管理者考虑的艰巨的工作了。机器的安全性取决于设置的安全措施，而许多安全方面的考虑，都会与人使用计算机的便利性相矛盾。一般来说，UNIX® 系统能胜任数目多进程并地理地管理各项任务，但其中的许多进程是以服务自身运行的 - 这意味着，外部实体能够与它交互并会产生交互。如今的桌面系统，已能够做到许多昔日的小型机甚至主机的性能，而随着某些计算机的联网和在更大范围内完成交互，安全也成了一个日益严峻的问题。

系统的安全也能对付各种形式的攻击，也包括那些使系统崩溃，或阻止其正常运行，并不限于窃取 root 帐号 (“破 root”) 的攻击形式。安全可大体分为以下几类：

1. 拒绝服务攻击。
2. 窃取其他用户的密码。
3. 通过可写服务窃取root密码。
4. 通过用root窃取root密码。
5. 建立后门。

拒绝服务攻击是侵占机器所需资源的一种行为。通常，DoS攻击采用暴力(brute-force)手段通过倒灌的流量来破坏服务器和网络，以使机器崩溃或无法使用。某些DoS攻击利用在网络中的漏洞，用一个小小的信息包就可以使机器崩溃，这种情况通常只能通过内核打补丁来修复。在一些不利的条件下，服务器的攻击能被修复，只要当地修改一下系统的配置来限制系统服务器的负荷。网络的攻击是很复杂的。例如，一个欺骗性信息包的攻击，无法阻止入侵者切断系统的系统与Internet的连接。它不会使你的机器死掉，但它会把Internet连接占满。

窃取用户密码要比D.o.S.攻击更加普遍。许多系统管理仍然在他的服务器上运行着基本的telnetd, rlogind, rshd和ftpd服务。这些服务在默认情况下不会以加密连接来操作。如果是如果的系统有中等规模大小的用户群，在通过远程登录的方式登录到系统的用户中，一些人的口令会被人窃取。仔细的系统管理会从那些成功登录系统的远程日志中可疑的源地址。

通常必须假定，如果一个入侵者已经得到了一个用户的密码，那么它就可能使自己成为root。然而，事情是在一个安全和做得很好的系统中，用户的密码不一定会使入侵者成为root。这个差别是很重要的，因为没有成为root的入侵者通常是无法隐藏它的踪迹的，而且，如果走运的话，除了用户的文件乱掉和系统崩溃之外，它不能做什么事情。窃取用户密码是很普遍的事情，因为用户往往不会对系统管理的警告采取措施。

系统管理必须牢牢记住，可能有多种潜在的方法会使他机器上的root用户受到威胁。入侵者可能知道root的口令，而如果在以root权限运行的服务器上找到一个缺陷(bug)，就可以通过网络连接到那台服务器上达到目的；另外，一旦入侵者已经侵入了一个用户的密码，可以在自己的机器上运行一个suid-root程序来利用服务器的漏洞，从而他侵入到服务器并获取root。攻击者到了入侵一台机器上root的途径之后，他就不再需要安装后门了。许多root漏洞被修复并修正之后，入侵者会想尽办法去删除日志来消除自己的痕迹，所以他不会安装后门。后门能入侵者提供一个小小的方法来重新获取系统的root权限，但它也会明显的系统管理一个入侵的简便方法。入侵者无法安装后门事实上系统的安全是有害的，因此并不会修复那些侵入系统的入侵者所开的新漏洞。

安全的管理方法应当使用像“洋葱皮”——多层次的方法来开，这些措施可以按下面的方式划分：

1. 保护root和普通人用户的安全。
2. 保护root - 以root用户权限运行的服务器和suid/sgid可执行程序的安全。
3. 保护用户的安全。
4. 保护口令文件的安全。
5. 保护内核中核心文件、直接内存和文件系统的安全。
6. 快速修复系统中产生的不当的变化。
7. 做个偏执狂。

一章的下一节将比这更深入地描述上面提到的每一个条目。

## 15.3. 保护 FreeBSD 的安全



命令与选项

在本文档中，我将使用 **粗体** 来表示用户程序，并使用 **倍距** 字体来表示命令。选项的排版区分能有效地区分类似 `ssh` 选项的概念，因为它既可以表示命令，又可以表示选项。

接下来的几章中，将介绍在下一章中[前一章](#)中所介绍的那些加固 FreeBSD 系统安全性的手段。

### 15.3.1. 保护 `root` 和普通人用户的安全

首先，如果没有保护 `root` 用户的安全，就没必要先考虑保护普通用户的安全了。大多数系统都会指派一个口令给 `root` 用户。我的第一个假定是，口令是不安全的。但这并不意味着要把口令去掉。口令通常是控制台的必需部分。也就是说，要避免允许在控制台以外的地方使用口令，甚至包括使用 `su(1)` 命令的情形。例如，相信的 `pty` 终端在 `/etc/ttys` 文件中被指定为 `insecure` (不安全)，将使直接通过 `telnet` 或 `rlogin` 登录 `root` 不会被接受。如果使用如 `sshd` 的其他登录服务，也要直接登录 `root` 是不可能的。可以通过 `/etc/ssh/sshd_config` 文件来做到这一点，相信 `PermitRootLogin` 被置成 `no`。考虑到一种方法 - 如 FTP 服务，以免因它而导致安全性的丢失。直接登录 `root` 只有通过系统控制台才被允许。

当然，作为一个系统管理员，当得到 `root` 身份，因此，我进行了一些后来允许自己进入。但此后只有在进行了额外的口令操作之后才能使用。一种 `root` 可行的方法是添加适当的用户到 `wheel` 组 (在 `/etc/group` 中)。`wheel` 组中的用户可以使用 `su` 命令来成为 `root`。不能通过口令中的行置来授予任何人天然的 `wheel` 组成员身份。普通人被放置在 `staff` 组中，然后通过 `/etc/group` 文件加入到 `wheel` 组。事实上，只有那些需要以 `root` 身份进行操作的用户才需要放入 `wheel` 组中。当然，也可以通过某些其它的手段，例如 Kerberos，可以通过 `root` 组中的 `.k5login` 文件来允许行 `ksu(1)` 成为 `root`，而不必把它放入 `wheel` 组。这可能是一种更好的解决方案，因为 `wheel` 机制仍然可能导致入侵者得到 `root`，如果他拿到了口令文件，并能进入系统的。尽管有 `wheel` 比什么都没有要好一些，但它并不是一种安全的办法。

可以使用 `pw(8)` 命令来完全禁止某一个用户：

```
# pw lock staff
```

将阻止用户使用任何方法登录，包括 `ssh(1)`。

一个阻止某个用户的方法是使用一个 `***` 字符替换掉加密后的口令。将不会与任何加密后的口令匹配，从而阻止了用户的登录。例如：

```
foobar:R9DT/Fa1/LV9U:1000:1000::0:0:Foo Bar:/home/foobar:/usr/local/bin/tcsh
```

被改为：

```
foobar:*:1000:1000::0:0:Foo Bar:/home/foobar:/usr/local/bin/tcsh
```

会阻止用 `foobar` 使用的方式登录。但是由于使用了 Kerberos 或者配置了 `ssh(1)` 公钥/密钥的情况下，用户依然可以登录。

一些安全机制同假定，从严格受限的机器向限制更松的机器上登。例如，如果的服务器行了所有的服务，那，工作站什么都不行。了工作站尽可能地安全，避免行任何没有必要的服务，甚至不行任何服务。外，也考虑使用口令保护功能的屏幕保护程序。毋庸置疑，如果攻者能物理地接触的 workstation，那他就有能力破坏任何安全措施，是我需要考的一个，但同地，真正能物理接触的 workstation 或服务并非施攻的人在日常生活中并不常，大多数攻来自于网，而攻者往往无法物理地接触服务器或工作站。

使用似 Kerberos 的工具，也我提供了使用一个工具来禁用某个用，或修改他的口令，并在所有机器上立即生效的方法。如果工的号被窃取，能在所有的其他机器上生效的口令更将很有意思。如果口令分散地保存在多个机器上，一次修改 N 台机器上的口令很可能是一件痛苦的事情。此外，Kerberos 能提供更多的限制，除了 Kerberos 令牌有很好的过期机制之外，它能限制用在某个特定的期限内修改口令(比如，月一次)。

### 15.3.2. 保以root用限制行的服务器和suid/sgid可行政程序的安全

慎的管理只行他需要的服务，不多，不少。要当心第三方的服务程序很可能有更多的。例如，行旧版的 `imapd` 或 `popper` 无于将 `root` 令牌拱手送全世界的攻者。永不要行那些没有仔的服务程序，外也要知道，多服务程序并不需要以 `root` 的身行。例如，`ntalk`、`comsat`，以及 `finger` 些服务，都能以一被称作 沙盒 的特殊用的身行。除非已解决掉了多麻烦的，否沙盒就不是完美的，但洋葱式安全仍然成立：如果有人法攻破了在沙盒中行的程序，那在做更多坏事之前，他必想法攻破沙盒本身的限制。攻者需要攻破的次数越多，他成功的可能性就越小。去，破解 `root` 的漏洞几乎在所有以 `root` 身行的服务上都，包括那些基本的系服务。如果的机器只打算向外界提供 `sshd` 登，而用不会使用 `telnetd` 或 `rshd` 甚至 `rlogind` 登，就毫不犹豫地它！

FreeBSD 在默在沙盒中行 `ntalkd`，`comsat`，以及 `finger`。此外，`named(8)` 也可以行。`/etc/defaults/rc.conf` 中包括了如何如此行 `named` 的方法，只是些内容被注掉了。如何升或安装系将决定些沙盒所使用的特殊用是否被自安装。慎的系管理将根据需要研究并沙盒。

此外，有一些服务通常并不在沙盒中行：`sendmail`，`popper`，`imapd`，`ftpd`，以及一些其他的服务。当然，它有一些替代品，但安装那些服务可能需要做更多外的工作。可能必以 `root` 身行些程序，并通其他机制来入侵。

系中一个比大的 `root` 漏洞是安装在其中的 `suid-root` 和 `sgid` 的可行文件。大多数程序，例如 `rlogin` 会放在 `/bin`、`/sbin`、`/usr/bin`，或 `/usr/sbin` 目中。尽管并没有 100% 的安全保，但系默的 `suid` 和 `sgid` 可行文件通常是相安全的。当然，偶也会一些存在于些可行文件中的 `root` 漏洞。1998年，`Xlib` 中了一个 `root` 漏洞，使得 `xterm` (通常是做了 `suid` 的) 得可以入侵。做得安全些，比比出再后悔要。因此，慎的管理通常会限制 `suid` 可行文件，并保只有工号能行它，或只放特定的用，甚至底干掉 (`chmod 000`) 任何 `suid` 可行文件，以至于没有人能行它。没有示的服务器通常不会需要 `xterm` 可行文件。`sgid` 可行文件通常同地危。一旦入侵者攻克了 `sgid-kmem`，那他就能取 `/dev/kmem` 并而取加密的口令文件，从而窃取任何包含口令的号。外，攻破了 `kmem` 的入侵者能通 `pty` 送的按序列，即使用使用的是安全的登方式。攻破了 `tty` 的用能向几乎所有用的 `tty` 写入数据。如果用正在行一个端程序，或包含了模功能的端真程序，那，入侵者能以那个用的身行任何命令。

### 15.3.3. 保用的安全

用的号的安全通常是最保的。然可以用的工置苛的登限制，并用 "星号" 替掉他的口令，但可能无法普通的用做。如果有足的决策，那在保用号安全的斗争中或会于，但如果不是

， 能做的只是警告地控制些号的。 用使用 `ssh` 或 `Kerberos` 可能会有更多的， 因需要更多的管理和技术支持， 尽管如此， 与使用加密的口令文件相比， 仍不失一个好法。

#### 15.3.4. 保口令文件的安全

能保起作用的唯一方法， 是将口令文件中尽可能多的口令用星号代替， 并通 `ssh` 或 `Kerberos` 来使用些号。 即使只有 `root` 用能取加密的口令文件 (`/etc/spwd.db`)， 入侵者仍然可能法到它的内容， 即使他无法写入个文件。

的安全脚本常并告口令文件的 (参后面的 [文件完整性](#) 一)。

#### 15.3.5. 保内核中内核、直接和文件系统的安全

如果攻者已拿到了 `root` 那他就有能力作任何事情， 当然， 有一些事情是他比喜干的。 例如， 大多数代的内核都包括一个内建的听包。 在 `FreeBSD` 中， 个被称作 `bpf`。 攻者通常会 在攻克的系上行它。 如果不需要 `bpf` 提供的功能， 那， 就不要把它入内核。

但是， 即使了 `bpf`， 仍需要注 `/dev/mem` 和 `/dev/kmem`。 就事地， 入侵者仍然能通直接 的方式写入磁。 外， 有一个称作模加器的内核机制， [kldload\(8\)](#)。 有取心的入侵者， 可以由 一机制， 在正在行的内核中通 `KLD` 模来安装自己的 `bpf`， 或其它听包。 了避免些， 必 将内核的安全提高到至少 1。

内核的安全可以通过多方式来置。 最的置正在行的内核安全的方法， 是使用 `sysctl` 来置内核量 `kern.securelevel`：

```
# sysctl kern.securelevel=1
```

默情况下， `FreeBSD` 内核的安全是 -1。 除非管理或 [init\(8\)](#) 由于脚本加以改， 安全会保持 -1。 在系程中， 可以在 `/etc/rc.conf` 文件中， 将量 `kern.securelevel_enable` 置 `YES` 并将 `kern.securelevel` 置希望的安全来提高它。

默情况下， 在脚本行完之后， `FreeBSD` 的安全置是 -1。 称作 "不安全模式"， 因文件的不可修改 (`immutable flag`) 可以改， 而且全部可以直接行写， 等等。

一旦将安全置 1 或更高， 只允追加 (`append-only`) 和不可修改会被行， 而且不可以。 直接裸会被拒。 更高的安全会施加一的限制。 于安全的完整介， 参机手册 [security\(7\)](#) (于 `FreeBSD 7.0` 之前的版本， 是机手册 [init\(8\)](#))。



将安全整到 1 或更高可能会致 X11 ( `/dev/io` 会被阻止)， 或从源代 `FreeBSD` (一程中的 `installworld` 部分需要取消一些文件上的只允追加和不可修改) 出一些， 并致一些其他小。 有些候， 例如 X11 的情况， 可以通在引程中早的 段 [xdm\(1\)](#) 来， 因安全很低。 似的方法， 于某些安全 或限制有可能不可用。 提前做好可能是个好主意。 理解不同的安全 所施加的限制非常重要， 因一些限制可能系得很使用。 外， 了解它 也有助于理性地配置默定。

如果内核的安全 1 或更高， 在重要的程序、 目和脚本文件上置 `schg` (也就是在系到 置安全之前行的程序和它的配置) 就有意了。 然而， 做也可能有些火， 而由于系行于



高的安全，提升系数也会得多。作妥协，可以系数以高的安全可行，但并不将所有的文件都配置 `schg`。一种方法是将 `/` 和 `/usr` 以只读模式挂。注意，过分苛的安全配置很可能限制入侵的能力。

### 15.3.6. 文件完整性: 可执行文件，配置文件和其他文件

当施严格的限制，往往会在使用的方便性上付出代价。例如，使用 `chflags` 来把 `schg` 用到 `/` 和 `/usr` 中的大多数文件上可能会起到反作用，尽管它能保护那些文件，但同时也使入侵无法进行。次化安全的最后一项可能也是最重要的 - 备份。如果无法看出潜在的入侵行，那安全的其他部分可能相来意可能就不那大了（或者，更糟的事情是，那些措施会安全的假象）。次化安全最重要的功能是入侵者，而不是底不他入侵，才可能当住入侵者。

入侵的一项好法是那些被修改、删除或添加的文件。文件修改的最佳方法是与某个（通常是中央的）受限的系数上的文件行比。在一台严格限制的系数上撰写的安全脚本通常不能被入侵者察，因此，非常重要。了最大限度地一策略的，通常会使用只读的 NFS，或者置 `ssh` 匙以便其他机器提供。除了网交互之外，NFS可能是一项很被察的方法 - 它允控制一台客机上的文件系统，而控制几乎是无法察的。如果一台格受限的服务器和客机是通交换机接的，那 NFS 将是一项非常好的方式。不，如果那台控制服务器和客机之通集器（Hub），或多多的路由来接，方式就很不安全了，此，考使用 `ssh`，即使可以在中到。

一旦个受限的机器予了至少取它控制的客系，就写的控制脚本。以 NFS 挂接例，可以用似 `find(1)` 和 `md5(1)` 的命令基来完成我所需的工作。最好能天天被控机的所有行文件算一遍 `md5`，同，以更高的率那些 `/etc` 和 `/usr/local/etc` 中的控制文件。一旦了不匹配的情形，控机立即通知系管理。好的安全脚本也在系分区，如 `/` 和 `/usr` 中是否有新或删除的可执行文件，以及不宜的 `suid`。

如果打算使用 `ssh` 来代替 NFS，那撰写安全脚本将得多。本上，需要在脚本中使用 `scp` 在客端制文件，一方面，用于的行文件（例如 `find`）也需要使用 `scp` 到客端，因 `ssh` 客程序很可能已被攻陷。之，在一条不安全的路上 `ssh` 可能是必的，但也必付它所来的。

安全脚本用以及成的限置文件：`.rhosts`、`.shosts`、`.ssh/authorized_keys` 等等。些文件可能并非通 MD5 来行。

如果的用磁空很大，分区上面的文件可能非常耗。情况下，采用志来禁止使用 `suid` 可执行文件将是一个好主意。可能会想看看 `nosuid`（参 `mount(8)`）。尽管如此，些描仍然至少周行一次，做的意并不是有效的攻，而是攻企。

程（参 `accton(8)`）是一项成本低的，可以助在被入侵后估失的机制。于出入侵者是如何入系的件事来，它会非常的有所助益，特是当入侵者什文件都没有修改的情况下。

最后，安全脚本理日志文件，而日志文件本身通尽可能安全的方法生成 - 程 `syslog` 可能非常有用。入侵者会掩他的踪迹，而日志文件于希望了解入侵生出的系管理来得尤重要。保持日志文件的永久性的一项方法是在串口上行系控制台，并在一台安全的机器上收集些信息。

### 15.3.7. 偏

点偏不会来害。作一例，系管理在不影使用的便利的前提下可以用任何安全特性，此外，在深思熟之后，也可以加一些会使用得不那方便的安全特性。更重要的是，有安全意的管理学会混合不同的安全策略 - 如果逐字逐句地按照文来配置的机器，那无于向那些同能得到文的攻者透露了更多的信息。

### 15.3.8. 拒服攻

一将介绍拒服攻。 DoS 攻通常是基于数据包的攻， 尽管几乎没有任何方法来阻止大量的构造数据包耗尽网源， 但通常可以通过一些手段来限制攻的害， 使它无法服器：

1. 限制服程 fork。
2. 限制 springboard 攻 (ICMP 攻， ping 广播， 等等)。
3. 使内核路由存。

一比较常的 DoS 攻情形， 是通攻制程 (fork) 的服， 使其生大量子程， 从而是其行的机器耗尽内存、 文件描述符等源， 直到服器底死掉。 inetd (参 [inetd\(8\)](#)) 提供了多来限制攻。 需要注意的是， 尽管能阻止一台机器底掉， 但通常无法防止服本身被。 仔 inetd 的机手册， 特是它的 `-c`、 `-C` 以及 `-R` 三个。 造 IP 攻能 inetd 的 `-C`， 因此， 些需要配合使用。 某些独立的服器也有似的限制参数。

例如， Sendmail 就提供了自己的 `-OMaxDaemonChildren`， 它通常比 Sendmail 的限制更有效， 因服器算有滞后性。 可以在 `sendmail` 指定一个 `MaxDaemonChildren` 参数； 把它的高以便承所需要的荷， 当然， 不要高到足以行 Sendmail 的机器死掉。 此外， 以列模式 (`-ODeliveryMode=queued`) 行 Sendmail 并把服程序 (`sendmail -bd`) 和列行程序分行 (`sendmail -q15m`) 也是一个好主意。 如果希望保列的性， 可以考使用更短的隔， 例如 `-q1m`， 但同也需要指定一个合理的子程数， 也就是通 `MaxDaemonChildren` 以免那个 Sendmail 造成重的故障。

Syslogd 可以被直接地攻， 因此， 烈建只要可行， 就在它的候加上 `-s` 参数， 其他情况下， 至少加上 `-a`。

于基于接的服， 例如 TCP Wrapper 的 `reverse-identd`， 都格外的小心， 因它都可能直接遭受攻。 一般情况下， 基于安全考， 不使用 TCP Wrapper 所提供的 `reverse-ident` 的功能。

此外， 将内部服保起来， 阻止来自其他主机的也十分重要， 些工作可以通过置界路由器来完成。 主要的想法， 是阻止来自的 LAN 以外的， 有助于避免 `root` 受到攻。 尽可能配置排他式的防火， 例如， "用防火阻止所有的网流量 除了 端口 A、 B、 C、 D， 以及 M-Z"。 通采用方法， 可以很容易地将低端口的阻止在外， 而又不配置使防火放那些明需要放的服， 例如 `named` (如果的机器准作域的主要解析服器)， `ntalkd`， `sendmail`， 以及其他可以从 Internet 的服。 如果以其他方式配置防火 - 采用比松的策略， 那将很有可能忘 "掉" 一个服， 或者在加了一些服之后忘更新防火策略。 尽管如此， 仍然可以考允数据入号高的那一部分端口， 将保那些需要特性的服能正常工作， 而又不影低端口服的安全性。 此外， 注意到 FreeBSD 允来控制定的端口的， 即一系列 `net.inet.ip.portrange` 量， 通 `sysctl` 来完成置。( `sysctl -a | fgrep portrange`)。 使得完成防火策略得易如反掌。 例如， 可能希望普通的高段端口的起止是 4000 到 5000， 而更高是 49152 到 65535， 随后在防火中阻止低于 4000 的所有端口 (当然， 除了那些特地 Internet 而的端口)。

一常被称作 springboard 的攻也是非常常的 DoS 攻 - 它通使服器生其无法理的来到的目的。 最常的攻就是 ICMP ping 广播攻。 攻者通造 ping 包， 将其源 IP 置希望攻的机器的 IP。 如果的界路由器没有行禁止 ping 广播地址的置， 的网将最陷于造的 ping 包之中， 特是当攻者同时使用了多个不同的网。 广播攻能生超 120 兆位的瞬流量。 一常的 ICMP 告系的 springboard 攻， 通建立可以生成 ICMP 出的包， 攻者能攻服器的网下行源， 并致其上行源耗尽。 型的攻也可以通耗尽内存来使得被攻的服器崩， 特是当些服器无法快地完成 ICMP 的候。 新的内核可以通过整 `sysctl` 量 `net.inet.icmp.icmplim` 来限制攻。 最后一主要的 springboard 是某些 `inetd` 的内部服， 例如 `udp echo` 服行的。 攻者地造一个来自服器 A 的



echo 口的 UDP 包，然后将这个包发到 B 的 echo 口。于是，这台服务器将不停地将包发回 A。攻击者能将这台服务器的服务器都耗竭，并且通过这种方式，只需要很少的包就可以让 LAN 超载。类似的攻击 chargin 口也是存在的。好的系统管理应该限制一些 inetd 的服务。

构造的包攻击也可以用来使内核的路由缓存。参考 `net.inet.ip.rtexpire`, `rtminexpire`, 以及 `rtmaxcache sysctl` 参数。构造的包可以用随机的源 IP 攻击，使得内核在路由表中产生一个空的缓存，它可以通过 `netstat -rna | fgrep W3` 看到。这些路由通常需要 1600 秒才会过期。如果内核的路由表变得太大，它会不断地降低 `rtexpire` 但以 `rtminexpire` 为限。引出了一个问题：

1. 在容量不大的服务器上，内核对于突然增加的反响不快。
2. `rtminexpire` 的值没有低到内核在此攻击存活下去的程度。

如果服务器的网络 T3 或更快的线路接入 Internet，那通过 `sysctl(8)` 来手动地降低 `rtexpire` 和 `rtminexpire` 就非常必要。当然，不要把它设置为零（除非你想让机器崩溃）将这两个参数设置 2 通常已足以抵御攻击了。

### 15.3.9. Kerberos 和 SSH 的认证

如果你打算使用，那 Kerberos 和 ssh 都有一些需要解决的问题。Kerberos 5 是一个很棒的认证，但使用了它的 telnet 和 rlogin 的程序有一些 bug，使得它不适合合理二流制流。而且，除非使用了 `-x` 选项，否则默认情况下 Kerberos 并不加密会话。ssh 在默认加密所有的会话内容。

除了默认加密会话之外，ssh 在所有的其他方面都做得很好。这意味着如果你持有供其他部分密码的工作站作了很好的安全防护，而得到了一台不安全的机器上，你的密码可能被入侵者得到。尽管你的密码并没有被泄漏，但由于 ssh 会在登录的过程中用一个端口，如果攻击者拿到那台不安全的机器上的 root 那他将会利用那个端口来使用你的密码，从而能够访问那些机器。

我建议在使用 ssh 时配合 Kerberos 来完成工作人员的登录过程。Ssh 在配置中可以加入 Kerberos 支持。在缺少了潜在地暴露 ssh 密码的机会的同时，它能够通过 Kerberos 来保护口令。Ssh 密码只有在做了安全防护的机器上自行操作时才使用（是 Kerberos 不适合的情形）。此外，我建议要在 ssh 配置中设置密码，要在 `authorized_keys` 中加 `from=IP/DOMAIN` 选项，来限制哪些密码能登录的来源机器。

## 15.4. DES、Blowfish、MD5，以及 Crypt

UNIX® 系统上的每个用户都有一个与其相关的口令。很自然，密码只需要被用户和操作系统知道。为了保护口令的私密性，采用了一种称“单向散列”的方法来管理口令，即地，很容易从口令推算出散列，反之却很。其，刚才那句可能并不十分确切：因操作系统本身并不真的知道你的口令。它只知道口令加密的形式。取口令“明文”的唯一方法是采用暴力在口令可能的区内。

不幸的是，当 UNIX® 出现时，安全地加密口令的唯一方法基于 DES，数据加密标准（the Data Encryption Standard）。于是那些非美国居民来了，因 DES 的源代码在当时不能被出口到美国以外的地方，FreeBSD 必须到符合美国法律，但又要与其他那些使用 DES 的 UNIX® 版本兼容的方法。

解决方案是把加密函数分割成两个，于是美国的用户可以安装并使用 DES 函数，而国外用户则使用另外一套提供的一套可以出口的加密算法。就是 FreeBSD 为什么使用 MD5 作为它的默认加密算法的原因。MD5 据信要比 DES 更安全，因此，安装 DES 更多地是出于兼容目的。

### 15.4.1. 系统采用的加密算法

在单个支持 DES、MD5 和 Blowfish 散列函数。默认情况下，FreeBSD 使用 MD5 来加密口令。

可以很容易地在 FreeBSD 使用加密方法。在 `/etc/master.passwd` 文件中的加密密码是一种方法。用 MD5 散列加密的密码通常要比用 DES 散列得到的密码一些，并且以 `$1$` 字符开始。以 `$2a$` 开始的口令是通过 Blowfish 散列函数加密的。DES 密码字符没有任何可以用于密码的特征，但他要比 MD5 短，并且以不包括 `$` 在内的 64 个可显示字符来表示，因此相比短的、没有以美元符号开头的字符串很可能是一个 DES 口令。

新口令所使用的密码格式是由 `/etc/login.conf` 中的 `passwd_format` 来控制的，可供选择的算法包括 `des`, `md5` 和 `blf`。参考 [login.conf\(5\)](#) 可以获得更详细的情况。

## 15.5. 一次性口令

默认情况下，FreeBSD 提供了 OPIE (One-time Passwords In Everything) 支持，它默认使用 MD5 散列。

下面将介绍三种不同的口令。第一种是常用的 UNIX® 风格或 Kerberos 口令；我在后面的章节中将称其为 "UNIX® 口令"。第二种是使用 OPIE 的 `opiekey(1)` 程序生成，并由 `opiepasswd(1)` 以及登录提示所接受的一次性口令，我称其为 "一次性口令"。第三种口令是输入 `opiekey` 程序（有些时候是 `opiepasswd` 程序）用以生成一次性口令的秘密口令，我称其为 "秘密口令" 或通俗地称其为 "口令"。

秘密口令和 UNIX® 口令毫无关系，尽管可以设置相同的，但不推荐这样做。OPIE 秘密口令并不像旧式的 UNIX® 口令那样只能限于 8 位以内。想要用多位的口令都可以。有六、七个位的短句是很常见的。在大多数时候，OPIE 系统和 UNIX® 口令系统完全相互独立地工作。

除了口令之外，对于 OPIE 还有两至三重要的数据。其一被称作 "种子" 或 "key"，它包括 6 个字符和五个数字。第二个被称作 "迭代次数"，它是一个 1 到 100 之间的数字。OPIE 通常将种子加到秘密口令后面，并进行迭代次数那多次的 MD4/MD5 散列计算来得到结果，并将结果表示为 6 个短的英文字母。这 6 个英文字母就是一次性口令。系统（主要是 PAM）会记住上次使用的一次性口令，如果用提供的口令的散列与上次一致，可以通过身份验证。由于使用了双向的散列函数，因此即使截取了上次使用的口令，也没有办法恢复出下次将要使用的口令；每次成功登录都将导致迭代次数增加，用户和登录程序将保持同步。当迭代次数至少到 1 时，都必须重新初始化 OPIE。

接下来将介绍三个系统有的三个程序。`opiekey` 程序能够接收迭代次数，种子和秘密口令，并生成一个一次性口令，或是一个包含所有的一次性口令的表格。`opiepasswd` 程序用于初始化 OPIE，并修改口令、迭代次数、种子和一次性口令。和 `opieinfo` 程序可以用于管理相关的数据库文件 (`/etc/opiekeys`) 并显示运行命令的当前迭代次数和种子。

我将介绍四种不同的操作。在安全的接口上通过 `opiepasswd` 来第一次设置一次性口令，或修改口令及种子。第二种操作是在不安全的接口上使用 `opiepasswd` 以在安全接口上运行的 `opiekey` 来完成同样的工作。第三种操作是在不安全的接口上使用 `opiekey` 来登录。最后一种操作是采用 `opiekey` 来生成大批的密码，以便抄下来或打印出来，在没有安全接口的地方使用。

### 15.5.1. 安全接口的初始化

第一次初始化 OPIE 时，可以使用 `opiepasswd` 命令：

```
% opiepasswd -c
[grimreaper] ~ $ opiepasswd -f -c
Adding unfurl:
Only use this method from the console; NEVER from remote. If you are using
telnet, xterm, or a dial-in, type ^C now or exit with no password.
Then run opiepasswd without the -c parameter.
Using MD5 to compute responses.
Enter new secret pass phrase:
Again new secret pass phrase:
ID unfurl OTP key is 499 to4268
MOS MALL GOAT ARM AVID COED
```

在 `Enter new secret pass phrase:` 或 `Enter secret password:` 提示之后， 输入一个密码或口令字。  
 留意， 这并不是用于登录的口令， 它用于生成一次性的登录密码。"ID" 一行给出了所需的参数： 的登录名，  
 迭代次数， 以及种子。 登录系统， 它能记住这些参数并呈现， 因此无需输入它。 最后一行给出了与  
 的秘密口令的、用于登录的一个一次性口令； 如果立即重新登录， 它将是需要使用的那个口令。

### 15.5.2. 不安全接口初始化

如果需要通过一个不安全的接口来初始化， 首先在安全接口上运行一次 `opiekey`；  
 可能希望在可信的机器的 `shell` 提示符下完成。 此外需要指定一个迭代次数（100 也是一个好的）  
 也可以自己的种子， 或计算机随机生成一个。 在不安全的接口上（当然是到希望初始化的机器上），  
 使用 `opiepasswd` 命令：

```
% opiepasswd

Updating unfurl:
You need the response from an OTP generator.
Old secret pass phrase:
    otp-md5 498 to4268 ext
    Response: GAME GAG WELT OUT DOWN CHAT
New secret pass phrase:
    otp-md5 499 to4269
    Response: LINE PAP MILK NELL BUOY TROY

ID mark OTP key is 499 gr4269
LINE PAP MILK NELL BUOY TROY
```

了接受默认的种子， 按下 `Return`（回车）。 在输入口令之前， 到一个有安全接口的机器上， 并让它同  
 的参数：

```
% opiekey 498 to4268
Using the MD5 algorithm to compute response.
Reminder: Don't use opiekey from telnet or dial-in sessions.
Enter secret pass phrase:
GAME GAG WELT OUT DOWN CHAT
```

在回到不安全的连接，并将生成的一次性口令粘贴到相关的应用程序中。

### 15.5.3. 生成一个一次性密码

一旦初始化 OPIE，当登录时将看到类似的提示：

```
% telnet example.com
Trying 10.0.0.1...
Connected to example.com
Escape character is '^J'.

FreeBSD/i386 (example.com) (ttyps)

login: <username>
otp-md5 498 gr4269 ext
Password:
```

另外，OPIE 提示有一个很有用的特性（这里没有表示出来）：如果在口令提示按下 **Return**（回车）系统将返回你输入的口令，你可以藉此看到自己所输入的内容。如果你手工输入一个一次性密码，它会非常有用。

此操作需要生成一个一次性密码来回答提示。这项工作必须在一个可信的系统上运行 **opiekey** 来完成。（也可以到 DOS、Windows® 以及 Mac OS® 等操作系统的版本）。该程序需要将迭代次数和种子提供给它。你可以从登录提示那里复制和粘贴它。

在可信的系统上：

```
% opiekey 498 to4268
Using the MD5 algorithm to compute response.
Reminder: Don't use opiekey from telnet or dial-in sessions.
Enter secret pass phrase:
GAME GAG WELT OUT DOWN CHAT
```

现在就可以用你得到的一次性口令登录了。

### 15.5.4. 生成多个一次性口令

有时，会需要到不能信任的机器或安全连接的地方。这种情况下，可以使用 **opiekey** 命令来一次生成多个一次性口令。例如：

```
% opiekey -n 5 30 zz99999
Using the MD5 algorithm to compute response.
Reminder: Don't use opiekey from telnet or dial-in sessions.
Enter secret pass phrase: <secret password>
26: JOAN BORE FOSS DES NAY QUIT
27: LATE BIAS SLAY FOLK MUCH TRIG
28: SALT TIN ANTI LOON NEAL USE
29: RIO ODIN GO BYE FURY TIC
30: GREW JIVE SAN GIRD BOIL PHI
```

`-n 5` 按顺序求 5 个口令，`30` 指定了最后一个迭代次数是多少。注意这些口令将按与使用顺序相反的序列来显示。如果它比偏，可以手工写下些结果；一般来把它粘到 `lpr` 就可以了。注意，每一行都显示迭代次数及其的一次性的密码；一些人建用完一个就掉一个。

### 15.5.5. 限制使用 UNIX® 口令

OPIE 可以 UNIX® 口令的使用行基于 IP 的登录限制。它的文件是 `/etc/opieaccess`，这个文件默认情况下就是存在的。参 [opieaccess\(5\)](#) 以了解于这个文件一的情况，以及安全方面需要行的一些考。

下面是一个示的 `opieaccess` 文件：

```
permit 192.168.0.0 255.255.0.0
```

行允许指定 IP 地址的用途（再次地址容易被造）在任何时候使用 UNIX® 口令登录。

如果 `opieaccess` 中没有匹配的，将默认拒任何非 OPIE 登录。

## 15.6. TCP Wrappers

一个熟悉 [inetd\(8\)](#) 都听 TCP Wrappers，但几乎没有人它在网络境中的作用有全面的理解。几乎个人都会安装防火墙来理网络接，然而然防火墙有非常广泛的用途，它却不是万能的，例如它无法理似向接起者送一些文本的任。而 TCP Wrappers 件能完成它以及更多的其他事情。接下来的几段中将多 TCP Wrappers 提供的功能，并且，出了一些配置例。

TCP Wrappers 件展了 `inetd` 受其控制的服务器程序施控制的能力。通使用的方法，它能提供日志支持、返回消息入的接、使得服务器程序只接受内部接，等等。尽管防火墙也能完成其中的某些功能，但它不增加了一外的保，也提供了防火墙无法提供的功能。

然而，由 TCP Wrappers 提供的一些外的安全功能，不被好的防火墙的替代品。TCP Wrappers 合防火墙或其他安全加施一并使用，系多提供一安全防。

由于些配置是于 `inetd` 的展，因此，者首先配置 `inetd`。



尽管由 [inetd\(8\)](#) 行的程序并不是真正的“服务器程序”，但上也把它称服务器程序。下面仍将使用一。

## 15.6.1. 初始配置

在 FreeBSD 中使用 TCP Wrappers 的唯一要求是确保 `inetd` 在从 `rc.conf` 中确实包含了 `-Ww` ；它是默认的。当然，它需要 `/etc/hosts.allow` 行适当的配置，但 [syslogd\(8\)](#) 在配置不当会在系统日志中相关消息。



与其它的 TCP Wrappers 不同，使用 `hosts.deny` 在那里被是不推荐和的做法。所有的配置都放到 `/etc/hosts.allow` 中。

在最的配置中，服务程序的策略是根据 `/etc/hosts.allow` 允许或阻止。FreeBSD 中的默认配置是允许一切到由 `inetd` 所的服务的接口。在基本配置之后将更改的情况。

基本配置的形式通常是 `服务 : 地址 : 动作`。这里 `服务` 是从 `inetd` 的服务程序的名字。而 `地址` 可以是任何有效的主机名、一个 IP 或由方括号 ( [ ] ) 括起来的 IPv6 地址。`动作` 字段可以使 `allow` 或 `deny`，分用于允许和禁止相关的。在配置时需要注意所有的配置都是按照第一个匹配的匹配的，表示配置文件将按照顺序匹配，而一旦匹配，搜索也就停止了。

外也有许多其他，些将在后面介绍。的配置行从上面些描述之中可以很容易得出。例如，允许 POP3 接通 [mail/qpopper](#) 服务，把下面的行添加到 `hosts.allow`：

```
# This line is required for POP3 connections:
qpopper : ALL : allow
```

添加之后，需要重新启动 `inetd`。可以通过使用 [kill\(1\)](#) 命令来完成工作，或使用 `/etc/rc.d/inetd` 的 `restart` parameter 参数。

## 15.6.2. 高级配置

TCP Wrappers 也有一些高级的配置；它能用来如何管理接口施更多的控制。一些时候，返回一个明确到特定的主机或服务接口可能是更好的方法。其他情况下，日志或者发送邮件管理可能更合适。此外，一些服务可能只希望本机提供。些需求都可以通过使用 [通配符](#)，转义字符以及外部命令来。接下来的将介绍些。

### 15.6.2.1. 外部命令

假如由于生了某种状况，而致接口被拒掉，而将其原因送回引起接口的人。如何完成的任务？的工作可以通过使用 `twist` 来。当起了接口，`twist` 将用一个命令或脚本。在 `hosts.allow` 文件中已出了一个例子：

```
# The rest of the daemons are protected.
ALL : ALL \
    : severity auth.info \
    : twist /bin/echo "You are not welcome to use %d from %h."
```

个例子将把消息 "You are not allowed to use `daemon` from `hostname`." 返回先前没有配置允许的服务客。于希望把消息反馈接口起者，然后立即切断的需求来，的配置非常有用。注意所有反信息必须被引号 " 包；一没有例外的。





如果攻击者向服务器程序发送大量的连接请求，它可能一次成功的拒绝服务攻击。

它可能是某些情况使用 `spawn`。类似 `twist`，`spawn` 也暗含拒绝连接，并可以用来执行外部命令或服务。与 `twist` 不同的是，`spawn` 不会向连接发起者送回。考虑下面的配置：

```
# 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
```

它将拒绝来自 `*.example.com` 域的所有连接；同时它将主机名，IP 地址，以及对方所连接的服务器名字到 `/var/log/connections.log` 文件中。

除了前面已介绍的符号，例如 `%a` 之外，还有一些其它的符号。参考 [hosts\\_access\(5\)](#) 手册可以获得完整的列表。

### 15.6.2.2. 通配符

前面的例子都使用了 `ALL`。其它功能将功能扩展到更。例如，`ALL` 可以被用来匹配一个服务器、域，或 IP 地址。一些可用的通配符包括 `PARANOID`，它可以用来匹配任何来自可能被伪造的 IP 地址的主机。换言之，`paranoid` 可以被用来定义来自 IP 与其主机名不符的客户。下面的例子将更多的感性：

```
# Block possibly spoofed requests to sendmail:
sendmail : PARANOID : deny
```

在这个例子中，所有连接 `sendmail` 的 IP 地址与其主机名不符的主机都将被拒绝。



如果服务器和客户机有一方的 DNS 配置不正确，使用 `PARANOID` 可能会严重地削弱服务。在配置之前，管理必须谨慎地考虑。

要了解关于通配符和它的功能，参考 [hosts\\_access\(5\)](#) 手册。

为了使配置能生效，你必须首先把 `hosts.allow` 的第一行配置注释掉。它的起始部分已说明了这一点。

## 15.7. Kerberos5

Kerberos 是一个附加的网络系统，用以通过一台安全服务器提供的服务来认证。包括进程登录、进程控制、在系统安全地控制文件，以及其它高危性的操作，由于其存在而显著地提高了安全型并且更加可控。

Kerberos 可以理解为一个身份代理系统。它也被描述为一个以受信第三方为主的身份系统。Kerberos 只提供一功能 - 在网络上安全地完成用身份。它并不提供授权功能（也就是用能做什么操作）或功能（用做了什么操作）。一旦客户和服务都使用了 Kerberos 来证明各自的身份之后，他们可以加密全部的通信以保证数据的私密性和完整性。

因此，强烈建议将 Kerberos 同其它提供授权和服务的安全手段用。

接下来的说明可以用来指导如何安装 FreeBSD 所附的 Kerberos。不过，仍然需要参考相关的机手册以获得完整的描述。

为了展示 Kerberos 的安装过程，我确定：

- DNS 域 ("zone") 为 example.org。
- Kerberos 域是 EXAMPLE.ORG。



在安装 Kerberos 时使用自己的域名即使只是想在内部网上用一用。这样可以避免 DNS 冲突并保证了同其它 Kerberos 之间的互操作性。

### 15.7.1. 历史

Kerberos 最早由 MIT 作为解决网络安全的一个方案提出。Kerberos 采用了加密，因此客户能够在不安全的网络上向服务器（以及相反地）证明自己的身份。

Kerberos 是网络的名字，同时也是用以表示它的程序的形容词。（例如 Kerberos telnet）。目前最新的版本是 5，在 RFC 1510 中有所描述。

有很多免费的，有些涵盖了多种不同的操作系统。最初研制 Kerberos 的麻省理工学院（MIT）也仍然在维护他的 Kerberos 软件包。在 US 它被作为加密产品使用，因而历史上曾受到 US 出口管制。MIT Kerberos 可以通过 port ([security/krb5](#)) 来安装和使用。Heimdal Kerberos 是第一版，并且明确地在 US 之外的地区，以避免出口管制（因此在许多非商业的 UNIX® 系统中非常常用。Heimdal Kerberos 软件包可以通过 port ([security/heimdal](#)) 安装，最新的 FreeBSD 的最小安装也会包含它。

即使尽可能多的用户从中受益，我们明确以 FreeBSD 附的 Heimdal 软件包为准。

### 15.7.2. 配置 Heimdal KDC

密钥分发中心 (KDC) 是 Kerberos 提供的集中式服务 - 它是 Kerberos tickets 的那台计算机。KDC 在 Kerberos 域中的其它机器看来是 "受信的"，因此必须格外注意其安全性。

需要明确 Kerberos 服务器只需要非常少的计算资源，尽管如此，基于安全理由仍然推荐使用独占的机器来扮演 KDC 的角色。

要开始配置 KDC，首先看看的 /etc/rc.conf 文件包含了作为一个 KDC 所需的设置（可能需要本地完整路径以自己系的情况）：

```
kerberos5_server_enable="YES"
kadmind5_server_enable="YES"
```

接下来需要修改 Kerberos 的配置文件，/etc/krb5.conf：



```
[libdefaults]
    default_realm = EXAMPLE.ORG
[realms]
    EXAMPLE.ORG = {
        kdc = kerberos.example.org
        admin_server = kerberos.example.org
    }
[domain_realm]
    .example.org = EXAMPLE.ORG
```

注意：个 `/etc/krb5.conf` 文件假定个的 KDC 有一个完整的主机名，即 `kerberos.example.org`。如果个的 KDC 主机名与它不同，添加一条 CNAME (名) 到 zone 中去。

于有正地配置个的 BINDDNS 服器的大型网，上述例子可以精：

```
[libdefaults]
    default_realm = EXAMPLE.ORG
```



将下面的内容加入到 `example.org` zone 数据文件中：

```
_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
```



要客机能到 Kerberos 服，就必首先配置完整或最小配置的 `/etc/krb5.conf` 并且正地配置 DNS 服器。

接下来需要建 Kerberos 数据。个数据包括了使用主密加密的所有体的密。并不需要住个密，它会保存在一个文件 (`/var/heimdal/m-key`) 中。要建主密，需要行 `kstash` 并入一个口令。

主密一旦建立，就可以用 `kadmin` 程序的 `-l` 参数 (表示 "local") 来初始化数据了。个 `kadmin` 直接地修改数据文件而不是通 `kadmind` 的网服。解决了在数据建之前接它的生蛋的。入 `kadmin` 提示符之后，用 `init` 命令来建域的初始数据。

最后，仍然在 `kadmin` 中，使用 `add` 命令来建第一个 principal。使用全部的默置，随后可以在任何候使用 `modify` 命令来修改些置。外，也可以用 `?` 命令来了解可用的。

典型的数据建程如下：

```
# kstash
Master key: xxxxxxxx
Verifying password - Master key: xxxxxxxx

# kadmin -l
kadmin> init EXAMPLE.ORG
Realm max ticket life [unlimited]:
kadmin> add tillman
Max ticket life [unlimited]:
Max renewable life [unlimited]:
Attributes []:
Password: xxxxxxxx
Verifying password - Password: xxxxxxxx
```

现在是在 KDC 服务的时候了。运行 `/etc/rc.d/kerberos start` 以及 `/etc/rc.d/kadmind start` 来启动这些服务。尽管此时没有任何正在运行的 Kerberos 服务，但仍然可以通过取并列出刚刚建的那个 principal（用 `k`）的 ticket 来测试 KDC 是否在正常工作，使用 KDC 本身的功能：

```
% kinit tillman
tillman@EXAMPLE.ORG's Password:

% klist
Credentials cache: FILE:/tmp/krb5cc_500
Principal: tillman@EXAMPLE.ORG

Issued                Expires                Principal
Aug 27 15:37:58      Aug 28 01:37:58      krbtgt/EXAMPLE.ORG@EXAMPLE.ORG
```

完成所需的操作之后，可以撤销一个 ticket：

```
% kdestroy
```

### 15.7.3. 用 Kerberos 用 Heimdal 服务

首先我们需要一个 Kerberos 配置文件 `/etc/krb5.conf` 的副本。只需简单地用安全的方式（使用类似 [scp\(1\)](#) 的网络工具，或通过 SSH）复制 KDC 上的版本，并覆盖掉客户机上的文件就可以了。

接下来需要一个 `/etc/krb5.keytab` 文件。它是提供 Kerberos 服务的服务器和工作站的一个主要区别 - 服务器必须有 keytab 文件。这个文件包括了服务器的主机密钥，这使得 KDC 得以通过它的身。此文件必须以安全的方式复制到服务器上，因为如果密钥被公之于众，安全也就毁于一旦。也就是，通过明文的通道，例如 FTP 是非常糟糕的想法。

一般来，我们会希望使用 `kadmin` 程序来把 keytab 复制到服务器上。由于也需要使用 `kadmin` 来在主机建立 principal（KDC 一端的 `krb5.keytab`），因此这并不简单。

注意你必须已经得到了一个 ticket 而且这个 ticket 必须可使用 `kadmind.acl` 中的 `kadmin` 接口。参考 Heimdal info 中的 "Remote administration(进程管理)" 一章 ([info heimdal](#)) 以了解如何配置控制表。如果不希望

用程序的 `kadmin` 操作，可以本地采用安全的方式连接 KDC (通过本机控制台，[ssh\(1\)](#) 或 Kerberos [telnet\(1\)](#)) 并使用 `kadmin -l` 在本地进行管理操作。

安装了 `/etc/krb5.conf` 文件之后，就可以使用 Kerberos 上的 `kadmin` 了。 `add --random-key` 命令可以用于添加主机 principal，而 `ext` 命令允许导出服务器的主机 principal 到它的 keytab 中。例如：

```
# kadmin
kadmin> add --random-key host/myserver.example.org
Max ticket life [unlimited]:
Max renewable life [unlimited]:
Attributes []:
kadmin> ext host/myserver.example.org
kadmin> exit
```

注意 `ext` 命令 (它是 "extract" 的缩写) 默认会把导出的密钥放到 `/etc/krb5.keytab` 中。

如果由于没有在 KDC 上运行 `kadmin` (例如基于安全理由) 因而无法本地地使用 `kadmin` 可以直接在 KDC 上添加主机 principal (`host/myserver.EXAMPLE.ORG`) 随后将其导出到一个文件中 (以免覆盖 KDC 上的 `/etc/krb5.keytab`)，方法是使用下面的命令：

```
# kadmin
kadmin> ext --keytab=/tmp/example.keytab host/myserver.example.org
kadmin> exit
```

随后需要把 keytab 复制到服务器上 (例如使用 `scp` 或 `rsync`)。一定要指定一个不同于默认的 keytab 名字以免覆盖 KDC 上的 keytab。

到在本地服务器上可以同 KDC 通信了 (因为它已配置了 `krb5.conf` 文件)，而且它能够证明自己的身份 (由于配置了 `krb5.keytab` 文件)。现在可以用一些 Kerberos 服务。在这个例子中，我将在 `/etc/inetd.conf` 中添加下面的行来启用 `telnet` 服务，随后用 `/etc/rc.d/inetd restart` 重启 `inetd(8)` 服务来使配置生效：

```
telnet    stream  tcp      nowait  root    /usr/libexec/telnetd  telnetd -a user
```

其中的部分是 `-a` (表示) 类型配置用户 (user)。参考 [telnetd\(8\)](#) 手册以了解详情。

#### 15.7.4. 使用 Heimdal 来用客户端 Kerberos

配置客户端是非常简单的。在正确配置了 Kerberos 的网络中，只需要将位于 `/etc/krb5.conf` 的配置文件运行一下配置就可以了。这可以本地通过安全的方式将文件从 KDC 复制到客户端上来完成。

可以在客户端上运行 `kinit`、`klist`，以及 `kdestroy` 来分别获取、显示并删除由 principal 建立的 ticket 是否能正常运行，如果能，用其它的 Kerberos 应用程序来连接用了 Kerberos 的服务。如果应用程序不能正常工作而取 ticket 正常，通常是服务本身，而非客户端或 KDC 有问题。

在类似 `telnet` 的应用程序，可以考虑使用包装程序 (例如 [tcpdump\(1\)](#)) 来确保的口令没有以明文方式。使用 `telnet` 的 `-x` 参数，它将加密整个数据流 (类似 `ssh`)。

多非核心的 Kerberos 客户端程序也是默认安装的。在 Heimdal 的 "最小" 安装理念下, `telnet` 是唯一一个采用了 Kerberos 的服务。

Heimdal port 提供了一些默认不安装的客户端程序, 例如用了 Kerberos 版本的 `ftp`、`rsh`、`rcp`、`rlogin` 以及一些更不常用的程序。MIT port 也包括了一整套 Kerberos 客户端程序。

### 15.7.5. 用户配置文件: `.k5login` 和 `.k5users`

在某个域中的用户往往都有自己的 Kerberos principal (例如 `tillman@EXAMPLE.ORG`) 并映射到本机用户 (例如本机上名为 `tillman` 的用户)。客户端程序, 如 `telnet` 通常并不需要用名或 principal。

不过, 有时可能会需要给予某些没有匹配 Kerberos principal 的人使用本地用户的权限。例如 `tillman@EXAMPLE.ORG` 可能需要本地的 `webdevelopers` 用户号。其它 principal 可能也会需要某个本地用户号。

用户 home 目录中的 `.k5login` 和 `.k5users` 两个文件可以配合 `.hosts` 和 `.rhosts` 来有效地解决这个问题。例如, 如果 `.k5login` 中有如下内容:

```
tillman@example.org
jdoe@example.org
```

并放到了本地用户 `webdevelopers` 的 home 目录中, 列出的两个 principals 都可以使用那个用户号, 而无须共享口令。

建库在实施之前首先有些命令的帮助。特别地, `ksu` 的用户手册包括了 `.k5users` 的相关内容。

### 15.7.6. Kerberos 提示、技巧和故障排除

- 当使用 Heimdal 或 MITKerberos ports 时, 需要设置 `PATH` 环境变量把 Kerberos 客户端列在系统自己的版本之前。
- 同一域内的所有计算机的设置是否相同? 如果不是的, 自身可能会失败。通过 `NTP` 运行同步描述了如何使用 NTP 来同步。
- MIT 和 Heimdal 能很好地互操作。一个例外是 `kadmin`, 因为这个没有被标准化。
- 如果改了主机名, 可能需要修改的 `host/` principal 并更新 keytab。它一律也用于类似 Apache 的 `www/mod_auth_kerb` 所使用的 `www/` principal 的特殊 keytab。
- 域中的一台主机必须在 DNS (或至少在 `/etc/hosts` 中) 可以解析 (同时包括正向和反向)。CNAME 能正常使用, 但必须有正向的 `A` 和 `PTR` 记录。此输出的信息可能很令人困惑: `Kerberos5 refuses authentication because Read req failed: Key table entry not found`。
- 某些客户端使用的 KDC 的操作系可能没有将 `ksu` 设置 `setuid root` 的权限。这意味着 `ksu` 将不能正常工作, 从安全角度这是一个不好的主意, 但可能令人烦恼。它并不是 KDC 的问题。
- 使用 MITKerberos 时, 如果希望允许一个 principal 有超过默认的十秒有效期的 ticket 必须使用 `kadmin` 中的 `modify_principal` 来修改 principal 本身以及 `krbtgt` 的 `maxlife`(最大有效期)。此后, principal 可以使用 `kinit` 的 `-l` 参数来请求一个有更有效期的 ticket。\*

如果在 KDC 上运行了听包程序，并在工作站上运行 `kinit`，你可能会注意到 TGT 是在 `kinit` 一开始运行的时候就出来的 - 甚至在输入口令之前！由于这个现象的解释是 Kerberos 服务器可以无限制地收到 TGT (Ticket Granting Ticket) 的任何未授权的请求；但是，每一个 TGT 都是使用用户的口令派生出来的密码进行加密的。因此，当用输入口令它并不会发送 KDC，而是直接用于解密 `kinit` 所拿到的 TGT。如果解密过程得到了一个包含合法的密码的有效 ticket，那么说明用户的 Kerberos 凭据有效。这些凭据包含了一个会密码用以在随后建立 Kerberos 服务器的加密通道，并由服务器自己的私钥加密的 ticket-granting ticket。第二个加密用于用来是看不到的，但它使得 Kerberos 服务器能够验证一个 TGT 的真实性。

- 如果需要有效期更长的 ticket (例如一周) 而且使用 OpenSSH 接收保存的 ticket 的机器，那么 `sshd_config` 中的 Kerberos `TicketCleanup` 被置为 `no` 否则在注册时会删除所有的 ticket。
- 切主机的 principals 的 ticket 有效期一定要比用的。如果用的用 principal 的有效期是一周，而所接收的主机的有效期是九个小时，那么存的主机 principal 将先行过期，那么是 ticket 保存无法正常工作。
- 当配置 `krb5.dict` 文件来防止使用特定的口令 (`kadmind` 的机器手册中要介绍了它)，切记只有指定了口令策略的 principals 才会使用它。 `krb5.dict` 文件的格式很简单：一个串占一行。建立一个到 `/usr/shared/dict/words` 的符号链接会很有用。

### 15.7.7. 与 MIT port 的区别

MIT 和 Heimdal 主要的区别在于 `kadmin` 程序使用不同 (尽管等价) 的命令和选项。如果你的 KDC 是 MIT 的，那么影响是不能使用 Heimdal 的 `kadmin` 程序来管理 KDC (或相反)。

完成同样工作的命令可能会有些不同。推荐按照 MITKerberos 的网站 (<http://web.mit.edu/Kerberos/www/>) 上的说明来操作。小心关于路径的选项，MIT port 会默认安装到 `/usr/local/`，因此可能会运行“普通的”系统应用程序而非 MIT，如果你的 `PATH` 环境变量把系统目录放在前面的话。



如果使用 FreeBSD 提供的 `MITsecurity/krb5` port，一定要仔细阅读 port 所安装的 `/usr/local/shared/doc/krb5/README.FreeBSD`，如果你想知道什么通过 `telnetd` 和 `klogin` 登录会出一些什么现象的。最重要地，“incorrect permissions on cache file(缓存文件权限不正)”一行需要使用 `login.krb5` 来进行，才能正确地修改你的凭据的属主。

除此之外，修改 `rc.conf` 并加入下列配置：

```
kerberos5_server="/usr/local/sbin/krb5kdc"
kadmind5_server="/usr/local/sbin/kadmind"
kerberos5_server_enable="YES"
kadmind5_server_enable="YES"
```

这样做的原因是，MIT kerberos 会将可执行文件装到 `/usr/local` 之下。

## 15.7.8. 解除 Kerberos 的限制

### 15.7.8.1. Kerberos 是一个 all-or-nothing 方式

在网络上用的每个服务都必须进行修改以便其能配合 Kerberos 工作 (否则就只能使用其它方法来保护它不受网络攻击的侵害), 如果不是这样, 用过的凭据就有可能被窃取并再次使用。 一个例子是所有运行 shell (例如通过 `rsh` 和 `telnet`) 用了 Kerberos 但没有将使用明文密码的 POP3 服务器 Kerberos 化。

### 15.7.8.2. Kerberos 是公用工作站用的

在多用环境中 Kerberos 的安全性会被削弱。 是因为它把 ticket 保存到 `/tmp` 目录中, 而该目录可以被任何用户读取。 如果有用户与其它人同时共享一台计算机 (也就是 multi-user), 该用户的 ticket 就可能被其它用户窃取 (复制)。

可以通过使用 `-c` 文件名选项的命令行选项, 或者(推荐的)改 `KRB5CCNAME` 环境变量来避免这个问题, 但很少有人这么做。 原则上, 将 ticket 保存到用户的 home 目录并正确地设置权限就能解决这个问题。

### 15.7.8.3. KDC 会成为单点崩溃故障点

根据这个, KDC 必须是安全的, 因为主密钥数据保存在它上面。 决不能在 KDC 上面运行其它服务, 而且必须保证它的物理安全。 由于 Kerberos 使用同一个密钥 (其中的那个 "主" 密钥) 来加密所有的密钥, 而将每个文件保存在 KDC, 因此其安全尤其重要。

不过, 主密钥的泄露并没有想象中的那么可怕。 主密钥只用来加密 Kerberos 数据以及生成随机数生成器的种子。 只要 KDC 是安全的, 即使攻击者拿到了主密钥也做不了什么。

另外, 如果 KDC 不可用 (例如由于拒绝服务攻击或网络故障) 网络服务将由于该服务无法运行而不能使用, 从而导致更大的拒绝服务攻击。 通常部署多个 KDC (一个主服务器, 配合一个或多个从服务器) 并采用仔细配置和冗余的备用方式可以避免这种情况 (PAM 是一个不错的例子)。

### 15.7.8.4. Kerberos 的不足

Kerberos 允许用户、主机和服务之间进行相互认证。 但它并没有提供机制来向用户、主机或服务提供 KDC。 这意味着某些木桶的程序, 例如 `kinit` 有可能用所有的用户名和密码。 尽管如此, 可以用类似 `security/tripwire` 这样的文件系统完整性工具来避免此情况的发生。

## 15.7.9. 相关资源和材料

- [The Kerberos FAQ](#)
- [Designing an Authentication System: a Dialog in Four Scenes](#)
- [RFC 1510, The Kerberos Network Authentication Service \(V5\)](#)
- [MIT Kerberos home page](#)
- [Heimdal Kerberos home page](#)

## 15.8. OpenSSL

许多用户可能并没有注意到 FreeBSD 所附带的 OpenSSL 工具包的功能。 OpenSSL 提供了建立在普通的通用基础上的加密功能; 这些功能被许多网络应用和服务程序所广泛使用。

OpenSSL 的一些常用用法包括加密邮件客户端的身边程序，基于 Web 的交易如信用卡等等。许多 ports 如 [www/apache13-ssl](http://www.apache13-ssl)，以及 [mail/claws-mail](mailto:claws-mail) 等等都提供了 OpenSSL 支持的方法。



大多数情况下 Ports Collection 会使用 [security/openssl](http://security/openssl) 除非明确地将 `WITH_OPENSSL_BASE` make 量置为 "yes"。

FreeBSD 中附带的 OpenSSL 版本能支持安全套接字 v2/v3 (SSLv2/SSLv3) 和安全 v1 (TLSv1) 三网，并可作通用的密码学函数使用。



尽管 OpenSSL 支持 IDEA 算法，但由于美国专利，它在默认情况下是不支持的。如果想使用它，需要相应的授权，如果该授权可以接受，可以在 `make.conf` 中置 `MAKE_IDEA`。

邮件提供是 OpenSSL 最常用的功能之一。它是一种能确保公司或个人有效身份不被伪造的凭据。如果没有被许多“权威机构”，或 CA 中的某一个，会产生一个警告。权威机构通常是一家公司，例如 [VeriSign](http://VeriSign)，它能通过签署来证明个人或公司身份的有效性。这个过程是需要付费的，当然，这不是使用邮件的必要条件；然而，这样做会让那些比偏见的用户感到轻松。

### 15.8.1. 生成

为了生成，需要使用下面的命令：

```
# openssl req -new -nodes -out req.pem -keyout cert.pem
Generating a 1024 bit RSA private key
.....+++++
.....+++++
writing new private key to 'cert.pem'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:PA
Locality Name (eg, city) []:Pittsburgh
Organization Name (eg, company) [Internet Widgits Pty Ltd]:My Company
Organizational Unit Name (eg, section) []:Systems Administrator
Common Name (eg, YOUR name) []:localhost.example.org
Email Address []:trhodes@FreeBSD.org

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:SOME PASSWORD
An optional company name []:Another Name
```

注意，在 "Common Name" 提示后面我输入的是一个域名。该提示要求输入服务器的名字，



这个名字今后将用于完成配置；如果在配置里输入域名以外的内容，那配置也就失去其意义了。配置可以指定一些其他的选项，比如配置的有效期，以及使用的加密算法等等。这些选项的完整列表，可以在 [openssl\(1\)](#) 手册中找到。

在配置行前述命令的目录中将生成两个文件。第一个是 `req.pem`，可以是一家公司的公钥，它将输入的凭据的真实性，并申请行名称，再把配置返回。第二个文件的名称将是 `cert.pem`，它包含了公司的私钥，并被全力保护；如果它落入坏人手中，它可以被用来伪造（或的服务器）。

如果不需要来自 CA 的名称，也可以自建自行名称的配置。首先，需要生成 RSA 密钥：

```
# openssl dsaparam -rand -genkey -out myRSA.key 1024
```

接下来，生成 CA 密钥：

```
# openssl gendsa -des3 -out myca.key myRSA.key
```

然后用这个密钥来建配置：

```
# openssl req -new -x509 -days 365 -key myca.key -out new.crt
```

上述配置将在当前目录中生成两个新文件：一个是权威公司的名称文件，`myca.key`；一个是配置本身，`new.crt`。这些文件放到同一个目录中，一般而言，推放到 `/etc`，并且只允许 `root` 取。建把权限置 0700，可以通过 `chmod` 工具来完成。

### 15.8.2. 使用配置的一个例子

那有了这些文件可以做什么？一个比典型的用法是用来加密 SendmailMTA 的通信接。可以解决用通信本地 MTA 发送邮件使用明文行身的配置。



这个用法可能并不完美，因某些 MUA 会由于没有在本地安装配置而向用配置出警告。参考那些邮件的说明了解于安装配置的信息。

下面的配置添加到本地的 `.mc` 文件

```
dn1 SSL Options
define(`confCACERT_PATH', `/etc/certs')dn1
define(`confCACERT', `/etc/certs/new.crt')dn1
define(`confSERVER_CERT', `/etc/certs/new.crt')dn1
define(`confSERVER_KEY', `/etc/certs/myca.key')dn1
define(`confTLS_SRV_OPTIONS', `V')dn1
```

这里，`/etc/certs/` 是准用来在本地保存配置和密钥的位置。最后，需要重新生成本地的 `.cf` 文件。工作可以通地在目录中行 `makeinstall` 来完成。接下来，可以使用 `makerestart` 来重新 Sendmail 服务程序。



如果一切正常的，在 `/var/log/maillog` 中就不会有提示，Sendmail 也不会出现在进程列表中。

做一个简单的测试，使用 [telnet\(1\)](#) 来连接邮件服务器：

```
# telnet example.com 25
Trying 192.0.34.166...
Connected to example.com.
Escape character is '^['.
220 example.com ESMTP Sendmail 8.12.10/8.12.10; Tue, 31 Aug 2004 03:41:22 -0400 (EDT)
ehlo example.com
250-example.com Hello example.com [192.0.34.166], pleased to meet you
250-ENHANCEDSTATUSCODES
250-PIPELINING
250-8BITMIME
250-SIZE
250-DSN
250-ETRN
250-AUTH LOGIN PLAIN
250-STARTTLS
250-DELIVERBY
250 HELP
quit
221 2.0.0 example.com closing connection
Connection closed by foreign host.
```

如果输出中出现了 "STARTTLS" 表明一切正常。

## 15.9. IPsec 上的 VPN

使用 FreeBSD 网桥在两个被 Internet 分隔的网桥之间架设 VPN。

### 15.9.1. 理解 IPsec

本文将指导你完成架设 IPsec。除了配置 IPsec，你应当熟悉如何在一个定制的内核的一些概念（参见 [配置 FreeBSD 的内核](#)）。

IPsec 是一个建立在 Internet 协议 (IP) 之上的协议。它能连接两个或更多主机以安全的方式来通信（并因此而得名）。FreeBSD IPsec "网桥" 基于 [KAME](#) 的协议，它支持协议族，IPv4 和 IPv6。

IPsec 包括了两个子协议：

- *Encapsulated Security Payload (ESP)*, 保护 IP 包数据不被第三方介入，通常使用对称加密算法（例如 Blowfish、3DES）。
- *Authentication Header (AH)*, 保护 IP 包不被第三方介入和篡改，通常计算校验和以及 IP 包的字段进行安全散列来验证。随后是一个包含了散列值的附加包，以便能验证包。

ESP 和 AH 可以根据环境的不同，分别或者一同使用。

IPsec 既可以用来直接加密主机之间的网络通信（也就是 隧道模式）；也可以用来在两个子网之间建造 "虚拟隧道"

用于两个网络之间的安全通信（也就是隧道模式）。后者更多的被称为是虚拟专用网（VPN）。[ipsec\(4\)](#) 手册提供了关于 FreeBSD 中 IPsec 子系统的信息。

要把 IPsec 支持放入内核，需要在配置文件中加入下面的内容：

```
options      IPSEC          IP security
device      crypto
```

如果需要 IPsec 的调试支持，添加：

```
options      IPSEC_DEBUG  debug for IP security
```

### 15.9.2. 简介

由于如何建立 VPN 并不存在标准，因此 VPN 可以采用多种不同的技术来实现，每种技术都有其优点和弱点。这篇文章将展示一个具体的应用情景，并说明它符合的 VPN。

### 15.9.3. 情景：两个网络，一个家庭的网络和一个公司的网络。都接入了 Internet，并且通过一条 VPN 就像在同一个网络一样。

现有条件如下：

- 至少有三个不同的站点
- 每个站点都使用内部的 IP
- 每个站点都通过运行 FreeBSD 的网络接入 Internet。
- 每个网络上的网络至少有一个公网的 IP 地址。
- 网络的内部地址可以是公网或私有的 IP 地址，它并不是公网。它并不冲突，比如它不同地使用 **192.168.1.x** 的地址。

### 15.9.4. 在 FreeBSD 上配置 IPsec

开始需先从 Ports Collection 安装 [security/ipsec-tools](#)。这个第三方软件提供了一些能帮助配置的应用程序。

下一步是建立一个 [gif\(4\)](#) 设备用来在两个网络间传输数据包，"隧道"。使用 **root** 身份运行以下命令，并用真实的内部外部网络替换命令中的 *internal* 和 *external*：

```
# ifconfig gif0 create
```

```
# ifconfig gif0 internal1 internal2
```

```
# ifconfig gif0 tunnel external1 external2
```

比如，公司 LAN 外的 IP 地址是 172.16.5.4，内部的 IP 地址 10.246.38.1。家庭 LAN 外的 IP 地址是 192.168.1.12，内部的 IP 地址 10.0.0.5。

看起来可能有些混乱，所以我通过 `ifconfig(8)` 命令再回顾一下：

Gateway 1:

```
gif0: flags=8051 mtu 1280
tunnel inet 172.16.5.4 --> 192.168.1.12
inet6 fe80::2e0:81ff:fe02:5881%gif0 prefixlen 64 scopeid 0x6
inet 10.246.38.1 --> 10.0.0.5 netmask 0xffffffff00
```

Gateway 2:

```
gif0: flags=8051 mtu 1280
tunnel inet 192.168.1.12 --> 172.16.5.4
inet 10.0.0.5 --> 10.246.38.1 netmask 0xffffffff00
inet6 fe80::250:bfff:fe3a:c1f%gif0 prefixlen 64 scopeid 0x4
```

一旦完成以后，两个私有的 IP 地址都能像下面 `ping(8)` 命令出那样互相。

```
priv-net# ping 10.0.0.5
PING 10.0.0.5 (10.0.0.5): 56 data bytes
64 bytes from 10.0.0.5: icmp_seq=0 ttl=64 time=42.786 ms
64 bytes from 10.0.0.5: icmp_seq=1 ttl=64 time=19.255 ms
64 bytes from 10.0.0.5: icmp_seq=2 ttl=64 time=20.440 ms
64 bytes from 10.0.0.5: icmp_seq=3 ttl=64 time=21.036 ms
--- 10.0.0.5 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 19.255/25.879/42.786/9.782 ms

corp-net# ping 10.246.38.1
PING 10.246.38.1 (10.246.38.1): 56 data bytes
64 bytes from 10.246.38.1: icmp_seq=0 ttl=64 time=28.106 ms
64 bytes from 10.246.38.1: icmp_seq=1 ttl=64 time=42.917 ms
64 bytes from 10.246.38.1: icmp_seq=2 ttl=64 time=127.525 ms
64 bytes from 10.246.38.1: icmp_seq=3 ttl=64 time=119.896 ms
64 bytes from 10.246.38.1: icmp_seq=4 ttl=64 time=154.524 ms
--- 10.246.38.1 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 28.106/94.594/154.524/49.814 ms
```

正如期的那，它们都有从私有地址发送和接受 ICMP 数据包的能力。下面，两个网都必须配置路由以正的网络流量。下面的命令可以设置：

```
# corp-net# route add 10.0.0.0 10.0.0.5 255.255.255.0
```

```
# corp-net# route add net 10.0.0.0: gateway 10.0.0.5
```

```
# priv-net# route add 10.246.38.0 10.246.38.1 255.255.255.0
```

```
# priv-net# route add host 10.246.38.0: gateway 10.246.38.1
```

此刻，不仅从网段是网段后的机器都能访问内部的网段。很容易通过以下的例子：

```
corp-net# ping 10.0.0.8
PING 10.0.0.8 (10.0.0.8): 56 data bytes
64 bytes from 10.0.0.8: icmp_seq=0 ttl=63 time=92.391 ms
64 bytes from 10.0.0.8: icmp_seq=1 ttl=63 time=21.870 ms
64 bytes from 10.0.0.8: icmp_seq=2 ttl=63 time=198.022 ms
64 bytes from 10.0.0.8: icmp_seq=3 ttl=63 time=22.241 ms
64 bytes from 10.0.0.8: icmp_seq=4 ttl=63 time=174.705 ms
--- 10.0.0.8 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 21.870/101.846/198.022/74.001 ms

priv-net# ping 10.246.38.107
PING 10.246.38.1 (10.246.38.107): 56 data bytes
64 bytes from 10.246.38.107: icmp_seq=0 ttl=64 time=53.491 ms
64 bytes from 10.246.38.107: icmp_seq=1 ttl=64 time=23.395 ms
64 bytes from 10.246.38.107: icmp_seq=2 ttl=64 time=23.865 ms
64 bytes from 10.246.38.107: icmp_seq=3 ttl=64 time=21.145 ms
64 bytes from 10.246.38.107: icmp_seq=4 ttl=64 time=36.708 ms
--- 10.246.38.107 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 21.145/31.721/53.491/12.179 ms
```

配置“隧道”是比较容易的部分。配置一条安全接口是个更加深入的工程。下面的配置是使用 pre-shared (PSK) RSA 密钥。除了 IP 地址外，文件的 /usr/local/etc/racoon/racoon.conf 也几乎相同。

```
path    pre_shared_key  "/usr/local/etc/racoon/psk.txt"; #location of pre-shared key
file
log      debug; #log verbosity setting: set to 'notify' when testing and debugging is
complete

padding # options are not to be changed
{
    maximum_length  20;
    randomize        off;
    strict_check     off;
    exclusive_tail   off;
}
```

```

timer    # timing options. change as needed
{
    counter        5;
    interval        20 sec;
    persend         1;
#    natt_keepalive 15 sec;
    phase1         30 sec;
    phase2         15 sec;
}

listen   # address [port] that racoon will listening on
{
    isakmp          172.16.5.4 [500];
    isakmp_natt     172.16.5.4 [4500];
}

remote   192.168.1.12 [500]
{
    exchange_mode   main,aggressive;
    doi             ipsec_doi;
    situation        identity_only;
    my_identifier    address 172.16.5.4;
    peers_identifier address 192.168.1.12;
    lifetime         time 8 hour;
    passive          off;
    proposal_check   obey;
#    nat_traversal   off;
    generate_policy  off;

                proposal {
                    encryption_algorithm  blowfish;
                    hash_algorithm        md5;
                    authentication_method  pre_shared_key;
                    lifetime time         30 sec;
                    dh_group               1;
                }
}

sainfo   (address 10.246.38.0/24 any address 10.0.0.0/24 any)    # address
$network/$netmask $type 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,des;
    authentication_algorithm    hmac_md5,hmac_sha1;
    compression_algorithm  deflate;
}

```

解所有可用的，同些例子里列出的都超越了文的。在 racoon 配置手册中有着丰富的相关信息。

SPD 策略也需要配置一下，FreeBSD 和 racoon 就能加密和解密主机的网流量了。

可以通过在公司的网上行一个似下面 shell 脚本。保存到 /usr/local/etc/racoon/setkey.conf，个文件会被在系初始化的候用到。

```
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;
```

一旦完成后，便使用下面的命令在的网上都 racoon：

```
# /usr/local/sbin/racoon -F -f /usr/local/etc/racoon/racoon.conf -l
/var/log/racoon.log
```

出将会似的：

```
corp-net# /usr/local/sbin/racoon -F -f /usr/local/etc/racoon/racoon.conf
Foreground mode.
2006-01-30 01:35:47: INFO: begin Identity Protection mode.
2006-01-30 01:35:48: INFO: received Vendor ID: KAME/racoon
2006-01-30 01:35:55: INFO: received Vendor ID: KAME/racoon
2006-01-30 01:36:04: INFO: ISAKMP-SA established 172.16.5.4[500]-192.168.1.12[500]
spi=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)
```

一下“隧道”能正常工作，切到外一个控制台用如下的 [tcpdump\(1\)](#) 命令看网流量。根据需要替掉下面的 em0 网界面。

```
# tcpdump -i em0 host 172.16.5.4 and dst 192.168.1.12
```

控制台上能看到如下类似的输出。如果不是这样的话，可能就有些问题了，这样的话就需要用到返回的数据。

```
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)
```

此刻，两个网络就好像是同一个网络的一部分一样。而且这两个网络很可能也有防火墙的保护。要使得这两个网络能互相通信，就需要添加一些路由包的配置。就 [ipfw\(8\)](#) 来说，加入下面的几行配置文件：

```
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
```



编号可能需要根据本机器上的配置做相应的修改。

对于 [pf\(4\)](#) 或者 [ipf\(8\)](#) 的用法，下面的几行配置可行：

```
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，要允许机器初始化的时候开始 VPN 支持，在 `/etc/rc.conf` 中加入以下的几行：

```
ipsec_enable="YES"
ipsec_program="/usr/local/sbin/setkey"
ipsec_file="/usr/local/etc/racoon/setkey.conf" # allows setting up spd policies on
boot
racoon_enable="yes"
```

## 15.10. OpenSSH

OpenSSH 是一个用于安全地远程登录计算机的接口工具。它可以作为 `rlogin`、`rsh`、`rcp` 以及 `telnet` 的直接替代品使用。更一般地，其他任何 TCP/IP 接口都可以通过 SSH 安全地进行隧道/封装。OpenSSH 对所有的通信进行加密，从而有效地阻止了窃听、接口劫持，以及其他网络的攻击。

OpenSSH 由 OpenBSD project 维护，它基于 SSH v1.2.12 并包含了最新的修复和更新。它同时兼容 SSH 的 1 和 2 两个版本。

### 15.10.1. 使用 OpenSSH 的好处

一般来说，在使用 [telnet\(1\)](#) 或 [rlogin\(1\)](#) 时，数据是以未加密的明文的形式发送的。这样一来，在客户端和服务器之间的网络上运行的监听程序，便可以在会话中窃取到用户名/密码和数据。OpenSSH 提供了多种的身份验证和加密方法来防止这种情况的发生。

### 15.10.2. 启用 sshd

sshd 的用途是作为 FreeBSD 安装中 **Standard** 安装过程中的一部分来运行的。要查看 sshd 是否已被启用，查看 rc.conf 文件中的：

```
ssh_enable="YES"
```

表示在下次系统启动时添加 OpenSSH 的服务程序 [sshd\(8\)](#)。此外，也可以手动使用 [rc\(8\)](#) 脚本 /etc/rc.d/sshd 来启动 OpenSSH：

```
/etc/rc.d/sshd start
```

### 15.10.3. SSH 客户端

[ssh\(1\)](#) 的工作方式和 [rlogin\(1\)](#) 非常相似。

```
# ssh user@example.com
Host key not found from the list of known hosts.
Are you sure you want to continue connecting (yes/no)? yes
Host 'example.com' added to the list of known hosts.
user@example.com's password: *****
```

登录过程和使用 [rlogin](#) 或 [telnet](#) 建立的会话非常相似。在连接时，SSH 会利用一个密钥指纹系来验证服务器的真实性。只有在第一次连接时，用户会被要求输入 **yes**。之后的连接将会首先保存下来的密钥指纹。如果保存的指纹与登录接收到的不符，将会发出警告。指纹保存在 ~/.ssh/known\_hosts 中，对于 SSH v2 指纹，则是 ~/.ssh/known\_hosts2。

默认情况下，新版本 OpenSSH 只接受 SSH v2 连接。如果能用版本 2 的客户端程序会自动使用，否则它会返回使用版本 1 的模式。此外，也可以通过命令行参数 **-1** 或 **-2** 来相应地控制使用版本 1 或 2。保持客户端的版本 1 能力是为了考虑旧版本的兼容性。

### 15.10.4. 安全限制

[scp\(1\)](#) 命令和 [rcp\(1\)](#) 的用法相似，它用于将文件复制到远程的机器上，或复制回来，区别是它是安全的。



```
# scp user@example.com:/COPYRIGHT COPYRIGHT
user@example.com's password: *****
COPYRIGHT          100% |*****| 4735
00:00
#
```

由于先前的例子中已保存了指，使用 `scp(1)` 会自地加以。

`scp(1)` 使用的参数同 `cp(1)` 似。第一个参数是一个或一文件，然后是制的目。由于文件是通 SSH 在网上的，因此某些文件的名字需要写成 用名@主机名:<程文件路径>。

### 15.10.5. 配置

OpenSSH 服程序和客端的系配置文件在 `/etc/ssh` 目中。

`ssh_config` 用于配置客端的定，而 `sshd_config` 用于配置服器端。

外 `sshd_program` (默是 `/usr/sbin/sshd`)，以及 `sshd_flags` 个 `rc.conf` 提供了更多的配置。

### 15.10.6. ssh-keygen

用于取代口令的一方法是使用 `ssh-keygen(1)` 来生成 DSA 或 RSA 密用于用的身：

```
% ssh-keygen -t dsa
Generating public/private dsa key pair.
Enter file in which to save the key (/home/user/.ssh/id_dsa):
Created directory '/home/user/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/user/.ssh/id_dsa.
Your public key has been saved in /home/user/.ssh/id_dsa.pub.
The key fingerprint is:
bb:48:db:f2:93:57:80:b6:aa:bc:f5:d5:ba:8f:79:17 user@host.example.com
```

`ssh-keygen(1)` 会生成一个包含公私用于身。私将保存到 `~/.ssh/id_dsa` 或 `~/.ssh/id_rsa`，而公被存放到 `~/.ssh/id_dsa.pub` 或 `~/.ssh/id_rsa.pub`，文件名取决于的 DSA 和 RSA 密型。RSA 或者 DSA 公必被存放到程机器上的 `~/.ssh/authorized_keys` 才能使系正。

将允从程接以基于 SSH 密的来代替口令。

如果在 `ssh-keygen(1)` 中使用了通行字，次使用私都需要入它。`ssh-agent(1)` 能解多次入通行字的力，并将在接下来的 `ssh-agent` 和 `ssh-add` 予以述。



和配置文件可能随 OpenSSH 的版本不同而不同；了避免出，参考 `ssh-keygen(1)` 机手册。

将使到程机器的接基于 SSH 密而不是口令。

如果在运行 `ssh-keygen(1)` 时使用了通行字，下次使用私钥的时候都将被要求输入通行字。`ssh-agent(1)` 能够重新输入通行字的负担，有更详细的探究在 `ssh-agent` 和 `ssh-add` 下一节。



随着系统上的 OpenSSH 版本的不同，各平台和配置文件也会不同；为了避免此问题，需要参考 `ssh-keygen(1)` 的手册。

### 15.10.7. `ssh-agent` 和 `ssh-add`

`ssh-agent(1)` 和 `ssh-add(1)` 是两个工具，提供了一组将 SSH 密钥添加到内存中以便使用，而不必每次都输入通行字的方法。

`ssh-agent(1)` 工具能够使用添加到其中的私钥来管理进程。`ssh-agent(1)` 被用于运行一个应用程序。最基本的用法是，使用它来运行 shell，而高级一些的用法是用它来运行窗口管理器。

要在 shell 中使用 `ssh-agent(1)`，首先把 shell 作为参数来运行它。随后，通过 `ssh-add(1)` 并输入通行字，来向它提供身份信息。一旦这些操作都做完了，用你就能通过 `ssh(1)` 到任何一个安装了公共密钥的机器了。例如：

```
% ssh-agent csh
% ssh-add
Enter passphrase for /home/user/.ssh/id_dsa:
Identity added: /home/user/.ssh/id_dsa (/home/user/.ssh/id_dsa)
%
```

要在 X11 中使用 `ssh-agent(1)`，用 `ssh-agent(1)` 的进程置于 `~/.xinitrc` 之中。将把 `ssh-agent(1)` 服务提供所有在 X11 中运行的程序。下面是一个 `~/.xinitrc` 文件的例子：

```
exec ssh-agent startxfce4
```

将运行 `ssh-agent(1)`，而后者将在下次 X11 时运行 XFCE。做完这些之后就可以重启 X11 以便使修改生效。随后就可以运行 `ssh-add(1)` 来添加全部 SSH 密钥了。

### 15.10.8. SSH 隧道

OpenSSH 能够建立隧道以使用加密的套接字来封装其他数据。

下面的命令告诉 `ssh(1)` 通过 telnet 建立一个隧道：

```
% ssh -2 -N -f -L 5023:localhost:23 user@foo.example.com
%
```

上述 `ssh` 命令使用了下面这些选项：

**-2**

控制 `ssh` 使用第2版的套接字 (如果需要和老的 SSH 一同工作就不要使用这个选项)。

-N

表示不使用命令行，或者只使用隧道。如果省略，ssh 将同初始化会。

-f

制 ssh 在后台行。

-L

表示生一条 本地端口:程主机:程端口 形式的隧道。

user@foo.example.com

程 SSH 服器。

SSH 隧道通听 localhost 上面指定端口来完成工作。它将把本机主机/端口上接收到的接通 SSH 接到程主机/端口。

本例中，位于 localhost 的 5023 端口被用于到 localhost 的接到程主机的 23 端口。由于 23 是 telnet 使用的，因此它将通 SSH 隧道完成 telnet 会。

可以用来封装任意不安全的 TCP 包，例如 SMTP、POP3、FTP 等等。

例 23. 使用 SSH 建 SMTP 安全隧道

```
% ssh -2 -N -f -L 5025:localhost:25 user@mailserver.example.com
user@mailserver.example.com's password: *****
% telnet localhost 5025
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 mailserver.example.com ESMTPE
```

可以与 ssh-keygen(1) 以及外的用号配合来建立一个更透明的 SSH 隧道境。密可以被用在需要入口令的地方，而且可以不同的用配置不同的隧道。

### 15.10.8.1. 用的 SSH 通道例子

#### 15.10.8.1.1. 加 POP3 服的安全

工作，有一个允外来接的 SSH 服器。同一个公网中有一个件服器提供 POP3 服。个网，或从家到公室的网可能不，或不完全可信。基于的原因，需要以安全的方式来看件。解决方法是建一个到公室 SSH 服器的接，并通个接来 POP3 服：

```
% ssh -2 -N -f -L 2110:mail.example.com:110 user@ssh-server.example.com
user@ssh-server.example.com's password: *****
```

当个通道上，可以把 POP3 求到 localhost 端口 2110。个接将通通道安全地到 mail.example.com。

#### 15.10.8.1.2. 防火墙

一些大型包的网管可能会使用一些端的防火墙策略，不开放的连接，而且也阻止出的连接。一些时候可能只能连接程序机器 22 端口，以及 80 端口用来运行 SSH 和网页。

可能希望一些其它的（也与工作无关的）服务，例如提供音频的 Ogg Vorbis 流媒体服务器。如果 Ogg Vorbis server 在 22 或 80 端口以外的端口播放音频，将无法访问它。

解决方法是建立一个到网的防火墙之外的网路上的 SSH 服务器，并通它提供的通道连接到 Ogg Vorbis 服务器上。

```
% ssh -2 -N -f -L 8888:music.example.com:8000 user@unfirewalled-system.example.org
user@unfirewalled-system.example.org's password: *****
```

在可以把客户端程序指定到 localhost 的 8888 端口，它将把请求到 music.example.com 的 8000 端口，从而绕过防火墙。

#### 15.10.9. 允许用户登录 AllowUsers

通常限制哪些用户能登录，以及从何处登录会是好主意。采用 AllowUsers 功能方便地达到这一目的。例如，想要只允许 root 用户从 192.168.1.32 登录，就可以在 /etc/ssh/sshd\_config 文件中加入下述配置：

```
AllowUsers root@192.168.1.32
```

要允许用户 admin 从任何地方登录，只需列出用户名：

```
AllowUsers admin
```

可以在同一行指定多个用户，例如：

```
AllowUsers root@192.168.1.32 admin
```



列出需要登录机器的用户很重要；否则他将被拒在外面。

在完成 /etc/ssh/sshd\_config 的修改之后必须告诉 sshd(8) 重新加载其配置文件，方法是运行：

```
# /etc/rc.d/sshd reload
```

#### 15.10.10. 进一步的资料

OpenSSH

ssh(1) scp(1) ssh-keygen(1) ssh-agent(1) ssh-add(1) ssh\_config(5)

sshd(8) sftp-server(8) sshd\_config(5)

## 15.11. 文件系统ACL控制表

与文件系统在其他方面的加，如快照等一道，FreeBSD 提供了通文件系统控制表 (ACL) 的安全机制。

控制表以高度兼容 (POSIX®.1e) 的方式展了准的 UNIX® 限模型。一特性使得管理能利用其更的安全模型。

如果想 UFS 文件系统用 ACL 支持，需要添加下列：

```
options UFS_ACL
```

并重新内核。如果没有将个内核，在挂载支持 ACL 的文件系统将会收到警告。个在 GENERIC 内核中已包含了。ACL 依赖于在文件系统上用展属性。在新一代的 UNIX® 文件系统，UFS2 中内建了支持。



在 UFS1 上配置展属性需要比 UFS2 更多的管理。而且，在 UFS2 上的展属性的性能也有大的提高。因此，如果想要使用控制表，推荐使用 UFS2 而不是 UFS1。

ACL 可以在挂载通 `acls` 来，它可以加入 `/etc/fstab`。外，也可以通使用 `tunefs(8)` 修改超中的 ACL 来持久性地置自的挂载属性。一般而言，后一方法是推荐的做法，其原因是：

- 挂载的 ACL 无法被重挂载 (`mount(8) -u`) 改，只有完整地 `umount(8)` 并做一次新的 `mount(8)` 才能改它。意味着 ACL 状在系之后就不可能在 `root` 文件系统上生了。外也没有法改正在使用的文件系统的个状。
- 在超中的置将使得文件系统被以用 ACL 的方式挂载，即使在 `fstab` 中的目没有作置，或序生也是如此。避免了不慎将文件系统以没有用 ACL 的状挂载，从而避免没有制 ACL 的安全。



可以修改 ACL 行，以允在没有行一次全新的 `mount(8)` 的情况下用它，但我，不鼓励在未用 ACL 做是有必要的，因如果用了 ACL，然后掉它，然后在没有刷新展属性的情况下重新用它是很容易造成。一般而言，一旦用了文件系统的 ACL 就不再再掉它，因此此的文件系统的保措施可能和用所期待的子不再兼容，而重新用 ACL 将重新把先前的 ACL 附着到文件上，而由于它的限生了，就很可能造成无法期的行。

在查看目，用了 ACL 的文件将在通常的属性后面示 + (加号)。例如：

```
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
```

里我看到了 `directory1`、`directory2`，以及 `directory3` 目使用了 ACL。而 `public_html` 没有。

### 15.11.1. 使用 ACL

文件系统 ACL 可以使用 `getfacl(1)` 工具来看。例如，如果想看 test 的 ACL 设置，所用的命令是：

```
% getfacl test
#file:
#owner:1001
#group:1001
user::rw-
group::r--
other::r--
```

要修改一个文件上的 ACL 设置，需要使用 `setfacl(1)` 工具。例如：

```
% setfacl -k test
```

`-k` 参数将把所有当前定义的 ACL 从文件或文件系统中删除。一般来使用 `-b` 因为它会保持 ACL 正常工作的那些不变。

```
% setfacl -m u:trhodes:rw,group:web:r--,o:---- test
```

在前面的命令中，`-m` 被用来修改默认的 ACL 设置。由于已被先前的命令删除，因此没有预先定义的，于是默认的设置被恢复，并附加上指定的。小心地，如果加入了一个不存在的用户或组，那么将会在 stdout 得到一条 `Invalid argument` 的提示。

## 15.12. 第三方安全

在过去的几年中，安全领域在如何管理漏洞的评估方面取得了足够的进步。几乎每一个操作系统都越来越多地安装和配置了第三方工具，而系统被入侵的威胁也随之增加。

漏洞的评估是安全的一个关键因素，尽管 FreeBSD 会发布基本系统的安全公告，然而没有一个第三方工具都发布安全公告超出了 FreeBSD Project 的能力。在这一前提下，一系列第三方漏洞的威胁，并警告管理存在已知的安全问题的方法也就应运而生。名为 Portaudit 的 FreeBSD 附加工具能辅助达成这一目的。

`ports-mgmt/portaudit` port 会下一个数据，这一数据是由 FreeBSD Security Team 和 ports 人员提供的，其中包含了已知的安全问题。

要开始使用 Portaudit，需要首先从 Ports Collection 安装它：

```
# cd /usr/ports/ports-mgmt/portaudit && make install clean
```

在安装过程中，`periodic(8)` 的配置文件将被修改，以便 Portaudit 能在每天的安全扫描过程中运行。一定要保证到 `root` 号码的每日安全扫描件有人在。除此之外不需要进行更多的配置了。

安装完成之后，管理员可以通过下面的命令来更新数据，并查看目前安装的软件包中所存在的已知安全漏洞：

```
# portaudit -fda
```



由于每天运行 [periodic\(8\)](#) 都会自动更新数据，因此，运行这条命令是可取的。在这里只是作个例子输出。

在任何时候，如果希望通过 Ports Collection 安装的第三方软件工具运行，管理员都可以使用下面的命令：

```
# portaudit -a
```

当存在漏洞的软件包，Portaudit 将生成如下面的输出：

```
Affected package: cups-base-1.1.22.0_1
Type of problem: cups-base -- HPGL buffer overflow vulnerability.
Reference: <http://www.FreeBSD.org/ports/portaudit/40a3bca2-6809-11d9-a9e7-0001020eed82.html>

1 problem(s) in your installed packages found.

You are advised to update or deinstall the affected package(s) immediately.
```

通过上面输出的 URL，管理员能够了解关于那个漏洞的一些信息。这些信息通常包括受到影响的 FreeBSD Port 版本，以及其他可能包含安全公告的网站。

总而言之，Portaudit 是一个强大的工具，并能配合 Portupgrade port 来非常有效地工作。

## 15.13. FreeBSD 安全公告

像其它具有产品品牌的操作系统一样，FreeBSD 会发布“安全公告”。通常公告会只有在相关的发行版本已正确地打补丁之后得到安全邮件列表并在勘误中说明。本文将介绍什么是安全公告，如何理解它，以及系统打补丁的具体步骤。

### 15.13.1. 安全公告看上去是什么样子？

FreeBSD 安全公告的格式如下面的例子，一个例子来自 [FreeBSD 安全公告通知邮件列表](#) 邮件列表。

Topic: denial of service due to some problem①

Category: core②

Module: sys③

Announced: 2003-09-23④

Credits: Person⑤

Affects: All releases of FreeBSD⑥  
FreeBSD 4-STABLE prior to the correction date

Corrected: 2003-09-23 16:42:59 UTC (RELENG\_4, 4.9-PRERELEASE)  
2003-09-23 20:08:42 UTC (RELENG\_5\_1, 5.1-RELEASE-p6)  
2003-09-23 20:07:06 UTC (RELENG\_5\_0, 5.0-RELEASE-p15)  
2003-09-23 16:44:58 UTC (RELENG\_4\_8, 4.8-RELEASE-p8)  
2003-09-23 16:47:34 UTC (RELENG\_4\_7, 4.7-RELEASE-p18)  
2003-09-23 16:49:46 UTC (RELENG\_4\_6, 4.6-RELEASE-p21)  
2003-09-23 16:51:24 UTC (RELENG\_4\_5, 4.5-RELEASE-p33)  
2003-09-23 16:52:45 UTC (RELENG\_4\_4, 4.4-RELEASE-p43)  
2003-09-23 16:54:39 UTC (RELENG\_4\_3, 4.3-RELEASE-p39)⑦

CVE Name: CVE-XXXX-XXXX⑧

For general information regarding FreeBSD Security Advisories, including descriptions of the fields above, security branches, and the following sections, please visit <http://www.FreeBSD.org/security/>.

#### I. Background⑨

#### II. Problem Description⑩

#### III. Impact⑪

#### IV. Workaround⑫

#### V. Solution⑬

#### VI. Correction details⑭

#### VII. References⑮

① **Topic**(主题) 一目了然地说明了到底是什么。它基本上是所受影响的安全问题及其所涉及的工具的描述。

② **Category** (分类) 是指系统中受到影响的组件，它可能是 **core**、**contrib**，或者 **ports** 之一。**core** 分类表示安全弱点影响到了 FreeBSD 操作系统的某个核心组件。**contrib** 分类表示弱点存在于某个捐赠给 FreeBSD Project 的组件，例如 sendmail。最后是 **ports**，它表示弱点影响了 Ports Collection 中的某个第三方组件。

③ **Module**(模块) 指出了组件的具体位置，例如 **sys**。在这个例子中，可以看到 **sys** 模块是存在的；因此，这个漏洞会影响某个在内核中的组件。



- ④ **Announced**(宣布) —反映了与安全公告有关的数据是什么时候公之于众的。说明安全漏洞已存在，而补丁已写入了 FreeBSD 的代码。
- ⑤ **Credits**(作者) —指出了注意到漏洞存在并报告它的个人或团体。
- ⑥ **The Affects**(影响) —指出了 FreeBSD 的哪些版本存在漏洞。对于内核来说，受影响的文件上执行的 `ident` 输出可以协助文件版本。对于 ports，版本号在 `/var/db/pkg` 里面的 `port` 的名字后面列出。如果系统没有与 FreeBSD CVS 代码同步并重建，它很可能是有问题的。
- ⑦ **Corrected**(修正) —指出了发行版本中修正的具体日期、时间和差。在公共漏洞数据库 (Common Vulnerabilities Database) 系统中留的，用于查看漏洞的信息。
- ⑧ **Background**(技术背景) —提供了受影响组件的作用。多数时候一部分会说明什么 FreeBSD 中包含了它，它的作用，以及它的一些原理。
- ⑨ **Problem Description**(问题描述) —深入描述安全漏洞的技术细节。部分有可能会包括有问题的代码相关的情况，甚至是哪个部件如何可能被恶意利用并打漏洞的。
- ⑩ **Impact**(影响) —描述了漏洞能造成的影响类型。例如，可能导致拒绝服务攻击，权限提升，甚至导致得到超用权限。
- ⑪ **Workaround**(应急方案) —指出了系统管理在无法升级系统可以采取的应急策略。有些原因可能包括限制，网络资源的限制，或其它因素。不过无论如何，安全不能被牺牲，有问题的系统要被打补丁，要实施应急方案。
- ⑫ **Solution**(解决方案) —提供了如何有问题的系统打补丁的方法。是逐步的和系统的打补丁并使其安全地工作的方法。
- ⑬ **Correction Details**(修正细节) —展示了 CVS 分支或某个发行版的修正特征。同时也提供了哪个分支上相关文件的版本号。
- ⑭ **References**(文献) —通常会指出其它信息的来源。可能包括 URL，书籍、邮件列表以及新闻。

## 15.14. 进程

进程是一种管理可以使用的跟踪系统资源使用情况的手段，包括它分配了哪些用、提供系统手段，并且可以精确到用行的一个命令。

当然，这种做法是兼有利弊的。它的好处是，入侵者可以迅速把系统小到攻击者入侵的时刻；而做的缺点，是会产生大量的日志，因而需要很多磁盘空间来存它。一旦将进程管理——地配置基本的进程。

### 15.14.1. 用并利用进程

在使用进程之前，必须先用它。要完成这项工作，需要运行下面的命令：

```
# touch /var/account/acct

# accton /var/account/acct

# echo 'accounting_enable="YES"' >> /etc/rc.conf
```

一旦用之后，系统就会开始跟踪 CPU 数据、命令，等等。所有的日志不是以可读的方式写的，要查看它，需要使用 [sa\(8\)](#) 个工具。如果没有给出其他参数，`sa` 将按用，以分单位示他所使用的、共的

CPU 和用□□□，以及平均的 I/O 操作数目，等等。

要□示□于□□□出的命令的相□信息，□□使用 [lastcomm\(1\)](#) 工具。[lastcomm](#) 命令可以用来□示在某一 [tty\(5\)](#) 上的用□信息，例如：

```
# lastcomm ls
trhodes ttyp1
```

将会□示出所有已知的 [trhodes](#) 在 [ttyp1](#) □端上□行 [ls](#) 的情况。

更多的可用□□在□机手册 [lastcomm\(1\)](#)、[acct\(5\)](#) 和 [sa\(8\)](#) 中有所介□。

# Chapter 16. Jails

## 16.1. 概述

这一章将介绍 FreeBSD jail 是什么，以及如何使用它。Jail，有时也被认为是 *chroot* 环境的一种类型替代品，对于管理而言是非常强大的工具，同时，它的一些基本用法，对于高级用户而言也相当有用。

读完本章，你将了解：

- jail 是什么，以及它在安装的 FreeBSD 中所能起到的作用。
- 如何创建、删除和停止 jail。
- 如何从 jail 内部或主机上执行管理的一些基础知识。

其他一些能够提供更多关于 jail 的有用信息的地方有：

- [jail\(8\)](#) 手册。它是关于 [jail](#) - 用于在 FreeBSD 中创建、停止和控制 FreeBSD jails - 工具的完整说明。
- 文件列表及其存储。由 [FreeBSD 文件列表服务器](#) 提供的 [FreeBSD 一般文件列表](#) 和其他文件列表的存储，已经包含了一系列关于 jails 的有价值的信息。通常搜索存储或访问 [freebsd-questions](#) 文件列表能够带来更多有用的信息。

## 16.2. 与 Jail 相关的一些概念

这将帮助你更好地理解与 jail 有关的 FreeBSD 系统知识，以及它如何与 FreeBSD 的其它部分相互作用，理解下列概念：

### [chroot\(8\)](#) (命令)

这个工具使用 FreeBSD 的系统调用 [chroot\(2\)](#) FreeBSD 来改变进程，以及进程的所有衍生进程所能看到的根目录。

### [chroot\(2\)](#) (系统调用)

在 "chroot" 中执行的进程环境。它包括类似文件系统树中的可执行部分、可用的用户及用户 ID、网络接口以及其他 IPC 机制等系统资源。

### [jail\(8\)](#) (命令)

用于在 jail 环境中执行进程的系統管理工具。

### 宿主 (系统调用、进程、用户等等)

能够控制 jail 环境的系统。宿主系统能够访问全部可用的硬件资源，并能控制 jail 境内外的进程。宿主系统与 jail 的一个重要区别是，在宿主系统中的超用户进程，并不像在 jail 中那样受到一系列限制。

### hosted (系统调用、进程、用户等等)

可访问源受 FreeBSD jail 限制的进程、用户或其他实体。

## 16.3. 介绍

由于系统管理是一项困难而又令人不解的任务，因此人们开发了一系列强大的工具，来管理的工作变得更加简单。

这些改动通常是系统能以更简单的方式安装、配置，并毫无代价地持续。其中，系统管理员希望能正确地执行安全方面的配置，使其能用于真正的用途，而阻止安全方面的。

FreeBSD 系统提供的一个用于改善安全的工具就是 *jail*。jail 是在 FreeBSD 4.X 中由 Poul-Henning Kamp <phk@FreeBSD.org> 引入的，它在 FreeBSD 5.X 中又进行了一系列改动，使得它成了一个强大而灵活的系统。目前仍然在持续行持的，以提高其可用性、性能和安全性。

### 16.3.1. Jail 是什么

BSD 的操作系统从 4.2BSD 开始即提供了 *chroot(8)*。*chroot(2)* 工具能改变一个程序的根目录的位置，从而建立一个与系统中其他部分相隔的安全环境：在 *chroot* 环境中的程序，将无法访问其外的文件或其他源。正是由于这个能力，即使攻击者攻破了某一个运行于 *chroot* 环境的服务器，也不能攻破整个系统。*chroot(8)* 对于那些不需要很多灵活性或高功能的应用而言相当好用。另外，在引入 *chroot* 概念的过程中，曾出现过多次跳出 *chroot* 环境的方法，尽管有些在新的 FreeBSD 版本中已修正，但很明确地，*chroot(8)* 并不是一个用于加固服务器安全的理想解决方案。因此，必须有一个新的子系统来解决这些问题。

这就是为什么要 *jail* 最主要的原因。

Jail 以多种方式改进了 *chroot(2)* 的环境概念。在 *chroot(2)* 的环境中，只限制了程序能访问文件系统的一些部分。其他部分的系统源（例如系统库、正在运行的程序，以及网络子系统）是由 *chroot* 程序与宿主系统中的其他程序共享的。jail 展示了一个模型，它不将文件系统的虚拟化，而且将用、FreeBSD 的网络子系统，以及一些其他系统源虚拟化。关于这些精细控制以及调整 jail 环境能力的更具体的介绍，可参看 [微和管理](#)。

jail 具有以下四个特点：

- 目录子 - 进入 jail 的起点。一旦进入了 jail，程序就不再被允许访问子目录以外的对象。这影响到最初 *chroot(2)* 的安全不会影响到 FreeBSD jail。
- 主机名 - 将用于 jail 的主机名。jail 主要用于存放网络服务器，因此在每个 mail 上能注册一个有意义的主机名，能在很大程度上简化系统管理的工作。
- IP 地址 - 一个地址是指定 jail 的，在 jail 的生命周期内都无法改变。通常 jail 的 IP 地址是某一个网络接口上的名称地址，但这并不是必需的。
- 命令 - 准在 jail 中运行的可执行文件的完整路径名。这个命令是相对于 jail 环境的根目录的，随 jail 环境的类型不同，可能会有很多不同之处。

除了这些之外，jail 也可以有自己的用户和自己的 *root* 用户。自然，这里的 *root* 用户的权力会受限于 jail 环境，并且，从宿主系统的观点看来，jail *root* 用户并不是一个无所不能的用户。此外，jail 中的 *root* 用户不能进行除了其 *jail(8)* 环境之外的系统中的一些操作。关于 *root* 用户的能力和限制，在后面的 [微和管理](#) 中将加以介绍。

## 16.4. 建立和控制 jail

一些系统管理员喜欢将 jail 分三类：“完整的” jail，通常包含真正的 FreeBSD 系统，以及“服务器” jail，用于运行一个可能使用特权的用户或服务。这只是一概念上的区分，并不影响如何建立 jail 的程序。在手册 *jail(8)* 中如何建立 jail 行了清晰的描述：

```
# 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 第一就是 `jail` 的一个位置。 个路径是在宿主系中 `jail` 的物理位置。 一常用的 是 `/usr/jail/jailname`, 此 `jailname` 是 `jail` 的主机名。 `/usr/` 文件系 通常会有足 的空 来保存 `jail` 文件系, 于 "完整" 的 `jail` 而言, 它通常包含了 FreeBSD 默 安装的基本系 中 个文件的副本。
- ② 如果 已通 使用 `make world` 或者 `make buildworld` 重新 了 的 `userland`, 可以跳 一 并把 有的 `userland` 安装 新的 `jail`。
- ③ 个命令将在 `jail` 目 中安装所需的可 行文件、函数 以及 机手册等。
- ④ `distribution` 个 `make target` 将安装全部配置文件, 或者 句 , 就是将 `/usr/src/etc/` 制到 `jail` 境中的 `/etc: $D/etc/`。
- ⑤ 在 `jail` 中不是必 要挂接 `devfs(8)` 文件系 。而 一方面, 几乎所有的 用程序都会需要 至少一个 , 主要取决于 用程序的性 和目的。控制 `jail` 中能 的 非常重要, 因 不正 的配置, 很可能允 攻 者在 `jail` 中 行一些 意的操作。通 `devfs(8)` 施的控制, 可以通 由 机手册 `devfs(8)` 和 `devfs.conf(5)` 介 的 集配置来 。

一旦装好了 `jail`, 就可以使用 `jail(8)` 工具来安装它了。 `jail(8)` 工具需要四个必填参数, 些参数在 `Jail 是什` 中 行了介 。除了 四个参数之外, 可以指定一些其他参数, 例如, 以特定用 身 来在 `jail` 中 行程序等等。里, `command` 参数取决于 希望建立的 `jail` 的 型; 于 虚 系 , 可以 `/etc/rc`, 因 它会完成真正的 FreeBSD 系 所需的操作。于 服 `jail`, 行的命令取决于将在 `jail` 中 行的 用程序。

`Jail` 通常 在系 中, 因此, FreeBSD `rc` 机制提供了一些很方便的机制来 化 些工作。

1. 在引导需要 jail 列表写入 `rc.conf(5)` 文件：

```
jail_enable="YES"    # 如果 NO 表示不自启 jail
jail_list="www"      # 以空格分隔的 jail 名字列表
```



在 `jail_list` 中的名字中，可以使用字母和数字，而不使用其他字符。

2. 对于 `jail_list` 中列出的 jail，指定一系列 `rc.conf(5)` 配置，用以描述具体的 jail：

```
jail_www_rootdir="/usr/jail/www"    # jail 的根目录
jail_www_hostname="www.example.org" # jail 的主机名
jail_www_ip="192.168.0.10"          # jail 的 IP 地址
jail_www_devfs_enable="YES"          # 在 jail 中挂接 devfs
jail_www_devfs_ruleset="www_ruleset" # 在 jail 中使用的 devfs 规则集
```

默认情况下，在 `rc.conf(5)` 中配置 jail 会执行其中的 `/etc/rc` 脚本，也就是，默认情况下将 jail 作为虚拟系统的方式来配置。对于服务 jail，另外指定命令，方法是配置 `jailjailnameexec_start` 配置。



如欲了解全部可用的配置，参考手册 `rc.conf(5)`。

`/etc/rc.d/jail` 脚本也可以用于手工启动或停止 `rc.conf` 中配置的 jail：

```
# /etc/rc.d/jail start www
# /etc/rc.d/jail stop www
```

目前，尚没有一种方法来很干净地 `jail(8)`。这是因为通常用于正常系统的命令，目前尚不能在 jail 中使用。目前，管理 jail 最好的方式，是在 jail 外通过 `jexec(8)` 工具，在 jail 中执行下列命令：

```
# sh /etc/rc.shutdown
```

更进一步说明，参考手册 `jail(8)`。

## 16.5. 微管理和

可以管理 jail 配置多种不同的方式，并在 FreeBSD 宿主系统以不同的方式与 jail 交互，以支持更高效率的应用。本文将介绍：

- 一些用于微管理 jail 运行和安全限制的命令。
- 一些可以通过 FreeBSD Ports 套件安装的高级 jail 管理程序，这些程序可以用于一般的基于 jail 的解决方案。

### 16.5.1. FreeBSD 提供的用于微 jail 的系工具

于 jail 的配置微，基本上都是通过置 [sysctl\(8\)](#) 量来完成的。系提供了一个特殊的 sysctl 子，全部相的均在子中；就是 FreeBSD 内核的 `security.jail.*` 子。下面是与 jail 有的主要 sysctl，以及些量的默。些名字都比容易理解，如欲了解一的，参机手册 [jail\(8\)](#) 和 [sysctl\(8\)](#)。

- `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`

系管理可以在 宿主系 中，透置些量的来默 `root` 用加或取消限制。需要注意的是，某些限制是不能取消的。在 [jail\(8\)](#) 中的 `root` 用，无法挂或卸下文件系统，此外在 jail 中的 `root` 用也不能加或卸 [devfs\(8\)](#) 集、配置防火，或行其他需要修改内核数据的管理操作，例如置内核的 `securelevel` 等等。

FreeBSD 的基本系包含一系列用于看目前正在使用的 jail 信息，以及接入 jail 并行管理命令所需的基本工具。[jls\(8\)](#) 和 [jexec\(8\)](#) 命令都是 FreeBSD 基本系的一部分，并可用于行的任：

- 列出在用的 jail 以及的 jail (JID)、IP 地址、主机名和路径。
- 从宿主系中接入正在行的 jail，并在其中行命令，以完成一系列 jail 管理任。在 `root` 希望干地 jail 非常有用。[jexec\(8\)](#) 工具也可以用于在 jail 中 shell 以便其行管理；例如：

```
# jexec 1 tcsh
```

### 16.5.2. 由 FreeBSD Ports 套件提供的高管理工具

在多第三方 jail 管理工具中，[sysutils/jailutils](#) 是最完整和好用的。它是一系列方便 [jail\(8\)](#) 管理的小工具。参其网站以了解一的。

## 16.6. Jail 的用

### 16.6.1. 服 Jail

一主要基于 Simon L. B. Nielsen <[simon@FreeBSD.org](mailto:simon@FreeBSD.org)> 的 <http://simon.nitro.dk/service-jails.html> 中的思路，以及由 Ken Tom [locals@gmail.com](mailto:locals@gmail.com) 更新的文。一中描述了如何配置 FreeBSD 系的 [jail\(8\)](#) 功能其加一个安全次。部分假定行 `RELENG_6_0` 或更新版本，并理解本章之前部分的内容。

### 16.6.1.1. 简介

jail 的一个主要问题是它如何运行和管理。由于每个 jail 都是从根文件系统开始的，对于每个 jail 而言升级也不是个很重的任务，因为升级不会太麻烦，而对于多个 jail 而言，升级不会消耗大量时间，并且是十分乏味的过程。



每个配置过程需要 FreeBSD 有更多的配置和使用时间。如果某些过程花得太久，可以考虑使用其他的系统，例如 [sysutils/ezjail](#)，它提供了更简单的管理 FreeBSD jail 的方法。

基本的想法是，在不同的 jail 中尽可能多地以安全的方式使用共享的资源 - 使用只读的 [mount\\_nullfs\(8\)](#) 挂载，这会升级时间多，从而使每个服务建立不同的 jail 方案变得更加可行。此外，它也增加、删除以及升级 jail 提供了更便捷的方法。



在服务器上的常见例子包括：HTTP 服务、DNS 服务、SMTP 服务等，即如此。

简介的配置的目的包括：

- 建立简单并易于理解的 jail 方案。也就是不必对每个 jail 进行完整的 installworld 操作。
- 使每个 jail 更容易。
- 使更新或升级 jail 更容易。
- 使自行升级的 FreeBSD 分支成为可能。
- 安全的更偏好的追求，尽可能减少被攻陷的可能。
- 尽可能节省空间和 inode。

如前面提到的那样，每个方案很大程度上依赖于将一只读的主模板 (known as nullfs) 挂载到一个 jail 中，并对每个 jail 配置一个可写的分区。分区可以是物理磁盘、分区，或以 vnode 后端的 [md\(4\)](#) 分区。在例子中，我将使用可写的 nullfs 挂载。

下面的表中描述了文件系统格局：

- 每个 jail 挂载到 /home/j 目录下的一个目录。
- /home/j/mroot 是每个 jail 共用的模板，对于所有的 jail 而言都是只读的。
- 在 /home/j 目录中，每个 jail 有一个自己的空目录。
- 每个 jail 中都有一个 /s 目录，该目录将挂载到系统中的可写部分。
- 每个 jail 基于 /home/j/skel 建立其可写空间。
- 每个 jailspace (jail 中的可写部分) 构建到 /home/js。



假定所有的 jail 都放置于 /home 分区中。当然，可以根据需要将每个配置改需要的任何子，但在接下来的例子中，也将相应地加以。

### 16.6.1.2. 建立模板

我们将介绍建立 jail 所需的只读主模板所需的时间。

一般来，我们将系统升级到最新的 FreeBSD -RELEASE 分支，具体做法参看本手册的相应章节。当更新不可行，需要完成 buildworld 过程，此外，还需要 [sysutils/cpdup](#) 软件包。我将使用



[portsnap\(8\)](#) 工具来下载 FreeBSD Ports 套件。在使用手册的 [Portsnap 章](#) 中，提供了对初学者的介绍。

1. 首先， 需要 将存放只读的 FreeBSD 运行文件的文件系统建立一个目录， 接着 输入 FreeBSD 源代码的目录， 并在其中安装 jail 模板：

```
# mkdir /home/j /home/j/mroot
# cd /usr/src
# make installworld DESTDIR=/home/j/mroot
```

2. 接着， 准备 FreeBSD Ports 套件， 以及用于运行 mergemaster 的 FreeBSD 源代码：

```
# cd /home/j/mroot
# mkdir usr/ports
# portsnap -p /home/j/mroot/usr/ports fetch extract
# cpdup /usr/src /home/j/mroot/usr/src
```

3. 构建系统中可写部分的骨架：

```
# 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
```

4. 使用 mergemaster 安装缺失的配置文件。接下来， 删除 mergemaster 构建的多余目录：

```
# mergemaster -t /home/j/skel/var/tmp/temproot -D /home/j/skel -i
# cd /home/j/skel
# rm -R bin boot lib libexec mnt proc rescue sbin sys usr dev
```

5. 现在， 将可写文件系统 接到只读文件系统中。 确保在 s/ 目录中建立了适当的符号链接。如果没有建立目录或建立的位置不正， 可能会导致安装失败。

```
# 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
```

6. 最后，创建一个默认为包含下列配置的 `/home/j/skel/etc/make.conf`：

```
WRKDIRPREFIX?= /s/portbuild
```

配置 `WRKDIRPREFIX` 使得在 jail 中分门别类 FreeBSD 成为可能。注意 `ports` 目录只是系统的一部分。而自己的 `WRKDIRPREFIX` 使得程序得以在 jail 中的可写部分完成。

### 16.6.1.3. 建立 Jail

在我已有了完整的 FreeBSD jail 模板，可以在 `/etc/rc.conf` 中安装并配置它了。例子中演示了建立 3 个 jail：“NS”、“MAIL”和“WWW”。

1. 在 `/etc/fstab` 文件中加入下列配置，以便系统自动挂载 jail 的只读模板和写空：

```
/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/j/ns       /home/j/ns/s    nullfs  rw  0  0
/home/j/mail     /home/j/mail/s  nullfs  rw  0  0
/home/j/www      /home/j/www/s   nullfs  rw  0  0
```



描述批次号 (pass number) 为 0 的分区不会在系统使用 `fsck(8)` 时运行，而存储批次号 (dump number) 为 0 的分区不会在 `dump(8)` 时备份。我不希望 `fsck` 对 `nullfs` 挂载，或对 `dump` 对 jail 中的只读 `nullfs` 挂载。这就是为什么在 `fstab` 条目的最后两列是 "0 0" 的原因。

2. 在 `/etc/rc.conf` 中配置 jail：

```
jail_enable="YES"
jail_set_hostname_allow="NO"
jail_list="ns mail www"
jail_ns_hostname="ns.example.org"
jail_ns_ip="192.168.3.17"
jail_ns_rootdir="/usr/home/j/ns"
jail_ns_devfs_enable="YES"
jail_mail_hostname="mail.example.org"
jail_mail_ip="192.168.3.18"
jail_mail_rootdir="/usr/home/j/mail"
jail_mail_devfs_enable="YES"
jail_www_hostname="www.example.org"
jail_www_ip="62.123.43.14"
jail_www_rootdir="/usr/home/j/www"
jail_www_devfs_enable="YES"
```



把 `jailnamerootdir` 变量置成 `/usr/home` 而不是 `/home` 的原因是 `/home` 目录在默认安装的 FreeBSD 上是指向 `/usr/home` 的一个符号链接。而 `jailnamerootdir` 变量必须是一个不包含符号链接的路径，否则 jail 将拒绝启动。可以使用 `realpath(1)` 工具来决定一个变量是否被给予一个符号链接。更多的信息请参考安全公告 `FreeBSD-SA-07:01.jail`

3. 为每个 jail 创建所需的只读文件系统挂载点：

```
# mkdir /home/j/ns /home/j/mail /home/j/www
```

4. 在 jail 中安装可写的模板。注意需要使用 `sysutils/cpdup`，它能帮助确保每个目录都是正确地复制的：

```
# mkdir /home/js
# cpdup /home/j/skel /home/js/ns
# cpdup /home/j/skel /home/js/mail
# cpdup /home/j/skel /home/js/www
```

5. 现在，就完成了 jail 的制作，可以运行了。首先让 jail 挂接文件系统，然后使用 `/etc/rc.d/jail` 脚本来启动它：

```
# mount -a
# /etc/rc.d/jail start
```

现在 jail 就启动起来了。要验证它是否运行正常，可以使用 `jls(8)` 命令。它的输出类似如下：

```
# 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
```

现在，就可以登入 jail 并添加用户和配置服务了。JID 列出了正在运行的 jail 的 ID 号。可以使用下面的命令来在 JID 号为 3 的 jail 中运行管理任务：

```
# jexec 3 tcsh
```

#### 16.6.1.4. 升级

有时，由于安全原因，或新功能有用，会希望将系统升级到一个新版本的 FreeBSD。安装方式的升级使得升级有 jail 变得很容易。另外，它也能最大限度地减小停机时间，因为 jail 只在最后时刻才需要升级。另外，它也提供了升级的回退到先前版本的方法。

1. 第一是按通常的方法升主机的系。接着，在 /home/j/mroot2 中建立一个新的模板：

```
# mkdir /home/j/mroot2
# cd /usr/src
# make installworld DESTDIR=/home/j/mroot2
# cd /home/j/mroot2
# cpdup /usr/src usr/src
# mkdir s
```

在行 **installworld** 会建一些不需要的目，将它除：

```
# chflags -R 0 var
# rm -R etc var root usr/local tmp
```

2. 重建到主系中的可写符号接：

```
# 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
```

3. 在是候 jail 了：

```
# /etc/rc.d/jail stop
```

4. 卸下原先的文件系：

```
# umount /home/j/ns/s
# umount /home/j/ns
# umount /home/j/mail/s
# umount /home/j/mail
# umount /home/j/www/s
# umount /home/j/www
```



可写的文件系统 (/s) 会在只系之后挂接，因此首先卸。

5. 将先前的只文件系统走，成新的系。做也同保留了先前系的，从而可以在出从中恢。里我根据新系的建来命名。此外我把先前的 FreeBSD Ports 套件直接移到新的文件系中，以省磁盘空间和 inode：

```
# cd /home/j
# mv mroot mroot.20060601
# mv mroot2 mroot
# mv mroot.20060601/usr/ports mroot/usr
```

6. 在新的只模板就可以用了，剩下的事情是重新挂接文件系统并 jails：

```
# mount -a
# /etc/rc.d/jail start
```

最后用 [jls\(8\)](#) 看 jail 是否正常。不要忘记在 jail 中行 mergemaster。配置文件和 rc.d 脚本在升级时行更新。

# Chapter 17. 制控制

## 17.1. 概要

FreeBSD 5.X 在 POSIX®.1e 草案的基础上引入了 TrustedBSD 项目提供的新的安全性扩展。新安全机制中最重要的一个，是文件系统访问控制列表 (ACL) 和制控制 (MAC) 机制。制控制允许添加新的控制模式，并借此实施新的安全策略，其中一部分一个很小的系统子集提供保护并添加特定的服务，其他的则所有的主体和客体提供全面的访问式安全保护。定制中有制的部分源于如下事项，控制的则由管理者和系统作出，而不像自主访问控制 (DAC, FreeBSD 中的默认文件以及 System V IPC 权限) 那是按照用户意图执行的。

本章将集中描述制控制框架 (MAC 框架) 以及一套用以实施多安全策略的组件式的安全策略模式。

本章之后，你将了解：

- 目前 FreeBSD 中具有哪些 MAC 安全策略模式，以及与之相关的机制。
- MAC 安全策略模式将实施何策略，以及访问式与非访问式策略之间的差异。
- 如何高效地配置系统令使其使用 MAC 框架。
- 如何配置 MAC 框架所提供的不同的安全策略模式。
- 如何用 MAC 框架构建更安全的系统，并例明。
- 如何配置 MAC 配置以保护正构建了框架。

本章之前，你：

- 了解 UNIX® 和 FreeBSD 的基础 ([UNIX 基础](#))。
- 熟悉内核配置/内核 ([配置FreeBSD的内核](#)) 的基础。
- 安全及其如何与 FreeBSD 相配合有些了解；([安全](#))。



本章信息的不当使用可能导致系统崩溃，激怒用户，或者无法使用 X11 提供的特性。更重要的是，MAC 不能用于保护一个系统。MAC 框架用于实施有安全策略；如果没有健全的安全条例以及定期的安全检查，系统将永远不会安全。

此外需要注意的是，本章中所包含的例子都是例子。我并不建议在任何一个生产系统上执行某些特定的配置。实施各安全策略模式需要谨慎的考虑与测试，因那些并不完全理解所有机制如何工作的人，可能会需要整个系统中很多的文件或目录进行重新配置。

### 17.1.1. 未涉及的内容

本章涵盖了与 MAC 框架有关的许多方面的安全问题；而新的 MAC 安全策略模式的结果不会涉及。MAC 框架中所包含的一部分安全策略模式，具有一些用于测试及新模式的特定属性，其中包括 [mac\\_test\(4\)](#)、[mac\\_stub\(4\)](#) 以及 [mac\\_none\(4\)](#)。对于某些安全策略模式及其提供的多种机制的更多信息，请参考手册中的内容。



## 17.2. 本章输出的重要概念

在本章之前，有些概念需要解释，希望能藉此理清可能的疑惑，并避免在文中对新概念、新信息行生硬的介绍。

- **区隔(compartment)：**(**注：**区隔即一隔，在一些文献中也称做类别(category)。此外，在其它一些翻文档中，区隔也翻做“象限”。)指一被划分或隔开的程序和数 据，其中，用区隔被明确地给予了特定系统件的权限。同时，区隔也能表示划分，例如工作区、部门、项目，或用户。可以通过使用区隔来实施 need-to-know 安全策略。
- **高水位(high water mark)：**高水位策略是一允许提高安全级别，以期得到更高级别的信息的安全策略。在多数情况下，当进程结束，又会回到原先的安全级别。目前，FreeBSD MAC 框架尚未提供该策略，在图里介绍其定义主要是希望得到一个完整的概念。
- **完整性(integrity)：**作为一个概念，完整性是数据可信性的一程度。若数据的完整性提高，数据的可信性相应提高。
- **标签(label)：**标签是一可用于文件、项目或系统其他客体的安全属性，它也可以被看做是一机密性印。当一个文件被施以标签，其标签会描述一文件的安全参数，并只允许有相似安全性设置的文件、用户、源等访问文件。标签的涵义及解释取决于相应的策略配置：某些策略会将标签当作某一客体的完整性和保密性的表述，而其它一些策略可能会用标签保存数据。
- **程度(level)：**某安全属性加强或削弱的程度。若程度增加，其安全性也相应增加。
- **低水位(low water mark)：**低水位策略允许降低安全级别，以得到安全性较差的信息。多数情况下，在进程结束，又会回到原先的安全级别。目前在 FreeBSD 中唯一的安全策略的是 [mac\\_lomac\(4\)](#)。
- **多重标签(multilabel)：**`multilabel` 属性是一个文件系统属性。该属性可在用户模式下通过 [tunefs\(8\)](#) 程序进行设置。可以在引导时使用的 [fstab\(5\)](#) 文件中，也可在创建新文件系统时进行配置。该属性将允许管理不同客体施以不同的 MAC 属性。该属性用于支持该的安全策略模式。
- **客体(object)：**客体或系统客体是一物体，信息随主体的流向在客体内部流动。客体包括项目、文件、区段、显示器、用户、存储器、磁存储器、打印机及其它数据存储/移动。基本上，客体就是指数据容器或系统源。一客体的流动上意味着数据的流动。
- **策略(policy)：**一套用以确定如何达成目标的规则。策略一般用以描述如何对特定客体进行操作。本章将在安全策略的范畴内讨论策略，一套用以控制数据和信息流并确定其流动者的规则，就是其中一例。
- **敏感性(sensitivity)：**通常在 MLS 中使用。敏感性程度曾被用来描述数据有何等的重要或机密。若敏感性程度增加，保密的重要性或数据的机密性相应增加。
- **单一标签(single label)：**整个文件系统使用一个属性数据流实施控制，叫做单一属性。当文件系统使用此设置，即无论何时当多重属性未被确定，所有文件都将遵守相同属性。
- **主体(subject)：**主体就是引起信息在多个客体中流动的任意活动体，比如用户，用进程(**注：**原文 processor)，系统进程等。在 FreeBSD 中，主体几乎总是代表用活动在某一进程中的一个进程。

## 17.3. 关于 MAC 的说明

在掌握了所有新概念之后，我从整体上来考察 MAC 是如何加系统安全性的。MAC 框架提供的多个安全策略模式可以用来保护网络及文件系统，也可以禁止用某些特定的端口、套接字及其它客体。将策略模式合在一起以建立一个有多次安全性的环境，也是其最佳的使用方式，可以通过一次性加多个安全策略模式来实现。在多次安全环境中，多重策略模式可以有效地控制安全性，这一点与硬化型(hardening)策略，即那通常只化系统中用于特定目的的元素策略是不同的。相比之下，

多重策略的唯一不足是需要系管理先期置好参数，如多重文件系安全志、一位用的网限等等。

与采用框架方式的期效果相比，些不足之是微不足道的。例如，系具有特定配置挑必需的策略的能力，有助于降低性能。而少无用策略的支持，不可以提高系的整体性能，而且提供了更活的空。好的施方案中考到整体的安全性要求，并有效地利用框架所提供的多安全策略模。

一个使用 MAC 特性的系，至少要保不允用任意更改安全属性；所有的用用工具、程序以及脚本，必在所安全策略模提供的的束下工作；并且系管理掌握 MAC 的一切控制。

心正的安全策略模是系管理有的。某些境也需要限制网的控制，在情况下，使用 `mac_portacl(4)`、`mac_ifoff(4)` 乃至 `mac_biba(4)` 安全策略模都会是不的始；在其他情况下，系客体也需要格的机密性，像 `mac_bsdeextended(4)` 和 `mac_mls(4)` 的安全策略模就是此而。

安全策略模的决定可依据网配置行，也只有特定的用才被允使用由 `ssh(1)` 提供的程序以网或互网，`mac_portacl(4)` 安全策略模成情况下的。但文件系又作些什？是由特定的用或群来定某些目的限，抑或是将特定客体保密以限制用或件特定文件？

在文件系的例子中，也客体的限某些用是保密的，但其他不是。比如，一个大的，也会被分成多由几人成的小，A 目中的的人可能不被允 B 目人作的客体，但同他需要由 C 目人作的客体，正符合上述情形。使用由 MAC 框架提供的不同策略，用就可以被分成小，然后被予当区域的，由此，我就不用担心信息泄漏的了。

因此，一安全策略模都有其理系整体安全的独特方法。安全策略模的在安全策略深思熟的基之上行。很多情况下，整体安全策略需要重新修正并在系上施。理解 MAC 框架提供的不同安全策略模会助管理就其面的情形最佳的策略模。

FreeBSD 的默内核并不包含 MAC 框架，因此，在使用本章中的例子或信息之前，添加以下内核：

```
options MAC
```

此外，内核需要重新并且重新安装。



尽管有 MAC 的多机手册中都声明它可以被到内核中，但些策略模的使用仍可能致死系的网及其他功能。使用 MAC 就像使用防火，因此必要小心防止将系完全死。在使用 MAC 时，考是否能回退到之前的配置，在程行配置更加倍小心。

## 17.4. 理解 MAC

MAC 是一安全属性，它可以被用于整个系中的主体和客体。

配置，用必能切理解其所行的操作。客体所具有的属性取决于被加的策略模，不同策略模解其属性的方式也差很大。由于缺乏理解或无法了解其系而致的配置不当，会引起意想不到的，也是不看到的系常。

客体上的安全是由安全策略模决定的安全控制的一部分。在某些策略模中，本身所包含的所有信息足以使其作出决策，而在其它一些安全策略模中，可能被作一个大的体系的一部分行理。

例如，在文件上定 `biba/low`，意味着此隶属 Biba 策略模，其 "low"。

某些在 FreeBSD 中支持特性的策略会提供三个定的，分是 `low`、`high` 及 `equal`。尽管些在不同安全策略模中会控制采取不同措施，但有一点是可以肯定的，那就是 `low` 表示最低限度的定，`equal` 会将主体或客体定被禁用的或不受影的，`high` 会用 Biba 及 MLS 安全策略模中允的最高的定。

在一文件系的境中，同一客体上只会用一个，于是，一套限将被用于整个系，也是很多境所全部需要的。一些用景中，我需要将多重用于文件系的客体或主体，如此一来，就需要使用 `tunefs(8)` 的 `multilabel`。

在使用 Biba 和 MLS 可以配置数，以示分控制中的程度。数的程度可以用来分或将信息按分，从而只允同程度或更高层次的其行。

多数情况下，管理将整个文件系定一。

等一下，看起来很像 *DAC*！但我 `MAC` 只将控制予了管理。此句依然是正的。在某程度上，`root` 是施控制的用，他配置安全策略模以使用被分配到当的/ levels 中。，很多安全策略模同可以限制 `root` 用。于客体的基本控制可能会下放群，但 `root` 用随可以除或更改些定。就是如 Biba 及 MLS 一些安全策略模所包含的 *hierarchal/clearance* 模型。

### 17.4.1. 配置

上，有式安全策略模配置的各都是用基系件。些命令客体和主体配置以及配置的施和提供了一个便的接口。

所有的配置都通 `setfmac(8)` 及 `setpmac(8)` 件施。`setfmac` 命令是用来系客体置 `MAC` 的，而 `setpmac` 是用来系主体置的。例如：

```
# setfmac biba/high test
```

若以上命令不生会直接返回命令提示符，只有当生，些命令才会出提示，和 `chmod(1)` 和 `chown(8)` 命令似。某些情况下，以上命令生的可能是 `Permission denied`，一般在受限客体上置或修改置会生此。系管理可使用以下命令解决此：

```
# setfmac biba/high test
Permission denied
# setpmac biba/low setfmac biba/high test
# getfmac test
test: biba/high
```

如上所示，通 `setpmac` 被用的程予不同的，以覆安全策略模的置。`getpmac` 件通常用于当前行的程，如 `sendmail`：尽管其使用程号来替代命令，其是相同的。如果用其无法的文件行操作，根据所加的安全策略模的，函数 `mac_set_link` 将会出 `Operation not permitted` 的

提示。

#### 17.4.1.1. 一般类型

`mac_biba(4)`、`mac_mls(4)` 及 `mac_lomac(4)` 策略模块提供了制定功能，其值是 `high`、`equal` 及 `low` 之一。以下是这些功能的描述：

- `low` 值被指定为主体或客体所具有的最低次的制定。当主体或客体采用此制定，将阻止其访问 `high` 的客体或主体。
- `equal` 值只能被用于不希望受策略控制的客体上。
- `high` 值当客体或主体采用可能的最高制定。

至于每个策略模块，制定都会产生不同的信息流指令。手册中相关的章节将一一说明一些一般配置的特点。

##### 17.4.1.1.1. 高配置

如下所示，用于 `比较方式:区+区` (`comparison:compartment+compartment`) 的等数：

```
biba/10:2+3+6(5:2+3-20:2+3+4+5+6)
```

其含义：

"Biba 策略"/"等 10"："区 2、3 及 6"：("等 5 ...")

本例中，第一个等将被指定为 "有效区" 的 "有效等"，第二个等是低等，最后一个等是高等。大多数配置中并不使用某些等，如上，它是更高等的配置准则。

当把它用在系统客体上，只有当前的等/区，因为它反映可以施加控制的系统中可用的等，以及网络接口。

等和区，可以用来在一主体和客体之间建立一种称 "支配 (dominance)" 的关系，该关系可能是主体支配客体，客体支配主体，互不支配或互相支配。"互相支配" 的情况会在两个相等时产生。由于 Biba 的信息流特性，可以设置一系列区，"need to know"，可能产生于目之，而客体也由其所属的区。用户可以使用 `su` 和 `setpmac` 来将他自己的限制一划分，以便在没有限制的区里访问客体。

#### 17.4.1.2. 用和配置

用户本身也需要配置，以使其文件和进程能正确地与系统上定义的安全策略交互，这是通过使用登录分在文件 `login.conf` 中配置的。每个使用策略模块都会执行分制定。

以下是一个使用所有策略模块的例子：

```
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/shared/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]:
```

**label** 用␣用␣定用␣分␣默␣， 将由 MAC 行。 用␣不会被允␣更改␣， 因此其从用␣的␣点看不是可␣的。 当然， 在真␣情况的配置中， 管理␣不会希望␣用所有策略模␣。 我␣建␣在␣施以上配置之前␣本章的其余部分。



用␣也␣会在首次登␣后更改其␣， 尽管如此， ␣␣␣是策略的主␣局限性。 上面的例子告␣ Biba 策略， ␣程的最小完整性是␣5， 最大完整性␣15， 默␣且有效的␣␣␣10。 ␣程将以 10的完整性␣行直至其决定更改␣， ␣可能是由于用␣使用了 `setpmac` 命令（␣操作将在登␣被 Biba 限制在一定用␣␣之内）。

在所有情况下， 修改 `login.conf` 之后， 都必␣使用 `cap_mkdb` 重␣登␣分␣ `capability` 数据␣， ␣在接下来的例子和␣中就会有所体␣。

很多站点可能␣有数目可␣的用␣需要不同的用␣分␣， 注意到␣点是大有裨益的。 深入来␣就是需要事先做好␣， 因␣管理起来可能十分困␣。

在 FreeBSD 以后的版本中， 将包含一␣将用␣映射到␣的新方式， 尽管如此， ␣也要到 FreeBSD 5.3 之后的某个␣才能␣。

### 17.4.1.3. 网␣接口和␣␣定

也可以在网␣接口上配置␣， 以控制␣出网␣的数据流。 在所有情况下， 策略都会以␣客体的方式␣作。 例如， 在 **biba** 中␣置␣高的用␣， 就不能␣␣␣␣低的网␣接口。

**maclabel** 可以作␣ `ifconfig` 的参数用于␣置网␣接口的 MAC ␣。 例如：

```
# ifconfig bge0 maclabel biba/equal
```

将在 `bge(4)` 接口上设置 `biba/equal` 的 MAC 标。当使用类似 `biba/high(low-high)` 的标时，整个标使用引号括起来；否则将生成标。

一个支持标的策略模式都提供了用于在网口接口上禁用标 MAC 标的系统控制量。将标置为 `equal` 的效果与此类似。参看 `sysctl` 的输出、策略模式的设备手册，或本章接下来的内容，以了解更详细的标情。

### 17.4.2. 用标——标是多重标？

默认情况下，系统采用的是 `singlelabel` 标。但标管理意味着什么？标策略之间存在很多的不同之处，它在系统安全模型的标活性方面，提供了不同的标。

`singlelabel` 只允许在一个主体或客体上使用一个标，如 `biba/high`。标降低了管理的标，但也同时降低了支持标的策略的标活性。标多管理标可能更希望在安全策略中使用 `multilabel`。

`multilabel` 标允许一个主体或客体有各自独立的 MAC 标，起作用与标准的、只允许整个分区上使用一个的 `singlelabel` 标类似。`multilabel` 和 `single` 标只有标了标功能的那些策略，如 Biba、Lomac、MLS 以及 SEBSD 才有意。

很多情况下是不需要设置 `multilabel` 的。考虑下列情形和安全模型：

- 使用了 MAC 以及标多混合策略的 FreeBSD web-服务器。
- 标台机器上的整个系统中只需要一个标，即 `biba/high`。此标的文件系统并不需要 `multilabel` 标，因为有效的 label 只有一个。
- 因标台机器将作为 Web 服务器使用，因此标以 `biba/low` 标行 Web 服，以杜标向上写。Biba 策略以及它如何标作将在标后予以标，因此，如果标感标前面的标明标以理解的标，标标标下面的内容，再回来标标些内容就会有标清晰的标了。服务器可以使用标置标 `biba/low` 的标独的分区，用于保持其标行标境的标。标个例子中省略了标多内容，例如，如何标数据配置标限制、参数配置和用标的标置；它只是标前述的内容提供一个标的例子。

如果打算使用非标式策略，就不需要 `multilabel` 标了。标些策略包括 `seeotheruids`、`portacl` 和 `partition`。

一个需要注意的事情是，在分区上使用 `multilabel` 并建立基于 `multilabel` 可能会提高系统管理的标，因标文件系统中的所有客体都需要指定标。标包括标目、文件，甚至标点。

接下来的命令将在需要使用多个标的文件系统上设置 `multilabel`。标一操作只能在标用标模式下完成：

```
# tuneefs -l enable /
```

交标区不需要如此配置。



某些用标可能会在根分区上配置 `multilabel` 标志标遇到困。如果标生标的情况，标标本章的 MAC 框架的故障排除。



## 17.5. MAC 安全配置

在实施新技术时，首先进行的工作都是非常好的。在阶段中，管理上一般都进行全面的考察，至少包括下列因素：

- 方案实施的必要条件；
- 方案实施的目的；

就实施 MAC 而言，包括：

- 如何在目标系统上信息和资源进行分割。
- 需要限制哪些信息或资源的访问，以及采用何种限制。
- 需要使用哪些 MAC 模式来完成哪些目的。

尽管重新配置并修改系统资源和安全配置是可行的，但整个系统并修保存的文件和用编号并不是一件而易的事情。这有助于完成无错误且有效的可信系统实施。事先采用 MAC 的可信系统，以及其配置做到十分有益，因为系统实施的成败至为重要。草率散漫地配置 MAC 通常是导致失败的根源。

不同的环境可能会有不同的需求。建立多次而完善的安全配置，可以减少系统正式安装之后所需要的微调。同时，接下来的章节将介绍管理功能使用的各种不同的模式；描述它的使用和配置；除此之外还有一些关于它最合宜的情景的介绍。例如，web 服务器可能希望使用 [mac\\_biba\(4\)](#) 和 [mac\\_bsdextended\(4\)](#) 策略，而其他情况下，例如一台机器上只有少量的本地用户，[mac\\_partition\(4\)](#) 是不错的。

## 17.6. 模式配置

在 MAC 框架中的各个模式，都可以像前述那样加入内核，或作运行内核模式加载。推荐的用法，是通过在 `/boot/loader.conf` 加入适当的配置，以便在系统启动的初始化操作过程中加载某些模式。

接下来的一些小节，将介绍许多 MAC 模式，并介绍它的功能。此外，一章将介绍一些具体环境中的用例。某些模式支持一种称做 (labeling) 的用法，它可以通过使用类似 "允许做这个而不允许做那个" 的语句来控制。这些配置文件可以控制允许的文件访问方式、网络访问，以及许多其他限制。在前一节中，我已展示了文件系统如何通 `multilabel` 标志来应用基于文件或分区的访问控制的方法。

这些配置在整个系统中只限制一个访问的限制，这也是 `tunefs` 为什么是 `multilabel` 的原因。

## 17.7. MAC seeotheruids 模式

模式名：`mac_seeotheruids.ko`

模式的内核配置：`options MAC_SEEOTHERUIDS`

引导选项：`mac_seeotheruids_load="YES"`

`mac_seeotheruids(4)` 模式模式并展示了 `security.bsd.see_other_uids` 和 `security.bsd.see_other_gids` `sysctl` 变量。该模式并不需要预先配置，它能透明地与其他模式同工作。

加载模式之后，下列 `sysctl` 变量可以用来控制其功能：

- `security.mac.seetheruids.enabled` 将用模的功能， 并使用默的配置。 些默置将阻止用看到其他用的程和 socket。
- `security.mac.seetheruids.specificgid_enabled` 将允特定的从一策略中和面。 要将某些排除在一策略之外， 可以用 `security.mac.seetheruids.specificgid=XXX sysctl` 量。 前述例子中， XXX 替希望不受限的 ID 的数形式。
- `security.mac.seetheruids.primarygroup_enabled` 可以用来将特定的主要排除在策略之外。 使用一量， 不能同置 `security.mac.seetheruids.specificgid_enabled`。

## 17.8. MAC `bsdextended` 模

模名： `mac_bsdextended.ko`

的内核配置： `options MAC_BSDEXTENDED`

引： `mac_bsdextended_load="YES"`

`mac_bsdextended(4)` 模能制文件系防火策略。 一模的策略提供了准文件系限模型的一展， 使得管理能建立一似防火的集， 以文件系次中的保文件、 用程序， 以及目。 在文件系客体， 会遍表， 直至到匹配的， 或到表尾。 一行可以通过修改 `sysctl(8)` 参数， `security.mac.bsdextended.firstmatch_enabled` 来行置。 与 FreeBSD 中的其他防火置似， 也可以建一个文件来配置控制策略， 并通过 `rc.conf(5)` 量的配置在系引加它。

表可以通过工具 `ugidfw(8)` 工具来入， 其法似 `ipfw(8)`。 此外可以通过使用 `libugidfw(3)` 来其他的工具。

当使用一模模其小心； 不正的使用将致文件系的某些部分无法。

### 17.8.1. 例子

在加了 `mac_bsdextended(4)` 模之后， 下列命令可以用来列出当前的配置：

```
# ugidfw list
0 slots, 0 rules
```

如希望的那， 目前没有定任何。 意味着一切都。 要建一个阻止所有用， 而保持 `root` 不受影的， 只需行下面的命令：

```
# ugidfw add subject not uid root new object not uid root mode n
```

本身可能是一个很糟的主意， 因它会阻止所有用行怕最的命令， 例如 `ls`。 更富于心的可能是：

```
# ugidfw set 2 subject uid user1 object uid user2 mode n
# ugidfw set 3 subject uid user1 object gid user2 mode n
```



将阻止任何 `user1` 和 `user2` 的主目录的全部，包括目录列表。

`user1` 可以用 `not uid user2` 代替。将同等的限制控制施在所有用户，而不是一个用户上。



`root` 用户不会受到某些的影响。

我们已给出了 [mac\\_bsdextended\(4\)](#) 模块如何帮助添加文件系统的大致介绍。要了解更多信息，请参考 [mac\\_bsdextended\(4\)](#) 和 [ugidfw\(8\)](#) 的手册。

## 17.9. MAC ifoff 模块

模块名：`mac_ifoff.ko`

模块的内核配置：`options MAC_IFOFF`

引导选项：`mac_ifoff_load="YES"`

[mac\\_ifoff\(4\)](#) 模块完全是为了立即禁止网络接口，以及阻止在系统初始时使用网络接口而设计的。它不需要再系统中配置任何，也不依赖于其他 MAC 模块。

大多数特性都可以通过调整下面的 `sysctl` 来加以控制。

- `security.mac.ifoff.lo_enabled` 表示启用/禁用回接口 ([lo\(4\)](#)) 上的全部流量。
- `security.mac.ifoff.bpfrecv_enabled` 表示启用/禁用 伯克利包过滤器 ([bpf\(4\)](#)) 接口上的全部流量。
- `security.mac.ifoff.other_enabled` 将在所有其他接口启用/禁用网络。

最常用的 [mac\\_ifoff\(4\)](#) 用法之一是在不允许引导过程中出现网络流量的环境中配置网络。一个常见的用法是撰写一个使用 [security/aide](#) 的脚本，以便自动地在受保护的目录中创建新的或修改的文件时切断网络。

## 17.10. MAC portacl 模块

模块名：`mac_portacl.ko`

模块的内核配置：`MAC_PORTACL`

引导选项：`mac_portacl_load="YES"`

[mac\\_portacl\(4\)](#) 模块可以用来通过一系列 `sysctl` 变量来限制本地 TCP 和 UDP 端口。本模块上 [mac\\_portacl\(4\)](#) 使得非-`root` 用户能够绑定到它所指定的特定端口，也就是那些编号小于 1024 的端口。

在加载之后，该模块将在所有的 socket 上应用 MAC 策略。可以调整下列一些配置：

- `security.mac.portacl.enabled` 将完全启用/禁用策略。
- `security.mac.portacl.port_high` 将设置 [mac\\_portacl\(4\)](#) 所保护的最高端口号。
- `security.mac.portacl.suser_exempt` 如果设置非零，表示将 `root` 用户排除在策略之外。
- `security.mac.portacl.rules` 将指定模块的 `mac_portacl` 策略；参考下文。

模块的 `mac_portacl` 策略，是在 `security.mac.portacl.rules` `sysctl` 所指定的一个下列形式的字符串：

`rule[,rule,...]` 其中可以出现任意多个规则。一个规则的形式都是：`idtype:id:protocol:port`。这里的 `idtype` 参数可以是 `uid` 或 `gid`，分号表示将 `id` 参数解释为 `uid` 或 `gid`。`protocol` 参数可以用来指定希望用到 TCP 或 UDP 上，方法是把参数置为 `tcp` 或 `udp`。最后的 `port` 参数给出了所指定的用或能绑定的端口号。



由于规则集会直接由内核加以解释，因此只能以数字形式表示用 `uid` ID、`gid` ID，以及端口等参数。换言之，不能使用 `www`、`pop3`，或端口服务的名字来指定它。

默认情况下，在类 UNIX 系统中，编号小于 1024 的端口只能由特等程序使用或绑定，也就是那些以 `root` 身份运行的程序。除了 `mac_portacl(4)` 能允许非特等程序绑定低于 1024 的端口，就必须首先禁用标准的 UNIX 限制。可以通过把 `sysctl(8)` 变量 `net.inet.ip.portrange.reservedlow` 和 `net.inet.ip.portrange.reservedhigh` 置为 0 来实现。

参见下面的例子，或 `mac_portacl(4)` 手册中的说明，以了解更多信息。

### 17.10.1. 例子

下面的例子更好地展示了前面规则的内容：

```
# sysctl security.mac.portacl.port_high=1023
# sysctl net.inet.ip.portrange.reservedlow=0 net.inet.ip.portrange.reservedhigh=0
```

首先我需要置使 `mac_portacl(4)` 管理标准的特等端口，并禁用普通的 UNIX 绑定限制。

```
# sysctl security.mac.portacl.suser_exempt=1
```

用 `root` 用或因此策略而失去特权，因此把 `security.mac.portacl.suser_exempt` 置为一个非零的值。在已成功地配置了 `mac_portacl(4)` 模式，并使其默认与类 UNIX 系统一致行了。

```
# sysctl security.mac.portacl.rules=uid:80:tcp:80
```

允许 UID 为 80 的用（正常情况下，它是 `www` 用）绑定到 80 端口。用 `www` 用就能运行 web 服务器，而不需要使用 `root` 限制了。

```
# sysctl security.mac.portacl.rules=uid:1001:tcp:110,uid:1001:tcp:995
```

允许 UID 为 1001 的用绑定 TCP 端口 110 ("pop3") 和 995 ("pop3s")。用就能接受来到 110 和 995 的接口的服务了。

## 17.11. MAC partition (分区) 模式

模式名：`mac_partition.ko`

内核配置：`options MAC_PARTITION`

引导：`mac_partition_load="YES"`

`mac_partition(4)` 策略将把进程基于其 MAC 放到特定的 "partitions" (分区) 中。它是一个特殊类型的 `jail(8)`，但两者运行比意图不大。

一个模块加到 `loader.conf(5)` 文件中，以便在进程中用些。

大多数策略的配置是通过 `setpmac(8)` 工具来完成的，它将在后面介绍。一个策略可以使用下面的 `sysctl`：

- `security.mac.partition.enabled` 将用制的 MAC 进程 partitions。

当用了个，用将只能看到他自己的，以及其他与他同个 partition 的进程，而不能使用能越 partition 的工具。例如，`insecure` class 中的用，就无法使用 `top` 命令，以及其他需要生新进程的工具。

要置或除 partition 中的工具，需要使用 `setpmac`：

```
# setpmac partition/13 top
```

将把 `top` 命令加入到 `insecure` class 中的用的集。注意，所有由 `insecure` class 中的用生的进程，仍然会留在 `partition/13` 中。

### 17.11.1. 例子

下面的命令将示 partition 以及进程列表：

```
# ps Zax
```

接下来的个命令将允察看其他用的进程 partition，以及那个用正在行的进程：

```
# ps -ZU trhodes
```



除非加了 `mac_seeotheruids(4)` 策略，否用就看不到 `root` 的。

非常手工化的，可能会在 `/etc/rc.conf` 中禁用所有的服，并用脚本来按不同的来它。



下面的几个策略支持基于所出的三的安全定。些，同它的限制，在模的机手册中行了介。

## 17.12. MAC 多 (Multi-Level) 安全模

模名：`mac_mls.ko`

的内核配置：`options MAC_MLS`

引：`mac_mls_load="YES"`

`mac_mls(4)` 策略，通格控制信息流向来控制系中主体和客体的。

在 MLS 环境中，"许可 (clearance)" 会在一个主体或客体上设置，同等的区别。由于某些透明度或敏感度可以有六千多个层次，因此一个主体或客体的配置将是一件任何系统管理都感到头疼的任务。所幸的是，一个策略中已经包含了三个"立即可用的"策略。

这些策略是 `mls/low`、`mls/equal` 以及 `mls/high`。由于这些策略已在手册中进行了介绍，这里只给出重要的说明：

- `mls/low` 策略包含了最低配置，从而允许其他客体支配它。任何具有 `mls/low` 的客体将是最低透明度的，从而不允许更高透明度的信息。此外，该策略也阻止有更高透明度的客体向其写入或读取信息。
- `mls/equal` 策略放到不希望使用另一策略的客体上。
- `mls/high` 策略是允许的最高透明度。指定了该策略的客体将支配系统中的其他客体；但是，它也将不允许向低透明度的客体泄露信息。

MLS 提供了：

- 提供了一些非层次分明的安全模型；
- 固定策略：不允许向上写，不允许向下写（主体可以读取同等或低透明度的客体，但不能读取高透明度的。类似地，主体可以向同等或高透明度写，而不能向下写）；
- 保密（防止不当的数据透露）；
- 系统的基本要点，是在多个敏感策略之间并行地管理数据（而不泄露秘密的和机密的信息）。

下列 `sysctl` 可以用来配置特殊服务和接口：

- `security.mac.mls.enabled` 用来启用/禁用 MLS 策略。
- `security.mac.mls.ptys_equal` 将所有的 `pty(4)` 策略设置为 `mls/equal`。
- `security.mac.mls.revocation_enabled` 可以用来在低 grade 撤销客体。
- `security.mac.mls.max_compartments` 可以用来设置客体的最大区域数；基本上，也就是系统中所允许的最大区域数。

要管理 MLS 策略，可以使用 `setfmac(8)` 命令。要在客体上指定策略，需要使用下面的命令：

```
# setfmac mls/5 test
```

下述命令用于取得文件 `test` 上的 MLS 策略：

```
# getfmac test
```

以上是关于 MLS 策略提供功能的概要。一种做法是在 `/etc` 中建立一个主策略文件，并在其中指定 MLS 策略信息，作为 `setfmac` 命令的输入。这种方法，将在其他策略之后进行介绍。

### 17.12.1. 策略托管敏感性

通过使用多个安全策略模型，管理员可以控制敏感信息的流向。默认情况下，由于其默认的禁止向上以及向下写的特性，系统会默认将所有客体置于最低的状态。因此，所有的客体都可以使用，而管理员可以在配置阶段慢慢地进行提高信息的敏感度策略的修改。

除了前面介绍的三个基本策略之外，管理策略可以根据需要将用和用进行分，以阻止它之间的信息流。一些人比较熟悉的信息限界，如 **机密**、**秘密**，以及 **密** 可以方便理解概念。管理策略也可以根据目的建立不同的分。无论采用何分方法，在实施限制性的策略之前，都必须首先想好如何行。

一个安全策略模型最典型的用例是电子商务的 web 服务器，其上的文件服务器保存公司的重要信息以及金融机的情况。对于只有三个用户的个人工作站而言，可能不甚用。

## 17.13. MAC Biba 模型

模型名：mac\_biba.ko

的内核配置：options MAC\_BIBA

引导：mac\_biba\_load="YES"

mac\_biba(4) 模型将加入 MAC Biba 策略。该策略与 MLS 策略非常类似，只是信息流的方向有些相反的地方。通俗地讲，就是防止敏感信息向下播，而 MLS 策略是防止敏感信息的向上播；因而，几乎所有的内容都可以同时用于该策略。

在 Biba 模型中，"integrity"（完整性）属性，将设置在一个主体或客体上。这些属性是按照层次建立的。如果客体或主体的属性被提升，其完整性也随之提升。

被支持的属性是 **biba/low**，**biba/equal** 以及 **biba/high**；解释如下：

- **biba/low** 属性是客体或主体所能拥有的最低完整性属性。在客体或主体上设置它，将阻止其在更高属性客体或主体上执行的写操作，当然仍被允许。
- **biba/equal** 属性只在那些希望排除在策略之外的客体上设置。
- **biba/high** 允许向低属性的客体上写，但不允许那些客体。推动在那些可能影响整个系统完整性的客体上设置该属性。

Biba 提供了：

- 层次式的完整性属性，并提供了一组非层次式的完整性分；
- 固定属性：不允许向上写，不允许向下（与 MLS 相反）。主体可以在它自己和低属性上写，但不能向更高属性施写操作。类似地，主体也可以在其自己的，或更高属性的客体，但不能读取低属性的客体；
- 完整性（防止数据行不正的修改）；
- 完整性属性（而不是 MLS 的敏感度属性）。

下列 sysctl 可以用于 Biba 策略。

- **security.mac.biba.enabled** 可以用来在机器上启用/禁用是否施 Biba 策略。
- **security.mac.biba.ptys\_equal** 可以用来在 **pty(4)** 上禁用 Biba 策略。
- **security.mac.biba.revocation\_enabled** 将在支配主体生成化制撤客体的属性。

要操作系统客体上的 Biba 策略，需要使用 **setfmac** 和 **getfmac** 命令：

```
# setfmac biba/low test
# getfmac test
test: biba/low
```

### 17.13.1. 托管完整性

与敏感性不同，完整性是要保护不受信方不能对信息行更改。它包括了在主体和客体之间的信息。它能保护只能修改甚至需要他的信息。

**mac\_biba(4)** 安全策略模式允许管理指定用户能看到和运行的文件和程序，并确保些文件能系及用或用所信任，而免受其他威胁。

在最初的阶段，管理必须做好将用户分成不同的等、和区域的准备。在前后，包括数据以及程序和使用工具在内的客体，用都会无法。一旦用了个策略模式，系将默认使用高的，而分用和等的工作交由管理来行配置。与前面介绍的界限不同，好的方法可能包括 topic。例如，只允许人修改代码、使用源代码器，以及其他工具，而其他用户分入其他，如人、人，以及普通用户，些用户可能只有些料的限。

通过其自然的安全控制，完整性低的主体，就会无法向完整性高的主体行写操作；而完整性高的主体，也不能观察或低完整性低的客体。通过将客体的最低，可以阻止所有主体其行的操作。一安全策略模式期的用合包括受限的 web 服务器、和机，以及源代码。而于个人端、作路由器的算机，以及网防火而言，它的用就不大了。

## 17.14. MAC LOMAC 模式

模式名：mac\_lomac.ko

的内核配置：options MAC\_LOMAC

引导：mac\_lomac\_load="YES"

和 MAC Biba 策略不同，**mac\_lomac(4)** 策略只允许在降低了完整性之后，才允许在不破坏完整性的前提下低完整性的客体。

MAC 版本的 Low-watermark 完整性策略不与早的 **lomac(4)** 相混，除了使用浮的支持主体通过助区降之外，其工作方式与 Biba 大体相似。一次要的区以 **[auxgrade]** 的形式出。当指定包含助的 lomac 策略，其形式类似于：**lomac/10[2]** 里数字二 (2) 就是助。

MAC LOMAC 策略依赖于系客体上存在普的，就允许主体从低完整性的客体取，并主体的降，以防止其在之后写高完整性的客体。就是前面的 **[auxgrade]**，因此个策略能提供更大的兼容性，而所需要的初始配置也要比 Biba 少。

### 17.14.1. 例子

与 Biba 和 MLS 策略似；**setfmac** 和 **setpmac** 工具可以用来在系客体上放置：

```
# setfmac /usr/home/trhodes lomac/high[low]
# getfmac /usr/home/trhodes
lomac/high[low]
```

注意，`low` 里的 `low` 是 `low`，`low` 特性只由 MAC LOMAC 策略提供。

## 17.15. MAC Jail 中的 Nagios

下面给出了通多 `MAC` 模式，并正确地配置策略来安全环境的例子。这只是一个例子，因此不能被看作四海一家的解决之道。这只是一个策略，而忽略它不能解决任何问题，并可能在生产环境中产生性的后果。

在开始些操作之前，必在在一个文件系统上置 `multilabel` 选项，些操作在第一章开始的部分进行了介绍。不完成些操作，将导致的结果。首先，已经安装了 `net-mngt/nagios-plugins`、`net-mngt/nagios`，和 `www/apache13` 些 ports，并对其进行了配置，且正常。

### 17.15.1. 建立一个 `insecure` (不安全) 用 `Class`

首先是在 `/etc/login.conf` 文件中加入一个新的用 `class`：

```
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/shared/man /usr/local/man:\
:nologin=/usr/sbin/nologin:\
:cputime=1h30m:\
:datasize=8M:\
:vmemoryuse=100M:\
:stacksize=2M:\
:memorylocked=4M:\
:memoryuse=8M:\
:filesize=8M:\
:coredumpsize=8M:\
:openfiles=24:\
:maxproc=32:\
:priority=0:\
:requirehome:\
:passwordtime=91d:\
:umask=022:\
:ignoretime@:\
:label=biba/10(10-10):
```

并在 `default` 用 `class` 中加入：

```
:label=biba/high:
```

一旦完成上述操作，就需要运行下面的命令来重建数据库：

```
# cap_mkdb /etc/login.conf
```

### 17.15.2. 引导配置

在引导时不要重新引导，我需要在 `/boot/loader.conf` 中添加下面几行，以便模式随系统初始化一同加载：

```
mac_biba_load="YES"
mac_seeotheruids_load="YES"
```

### 17.15.3. 配置用户

使用下面的命令将 `root` 用户属于默认的 class：

```
# pw usermod root -L default
```

所有非 `root` 或系统的用户，都需要一个登录 class。登录 class 是必需的，否则一些用户将被禁止使用类似 `vi(1)` 的命令。下面的 `sh` 脚本能完成这个工作：

```
# for x in `awk -F: '($3 >= 1001) && ($3 != 65534) { print $1 }' \
# /etc/passwd`; do pw usermod $x -L default; done;
```

将 `nagios` 和 `www` 用户加入不安全 class：

```
# pw usermod nagios -L insecure
```

```
# pw usermod www -L insecure
```

### 17.15.4. 创建上下文文件

接下来需要创建一个上下文文件；可以把下面的示例放到 `/etc/policy.contexts` 中。



```
# This is the default BIBA policy for this system.
```

```
# System:
```

```
/var/run          biba/equal
/var/run/*        biba/equal
```

```
/dev              biba/equal
/dev/*            biba/equal
```

```
/var              biba/equal
/var/spool        biba/equal
/var/spool/*      biba/equal
```

```
/var/log          biba/equal
/var/log/*        biba/equal
```

```
/tmp              biba/equal
/tmp/*            biba/equal
/var/tmp          biba/equal
/var/tmp/*        biba/equal
```

```
/var/spool/mqueue biba/equal
/var/spool/clientmqueue biba/equal
```

```
# For Nagios:
```

```
/usr/local/etc/nagios
/usr/local/etc/nagios/*      biba/10
```

```
/var/spool/nagios          biba/10
/var/spool/nagios/*        biba/10
```

```
# For apache
```

```
/usr/local/etc/apache      biba/10
/usr/local/etc/apache/*    biba/10
```

这个策略通过在信息流上设置限制来强化安全。在这个配置中，包括 **root** 和其他用户在内的用户，都不允许访问 Nagios。作为 Nagios 一部分的配置文件和进程，都是完全独立的，也称为 jailed。

接下来可以用下面的命令将其加入系统：

```
# setfsmac -ef /etc/policy.contexts /
# setfsmac -ef /etc/policy.contexts /
```



随环境不同前述的文件系统布局可能会有所不同；不过无论如何，都只能在一个文件系统上运行它。

在 `/etc/mac.conf` 文件中的 `main` 小节需要进行下面的修改：

```
default_labels file ?biba
default_labels ifnet ?biba
default_labels process ?biba
default_labels socket ?biba
```

### 17.15.5. 用网

在 `/boot/loader.conf` 中添加下列内容：

```
security.mac.biba.trust_all_interfaces=1
```

将下述内容加入 `rc.conf` 中的网接口配置。如果主 Internet 配置是通过 DHCP 完成的，需要在下次系统启动之后手工进行类似的配置：

```
maclabel biba/equal
```

### 17.15.6. 配置

首先要 web 服务以及 Nagios 不会随系统的初始化和重启而自动启动。在此之前，在此 `root` 用户不能编辑 Nagios 配置目录中的任何文件。如果 `root` 能在 `/var/spool/nagios` 中进行 `ls(1)`，则表示配置有问题。如果配置正确的，则会收到一条 "permission denied" 信息。

如果一切正常，Nagios、Apache，以及 Sendmail 就可以按照安全策略的方式运行了。下面的命令将完成此工作：

```
# 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
```

再次检查是否一切正常。如果不是的，查看日志文件和相关信息。此外，可以用 `sysctl(8)` 来禁用 `mac_biba(4)` 安全策略模型的强制措施，并象之前那样进行配置和服务。



`root` 用户可以放心大胆地修改安全强制措施，并编辑配置文件。下面的命令可以安全策略进行降级，并设置一个新的 shell：

```
# setpmac biba/10 csh
```

要阻止这种情况发生，就需要配置 `login.conf(5)` 中可执行的命令了。如果 `setpmac(8)` 进行超越可执行的命令，会返回一个错误，而不是执行命令。在这个例子中，可以把 `root` 设置 `biba/high(high-high)`。

## 17.16. User Lock Down

这个例子的是一个相当小的文件系统，其用户数少于五十。用户能够在其上登录，除了存储数据之外，它可以隐藏其他用户的程序的目的。

在这个场景中，`mac_bsextended(4)` 可以与 `mac_seeotheruids(4)` 并存，以达到禁止非授权源，同时隐藏其他用户的程序的目的。

首先，在 `/boot/loader.conf` 中加入：

```
mac_seeotheruids_load="YES"
```

随后，可以通过下述 `rc.conf` 变量来用 `mac_bsextended(4)` 安全策略模式：

```
ugidfw_enable="YES"
```

默认保存在 `/etc/rc.bsextended` 中，并在系统初始化时加载；但是，其中的默认可能需要进行一些改动。因为每台机器只得到了授权的用户提供服务，因此除了最后项之外，其它内容都应保持注释的状态。这会将默认制加载属于用户的系统客体。

在每台机器上添加需要的用户并重新配置。出于这个目的，可以在一个控制台上分别以不同的用户身份登录。运行 `ps aux` 命令来看看是否能看到其他用户的程序。此外，在其他用户的主目录中运行 `ls(1)` 命令，如果配置正确，这个命令会失败。

不要以 `root` 用户的身份运行，除非你已经修改了特定的 `sysctl` 来阻止超用户的操作。



在添加新用户时，他的 `mac_bsextended(4)` 不会自动出现在组表中。要迅速更新组集，只需使用 `kldunload(8)` 和 `kldload(8)` 工具来卸载并重新加载安全策略模式。

## 17.17. MAC 框架的故障排除

在配置过程中，有一些用户报告了正常配置下出现的错误。其中的一些如下所示：

### 17.17.1. 无法在 / 上用 `multilabel`

`multilabel` 标志在根 (/) 分区上没有保持有效！

看起来五十个用户中就有一个遇到这样的问题，当然，在我的初始配置过程中也出现这样的问题。更进一步的观察使得我相信这个问题的 "bug" 是由于文档中不准确的描述，或其产生的误解造成的。无论它是因何引起的，下面的步骤可能解决此问题：

1. 编辑 `/etc/fstab` 并将根分区置为 `ro`，表示只读。
2. 重新挂载并进入只读模式。
3. 在 `/` 上运行 `tunefs -l enable`
4. 重新挂载并进入正常的模式。
5. 运行 `mount -urw/` 并把 `/etc/fstab` 中的 `ro` 改回 `rw`，然后再次重新挂载。
6. 再次挂载来自 `mount` 的输出，已挂载根文件系统上正确地置了 `multilabel`。

### 17.17.2. 在 MAC 之后无法使用 X11 了

在使用 MAC 建立安全的环境之后，就无法使用 X 了！

这可能是由于 MAC `partition` 策略，或者某个 MAC 策略执行了错误的配置导致的。要解决这个问题，步骤：

1. 检查信息；如果用是在 `insecure` class 中，`partition` 策略就可能导致问题。将使用的 class 重新改为 `default` class，并使用 `cap_mkdb` 命令重建数据库。如果无法解决问题，进入第二步。
2. 仔细检查策略。策略有错误的用的策略是正确的，特别是 X11 应用，以及 `/dev`。
3. 如果这些都无法解决问题，将输出消息和错误的环境的描述，发送到 [TrustedBSD](http://TrustedBSD.org) 网站上的 TrustedBSD 邮件列表，或者 [FreeBSD](http://FreeBSD.org) 一般邮件列表。

### 17.17.3. Error: `_secure_path(3)` cannot stat `.login_conf`

当我从 `root` 切换到其同中的其他用户，出现了提示 `_secure_path: unable to state .login_conf`。

这个提示通常在使用有高于它将要成为的那个用户的权限定出。例如，如果系统上的一个用户 `joe` 有默认的 `biba/low` 策略，而 `root` 用户有 `biba/high`，它也就不能查看 `joe` 的主目录，无论 `root` 是否使用了 `su` 来成为 `joe`。这种情况下，Biba 完整性模型，就不会允许 `root` 查看在低完整性策略中的客体。

### 17.17.4. `root` 用户名被破坏了！

在普通模式，甚至是在只读模式中，`root` 不被限制。 `whoami` 命令返回了 `0` (零) 而 `su` 提示 `who are you?`。到底发生了什么？

策略被禁用可能会致问题，无论是通过 `sysctl(8)` 或是卸除了策略模块。如果打算禁用策略，或者禁用它，登录性能数据需要重新配置，在其中删除 `label` 策略。仔细检查 `login.conf` 以确保所有的 `label` 策略都已删除，然后使用 `cap_mkdb` 命令来重建数据库。

这种情况也可能在通过策略来限制 `master.passwd` 文件或那个数据产生。这主要是由于管理修改受某一 `label` 限制的文件，而与系统的通用策略产生了冲突。因此，用户信息将由系统直接获取，而在文件承载了新的 `label` 之后会拒绝访问。此时，只需使用 `sysctl(8)` 禁用该策略，一切就会恢复正常了。

# Chapter 18. 安全事件

## 18.1. 概述

FreeBSD 中包含了对原子粒度安全事件的支持。事件能够支持可变的、原子粒度且可配置的，对于各与安全有关的系统事件，包括登录、配置更改，以及文件和网络等的日志。这些日志对于正在运行的系统上实施监控、入侵检测和事后分析都十分重要。FreeBSD 遵循了 Sun 所颁布的 BSM API 和文件格式，并且与 Sun™ 的 Solaris™ 和 Apple® 的 Mac OS® X 兼容。

本章的重点是安装和配置事件。它介绍了事件策略，并提供了一个配置例子。

读完本章，你将了解：

- 事件是什么，以及它如何工作。
- 如何在 FreeBSD 上启用和配置事件。
- 如何使用摘要和工具来管理运行。

本章之前，需要：

- 理解 UNIX® 和 FreeBSD 的基础知识 ([UNIX 基础](#))。
- 熟悉内核配置和基本方法 ([配置 FreeBSD 的内核](#))。
- 熟悉安全知识以及如何在 FreeBSD 中使用它 ([安全](#))。



审计机制中存在一些已知的限制，例如并不是所有与安全有关的系统事件都可以审计，此外某些登录机制，例如基于 X11 的显示管理器，以及第三方服务的登录机制，都不会在用户的登录会话中正配置。

安全审计机制能够系统性地生成非常大量的信息：在繁忙的系统中，审计数据如果配置不当会非常的大，并在一周内迅速超过几个 GB 的尺寸。管理审计配置中的磁盘空间需求的某些。例如，可能需要 `/var/audit` 目录单独分配一个文件系统，以防止在审计日志所用的文件系统被填满影响其它文件系统。

## 18.2. 本章中的一些概念

在开始本章之前，我需要解释一下与审计有关的一些概念：

- 事件 (event)：可审计事件是指能被审计子系统记录的任何事件。例如来，与安全有关的事件包括创建文件、建立网络连接，以及以某一用户身份登录，等等。任何事件必要是 "有主 (attributable)" 的，即可以最归于某一已通道的用户的身，反之，则称事件是 "无主 (non-attributable)" 的。无主事件可以是发生在登录过程成功之前的任何事件，例如一次无效密码等。
- 类 (class)：事件类是指相关事件的一个命名集合，通常在列表式中使用。常用的事件类包括 "创建文件" (fc)、"执行" (ex) 和 "登录和注销" (lo)。
- 记录 (record)：记录是指描述一个安全事件的日志。记录包括事件类型、执行操作的主体 (用户) 信息、日期和事件信息，以及与之相关的现象或参数信息，最后是操作成功或失败。
- 轨迹 (trail)：轨迹，或日志文件，包含了一系列描述安全事件的记录。典型情况下，

目录基本上是以事件发生的顺序的。只有获得锁的进程，才能向目录中提交。

- 表达式 (*selection expression*)：表达式是包含一系列前缀和事件名字，用以匹配事件的字符串。
- 预选 (*preselection*)：系统通过一进程来预选事件是否是管理感兴趣，从而避免他不感兴趣的事件生成。配置使用一系列表达式，用以预选事件、要用的用，以及用于自身，以及未用自身的进程的全局配置。
- 约简 (*reduction*)：从已有的数据中选出用于保留、打印或分析的进程。除此之外，它也表示从数据中去不需要的数据。通常使用约简操作，管理员可以保留数据的策略。例如，进程的某些信息，可能会保留一个月之久，但在约简之后，某些信息约简操作，只保留登录信息用于存。

## 18.3. 安装支持

关于事件的支持，已随标准的 `installworld` 进程完成。管理员可以通过看 `/etc/security` 的内容来了解一下。能看到一些名字以 `audit` 的文件，例如 `audit_event`。

关于功能的用支持目前是作为 `FreeBSD` 基本系统的一部分来安装的。默认内核中也包含了关于事件的内核支持，但如果使用的是定制内核，就必须在内核配置文件中明确指定希望添加一支持：

```
options AUDIT
```

接下来，按照 [配置FreeBSD的内核](#) 中所介绍的来完成一次内核的编译和安装。

在编译好并安装了内核，并重新加载了系统之后，就可以在 `rc.conf(5)` 中添加下列配置来启用服务了：

在编译、安装了带功能的内核，并重新启动计算机之后，就可以在 `rc.conf(5)` 中添加下列配置来启用服务了：

```
auditd_enable="YES"
```

此后，必须重新加载系统，或通过下面的命令手工启动服务来支持：

```
/etc/rc.d/auditd start
```

## 18.4. 配置文件配置

所有用于安全服务的配置文件，都可以在 `/etc/security` 找到。要启动服务，下面些文件必须存在：

- `audit_class` - 包含关于事件的定义。
- `audit_control` - 控制子系统的特性，例如默认、在日志所在的卷上保留的最小空间、日志的最大尺寸，等等。
- `audit_event` - 文字化的系统事件名称和描述，以及每个事件属于哪个。
- `audit_user` - 特定用的需求，些配置在登录会与全局的默认合并。
- `audit_warn` - 由 `auditd` 用的一个可定制的 `shell` 脚本，用于在意外情况，如用于日志的空间少，或日志文件被翻，生成警告信息。



在 `audit` 和 `auditd` 配置文件时一定要小心，因配置文件中的 `+` 会致事件不正。

### 18.4.1. 事件表式

在 `audit` 配置文件中的很多地方会用到表式来定哪些事件是需要。表式中需要指定要匹配的事件型，并使用前指定是否接受或忽略匹配的事件，此外，可以指定一个可指明匹配成功或失败的操作。表式是按从左到右的顺序算的，而对于一个表式的情形，是通将后一个追加到前一个之后来。

下面列出了在 `audit_class` 中的默认事件型：

- **all** - *all* (全部) - 表示匹配全部事件。
- **ad** - *administrative* (管理) - 所有在系统上所行的管理性操作。
- **ap** - *application* (应用) - 用程序定的作。
- **cl** - *file close* (文件) - 系用的操作。
- **ex** - *exec* (行) - 程序的行。于命令行参数和环境量的是通在 `audit_control(5)` 中 `policy` 的 `argv` 和 `envv` 参数来控制的。
- **fa** - *file attribute access* (造文件属性) - 象属性，例如 `stat(1)`、`pathconf(2)` 以及似事件。
- **fc** - *file create* (建文件) - 建了文件的事件。
- **fd** - *file delete* (除文件) - 所生的文件除事件。
- **fm** - *file attribute modify* (修改文件属性) - 文件属性生化的事件，例如 `chown(8)`、`chflags(1)`、`flock(2)`，等等。
- **fr** - *file read* (文件数据) - 取数据、文件以方式打等事件。
- **fw** - *file write* (写文件数据) - 写入数据、文件以写方式打等事件。
- **io** - *ioctl* - 系用的使用。
- **ip** - *ipc* - 各形式的程通信 (IPC)，包括 POSIX 管道和 System V IPC 操作。
- **lo** - *login\_logout* - 系中生的 `login(1)` 和 `logout(1)` 事件。
- **na** - *non attributable* (无主) - 无法的事件。
- **no** - *invalid class* (无效) - 表示不匹配任何事件。
- **nt** - *network* (网) - 与网操作有的事件，例如 `connect(2)` 和 `accept(2)`。
- **ot** - *other* (其它) - 各事件。
- **pc** - *process* (程) - 程操作，例如 `exec(3)` 和 `exit(3)`。

些事件，可以通修改 `audit_class` 和 `audit_event` 个配置文件来行定制。

个列表中，个均包含一个表示匹配成功/失败操作的前，以及一是否是加或去事件或型的匹配。

- (none) 事件的成功和失败例。
- + 的成功事件。
- - 的失败事件。

- $\wedge$  不本中的成功或失事件。
- $\wedge+$  不本中的成功事件。
- $\wedge-$  不本中的失事件。

下面例子中的字符串表示成功和失的登/注事件，而行事件，只成功的：

```
lo,+ex
```

## 18.4.2. 配置文件

多数情况下，在配置系，管理只需修改个文件：`audit_control` 和 `audit_user`。前者控制系的属性和策略，而后者用于具体的用来微。

### 18.4.2.1. audit\_control 文件

`audit_control` 文件指定了一系列用于子系的默置。通看个文件，我可以看到下面的内容：

```
dir:/var/audit
flags:lo
minfree:20
naflags:lo
policy:cnt
filesz:0
```

里的 `dir` 可以用来置用于保存日志的一个或多个目。如果指定了多个目，将在填一个之后用下一个。一般而言，通常都会配置保存在一个用的文件系之下，以避免系与其它子系在文件系的候所生的冲突。

`flags` 字段用于有主事件配置系的条件。在前面的例子中，所有用成功和失的登和注都会被。

`minfree` 参数用于定保存日志的文件系上剩余空的最小百分比。当超一，将生一个警告。前面的例子中，最小剩余空比例置成了成。

`naflags` 表示无主事件，例如作登程和系服的那些程的事件。

`policy` 用于指定一个以逗号分隔的策略志表，以控制一系列行。默的 `cnt` 志表示系在失行（烈建使用个志）。一个常用的志是 `argv`，它表示在命令行操作，同 `execve(2)` 系用的命令行参数。

`filesz` 指明了日志在自停止和翻之前允的最大尺寸。默 0 表示禁用自日志翻。如果配置的不是零，但小于最小 512k，个配置会被忽略，并在日志中一消息。

### 18.4.2.2. audit\_user 文件

`audit_user` 文件允管理特定用指定一的需。一行使用个字段来配置用的：第一个是 `alwaysaudit` 字段，它指明了一用会行的事件；而第二个是 `neveraudit` 字段，它指明了一系列用不的事件。



在下述 `audit_user` 示例文件中，记录了 `root` 用户的登录/注销事件，以及成功的命令执行事件，此外，记录了 `www` 用户的文件创建和成功的命令执行事件。如果与前面的示例 `audit_control` 文件配合使用，`root` 的 `lo` 就是多余的，而 `www` 用户而言，其登录/注销事件也会被记录：

```
root:lo,+ex:no
www:fc,+ex:no
```

## 18.5. 管理子系

### 18.5.1. 查看日志

日志是以 BSM 二进制格式保存的，因此必须使用工具来对其进行修改，或将其转换为文本。`praudit(1)` 命令能将日志文件转换为文本格式；而 `auditreduce(1)` 命令可以分析、存储或打印目的来查看日志文件。`auditreduce` 支持一系列参数，包括事件类型、事件ID、用户、事件的日期和时间，以及文件路径或操作对象。

例如，`praudit` 工具会将指定的日志转换为文本格式的日志：

```
# praudit /var/audit/AUDITFILE
```

此 `AUDITFILE` 是要查看的日志文件。

日志中包括一系列记录，有些记录由一系列短字 (token) 组成，而 `praudit` 能把它按顺序显示一行。每个短字都属于某个特定的类型，例如 `header` 表示记录头，而 `path` 表示在一次名字空间中的文件路径。下面是一个 `execve` 事件的例子：

```
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
```

这个记录表示一次成功的 `execve` 调用，执行了 `finger` `doug`。在参数短字中是由 shell 提交给内核的命令行。`path` 短字包含了由内核返回得到的可执行文件路径。`attribute` 短字中包含了可执行文件的描述，特别地，它包括了文件的权限模式，用以确定用程序是否是 `setuid` 的。`subject(主体)` 短字描述了主体进程，并顺序记录了用户 ID、生效用户 ID 和组 ID、调用用户 ID 和组 ID、进程 ID、会话 ID、端口 ID，以及登录地址。注意用户 ID 和调用用户 ID 是不同的：用户 `robert` 在运行这个命令之前已切换为 `root`，但它会以最初运行进程的调用用户身份运行。最后，`return` 短字表示运行成功，而 `trailer` 表示记录结束。

`praudit` 可以使用 `-x` 参数来支持 XML 格式的输出。

### 18.5.2. 管理子系

由于日志可能会很大，管理员可能会希望查看的一个子集来使用，例如与特定用户相关的记录：

```
# auditreduce -u trhodes /var/audit/AUDITFILE | praudit
```

将保存在 AUDITFILE 中的所有由 **trhodes** 生的日志。

### 18.5.3. 委派权限

在 **audit** 中的用，有取 `/var/audit` 下的的限；默情况下，个是空的，因此只有 **root** 用可以取。如果希望某个用指定，可以将其加入 **audit**。由于看日志的内容可以提供于用和程行的大量深度信息，在委派些力，必慎行事。

### 18.5.4. 通过管道来控制

管道是位于文件系统自制 (cloning) 的虚，用于用程序控制正在行的流，主要是了足入侵和系控件作者的需要。不，管理而言，管道也提供了一无需冒文件属主出流的麻，或由于日志翻而打断事件流的麻，而控的方便途径。要跟踪事件流，使用下面的命令行：

```
# praudit /dev/auditpipe
```

默情况下，管道点只有 **root** 用才能。如果希望 **audit** 的成能它，在 `devfs.rules` 中加入下述 **devfs**：

```
add path 'auditpipe*' mode 0440 group audit
```

参 [devfs.rules\(5\)](#) 以了解于配置 `devfs` 文件系的—信息。



很容易配置出事件反循环，也就是看事件的操作本身会生更多的事件。例如，如果所有的网 I/O 均被，又在 SSH 会中行 [praudit\(1\)](#)，就会以很高的速率持的流，因示一个事件都会生新的事件。建在需要在管道上行 **praudit**，一个没有行粒度 I/O 的会来行。

### 18.5.5. 日志文件的管理

只由内核写入，且只能由 **auditd** 管理。管理不使 [newsyslog.conf\(5\)](#) 或其它工具来完成日志的工作。可以使用 **audit** 管理工具来、重新配置系，并完成日志。下面的命令将服建新的日志，并信号内核要求其使用新的日志。旧日志将止并被改名，此，管理就可以操作它了。

```
# audit -n
```



如果 **auditd** 服程序没有在行，个命令将失并出提示。

在 `/etc/crontab` 加入如下置，将使 [cron\(8\)](#) 十二小将日志一次。

```
0    */12    *    *    *    root    /usr/sbin/audit -n
```

些修改会在保存 `/etc/crontab` 后生效。

于文件基于尺寸的自翻，可以通过 `audit_control(5)` 中的 `filesz` 来配置，个在第一章的配置文件一中已介。

### 18.5.6. 清理操作

由于文件会得很大，通常会希望在服它，其行或。`audit_warn` 脚本可以用来在一系列与有的事件生，行一些用定的操作，也包括在翻行清理操作。例而言，可以在 `audit_warn` 脚本中加入下列内容来在它：

```
#
# Compress audit trail files on close.
#
if [ "$1" = closefile ]; then
    gzip -9 $2
fi
```

其它存操作也包括将制到一个中央的服器，除旧的的文件，或并除不需要的等。个脚本会在文件正常行一次，因此在非正常系，就不会行它了。

# Chapter 19. 存

## 19.1. 概述

本章介绍了 FreeBSD 中磁的使用方法。包括内存，网附属磁和标准的 SCSI/IDE 存，以及使用 USB 的。

读完章，将了解到：

- FreeBSD 中用来描述硬上数据 (partitions and slices)。
- 如何在的系上加硬。
- 如何配置 FreeBSD 来使用 USB 存。
- 如何置虚文件系统，例如内存磁。
- 如何使用配来限制磁空的使用。
- 如何加磁安全来防功。
- 如何刻 CD 和 DVD 。
- 用于的多存媒介。
- 如何在 FreeBSD 上使用程序。
- 如何到磁。
- 文件系统快照是什，以及如何有效地使用它。

在章之前，：

- 知道去配置和安装新的 FreeBSD 内核 ([配置FreeBSD的内核](#))。

## 19.2. 命名

下面是在 FreeBSD 上被支持的物理存和它被分配的命名。

表 7. 物理磁命名

| 器型                | 命名  |
|-------------------|---|
| IDE 硬器            | ad  |
| IDE CDROM 器       | acd   |
| SCSI 硬以及 USB 大容量存 | da  |
| SCSI CDROM 器      | cd  |
| 各非标准 CDROM 器      | 用于 Mitsumi CD-ROM 的 mcd 以及用于 Sony CD-ROM 器的 scd |
| Floppy drives     | fd  |
| SCSI tape drives  | sa  |
| IDE tape drives   | ast   |

| 设备类型         | 设备命名  |
|--------------|---|
| Flash drives | <b>fla</b> for DiskOnChip® Flash device   |
| RAID drives  | <b>aacd</b> for Adaptec® AdvancedRAID, <b>mlx</b> d and <b>mly</b> d for Mylex®, <b>amr</b> d for AMI MegaRAID®, <b>ida</b> d for Compaq Smart RAID, <b>tw</b> e for 3ware® RAID. |

## 19.3. 添加磁盘

下面将会介绍如何在一台只有一块磁盘的机器上新增一块 SCSI 磁盘。首先 需要关掉计算机，然后按操作程序来安装设备，控制器和驱动程序。由于 各厂家生产的设备各不相同，具体的安装不在此文介绍之内。

以 **root** 用登录。安装完后，看一下 `/var/run/dmesg.boot` 有没有看到新的磁盘。在我 的例子中新加的磁盘就是 `da1`，我 从 `/1` 挂上它。(如果 正添加 IDE 设备，设备名是 `ad1`)。

因 FreeBSD 运行在 IBM-PC 兼容机上，它必须遵循 PC BIOS 分区。 与 的 BSD 分区是不同的。一个 PC 的磁盘最高只能有四个 BIOS 主分区。如果磁盘只安装 FreeBSD 可以使用 *dedicated* 模式。 外，FreeBSD 必须安装在 PC BIOS 支持的分区内。FreeBSD 把分区叫作 *slices* 可能会把人 糊。 也可以在只安装 FreeBSD 的磁盘上使用 *slices*，也可以在安装有其它操作系统的磁盘上使用 *slices*。 不会影 其它操作系统的 **fdisk** 分区工具。

在 slice 方式表示下，设备被添加到 `/dev/da1s1e`。可以作：SCSI 磁盘，号 1 (第二个 SCSI 磁盘)，slice 1 (PC BIOS 分区 1)，的 BSD 分区 e。在有些例子中，也可以化 `/dev/da1e`。

由于 **bsdlabel(8)** 使用 32-位 的整数来表示扇区号， 因此在多数情况下它的表力限于 个磁盘  $2^{32}-1$  个扇区， 或 2TB。 **fdisk(8)** 格式允 的起始扇区号不能高于  $2^{32}-1$ ， 而分区尺寸也不能超  $2^{32}-1$ ， 一来通常情况下分区尺寸不能超 2TB， 而磁盘尺寸 不能超 4TB。 **sunlabel(8)** 格式的限制是 个分区  $2^{32}-1$  个扇区， 但可以有 8 个分区， 因而可以支持最大 16TB 的磁盘。 于更大的磁盘， 可以使用 **gpart(8)** 来建 GPT 分区。GPT 除了支持大磁盘之外， 不受 4 个 slice 的限制。

### 19.3.1. 使用 **sysinstall(8)**

## 1. 使用 Sysinstall

可以使用 `sysinstall` 命令的菜单来分区和创建一个新的磁盘。这一操作需要有 `root` 权限，可以直接使用 `root` 登录或者使用 `su` 命令来切换到 `root` 用户。运行 `sysinstall`，然后选择 `Configure` 菜单。在 `FreeBSD Configuration Menu` 下，上下移动，选择 `Fdisk` 条目。

## 2. fdisk 分区器

进入 `fdisk` 分区器后，选择 `A`，FreeBSD 将使用全部的磁盘。当被告知 "remain cooperative with any future possible operating systems"，回答 `YES`。使用 `W` 保存刚才的修改。在使用 `Q` 退出 `FDISK` 器。下面会看到有 "主引导区" 的信息。在已存在的行的系统上添加了一个磁盘，因此显示 `None`。

## 3. Disk Label 器

接下来，退出 `sysinstall` 并且再次运行它，并按照上面的步骤直接输入 `Label`。进入磁盘器。就是要建的 BSD 分区。一个磁盘最多可以有 8 个分区，用 `a-h`。有几个分区有特殊的用途。`a` 分区被用来作根分区 (/)。系统磁盘（例如：从那儿启动的分区）必须有一个 `a` 分区。`b` 分区被用作交换分区，可以用很多磁盘用作交换分区。`c` 分区代表整个硬盘，或在 FreeBSD `slice` 模式下代表整个 `slice`。其它分区作一般分区来使用。

`sysinstall` 的磁盘器用 `e` 表示非 `root` 和非 `swap` 分区。在磁盘器中，可以使用 `C` 建立一个文件系统。当提示是否是一个 FS（文件系统）或 `swap`，选择 `FS`，然后输入一个加号（如：`/mnt`）。当在 `post-install` 模式添加一个磁盘，`sysinstall` 不会在 `/etc/fstab` 中建立条目，所以是否指定加号并不重要。

在已准备把新数据写到磁盘上，然后建立一个文件系统，可以按下 `W`。输出任何条目都会不能建立新的分区。可以退出磁盘器然后重新运行 `sysinstall`。

## 4. 完成

下面一步就是编辑 `/etc/fstab`，为磁盘添加一个新条目。

## 19.3.2. 使用命令行工具

### 19.3.2.1. 使用 Slices

安装将允许磁盘与可能安装在计算机上的其它操作系统一起正常工作，而不会扰乱其它操作系统的分区。推荐使用这种方法来安装新磁盘，除非有更好的理由再使用 `dedicated` 模式！

```
# dd if=/dev/zero of=/dev/da1 bs=1k count=1
# fdisk -BI da1 #初始化新磁盘
# bsdlabel -B -w da1s1 auto #加上
# bsdlabel -e da1s1 # 在刚才建的磁盘分区
# mkdir -p /1
# newfs /dev/da1s1e # 建的分区重新个操作
# mount /dev/da1s1e /1 # 挂上分区
# vi /etc/fstab # 完成之后，添加合到的 /etc/fstab文件。
```

如果有一个 IDE 磁盘，它得要用 `ad` 替换前面的 `da`。

### 19.3.2.2. 专用模式

如果它并没有安装其它的操作系统，可以使用 **dedicated** 模式。专用模式可能会弄乱 Microsoft 的操作系统，但不会对它进行破坏。它不适用于 IBM OS/2® 的 "appropriate" 分区。

```
# dd if=/dev/zero of=/dev/da1 bs=1k count=1
# bsdlabel -Bw da1 auto
# bsdlabel -e da1           # 建立 'e' 分区
# newfs /dev/da1e
# mkdir -p /1
# vi /etc/fstab             # 在 /dev/da1e 添加一个条目
# mount /1
```

另一种方法：

```
# dd if=/dev/zero of=/dev/da1 count=2
# bsdlabel /dev/da1 | bsdlabel -BR da1 /dev/stdin
# newfs /dev/da1e
# mkdir -p /1
# vi /etc/fstab             # 在 /dev/da1e 添加一个条目
# mount /1
```

## 19.4. RAID

### 19.4.1. 软件 RAID

#### 19.4.1.1. 连接磁盘控制器配置 (CCD)

需要一个大容量存储比它好的解决方案，最重要的因素是产品的速度、性能和成本。通常三者不可能都满足；要得到比它快和可靠的大容量存储，就比它昂贵。但如果将成本降下来，那它的速度或可靠性就会打折扣。

在下面描述的系统中，价格被它最重要的因素，接下来是速度和性能。这个系统的数据传输速度基本上受限于网络。性能也非常重要，CCD 控制器上的所有数据都被写到了 CD-R 上，可以很容易地对数据进行恢复。

在一个大容量的存储解决方案中，第一是要满足自己的需求。如果它的需求更偏重于速度和性能，那它的解决方案将不同于上面的系统。

##### 19.4.1.1.1. 安装硬件

除了 IDE 系统磁盘外，它有三个 Western Digital 30GB、5400 RPM 的 IDE 磁盘成了大约 90G 的连接磁盘存储空间。理想情况是三个 IDE 硬盘都独占 IDE 控制器和数据，但它了尽可能降低成本，通常并不会安装更多的控制器，而是通过配置跳线，使三个 IDE 控制器都管理一个主盘和一个从盘。

重新后，系统 BIOS 被配置成自动检测硬盘。FreeBSD 识别到它：

```
ad0: 19574MB <WDC WD205BA> [39770/16/63] at ata0-master UDMA33
ad1: 29333MB <WDC WD307AA> [59598/16/63] at ata0-slave UDMA33
ad2: 29333MB <WDC WD307AA> [59598/16/63] at ata1-master UDMA33
ad3: 29333MB <WDC WD307AA> [59598/16/63] at ata1-slave UDMA33
```



如果 FreeBSD 没有看到它，确定它的跳是否设置正。大多数 IDE 磁盘有一个 "Cable Select" 跳。它不是设置 master/slave 硬盘的跳。本文信息来制定正的跳设置。

接下来考的是，如何建立文件系统。好好研究一下 [vinum\(4\)](#) ([Vinum 卷管理程序](#))和 [ccd\(4\)](#) 的方式，在这里我看看 [ccd\(4\)](#)

#### 19.4.1.1.2. 安装 CCD

[ccd\(4\)](#) 允许将几个相同的磁盘通一个文件系统连接起来。要使用 [ccd\(4\)](#)，需要在内核中配置 [ccd\(4\)](#) 支持。把行加入到内核配置文件中，然后重建内核：

```
device    ccd
```

[ccd\(4\)](#) 的支持也可以内核模块的形式入。

要安装 [ccd\(4\)](#)，首先需要使用 [bsdlablel\(8\)](#) 来硬盘：

```
bsdlablel -w ad1 auto
bsdlablel -w ad2 auto
bsdlablel -w ad3 auto
```

此将整个硬盘建 `ad1c`, `ad2c` 和 `ad3c`。

下一是改 `disklable` 的型。也可以使用 [bsdlablel\(8\)](#) 来：

```
bsdlablel -e ad1
bsdlablel -e ad2
bsdlablel -e ad3
```

儿在个已置了 [EDITOR](#) 境量的磁上打了 `disklable`，在我我例子中使用的是 [vi\(1\)](#)。

可以看到：

```
8 partitions:
#          size  offset  fstype  [fsize bsize bps/cpg]
c: 60074784      0   unused      0      0      0 # (Cyl.  0 - 59597)
```

添加一个新的 `e` 分区 [ccd\(4\)](#) 用。可以是 `c` 分区的一个副本，但 `fstype` 必是 `4.2BSD`。做完之后，会看到一面些：



```

8 partitions:
#          size  offset   fstype   [fsize bsize bps/cpg]
c: 60074784      0   unused      0      0      0 # (Cyl.   0 - 59597)
e: 60074784      0  4.2BSD      0      0      0 # (Cyl.   0 - 59597)

```

#### 19.4.1.1.3. 建立文件系统

在已有个磁盘都加上了，下面需要建立 [ccd\(4\)](#)。要这么做，需要使用 [ccdconfig\(8\)](#) 工具，同时需要提供以下的：

```
ccdconfig ccd0 32 0 /dev/ad1e /dev/ad2e /dev/ad3e
```

这个命令的意和用法如下所示：配置命令的第一个参数，在是 `/dev/ccd0c`。 `/dev/` 部分是任。下一个参数是文件系统的入入(interleave)。入入定了一个磁盘中一个分段或条(stripe)的大小，通常是 512 个字。所以一个 32 的入入将是 16,384 字。入入 [ccdconfig\(8\)](#) 附了。如果要用设备像，需要在儿指定它。在这个配置中没有做 [ccd\(4\)](#) 的像，所以把它 0 (zero)。 [ccdconfig\(8\)](#) 的最后配置是排列。使用完整的路径名。

行 [ccdconfig\(8\)](#) 后 [ccd\(4\)](#) 就配置好了。在要建文件系统了，参考 [newfs\(8\)](#) ，行下命的命令：

```
newfs /dev/ccd0c
```

#### 19.4.1.1.4. 自建

最后，要挂上 [ccd\(4\)](#)，需要先配置它。把当前的配置文件写入 `/etc/ccd.conf` 中，使用下面的命令：

```
ccdconfig -g > /etc/ccd.conf
```

当重新系，如果 `/etc/ccd.conf` 存在，脚本 `/etc/rc` 就行 [ccdconfig -C](#)。就能自配置 [ccd\(4\)](#) 以到它能被挂上。



如果入了用模式，在 [mount\(8\)](#) 上 [ccd\(4\)](#) 之前，需要行下面的命令来配置列：

```
ccdconfig -C
```

要自挂接 [ccd\(4\)](#)，需要 [ccd\(4\)](#) 在 `/etc/fstab` 中配置一个，以便在它能被挂上。如下所示：

```
/dev/ccd0c          /media             ufs                rw                 2                 2
```

### 19.4.1.2. Vinum 卷管理

Vinum 卷管理是一个虚磁的工具。它使磁从 的接口和数据映射中独立出来。与的存相比，加了 活性、性能和可性。 [vinum\(4\)](#) 了 RAID-0、RAID-1 和 RAID-5 三模式，它

既可以独立使用，也可合使用。

参考 [Vinum 卷管理程序](#) 得到更多 [vinum\(4\)](#) 的信息。

### 19.4.2. 硬件 RAID

FreeBSD 支持很多硬件 RAID 控制器。有些硬件不需要 FreeBSD 指定软件来管理 RAID 系统。

使用 BIOS 支持的硬件，一般情况下有些硬件可以自行操作。下面是一个明确的描述设置一个 Promise IDERAID 控制器。当硬件安装好且系重启后，屏幕上显示一个信息。接着输入硬件设置屏幕。在那里，可以把所有的磁盘合在一起使用。FreeBSD 将磁盘看作一个设备。其它种的 RAID 也可以相的行设置。

### 19.4.3. 重建 ATA RAID1 阵列

FreeBSD 允许拔出阵列中坏的磁盘。在重新系之前注意一点。

可能会在 /var/log/messages 或者在 [dmesg\(8\)](#) 的输出中看到似下面些的内容：

```
ad6 on monster1 suffered a hard error.
ad6: READ command timeout tag=0 serv=0 - resetting
ad6: trying fallback to PIO mode
ata3: resetting devices .. done
ad6: hard error reading fsbn 1116119 of 0-7 (ad6 bn 1116119; cn 1107 tn 4 sn 11)\\
status=59 error=40
ar0: WARNING - mirror lost
```

使用 [atacontrol\(8\)](#)，看更多的信息：

```
# atactrol list
ATA channel 0:
  Master:      no device present
  Slave:   acd0 <HL-DT-ST CD-ROM GCR-8520B/1.00> ATA/ATAPI rev 0

ATA channel 1:
  Master:      no device present
  Slave:      no device present

ATA channel 2:
  Master:   ad4 <MAXTOR 6L080J4/A93.0500> ATA/ATAPI rev 5
  Slave:    no device present

ATA channel 3:
  Master:   ad6 <MAXTOR 6L080J4/A93.0500> ATA/ATAPI rev 5
  Slave:    no device present

# atactrol status ar0
ar0: ATA RAID1 subdisks: ad4 ad6 status: DEGRADED
```

1. 首先将包含故障的 ata 通道卸下，以便安全地将其拆除：

```
# atacontrol detach ata3
```

2. 上磁
3. 重新挂接 ata 通道：

```
# atacontrol attach ata3
Master:  ad6 <MAXTOR 6L080J4/A93.0500> ATA/ATAPI rev 5
Slave:   no device present
```

4. 将新作加入列：

```
# atacontrol addspare ar0 ad6
```

5. 重建列：

```
# atacontrol rebuild ar0
```

6. 可以通过下面的命令来看度：

```
# dmesg | tail -10
[output removed]
ad6: removed from configuration
ad6: deleted from ar0 disk1
ad6: inserted into ar0 disk1 as spare

# atacontrol status ar0
ar0: ATA RAID1 subdisks: ad4 ad6 status: REBUILDING 0% completed
```

7. 等待操作完成。

## 19.5. USB 存储

到目前为止，有多种外部存储解决方案，例如：通用串行总线（USB）：硬盘、USB thumbdrives、CD-R burners 等等。FreeBSD 对这些提供了支持。

### 19.5.1. 配置

USB 大容量存储，在 [umass\(4\)](#)，中提供了 USB 存储的支持。如果使用 GENERIC 内核，不必要改配置文件里的任何内容。如果使用了定制的内核，就要定下面的行出在的内核配置文件里：

```
device scbus
device da
device pass
device uhci
device ohci
device ehci
device usb
device umass
```

[umass\(4\)](#) 程序使用 SCSI 子系来 USB 存， 的 USB 将被系看成 一个 SCSI 。依 主板上的 USB 芯片， 只 `device uhci` 或用于 USB 1.X 支持的 `device ohci` 二者之一即可， 但是 者都加入内核配置文件当中也是无害的。 于 USB 2.X 控制器的支持由 [ehci\(4\)](#) 提供 (`device ehci` 一行)。不要忘了如果 加入了上面的几行要重新 和安装内核。

如果 的 USB 是一个 CD-R 或 DVD 刻机， SCSI CD-ROM 程序， [cd\(4\)](#)，就必 加入内核中通 下面行：



```
device cd
```

由于刻机被 SCSI ， 因此， 不在内核配置文件中 使用 [atapicam\(4\)](#) 程序。

### 19.5.2. 配置

配置好后准行： 入 的 USB ， 在系信息中 ([dmesg\(8\)](#)), 会出 像下面的：

```
umass0: USB Solid state disk, rev 1.10/1.00, addr 2
GEOM: create disk da0 dp=0xc2d74850
da0 at umass-sim0 bus 0 target 0 lun 0
da0: <Generic Traveling Disk 1.11> Removable Direct Access SCSI-2 device
da0: 1.000MB/s transfers
da0: 126MB (258048 512 byte sectors: 64H 32S/T 126C)
```

当然，商， (da0) 和其它的 信息会根据 的配置不同 而有所不同。

因 USB 被看作 SCSI 中的一个， `camcontrol` 命令也能 用来列出 USB 存 和系 的：

```
# camcontrol devlist
<Generic Traveling Disk 1.11>      at scbus0 target 0 lun 0 (da0,pass0)
```

如果 上已 包含了文件系统， 在 就可以挂接它了。 如果需要， 参 [添加磁](#) 来了解如何在 USB 器上格式化和 建分区。



允 非可信用挂 任意介， 例如通 使用前面介 的 `vfs.usermount` 来 用的功能，从安全角度来看是很不保 的。 FreeBSD 中的 大多数文件系统并不提供 意 的内建防 能力。

如果希望磁盘能被普通用户挂载，需要做一些其它操作。首先，在 USB 存储设备接到计算机上，系统自动生成文件，必须是用户能写的。一做法是所有属于 `operator` 的用户都可以读写。要完成工作，首先需要用 `pw(8)` 来用户指定。其次，在生成文件，`operator` 能写它。可以通过在 `/etc/devfs.rules` 中加一些相关的设置来：

```
[localrules=5]
add path 'da*' mode 0660 group operator
```



如果系统中已有其它 SCSI 磁盘，上述操作必须做一些变化。例如，如果系统中已存在了名称 `da0` 到 `da2` 的磁盘，第二行改：

```
add path 'da[3-9]*' mode 0660 group operator
```

会将系统中已存在的磁盘，排除在属于 `operator` 的用户之外。

外，需要在 `/etc/rc.conf` 文件中，用 `devfs.rules(5)` 设置：

```
devfs_system_ruleset="localrules"
```

接下来，需要配置内核，令普通用户能挂载文件系统。最简的方法是将下面的配置加入到 `/etc/sysctl.conf`：

```
vfs.usermount=1
```

注意，这个设置只有在下次重启系统才会生效。外，也可以使用 `sysctl(8)` 来设置这个量。

最后一是建立将要挂载文件系统的目录。这个目录必须是属于将要挂载文件系统的用户的。以 `root` 身份建立属于用户的 `/mnt/username` (此 `username` 替换成用户的登录名，并把 `usergroup` 替换成用户所属的组)：

```
# mkdir /mnt/username
# chown username:usergroup /mnt/username
```

假如已插入了一个 USB 存储设备，并且系统将其识别为 `/dev/da0s1`，由于有些设备通常是 FAT 文件系统，用户可以挂载它：

```
% mount -t msdosfs -o -m=644,-M=755 /dev/da0s1 /mnt/username
```

如果拔出设备 (必须首先将其磁盘卷卸下)，会在系统消息缓冲区中看到类似下面的信息：

```
umass0: at uhub0 port 1 (addr 2) disconnected
(da0:umass-sim0:0:0:0): lost device
(da0:umass-sim0:0:0:0): removing device entry
GEOM: destroy disk da0 dp=0xc2d74850
umass0: detached
```

### 19.5.3. 深入

除了 [Adding Disks](#) 和 [Mounting and Unmounting File Systems](#) 章之外， 各手册也是有益的：[umass\(4\)](#), [camcontrol\(8\)](#), 和 FreeBSD 8.X 的 [usbconfig\(8\)](#) 或者于更早期 FreeBSD 版本的 [usbdevs\(8\)](#)。

## 19.6. 建和使用光学介(CD)

### 19.6.1. 介

CD 与普通的磁相比有很多不同的特性。最初它是不能被写入的。 由于没有磁和磁道移动的延迟，所以它可以的行取。方便的在个系之行数据的，比起相同大小的存介来。

CD 有磁道，系到数据取的性而不是物理磁的性能。 要在 FreeBSD 中制作一个 CD，要准备好要写到 CD 上的数据文件，然后根据个 tracks 写入到 CD。

ISO 9660 文件系统被用来理些差。 但令人憾的是， 它也有一些其他文件系统所没有限制，不幸的是， 它提供了一展机制， 使得正写入的 CD 能超越些限制， 而又能在不支持些展的系上正常使用。

[sysutils/ port](#) 包括了 [mkisofs\(8\)](#)， 是一个可以用来生成包含 ISO 9660 文件系统的数据文件的程序。 他也提供了于一些展的支持， 下面将介。

使用个工具来刻 CD 取决于的 CD 刻机是 ATAPI 的， 是其他型的。 于 ATAPI CD 刻机， 可以使用基本系附的 [burncd](#) 程序。 SCSI 和 USB CD 刻机， 需要配合 [cdrecord](#) 程序使用， 它可以通过 [sysutils/cdrtools port](#) 安装。 除此之外， 在 ATAPI 接口的刻机上， 也可以配合 [ATAPI/CAM 模](#) 来使用 [cdrecord](#) 以及其它 SCSI 刻机撰写的工具。

如果想使用形界面的 CD 刻件， 可以考一下 X-CD-Roast 或 K3b。 些工具可以通过使用安装包， 或通过 [sysutils/xcdroast](#) 和 [sysutils/k3b ports](#) 来安装。 X-CD-Roast 和 K3b 需要 [ATAPI/CAM 模](#) 配合 ATAPI 硬件。

### 19.6.2. mkisofs

[mkisofs\(8\)](#) 程序作 [sysutils/cdrtools port](#) 的一部分， 将生成 ISO 9660 文件系统， 其中包含 UNIX® 命名空的文件名。 最的用法是：

```
# mkisofs -o imagefile.iso /path/to/tree
```

个命令将建一个包含 ISO9660 文件系的 *imagefile.iso* 文件， 它是目 */path/to/tree* 的一个副本。 在理程中， 它将文件名称映射准的 ISO9660 文件系的文件名， 将排除那些不典型的 ISO 文件系的文件。

有很多选项用来克服那些限制。特别的，**-R** 选项能用 Rock Ridge 扩展一般的 UNIX® 系，**-J** 选项用于 Microsoft 系的 Joliet 扩展，**-hfs** 选项用来建用于 Mac OS® 系的 HFS 文件系统。

对于那些即将要在 FreeBSD 系中使用 CD 的人来，**-U** 选项用来消除所有文件名的限制。当使用 **-R** 选项，它会生成一个文件系统映像，它与从那儿 FreeBSD 是一致的，然它在多方面也反了 ISO 9660 的标准。

最后一个常用的选项是 **-b**。它用来指定映像的位置，用以生成 "El Torito" 的 CD。这个选项使用一个参数，用以指定将写入 CD 的目录的根。默认情况下，[mkisofs\(8\)](#) 会以常有的 "模式" 方式来建 ISO，因此它希望引导映像文件的尺寸恰好是 1200，1440 或 2880 KB。某些引导加载器，例如 FreeBSD 的行版磁碟，并不使用模式；这种情况下，需要使用 **-no-emul-boot** 选项。因此，如果 `/tmp/myboot` 是一个包含了映像文件 `/tmp/myboot/boot/cdboot` 的可引导的 FreeBSD 系，就可以使用下面的命令生成 ISO 9660 文件系统映像 `/tmp/bootable.iso`：

```
# mkisofs -R -no-emul-boot -b boot/cdboot -o /tmp/bootable.iso /tmp/myboot
```

完成些工作之后，如果的内核中配置了 `md`，就可以用下列命令来挂接文件系统了：

```
# mdconfig -a -t vnode -f /tmp/bootable.iso -u 0
# mount -t cd9660 /dev/md0 /mnt
```

可以 `/mnt` 和 `/tmp/myboot` 是一致的。

可以使用 [mkisofs\(8\)](#) 的其它选项来调整它的行。特别是修改 ISO 9660 的分格式，建 Joliet 和 HFS 格式的磁碟。看 [mkisofs\(8\)](#) 手册得到更多的帮助。

### 19.6.3. burncd

如果用的是 ATAPI 的 CD 刻机，可以使用 **burncd** 命令来刻的 CD ISO 映像文件。**burncd** 命令是基本系的一部分，中以使用 `/usr/sbin/burncd` 来安装。用法如下：

```
# burncd -f cddevice data imagefile.iso fixate
```

在 `cddevice` 上刻一个 `imagefile.iso` 的副本。默认的是 `/dev/acd0`。参考 [burncd\(8\)](#) 以了解置写入速度的参数，如何在刻完成之后自出 CD，以及刻音数据。

### 19.6.4. cdrecord

如果没有一个 ATAPI CD 刻机，必使用 **cdrecord** 来刻的 CD。**cdrecord** 不是基本系的一部分；必从 [sysutils/cdrtools](#) 或当的 package 安装它。基本系的化可能会引起个程序的。可能是由 "coaster" 引起的。当升系，同需要升 port，或者如果使用 **-STABLE**，那在升到新版本也要升 port。

**cdrecord** 有多样，基本用法与 **burncd** 相似。刻一个 ISO 9660 映像文件只需做：

```
# cdrecord dev=device imagefile.iso
```

使用 `cdrecord` 的比巧妙的方法是到使用的 `dev` 。要到正的位置，可以使用 `cdrecord` 的 `-scanbus` 选项，会产生的结果：

```
# cdrecord -scanbus
Cdrecord-Clone 2.01 (i386-unknown-freebsd7.0) Copyright (C) 1995-2004 Jörg Schilling
Using libscg version 'schily-0.1'
scsibus0:
  0,0,0 0) 'SEAGATE ' 'ST39236LW      ' '0004' Disk
  0,1,0 1) 'SEAGATE ' 'ST39173W      ' '5958' Disk
  0,2,0 2) *
  0,3,0 3) 'iomega ' 'jaz 1GB        ' 'J.86' Removable Disk
  0,4,0 4) 'NEC      ' 'CD-ROM DRIVE:466' '1.26' Removable CD-ROM
  0,5,0 5) *
  0,6,0 6) *
  0,7,0 7) *
scsibus1:
  1,0,0 100) *
  1,1,0 101) *
  1,2,0 102) *
  1,3,0 103) *
  1,4,0 104) *
  1,5,0 105) 'YAMAHA ' 'CRW4260      ' '1.0q' Removable CD-ROM
  1,6,0 106) 'ARTEC  ' 'AM12S        ' '1.06' Scanner
  1,7,0 107) *
```

个列表列出了的的的的 `dev` 选项。到的 `CD burner` ,使用三个用逗号分隔的数来表示 `dev`。在个例子中, `CRW` 是 `dev=1,5,0` ,所以正的位置是 `dev=1,5,0` 。有一个很容易的方法可以指定个;看看 [cdrecord\(1\)](#) 的介绍了解有选项,控制速度和其他的选项。

### 19.6.5. 制音 CD

可以制 CD, 把 CD 上面的音数据解出一系列的文件, 再把些文件写到一空白 CD 上。个程序于 ATAPI 和 SCSI 器来有些微的不同。

#### Procedure: SCSI 器

1. 使用 `cdda2wav` 来解音。

```
% cdda2wav -vall -D2,0 -B -Owav
```

2. 使用 `cdrecord` 来写 .wav 文件。

```
% cdrecord -v dev=2,0 -dao -useinfo *.wav
```

保 2,0 被当地置了, 具体方法在 [cdrecord](#) 中有所描述。



## Procedure: ATAPI 驱动器



借助于 [ATAPI/CAM 模型](#)，[cdda2wav](#) 同样也能在 ATAPI 驱动器上使用。此工具比起下面推荐的方法通常是个更好的选择(纠错，字节序等，等等)。

1. ATAPI CD 驱动器用 `/dev/acdtnn` 表示，其中 `d` 是驱动器号，`nn` 是通道号，由两位小数位组成，省略前零。所以第一个光盘上的第一个通道就是 `/dev/acd0t01`，第二个就是 `/dev/acd0t02`，第三个就是 `/dev/acd0t03`，等等。

你必须要在 `/dev` 中出了驱动器的文件。如果系统有某些目录缺失，系统制作者重新介绍：

```
# dd if=/dev/acd0 of=/dev/null count=1
```

2. 使用 [dd\(1\)](#) 解每个通道。当解文件的时也必须使用一个特殊的块大小。

```
# dd if=/dev/acd0t01 of=track1.cdr bs=2352
# dd if=/dev/acd0t02 of=track2.cdr bs=2352
...
```

3. 使用 [burncd](#) 把解出的文件刻到光盘上。你必须指定 哪些文件是音频文件，而 [burncd](#) 会在刻盘完成后束光。

```
# burncd -f /dev/acd0 audio track1.cdr track2.cdr ... fixate
```

### 19.6.6. 制作数据 CD

你可以把数据 CD 制作成一个与之等价的映像文件，可以使用 [mkisofs\(8\)](#) 创建文件，或使用它来制作任何数据 CD。这里给出的例子假定你的 CDROM 是 `acd0`，将其替换你使用的 CDROM。

```
# dd if=/dev/acd0 of=file.iso bs=2048
```

现在有一个映像文件了，可以像上面描述的那样把它刻成 CD。

### 19.6.7. 使用数据 CD

现在已创建了一个标准的数据 CDROM，或你想要挂起来取上面的。默认情况下，[mount\(8\)](#) 假定文件系统是 `ufs` 型的。如果运行下面的命令：

```
# mount /dev/cd0 /mnt
```

会得到一条 `Incorrect super block` 的信息，没有挂载成功。CDROM 不是 `UFS` 文件系统，所以挂载它是不可行的。需要告诉 [mount\(8\)](#) 文件系统是 `ISO9660` 型的，就可以了。只需要指定 [mount\(8\)](#) 的 `-t`

cd9660 的。例如，如果想要挂载 CDROM 的，/dev/cd0 到 /mnt 目录，需要运行：

```
# mount -t cd9660 /dev/cd0 /mnt
```

注意设备名（在这个例子中是 /dev/cd0）可能有所不同，取决于 CDROM 使用的接口。另外，`-t cd9660` 等同于运行 [mount\\_cd9660\(8\)](#)。上面的例子可以简短：

```
# mount_cd9660 /dev/cd0 /mnt
```

用这种方法基本可以使用任何到的数据 CDROM。然而某些有 ISO 9660 扩展的光盘可能会行古怪。例如，joliet 光盘用 8 个字的 unicode 字符存所有的文件名。FreeBSD 内核并不使用 Unicode，但 FreeBSD CD9660 可以将 Unicode 字符自内核可以的形式。如果有些非英文字符示号，就要使用 `-C` 来指定字符集了。欲了解一的情况，参手册 [mount\\_cd9660\(8\)](#)。



如果希望通 `-C` 来行字符集，内核会需要加载 `cd9660_iconv.ko` 模块。工作可以通在 `loader.conf` 中加入下列配置：

```
cd9660_iconv_load="YES"
```

并重新计算机来完成，除此之外，也可以通 [kldload\(8\)](#) 来手加载。

有时候，当挂载 CDROM 的时候，会得到一条 `Device not configured` 的信息。通常表明 CDROM 的托里没有光盘，或者器在不可用。需要几秒钟等待 CDROM 器辨已接到反的信息，耐心等待。

有时候，SCSI CDROM 可能会不到，因没有足的 来答的 reset 信号。如果有一个 SCSI CDROM 将下面的添加到的内核配置文件并 [重建的内核](#)。

```
options SCSI_DELAY=15000
```

个告的 SCSI 器停 15 秒，的 CDROM 器足的机会来答 reset 信号。

### 19.6.8. 刻原始数据 CD

可以一个文件目录刻到 CD 上而不用建 ISO 9660 文件系统。有些人做是了的目的。个行的比刻一个准 CD 速度要快得多：

```
# burncd -f /dev/acd1 -s 12 data archive.tar.gz fixate
```

要重新回刻到 CD 上的数据，必从原始点取数据：

```
# tar xzvf /dev/acd1
```

它不能像挂一个通常的 CDROM 一样挂到光驱。它的 CDROM 也不能在除了 FreeBSD 之外的任何操作系统上运行。如果想要可以挂 CD，或者和另一个操作系统共享数据，必须像上面描述的那样使用 [mkisofs\(8\)](#)。

### 19.6.9. 使用 ATAPI/CAM 设备

它允许 ATAPI 设备(CD-ROM, CD-RW, DVD 设备等...)通过 SCSI 子系统，允许使用像 [sysutils/cdrdao](#) 或者 [cdrecord\(1\)](#) 的程序。

要使用它，需要把下面一行添加到 `/boot/loader.conf` 文件中：

```
atapicam_load="YES"
```

接下来，重新引导计算机。

如果希望将 [atapicam\(4\)](#) 以静默的形式加入内核，需要在内核配置文件中加入一行：

```
device atapicam
```

此外需要在内核配置文件中加入：



```
device ata
device scbus
device cd
device pass
```

有些设备已经有了。然后，重新引导并安装新内核，并重新引导计算机。

在引导过程中，刻录机将会出现在内核的提示信息中，就像这样：

```
acd0: CD-RW <MATSHITA CD-RW/DVD-ROM UJDA740> at ata1-master PI04
cd0 at ata1 bus 0 target 0 lun 0
cd0: <MATSHITA CDRW/DVD UJDA740 1.00> Removable CD-ROM SCSI-0 device
cd0: 16.000MB/s transfers
cd0: Attempt to query device size failed: NOT READY, Medium not present - tray closed
```

设备现在可以通过 `/dev/cd0` 命名了，例如要挂 CD-ROM 到 `/mnt`，只需要输入下面的命令：

```
# mount -t cd9660 /dev/cd0 /mnt
```

作为 `root`，可以运行下面的命令来得到刻录机的 SCSI 地址：

```
# camcontrol devlist
<MATSHITA CDRW/DVD UJDA740 1.00> at scbus1 target 0 lun 0 (pass0,cd0)
```

00 1,0,0 就是 SCSI 地址了，可以被 [cdrecord\(1\)](#) 和其他的 SCSI 程序使用。

有关 ATAPI/CAM 和 SCSI 系的更多信息，可以参 [atapicam\(4\)](#) 和 [cam\(4\)](#) 手册。

## 19.7. 创建和使用光学介质(DVD)

### 19.7.1. 介绍

和 CD 相比，DVD 是下一代光学存储技术。DVD 可以容纳比任何 CD 更多的数据，已成为现今出版的基准。

我们称作可 DVD 的有五物理格式：

- DVD-R：是第一可用的 DVD 可格式。DVD-R 标准由 [DVD Forum](#) 定义。格式是一次可写的。
- DVD-RW：是 DVD-R 标准的可覆写版本。一个 DVD-RW 可以被覆写大约 1000 次。
- DVD-RAM：也是一被 DVD Forum 所支持的可覆写格式。DVD-RAM 可以被看作一个可移动硬盘。然而，介绍和大部分 DVD-ROM 驱动器以及 DVD-Video 播放器不兼容；只有少数 DVD 刻机支持 DVD-RAM。参 [使用 DVD-RAM](#) 以了解于如何使用 DVD-RAM 的详情。
- DVD+RW：是一个由 [DVD+RW Alliance](#) 定义的可覆写格式。一个 DVD+RW 可以被覆写大约 1000 次。
- DVD+R：格式是 DVD+RW 格式的一次可写。

一个可 DVD 可以存 4,700,000,000 字节，相当于 4.38 GB 或者 4485 MB (1 千字等于 1024 字节)。



必须明一下物理介质与应用程序的分。例如 DVD-Video 是一个特殊的文件系统，可以被覆写到任何可的 DVD 物理介质上：DVD-R、DVD+R、DVD-RW 等等。在介绍类型之前，一定要刻机和 DVD-Video 播放器（一个独立的播放器或者计算机上的 DVD-ROM 驱动器）是和介绍兼容的。

### 19.7.2. 配置

[growisofs\(1\)](#) 将被用来施 DVD 刻。这个命令是 `dvd+rw-tools` 工具集 (`sysutils/dvd+rw-tools`) 的一部分。`dvd+rw-tools` 支持所有的 DVD 介绍型。

些工具将使用 SCSI 子系来，因此 [ATAPI/CAM 支持](#) 必须加入内核。如果的刻机采用 USB 接口不需要做，参 [USB 存储](#) 来了解 USB 配置的细节。

此外，需要 ATAPI 的 DMA 支持。一工作可以通在 `/boot/loader.conf` 文件中加入下面的行来完成：

```
hw.ata.atapi_dma="1"
```

使用 `dvd+rw-tools` 之前参 [dvd+rw-tools 硬件兼容性列表](#) 是否有与的 DVD 刻机有的信息。



如果想要一个形化的用界面，看一看 `K3b` (`sysutils/k3b`)，它提供了 [growisofs\(1\)](#) 的一个友好界面和许多其他刻工具。

### 19.7.3. 刻数据 DVD

`growisofs(1)` 命令是 `mkisofs` 的前端，它会用 `mkisofs(8)` 来建文件系统布局，完成到 DVD 上的刻。这意味着不需要在刻之前建数据映像。

要把 `/path/to/data` 目录的数据刻到 DVD+R 或者 DVD-R 上面，使用下面的命令：

```
# growisofs -dvd-compat -Z /dev/cd0 -J -R /path/to/data
```

`-J -R` 选项 `mkisofs(8)` 用于文件系统建 (表示建有 joliet 和 Rock Ridge 展的 ISO 9660 文件系统)，参考 `mkisofs(8)` 手册了解更多。

`-Z` 用来在任何情况下初始刻：不管多会与否。DVD 的 `/dev/cd0`，必依照的配置做出改。`-dvd-compat` 参数会束光，光成不可附加的。会提供更多的和 DVD-ROM 器的介兼容性。

也可以刻成一个 pre-mastered 映像，例如一个映像文件 `imagefile.iso`，我可以行：

```
# growisofs -dvd-compat -Z /dev/cd0=imagefile.iso
```

刻的速度可以被到并自行整，根据介和器的使用情况。如果想制改速度，可以使用 `-speed=` 参数。更多的信息，看 `growisofs(1)` 手册。

如果需要在刻的中添加超 4.38GB 的文件，就必须使用 `mkisofs(8)` 或其他相工具 (例如 `growisofs(1)`) 的 `-udf -iso-level 3` 参数来建 UDF/ISO-9660 混合文件系统。只有在建 ISO 映像文件或直接在光上写数据才需要。以这种方式建的光必通 `mount_udf(8)` 工具以 UDF 文件系统挂，因此只有操作系统支持 UDF 才可以做，否则光上的文件数据可能会无法正出。

要建的 ISO 文件：

```
% mkisofs -R -J -udf -iso-level 3 -o imagefile.iso /path/to/data
```



直接将文件刻到光上：

```
# growisofs -dvd-compat -udf -iso-level 3 -Z /dev/cd0 -J -R /path/to/data
```

假如只是使用包含巨型文件的 ISO 映像文件，就不需要在行 `growisofs(1)` 来将映像文件刻成光指定任何外的了。

外，在映像文件中加或直接刻巨型文件，需要注意使用最新的 `sysutils/cdrtools` (包含了 `mkisofs(8)`)，因旧版并不提供巨型文件支持。如果遇到，也可以一下版本的件包，例如 `sysutils/cdrtools-devel` 并参 `mkisofs(8)` 手册。

### 19.7.4. 刻 DVD-Video

DVD-Video 是一个特殊的基于 ISO 9660 和 micro-UDF (M-UDF) 的文件系统。DVD-Video 也呈现了一个特殊的数据格式，这就是为什么需要一个特殊的程序像 [multimedia/dvdauthor](#) 来制作 DVD 的原因。

如果你已经有了 DVD-Video 文件系统的映像，就可以以同样的方式制作一个映像，可以参看前面章节的例子。如果你想制作 DVD 并想放在特定的目录中，如在目录 `/path/to/video` 中，可以使用下面的命令来刻 DVD-Video：

```
# growisofs -Z /dev/cd0 -dvd-video /path/to/video
```

`-dvd-video` 选项将选项 `mkisofs(8)` 并指示它建立一个 DVD-Video 文件系统布局。除此之外，`-dvd-video` 选项也包含了 `-dvd-compat growisofs(1)` 选项。

### 19.7.5. 使用 DVD+RW

不像 CD-RW，一个空白的 DVD+RW 在一次使用前必须先格式化。`growisofs(1)` 程序将会自动地执行适当的处理，这是 *recommended* 的方式。也可以使用 `dvd+rw-format` 来对 DVD+RW 进行格式化：

```
# dvd+rw-format /dev/cd0
```

只需要执行一次的操作，除非只有空白的 DVD+RW 介质才需要格式化。可以以前面章节同样的方式来刻 DVD+RW。

如果你想刻新的数据（刻一个新的完整的文件系统而不是追加一些数据）到 DVD+RW，不必再将其格式化成空白，只需要直接覆盖掉以前的即可。（执行一个新的初始化），像：

```
# growisofs -Z /dev/cd0 -J -R /path/to/newdata
```

DVD+RW 格式化程序向以前的追加数据提供了可能性。这个操作有一个新的会和一个已存在的会合并而成。它不需要多个写会话，`growisofs(1)` 将在介质上添加 ISO 9660 文件系统。

例如，我想追加一些数据到我以前的 DVD+RW 上，我可以使用下面的命令：

```
# growisofs -M /dev/cd0 -J -R /path/to/nextdata
```

在以后的写操作，使用与最初的刻会话相同的 `mkisofs(8)` 选项。



如果你想得与 DVD-ROM 有更好的兼容性，可以使用 `-dvd-compat` 选项。在 DVD+RW 情况下，这样做并不妨碍添加数据。

如果出于某些原因真的想要空白介质，可以执行下面的命令：

```
# growisofs -Z /dev/cd0=/dev/zero
```

### 19.7.6. 使用 DVD-RW

DVD-RW 接受两种光碟格式：顺序写入和受限式覆写。默认的 DVD-RW 是顺序写入格式。

空白的 DVD-RW 能够直接刻行刻而不需要格式化操作，然而非空的顺序写入格式的 DVD-RW 需要格式化才能写入新的初始区段。

要格式化一个 DVD-RW 顺序写入模式，执行：

```
# dvd+rw-format -blank=full /dev/cd0
```

一次完全的格式化 (**-blank=full**) 在 1x 倍速的介质上将会花费大约 1 个小时。快速格式化可以使用 **-blank** 选项来执行，如果 DVD-RW 要以 Disk-At-Once (DAO) 模式刻录的。要以 DAO 模式刻录 DVD-RW，使用命令：



```
# growisofs -use-the-force-luke=dao -Z /dev/cd0=imagefile.iso
```

**-use-the-force-luke=dao** 选项不是必需的，因为 [growisofs\(1\)](#) 选项最低限度的选项 (快速格式化) 介质并执行 DAO 写入。

事实上对于任何 DVD-RW 都使用受限式覆写模式，该格式比默认的顺序写入更加灵活。

在一个顺序 DVD-RW 上写入数据，使用和其他 DVD 格式相同的指令：

```
# growisofs -Z /dev/cd0 -J -R /path/to/data
```

如果想在以前的刻录上附加数据，必须使用 [growisofs\(1\)](#) 的 **-M** 选项。然而，如果在一个顺序写入模式的 DVD-RW 上附加数据，将会在介质上建立一个新的区段，结果就是一个多区段光碟。

受限式覆写格式的 DVD-RW 在新的初始化区段前不需要格式化，只是要用 **-Z** 选项覆写光碟，和 DVD+RW 的情形是相似的。也可以用和 DVD+RW 同样方式的 **-M** 选项把保存的 ISO 9660 文件系统写入光碟。结果会是一个区段 DVD。

要把 DVD-RW 置于受限式覆写格式，必须使用下面的命令：

```
# dvd+rw-format /dev/cd0
```

更改回顺序写入模式使用：

```
# dvd+rw-format -blank=full /dev/cd0
```

### 19.7.7. 多区段

几乎没有一个 DVD-ROM 驱动器支持多区段 DVD，它在大多数时候都只取第一个区段。顺序写入格式的 DVD+R、DVD-R 和 DVD-RW 可以支持多区段，DVD+RW 和 DVD-RW 受限式覆写格式不存在多区段的概念。

在 DVD+R、DVD-R 或者 DVD-RW 的顺序写入格式下，一次初始化（未写）区段之后使用下面的命令，将会在光盘上添加一个新的区段：

```
# growisofs -M /dev/cd0 -J -R /path/to/nextdata
```

DVD+RW 或者 DVD-RW 在受限式覆写模式下使用这条命令，会合并新区段到存在的区段中来附加数据。结果就是一块区段光盘。这是在光盘上用于在最初的写操作之后添加数据的方式。



光盘上的一些空间用于区段之间的开始与结束。因此，需要用大量的数据添加区段来优化光盘空间。对于 DVD+R 来区段的数量限制是 154，对于 DVD-R 来最大是 2000，对于双 DVD+R 来是 127。

### 19.7.8. 更多的信息

要获得更多的关于 DVD 的信息 `dvd+rw-mediainfo /dev/cd0` 命令可以运行来获得更多的信息。

更多的关于 `dvd+rw-tools` 的信息可以在 [growisofs\(1\)](#) 手册得到，在 [dvd+rw-tools web site](#) 和 [cdwrite mailing list](#) 中也可得到。



`dvd+rw-mediainfo` 命令的输出结果，以及介绍的输出会被用来做报告。如果没有输出，就很可能解决。

### 19.7.9. 使用 DVD-RAM

#### 19.7.9.1. 配置

DVD-RAM 刻录机通常使用 SCSI 或 ATAPI 接口之一。对于 ATAPI 接口，DMA 模式必须手工启用。这一工作可以通过在 `/boot/loader.conf` 文件中添加下述配置来完成：

```
hw.ata.atapi_dma="1"
```

#### 19.7.9.2. 初始化接口

如本章前面的接口所言，DVD-RAM 可以像一块移动硬盘。与任何其它型号的移动硬盘类似，首次使用它之前，首先“初始化”DVD-RAM。在下面的例子中，我们将在全部空间上使用标准的 UFS2 文件系统：

```
# dd if=/dev/zero of=/dev/acd0 bs=2k count=1
# bsdlabel -Bw acd0
# newfs /dev/acd0
```



根据情况将 `acd0` 改其所使用的名。

### 19.7.9.3. 使用介

一旦在 DVD-RAM 上完成了前面的操作，就可以像普通的硬一挂接它了：

```
# mount /dev/acd0 /mnt
```

然后就可以正常地 DVD-RAM 行写了。

## 19.8. 建和使用

把数据存在上也是十分有用的。例如，在没有其它可的存介，或只需将少量数据到其他计算机。

一章将介在 FreeBSD 上使用。在使用 DOS 3.5 英寸首要要及的就是格式化，但其概念与其它的格式化似。

### 19.8.1. 格式化

#### 19.8.1.1. 通

的通像其它一是在 `/dev` 中的条目来的。直接，只需地使用 `/dev/fdN` 来表示。

#### 19.8.1.2. 格式化

一在使用前必先被低格式化。通常主已做了，但格式化是介完整性的一好方法。尽管有可能会取大量（或少量）的硬大小，但大部分磁都能被格式化 1440kB。

低格式化需要使用 `fdformat(1)` 命令。个程序需要名作参数。

要留意一切信息，些信息能助定磁的好与坏。

##### 19.8.1.2.1. 的格式化

使用 `/dev/fdN` 来格式化。入一新的 3.5 英寸的在的：

```
# /usr/sbin/fdformat -f 1440 /dev/fd0
```

### 19.8.2. 磁

低格式化后，需要它分配一个。个磁以后会被去，但系需要使用它来定磁的尺寸。

新的磁将会接管整个磁，会包括所有合的于的 `geometry` 信息。磁的 `geometry` 列在 `/etc/disktab` 中。

在可以用下面的方法来使用 `bsdlabeled(8)` 了：

```
# /sbin/bsdlabel -B -w /dev/fd0 fd1440
```

### 19.8.3. 文件系统

在磁盘进行高格式化的。它会在它上面安置一个新的文件系统，可使 FreeBSD 来对它读写。在创建完新的文件系统后，磁道将被消，所以如果你想重新格式化磁道，你必须重新建磁道。

新的文件系统可以 UFS 或 FAT。FAT 是通常情况下比 UFS 好的。

要制作新的文件系统在磁道上，可以使用下面的命令：

```
# /sbin/newfs_msdos /dev/fd0
```

在磁道已可以读写和使用。

### 19.8.4. 使用

要使用，需要先使用 `mount_msdosfs(8)` 挂接它。除此之外，也可以使用在 `ports` 套件中的 `emulators/mttools` 程序。

## 19.9. 用磁头机

主流的磁头机有 4mm, 8mm, QIC, mini-cartridge 和 DLT。

### 19.9.1. 4mm (DDS: Digital Data Storage)

4mm 磁头机正在逐渐取代 QIC 成为工作站数据的首选。在 Conner 收购了 QIC 磁头机领域的先的制造商 Archive 之后不久，即再生磁头机，使得它一得愈加明显。4mm 的磁头器更加小和安静，但于数据保存的可性仍不及 8mm 磁头器。它要比 8mm 的便宜和小得多 (3 x 2 x 0.5 inches, 76 x 51 x 12 mm)。和 8mm 的一，写的寿命都不，因为它同使用螺旋式的方式来写。

些数据的速度在 ~150 kB/s 到 ~500 kB/s 之间，存储空间从 1.3 GB 到 2.0 GB 之间，硬件可使空间加倍。磁头单元可以有 6 台磁头机，120 个磁头匣，以自切的方式使用同一个磁头，磁头的容量可 240 GB。

DDS-3 标准在支持的磁头机容量最高可到 12 GB (或 24 GB)。

4mm 和 8mm 同都使用螺旋式写的方式，所有螺旋式写的点及缺点，都可以在 4mm 和 8mm 磁头机上看到。

磁头在 2,000 次的使用或 100 次的全部后，就退休了。

### 19.9.2. 8mm (Exabyte)

8mm 磁头机是最常见的 SCSI 磁头机，也是磁头交的最佳。几乎个工作站都有一台 2 GB 8mm 磁头机。8mm 磁头机可信度高、方便、安静。匣小 (4.8 x 3.3 x 0.6 inches; 122 x 84 x 15 mm)而且不。8mm 磁头机的下是一个短短的写，而写的寿命取决于磁头写的情况，相高速情况。

数据传输速度在 250 kB/s 到 500 kB/s 之间，可存储的空间从 300 MB 到 7 GB，硬件可使空间加倍。磁头单元可以有 6 台磁头机，120 个磁头匣，以自切的方式使用同一个磁头，磁头的容量可达 840+ GB。

Exabyte "Mammoth" 模型支持 12 GB 的容量在一个磁头上(读后可达 24 GB)相当于普通磁头的二倍。

数据是使用螺旋式写入的方式写在磁头上的，写入和磁头相差 6 度，磁头以 270 度绕着，并抵住写入，原地旋转，使得磁头具有高密度，从一端到另一端并可使磁道紧密地分布。

### 19.9.3. QIC

QIC-150 磁头和磁头机可能是最常用的磁头和介绍了。QIC 磁头机是最便宜的 "正" 磁头。它的缺点在于介绍的价格高。QIC 磁头要比 8mm 或 4mm 磁头，1 GB 的数据存储价格可能最高高出 5 倍。但是，如果需求能半打磁头所满足的，那么 QIC 可能是明智之选。QIC 是最常用的磁头机。每个站点都会有某密度的 QIC。它有是一麻，QIC 有很多在外上相似(有其一)，但是密度不同的磁头。QIC 磁头机噪音很大。它在地址以及写入都会发出声音。QIC 磁头的规格是 6 x 4 x 0.7 英寸(152 x 102 x 17 毫米)。

数据传输的速度介于 150 kB/s 到 500 kB/s 之间，可存储的空间从 40 MB 到 15 GB。新的 QIC 磁头机具有硬件的功能。QIC 的使用率愈来愈低，被 DAT 所取代。

数据以磁道的方式写在磁头上，磁道数及磁道的宽度会根据容量而有所不同。通常新的磁头机具有的向后兼容的读取功能(通常也具有写入的功能)。关于数据的安全性，QIC 具有不错的价格。

磁头机在约 5,000 次的使用后，就退休了。

### 19.9.4. DLT

在第一章列出的磁头机中 DLT 具有最快的数据传输率。1/2" (12.5mm) 的磁头包含在约的磁头匣(4 x 4 x 1 inches; 100 x 100 x 25 mm)中。磁头匣的一边是一个旋转匣道，通过匣道的配合，可以磁头卷。磁头匣内只有一个，而本章中所提到的其他磁头匣都是有二个的(9磁道磁头机例外)。

数据传输的速度 1.5 MB/s，是 4mm, 8mm, 或 QIC 磁头机的三倍。可存储的空间从 10 GB 到 20 GB，具有磁头机数据。磁头机数据单元可以有 1 to 20 台磁头机，5 到 900 个磁头匣，磁头机数据的容量可达 50 GB 到 9 TB。

如果要读的，DLT 型 IV 格式的磁头机最高可支持 70 GB 的存储空间。

数据存储在平行于磁头行方向的磁道上(就像 QIC 磁头)，一次写入一个磁道。写入的寿命相当，当磁头停止前，磁头与写入之间没有相碰。

### 19.9.5. AIT

AIT 是 Sony 的一种新格式，一个磁头最高可以存储 50 GB。磁头机使用内存芯片来保存磁头上的索引内容。一个索引能被磁头机快速来搜索磁头机上文件所在的位置，而不像其他的磁头机需要花几分秒的时间才能到文件。像 SAMS:Alexandria 的附件：能操作四十或者更多的 AIT 磁头，直接使用内存芯片来进行通信把内容显示在屏幕上，以决定把什么文件写到哪个磁头上，加载和恢复数据。

像它的成本大概在 \$20,000 美元左右，零售市可能贵一点。

### 19.9.6. 第一次使用新的磁机

当在一个完全空白的磁带上写入数据，会得到以下面的信息：

```
sa0(ncr1:4:0): NOT READY asc:4,1
sa0(ncr1:4:0): Logical unit is in process of becoming ready
```

信息指出磁没有块号 (block 号 0)。在 QIC-525 之后的所有 QIC 磁，都采用 QIC-525 标准，必须写入一个 Identifier Block。对于磁，有以下解决的方法：

- 用 `mt fsf 1` 可以磁机磁写入 Identifier Block。
- 使用面板上的按钮磁。

再入一次，并存 `dump` 数据到磁上。

`dump` 将回 `DUMP: End of tape detected`，然后会得到磁的信息：`HARDWARE FAILURE info:280 asc:80,96`。

用 `mt rewind` 来倒磁。

磁操作的后操作就完成了。

## 19.10. 用磁

### 19.10.1. 能用磁来数据

磁通常是用来磁的磁中不太合的磁：

- 磁不太可靠，特别是期使用。
- 磁和恢都很慢
- 它只有非常有限的存容量。

然而，如果没有其它的磁数据的方法，那磁比没有磁要好。

如果必须使用磁的，必须保片的量。在办公室中使用已有多年了。最好使用一些名牌厂商的品以保量。

### 19.10.2. 如何数据到磁

最好的磁数据到磁的方法是使用 `tar(1)` 程序加上 `-M` 磁，它可以允磁数据到多磁上。

要磁当前目磁中所有的文件可以使用磁个命令 (需要有 `root` 限)：

```
# tar Mcvf /dev/fd0 *
```

当第一磁的时候，`tar(1)` 会指示磁入下一磁，入第二磁之后就按回。

```
Prepare volume 2 for /dev/fd0 and hit return:
```

它可能需要重复很多次，直到某些文件完成为止。

### 19.10.3. 可以做什么

不幸的是，`tar(1)` 在多个卷文件操作是不允许使用 `-z` 的。当然，可以用 `gzip(1)` 压缩所有的文件，把它打包到磁盘中，以后在用 `gunzip(1)` 解包。

### 19.10.4. 如何恢复

要恢复所有文件：

```
# tar Mxvf /dev/fd0
```

有另一种方法来恢复磁盘中的单个文件。首先，就要用第一块磁盘：

```
# tar Mxvf /dev/fd0 filename
```

`tar(1)` 程序会提示输入后面的磁盘，直到它达到所需要的文件。

如果你知道哪个文件在哪个磁盘上，你就可以输入那个磁盘，然后使用上同同的命令。

如果

磁盘上的第一个文件与前面的文件是不同的，那 `tar(1)` 命令会警告它无法恢复，即使你不要求它这样做。

## 19.11. 备份策略

备份策略的第一要义是以下诸条皆已考虑到：

- 磁盘故障
- 文件的意外删除
- 随机的文件丢失
- 机器完全损坏 (例如火灾)，包括破坏全部在磁盘上。

上述的每个问题采用完全不同的技术来解决是完全可行的。除了只包含少量几乎没有价值数据的个人系统之外，一般来说很少有一种技术能同时兼顾前面所有的需要。

可以采用的技术包括：

- 将整个系统的数据并行存盘，并写到永久性的介质上。这种方法能提供前面所有问题的保护，但通常很慢，而且恢复会比较麻烦。可以将系统置于近后备或在后备的状态，然而恢复文件仍然是一个问题，特别是对于没有特殊的那些用户而言。
- 文件系统快照。这项技术上只无意中删除文件一种情况有用，但在其他情况下它会提供非常大的帮助，而且迅速，操作容易。
- 直接控制整个文件系统或磁盘 (例如周期性地对整个机器做 `rsync(1)`)。通常用于在网络上的单一需求最

用。要磁故障提供更通用的保，通常方法要于 RAID。于恢无意中除的文件来，方法基本上与 UFS 快照属于同一，使用一个取决于的喜好。

- RAID。它能最大限度地少磁故障致的停机。其代价是需要理更繁的磁故障（因磁的数量加了），尽管故障不再需要作非常急的事来理。
- 文件的指。 `mtree(8)` 工具于操作非常有用。尽管并不是一的技，但它能保有机会注意到那些需要求助于的事情。于非常重要，而且有地加以。

很容易列更多的技，它中有多上是前面所列出的方法的。特的需求通常会需要采用特的技（例如，在行的数据，往往需要数据件提供某方法来完成中）来足。最重要的事情是，一定要了解需要将数据保起来免受何，以及生如何理。

## 19.12. 程序

有三个主要的程序 `dump(8)`、`tar(1)` 和 `cpio(1)`。

### 19.12.1. Dump 和 Restore

`dump` 和 `restore` 是 UNIX® 的程序。它以 block 而不是以文件位来数据、接或目。`dump` 的是上的整个文件系，不能只一个文件系的部分或是用到个以上文件系的目。与其他件不同的是，`dump` 不会写文件和目到磁机，而是写入包含文件和目的原始数据。当需要恢数据的时候，`restore` 默在 `/tmp/` 下保存数据 - 如果正在操作的恢只有比小的 `/tmp` 的，可能需要把境量 `TMPDIR` 置到一个有更多空的目，使得此程更容易成功。



如果在的 `root` 目使用 `dump`，将不需要 `/home`、`/usr` 或其他目，因些是典型的其他文件系或符号接到那些文件系的加。

`dump` 是最早出于 AT&T UNIX 的 Version 6 (1975)。默的参数用于 9-track 磁(6250 bpi)，所以如果要用高密度的磁（最高可 62,182 ftpi），就不能用默的参数，而要外指定参数。些默必在命令行被修改以更好地利用当前磁机的功能。

`rdump` 和 `rrestore` 可以通网在一台算机的磁机上数据。个程序都是依 `rcmd(3)` 和 `ruserok(3)` 来程的磁机。因此，行用的用必要有程主机的 `.rhosts` 。`rdump` 和 `rrestore` 的参数必用于程主机例如，当从 FreeBSD 到一台 SUN 工作站 `knomodo` 去使用磁机，使用：

```
# /sbin/rdump 0dsbfu 54000 13000 126 komodo:/dev/nsa8 /dev/da0a 2>&1
```

要注意的是：必在使用 `.rhosts` 的安全情况。

也可以通使用 `ssh` 用一个更安全的方式来使用 `dump` 和 `restore`。

例 24. 通 `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
```

或使用 `dump` 的 built-in 方法，`rsync` 使用 `rsync`：

例 25. 通过 `ssh` 使用 `rsync` 使用 `dump`

```
# RSH=/usr/bin/ssh /sbin/dump -0uan -f
targetuser@targetmachine.example.com:/dev/sa0 /usr
```

### 19.12.2. tar

`tar(1)` 也是在第 6 版 AT&T UNIX（大约是 1975 前后）出的。`tar` 对文件系统直接操作；其作用是把文件和目录写入磁。 `tar` 并不支持 `cpio(1)` 所提供的全部功能，但也不需要 `cpio` 所需要使用的 `cpio` 的命令行管道。

要 `tar` 到接在名 `komodo` 的 Sun 机器上的 Exabyte 磁机，可以使用：

```
# tar cf - . | rsh komodo dd of=tape-device obs=20b
```

如果担心通过网络会有安全问题，应当使用 `ssh`，而不是 `rsh`。

### 19.12.3. cpio

`cpio(1)` 是 UNIX® 最早用来作文件交换的磁机程序。它有行字交换的，可以用几种不同的格式写入，并且可以将数据用管道到其他程序。`cpio` 没法自目录内的文件列表，必须通过标准输入 `stdin` 来指定。

`cpio` 不支持网络的方式。可以使用 pipeline 和 `rsh` 来送数据到磁机。

```
# for f in directory_list; do
find $f >> backup.list
done
# cpio -v -o --format=newc < backup.list | ssh user@host "cat > backup_device"
```

里的 `directory_list` 是要备份的目录列表，`user@host` 合了将要备份的用户名和主机名，`backup_device` 是写入的（如 `/dev/nsa0`）。

### 19.12.4. pax

`pax(1)` 是符合 IEEE/POSIX® 标准的程序。多年来各种不同版本的 `tar` 和 `cpio` 有些兼容。为了防止情况，并使其标准化，POSIX® 出了套新的工具程序。`pax` 可以写各种 `cpio` 和 `tar` 的格式，并可以自己加新的格式。它的命令集比 `tar` 更接近 `cpio`。

### 19.12.5. Amanda

Amanda (Advanced Maryland Network Disk Archiver) 并非一的程序，而是一个客户机/服务器模式的系统。一台 Amanda 服务器可以任意数量行 Amanda 的客户机或是将上 Amanda 服务器的

计算机上的数据备份到一台磁带上。一个常见的缺点是，数据写入磁带的速度将超过取行数据的速度，而 Amanda 解决了这个问题。它使用一个 "holding disk" 来同时备份几个文件系统。Amanda 建立 "archive sets" 的一个磁带，用来备份在 Amanda 的配置文件中所列出的完整的文件系统。

Amanda 配置文件提供完整的备份控制及 Amanda 生成的网络备份。Amanda 可以使用上述任何一个备份程序来向磁带写入数据。Amanda 可以从 port 或 package 取得，它并非系统默认安装的。

### 19.12.6. Do Nothing 备份策略

"Do nothing" 不是一个程序，而是被广泛使用的备份策略。不需要计算，不需要备份的列表，全部都不需要。如果备份的数据发生了什么问题，忽略它！

如果备份的数据不得不做一些事，那么 "Do nothing" 将是最好的备份程序。要注意的是，UNIX® 是相当好用的工具，可能在几个月内，就备份已收集了不少数据来备份相当具有价值的文件和程序。

"Do nothing" 对于像 /usr/obj 和其他可由计算机生成的文件来备份，是最好的方法。例如本手册包含有 HTML 或 PostScript® 格式的文件。有些文档格式是从 SGML 输入文件构建的。构建 HTML 或 PostScript® 格式的文件的速度就没有必要了。只要经常备份 SGML 文件就可以了。

### 19.12.7. 哪个备份程序最好？

在 [dump\(8\)](#) 期 Elizabeth D. Zwicky 测试了所有以上列出的备份程序。在各各怪怪的文件系统中，[dump](#) 是最明智的。Elizabeth 建立起各各怪、奇怪或常见的文件系统，并用各备份程序，在各文件系上备份及恢复数据。这些怪怪之怪包括：具有 holes 和一个 nulls block 的文件，文件名具有有趣字符，无法写入的文件及文件，在备份时改变文件大小，在备份时建立或删除的文件。测试结果写在：LISA V in Oct. 1991. 参看 [torture-testing Backup and Archive Programs](#).

### 19.12.8. 紧急恢复程序

#### 19.12.8.1. 在出事前

在遇到事前，只需要进行以下四个步骤：

第一，打出备份的磁带的磁头（例如：`bsdlabel da0 | lpr`），文件系统表，（`/etc/fstab`），以及所有备份信息，并将其控制。

第二，刻一个 "livefs" CDROM。这个 CDROM 包含了用于引导入 FreeBSD "livefs" 修复模式的支持，该模式允许用一行多任务，例如运行 [dump\(8\)](#)、[restore\(8\)](#)、[fdisk\(8\)](#)、[bsdlabel\(8\)](#)、[newfs\(8\)](#)、[mount\(8\)](#)，等等。Livefs CD 映像文件随 FreeBSD/i386 12.0-RELEASE 提供，可以从 <ftp://ftp.FreeBSD.org/pub/FreeBSD/releases/i386/ISO-IMAGES/12.0/FreeBSD-12.0-RELEASE-i386-livefs.iso> 获得。

第三，定期将数据备份到磁带。任何在上次备份后的改变都无法恢复。必须将磁带写保护。

第四，将第二步骤所建立的 "livefs" CDROM 及备份的磁带。写下备份，并和备份 CDROM、打印副本以及磁带放在一起。在需要恢复数据时可能正心慌意乱，而有些备份可能会帮助避免掉磁带（备份时生什么情况？例如来，本行 `tar xvf /dev/sa0` 命令，可能会不小心输入 `tar cvf /dev/sa0`，从而覆盖磁带）。

备份起，可以制作一个 "livefs" CDROM 和备份磁带。其中一备份放到其它地方，备份里的其他地方当然不是指同一办公楼的地下室，世界中心的一大批公司已学到了血的教训。保存备份



的位置与的计算机和磁器越好。

### 19.12.8.2. 出后

是：的硬件是否幸免于？由于已做好了定期的工作，因此并不需要担心件的。

如果硬件已坏，些部分在使计算机之前掉。

如果硬件能用，将 "livefs" CDROM 入 CDROM 器并引系。将看到最初安装系的菜。正的国家之后， Fixit — Repair mode with CDROM/DVD/floppy or start a shell ，然后再 CDROM/DVD — Use the live filesystem CDROM/DVD 。可以使用 `restore` 以及其他位于 `/mnt2/rescue` 的工具。

分恢一个文件系

着 `mount`（例如：`mount /dev/da0a /mnt`）第一个磁上的 `root` 分区。如果 `bsdlabel` 已坏，需要使用 `bsdlabel` 根据先前打印存的来重新分区并分配磁。接着使用 `newfs` 重建文件系。以写方式重新挂磁的根分区 (`mount -u -o rw /mnt`)。使用的程序以及磁恢文件系数据（例如 `restore vrf /dev/sa0`）。最后卸下文件系（例如 `umount /mnt`）。于掉的其他文件系，重行前面些操作。

当的系正常后，将的数据到新的磁。任何造成数据失的都可能再次生。在花一些，也可以在下次生救一把。

## 19.13. 网、内存和以及映像文件介的虚文件系

除了在计算机上的物理磁：、CD、硬器，等等之外，FreeBSD 能一些其他的磁形式 - 虚磁。

包括，如 [网文件系 \(Network File System\)](#) 和 Coda—的网文件系、内存以及映像文件介的虚文件系。

随行的 FreeBSD 版本不同，用来建和使用以映像文件介文件系和内存文件系的工具也不尽相同。



系会使用 `devfs(5)` 来建点，用是透明的。

### 19.13.1. 以映像文件介的文件系

在 FreeBSD 系中，可以用 `mdconfig(8)` 程序来配置和用内存磁，`md(4)`。要使用 `mdconfig(8)`，就需要在内核配置文件中添加 `md(4)` 模来支持它：

```
device md
```

`mdconfig(8)` 命令支持三型的虚文件系：使用 `malloc(9)`，来分配内存文件系，内存文件系作文件或作用的交分区。一使用方式是在文件中来挂一个和 CD 像。

将一个存的映像文件作文件系挂：

例 26. 使用 `mdconfig` 挂已存在的映像文件

```
# mdconfig -a -t vnode -f diskimage -u 0
# mount /dev/md0 /mnt
```

使用 `mdconfig(8)` 来建新的映像文件:

例 27. 使用 `mdconfig` 将映像文件作文件系统挂

```
# dd if=/dev/zero of=newimage bs=1k count=5k
5120+0 records in
5120+0 records out
# mdconfig -a -t vnode -f newimage -u 0
# bsdlabel -w md0 auto
# newfs md0a
/dev/md0a: 5.0MB (10224 sectors) block size 16384, fragment size 2048
      using 4 cylinder groups of 1.25MB, 80 blks, 192 inodes.
super-block backups (for fsck -b #) at:
 160, 2720, 5280, 7840
# mount /dev/md0a /mnt
# df /mnt
Filesystem 1K-blocks Used Avail Capacity Mounted on
/dev/md0a      4710    4 4330      0%    /mnt
```

如果没有通 `-u` 指定一个号 `mdconfig(8)` 将使用 `md(4)` 它自一个未用的号。分配它的名字将被出到输出，其形式是与 `md4` 似。如果希望了解更多相关信息，参机手册 `mdconfig(8)`。

`mdconfig(8)` 功能很大，但在将映像文件作文件系统挂，仍需使用多行的命令。此 FreeBSD 也提供了一个名 `mdmfs(8)` 的工具，程序使用 `mdconfig(8)` 来配置 `md(4)`，并用 `newfs(8)` 在其上建 UFS 文件系统，然后用 `mount(8)` 来完成挂操作。例如，如果想建和挂像上面那的文件系映像，只需地行下面的：

例 28. 使用 `mdmfs` 命令配置和挂一个映像文件文件系统

```
# dd if=/dev/zero of=newimage bs=1k count=5k
5120+0 records in
5120+0 records out
# mdmfs -F newimage -s 5m md0 /mnt
# df /mnt
Filesystem 1K-blocks Used Avail Capacity Mounted on
/dev/md0      4718    4 4338      0%    /mnt
```

如果使用没有加号的 `md`，`mdmfs(8)` 将使用 `md(4)` 的自示号特性来自其一个未使用的。更的 `mdmfs(8)`，参考机手册。

### 19.13.2. 以内存作介质的文件系统

一般来，在建立以内存作介质的文件系统，使用“交换区作介质 (swap backing)”。使用交换区作介质，并不意味着内存将被无条件地换出到交换区，它只是表示将根据需要从可换出的内存池中分配内存。此外，也可以使用 [malloc\(9\)](#) 建立以内存作介质的文件系统。不过在内存不足，这种方式可能引致系统崩溃。

例 29. 用 `mdconfig` 建立新的内存介质

```
# 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
```

例 30. 使用 `mdmfs` 来新建内存介质文件系统

```
# mdmfs -s 5m md2 /mnt
# df /mnt
Filesystem 1K-blocks Used Avail Capacity Mounted on
/dev/md2      4846    2 4458      0%    /mnt
```

### 19.13.3. 从系统中移除内存介质

当不再使用内存介质，将其源放回系统。第一操作是卸下文件系统，然后使用 [mdconfig\(8\)](#) 把虚拟磁盘从系统中分，以放回源。

例如，要分并放回所有 `/dev/md4` 使用的源，使用命令：

```
# mdconfig -d -u 4
```

`mdconfig -l` 命令可以列出关于配置 [md\(4\)](#) 的信息。

## 19.14. 文件系统快照

FreeBSD 提供了一个和 [Soft Updates](#) 的新功能: 文件系统快照

快照允许建立指定文件系统的映像，并把它当作一个文件来对待。快照文件必须在文件系统正在使用建立，一个用多少个文件系统建的快照不能大于20个。活着的快照文件被放在超块中，所以它可以在系

的候一行挂接后摘掉。当一个快照不再需要，可以使用准的除。快照可以以任何序行移除，但所有使用其它的快照将有可能互相引用一些。

[rm\(1\)](#) 使用来使其的快照不可能同行移除，因

不可改的 [snapshot](#) 文件志，是由 [mksnap\\_ffs\(8\)](#) 在完成建快照文件置的。[unlink\(1\)](#) 命令是一个特例，以允除快照文件。

快照可以通过 [mount\(8\)](#) 命令建。将文件系统 /var 的快照放到 /var/snapshot/snap 可以使用下面的命令：

```
# mount -u -o snapshot /var/snapshot/snap /var
```

作，也可以使用 [mksnap\\_ffs\(8\)](#) 来建一个快照：

```
# mksnap_ffs /var /var/snapshot/snap
```

可以文件系统中的快照文件 (例如 /var)，方法是使用 [find\(1\)](#) 命令：

```
# find /var -flags snapshot
```

当快照文件被建好后，可以用于下面一些目的：

- 有些管理用文件快照来行，因快照可以被移到 CD 或磁上。
- 文件系统一致性程序 [fsck\(8\)](#) 可以用来快照文件。如果文件系统在挂接前是一致的，快照果也一定是一致的 (也就是不会做任何修改)。上正是后台 [fsck\(8\)](#) 的操作程。
- 在快照上行 [dump\(8\)](#) 程序。dump 将返回包含文件系统和快照的。 [dump\(8\)](#) 也能取快照，使用 [-L](#) 志可以首先建快照，完成 dump 映像之后再自除它。
- 用 [mount\(8\)](#) 来挂接快照作文件系统的一个的像。要 [mount\(8\)](#) 快照 /var/snapshot/snap 行：

```
# mdconfig -a -t vnode -f /var/snapshot/snap -u 4
# mount -r /dev/md4 /mnt
```

在就可以看到挂接在 /mnt 目下的 /var 文件系统的快照。一西都保存的像它建的状态。唯一例外的是更早的快照文件将表度 0 的文件。用完快照文件之后可以把它卸下，使用：

```
# umount /mnt
# mdconfig -d -u 4
```

想了解更多于 [softupdates](#) 和 文件系统快照的信息，包括技明，可以 Marshall Kirk McKusick 的 WWW 站点 <http://www.mckusick.com/>。

## 19.15. 文件系统配额

配额是操作系统的一个可选项的功能，它允许管理以文件系统为单元，限制分派给用户或进程所使用的磁盘空间大小或是使用的文件数量。它常被用于那些分区操作的系统上，对于某些系统而言，通常希望限制分派到一个用户或进程的总容量，从而可以防止某个用户占用所有可用的磁盘空间。

### 19.15.1. 配置系统来用磁盘配额

在决定使用磁盘配额前，相信磁盘配额已在内核中配置好了。只要在在内核 中配置文件中添加下面一行就行了：

```
options QUOTA
```

在默认情况下 GENERIC 内核是不会启用这个功能的，所以必须配置、重建和安装一个定制的内核。请参考 FreeBSD 内核配置 [配置FreeBSD的内核](#) 章节了解更多有关内核配置的信息。

接下来，需要在 /etc/rc.conf 中启用磁盘配额。可以通过添加下面一行来完成：

```
enable_quotas="YES"
```

为了更好的控制配额的总量，还有一个可配置的变量。通常 `quotacheck(8)` 集成在某个文件系统上的配额会被配额程序 `quotacheck(8)` 自启动。配额功能能够保存在配额数据中的信息，数据正确地反映了文件系统的数量情况。它是一个很耗资源的处理程序，它会影系统性能。如果想跳过它，可以在文件 /etc/rc.conf 加入下面一行来达到目的：

```
check_quotas="NO"
```

最后，要编辑 /etc/fstab 文件，以在一个文件系统基础上启用磁盘配额。它是启用用户配额，或同时启用用户和组配额的地方。

要在一个文件系统上启用用户配额，可以在 /etc/fstab 里添加 `userquota` 在要雇用配额文件的系统上。例如：

```
/dev/da1s2g /home ufs rw,userquota 1 2
```

同样的，要启用组配额，使用 `groupquota` 来代替 `userquota`。要同时启用用户和组配额，可以这样做：

```
/dev/da1s2g /home ufs rw,userquota,groupquota 1 2
```

默认情况下，配额文件是存放在文件系统的以 `quota.user` 和 `quota.group` 命名的根目录下。可以看看 [fstab\(5\)](#) 手册了解更多信息。尽管手册 [fstab\(5\)](#) 提到，可以配额文件指定其他的位置，但并不推荐这样做，因为不同的配额工具并不一定遵循此规则。

到这儿，可以用新内核重新安装系统。/etc/rc 将自行执行适当的命令来创建最初的配额文件，所以并不需要手动来创建任何零度的配额文件。

在通常的操作过程中，并不要求手动运行 `quotacheck(8)`、`quotaon(8)`，或 `quotaoff(8)` 命令，然而可能需要与他操作相似的手册。

### 19.15.2. 配置限制

一旦配置好了用配的系统，可以看一下它是否真的有用。可以这样做：

```
# quota -v
```

可能看到一行当前正在使用的文件系所用的磁盘配额使用情况的摘要信息。

在可以使用 `edquota(8)` 命令准用配额限制。

有几个有如何限制用或可以分配到的磁盘空间大小的。可以限制磁盘存量的配额，或文件的数量，甚至同限制者。些限制最可分：硬限制和限制。

硬性限制是一不能越的限制。一旦用到了系指定的硬性限制，他就无法在的文件系分配到更多的源。例如，如果文件系上分用的硬性限制是 500 KB，而在已用掉了 490 KB，那这个用最多能再分配 10 KB 的空间。言之，如果再分配 11 KB，会失。

而与此相反，性限制在一段内是允越的。段也称期限，其默认是一周。如果一个用延得太的，限制将会成硬限制，而分配磁盘空间的操作将被拒。当用占用的空间回到性限制以下，期限将重新始算。

下面是一个行 `edquota(8)` 看到的例子。当 `edquota(8)` 命令被用，会被移 `EDITOR` 境量指派的器中，允配额限制。如果境量没有置，默认在 `vi` 器上行。

```
# 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)
```

在一个用了磁盘配的文件系上，通常会看到行。一行是 `block` 限制，一行是 `inode` 限制。地改要修改的配额限制的。例如，提高个用限制的数到 500，硬限制到 600：

```
/usr: kbytes in use: 65, limits (soft = 50, hard = 75)
```

to:

```
/usr: kbytes in use: 65, limits (soft = 500, hard = 600)
```

当器的候，新的配额限制置将会被保存。

有，在UIDs的上面配置限制是非常必要的。可以通过在 `edquota(8)` 命令后面加上 `-p` 来完成。首先，用分配所需要的配额限制，然后运行命令 `edquota -p protouser startuid-enduid`。例如，如果用 `test` 已经有了所需要的配额限制，下面的命令可以被用来限制那些UIDs 10,000 到 19,999 的配额限制：

```
# edquota -p test 10000-19999
```

更多参考 `edquota(8)` 手册。

### 19.15.3. 配额限制和磁盘使用

既可以使用 `quota(1)` 也可以使用 `repquota(8)` 命令来查看配额限制和磁盘使用情况。`quota(1)` 命令能查看一个用户和它的配置使用情况。只有超用户才可以查看其它用户的配额和磁盘使用情况。`repquota(8)` 命令可以用来了解所有配额和磁盘的使用情况。

下面是一个使用 `quota -v` 命令后的输出情况：

```
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
```

前面以 `/usr` 作例子。此用户目前已比配额限制 50 KB 超出了 15 KB，剩下 5 天的配额限期。注意，星号 \* 表明用户已超出了其配额限制。

通常，如果用户没有使用文件系统上的磁盘空间，就不会在 `quota(1)` 命令的输出中显示，即使已给那个用户指定了配额。而使用 `-v` 选项会显示它，例如前面例子中的 `/usr/var`。

### 19.15.4. 通过 NFS 使用磁盘配额

配额能在 NFS 服务器上被配额子系统强迫使用。在 NFS 客户端，`rpc.rquotad(8)` 命令可以使用 quota 信息用于 `quota(1)` 命令，可以允许用户查看它的 quota 信息。

可以在 `/etc/inetd.conf` 中使用 `rpc.rquotad`：

```
rquotad/1      dgram rpc/udp wait root /usr/libexec/rpc.rquotad rpc.rquotad
```

在重启 `inetd`：

```
# /etc/rc.d/inetd restart
```

## 19.16. 加密磁盘分区

FreeBSD 提供了好的数据保护措施，防止未受的数据。文件限制和控制(MAC)(看 `制控制`) 可以帮助防止操作系统于运行状态和计算机加未受的第三方数据。但是，和操作系统制受不相的是，如果黑客有物理上计算机的可能，那他就可以把计算机的硬件安装到一个系统上制出敏感的数据。

无论攻击者如何取得停机后的硬件或硬盘设备本身，FreeBSD GEOM Based Disk Encryption (基于 GEOM 的磁盘加密，`gbde`) 和 `geli` 加密子系统都能保护计算机上的文件系统数据，使它免受哪怕是专业的攻击者得到有用的来源。与那些只能加密一个文件的传统的加密方法不同，`gbde` 和 `geli` 能透明地加密整个文件系统。明文数据不会出现在硬盘的任何地方。

### 19.16.1. 使用 `gbde` 对磁盘进行加密

#### 1. 成为 `root`

配置 `gbde` 需要超级用户的权力。

```
% su -  
Password:
```

#### 2. 在内核配置文件中添加 `gbde(4)` 的支持

在您的内核配置中加入下面一行：

```
options GEOM_BDE
```

按照 [配置FreeBSD的内核](#) 所介绍的重新编译并安装内核。

重新引导入新的内核。

#### 3. 另一种无需重新编译内核的方法，是使用 `kldload` 来加载 `gbde(4)`：

```
# kldload geom_bde
```

#### 19.16.1.1. 准备加密

下面这个例子假设添加了一个新的硬盘到您的系统并将它有一个独立的加密分区。这个分区将挂载在 `/private` 目录下。`gbde` 也可以用来加密 `/home` 和 `/var/mail`，但是它需要更多的命令来执行。



## 1. 添加新的硬盘

添加新的硬盘到系统中可以查看 [添加磁盘](#) 中的说明。 这个例子的目的是说明一个新的硬盘分区已添加到系统中如： /dev/ad4s1c。在例子中 /dev/ad0s1\* 代表系统中存在的标准 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
```

## 2. 建立一个目录来保存 gbde Lock 文件

```
# mkdir /etc/gbde
```

gbde lock 文件包含了 gbde 需要知道的加密分区的信息。 没有 lock 文件， gbde 将不能解密包含在加密分区上的数据。 一个加密分区使用一个独立的 lock 文件。

## 3. 初始化 gbde 分区

一个 gbde 分区在使用前必须被初始化， 一个初始化过程只需要进行一次：

```
# gbde init /dev/ad4s1c -i -L /etc/gbde/ad4s1c.lock
```

**gbde(8)** 将打印的提示去放置在一个模板文件中的配置变量。 使用 UFS1 或 UFS2， 设置扇区大小 2048：

```
$FreeBSD: src/sbin/gbde/template.txt,v 1.1 2002/10/20 11:16:13 phk Exp $
#
# Sector size is the smallest unit of data which can be read or written.
# Making it too small decreases performance and decreases available space.
# Making it too large may prevent filesystems from working. 512 is the
# minimum and always safe. For UFS, use the fragment size
#
sector_size      =      2048
[...]
```

**gbde(8)** 将输入四次用来加密数据的密码短。 每次输入的密码必须相同。 gbde 保护数据的能力依赖于输入的密码的强度。

**gbde init** 命令创建的 gbde 分区创建了一个 lock 文件， 在例子中存储在 /etc/gbde/ad4s1c.lock 中。 gbde lock 文件必须使用 ".lock" 扩展名才能被 /etc/rc.d/gbde 脚本正确识别。



gbde lock 文件 必须和加密分区上的内容相同。如果生成只有 lock 文件遭到删除的情况，就没有办法确定 gbde 分区上的数据是否是解密过的。另外，如果没有 lock 文件，即使磁盘的合法主人，不能大量导致的工作也无法解密加密分区上的数据，而是在 [gbde\(8\)](#) 完全没有考虑的。

#### 4. 把加密分区和内核运行

```
# gbde attach /dev/ad4s1c -l /etc/gbde/ad4s1c.lock
```

在加密分区的初始化过程中将被要求提供一个密码短。新的加密将在 `/dev` 中显示 `/dev/device_name.bde`：

```
# ls /dev/ad*
/dev/ad0          /dev/ad0s1b      /dev/ad0s1e      /dev/ad4s1
/dev/ad0s1        /dev/ad0s1c      /dev/ad0s1f      /dev/ad4s1c
/dev/ad0s1a       /dev/ad0s1d      /dev/ad4          /dev/ad4s1c.bde
```

#### 5. 在加密上建立文件系统

当加密和内核运行后，就可以使用 [newfs\(8\)](#) 在此上建立文件系统，使用 [newfs\(8\)](#) 来初始化一个 UFS2 文件系统比初始化一个 UFS1 文件系统要快，推荐使用 `-O2`。

```
# newfs -U -O2 /dev/ad4s1c.bde
```



[newfs\(8\)](#) 命令必须在一个 gbde 分区上运行，每个分区通过一个存在的 \*.bde 命名运行。

#### 6. 挂载加密分区

加密文件系统建立一个挂载点。

```
# mkdir /private
```

挂载加密文件系统。

```
# mount /dev/ad4s1c.bde /private
```

#### 7. 校验加密文件系统是否有效

加密的文件系统在 [df\(1\)](#) 中可并可以使用。

```
% 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
```

### 19.16.1.2. 挂接已有的加密文件系统

初次系启动后，在使用加密文件系统前必须和内核重新行，校和再次挂接。使用的命令必须由 **root** 用来自行。

#### 1. 将 gbde 分区到内核

```
# gbde attach /dev/ad4s1c -l /etc/gbde/ad4s1c.lock
```

接下来系统将提示入在初始化加密的 gbde 分区所用的密码。

#### 2. 校文件系统

加密文件系统不能列在 `/etc/fstab` 文件中自行加，在加前必须手行 **fsck(8)** 命令文件系统行。

```
# fsck -p -t ffs /dev/ad4s1c.bde
```

#### 3. 挂接加密文件系统

```
# mount /dev/ad4s1c.bde /private
```

加密后的文件系统可以有效使用。

#### 19.16.1.2.1. 自挂接加密分区

可以建脚本来自地附加、，并挂接加密分区，然而，于安全考，个脚本不包 **gbde(8)** 密码。因而，我建脚本在控制台或通 **ssh(1)** 行并要求用入口令。

除此之外，系提供了一个 `rc.d` 脚本。个脚本的参数可以通 **rc.conf(5)** 来指定，例如：

```
gbde_autoattach_all="YES"
gbde_devices="ad4s1c"
gbde_lockdir="/etc/gbde"
```

在`root`将要求`root`入 `gbde` 的口令。在`root`入正`root`的口令之后，`gbde` 加密分区将被自`root`挂接。`root`于将 `gbde` 用在`root`本`root`上`root`，`root`就很有用了。

#### 19.16.1.3. `gbde` 提供的密`root`学保`root`

`gbde(8)` 采用 CBC 模式的 128-位 AES 来加密扇区数据。磁`root`上的`root`个扇区都采用不同的 AES 密`root`来加密。要了解`root`于 `gbde` 的密`root`学`root`，包括扇区密`root`如何从用`root`提供的口令字中生成等`root`，`root`参考 `gbde(4)`。

#### 19.16.1.4. 兼容性`root`

`sysinstall(8)` 是和 `gbde` 加密`root`不兼容的。在`root` `sysinstall(8)` `root`必`root`将 `*.bde` `root`和内核`root`行分`root`，否`root`在初始化探`root`将引起冲突。与加密`root`行分`root`在我`root`的例子中使用如下的命令：

```
# gbde detach /dev/ad4s1c
```

`root`需要注意的是，由于 `vinum(4)` 没有使用 `geom(4)` 子系`root`，因此不能同`root`使用 `gbde` 与 `vinum` 卷。

### 19.16.2. 使用 `geli` `root`磁`root`行加密

`root`有`root`一个可用于加密的 GEOM class - `geli`。它目前由 Paweł Jakub Dawidek <[pjd@FreeBSD.org](mailto:pjd@FreeBSD.org)> `root`。`Geli` 工具与 `gbde` 不同；它提供了一些不同的功能，并采用了不同的方式来`root`行密`root`学`root`算。

`geli(8)` 最重要的功能包括：

- 使用了 `crypto(9)` 框架 - 如果系`root`中有加解密硬件加速`root`，`root` `geli` 会自`root`加以利用。
- 支持多`root`加密算法 (目前支持 AES、Blowfish，以及 3DES)。
- 允`root`根分区`root`行加密。在系`root`，将要求`root`入用于加密根分区的口令。
- 允`root`使用`root`个不同的密`root` (例如，一个 "个人密`root`" 和一个 "公司密`root`")。
- `geli` 速度很快 - 它只`root`行`root`的扇区到扇区的加密。
- 允`root`和恢`root`主密`root`。当用`root`必`root`其密`root`，仍然可以通`root`从`root`中恢`root`密`root`来存取数据。
- 允`root`使用随机的一次性密`root`来挂接磁`root` - `root`于交`root`区和`root`文件系`root`非常有用。

更多 `geli` 功能介`root`可以在 `geli(8)` `root`机手册中`root`到。

下面的`root`介`root`了如何`root`用 FreeBSD 内核中的 `geli` 支持，并解`root`了如何`root`建新和使用 `geli` 加密 provider。

由于需要修改内核，`root`需要`root`有超`root`用`root`限。

## 1. 在内核中加入 geli 支持

在内核配置文件中加入下面行：

```
options GEOM_ELI
device crypto
```

按照 [配置FreeBSD的内核](#) 介绍的重新编译并安装内核。

另外，**geli** 也可以在系统引导时添加。这是通过在 `/boot/loader.conf` 中添加下面的配置来实现的：

```
geom_eli_load="YES"
```

[geli\(8\)](#) 在已安装内核所支持了。

## 2. 生成主密钥

下面的例子描述如何生成密钥文件，它将作为主密钥 (Master Key) 的一部分，用于挂接到 `/private` 的加密 provider。一个密钥文件将提供一些随机数据来加密主密钥。同时，主密钥也会使用一个口令字来保护。Provider 的扇区尺寸是 4kB。此外，这里将介绍如何挂接 **geli** provider，在其上建立文件系统，如何挂接并在其上工作，最后将其卸下。

建议使用较大的扇区尺寸 (例如 4kB)，以获得更好的性能。

主密钥将由口令字保护，而密钥文件的数据来源将是 `/dev/random`。我们称之为 provider 的 `/dev/da2.eli` 的扇区尺寸将是 4kB。

```
# dd if=/dev/random of=/root/da2.key bs=64 count=1
# geli init -s 4096 -K /root/da2.key /dev/da2
Enter new passphrase:
Reenter new passphrase:
```

同时使用口令字和密钥文件并不是必需的；也可以只使用其中之一来加密主密钥。

如果密钥文件写作 `"-"`，则表示使用标准输入。下面是关于如何使用多个密钥文件的例子：

```
# cat keyfile1 keyfile2 keyfile3 | geli init -K - /dev/da2
```

## 3. 将 provider 与所生成的密钥绑定

```
# geli attach -k /root/da2.key /dev/da2
Enter passphrase:
```

新的明文将被命名为 `/dev/da2.eli`。

```
# ls /dev/da2*
/dev/da2  /dev/da2.eli
```

#### 4. 新建的文件系

```
# dd if=/dev/random of=/dev/da2.eli bs=1m
# newfs /dev/da2.eli
# mount /dev/da2.eli /private
```

在加密的文件系已可以被 [df\(1\)](#) 看到，并处于可用状态了：

```
# 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
```

#### 5. 卸下卷并断 provider

一旦在加密分区上的工作完成，并且不再需要 `/private` 分区，就考将其卸下并将 `geli` 加密分区从内核上断。

```
# umount /private
# geli detach da2.eli
```

于如何使用 [geli\(8\)](#) 的更多信息，可以在其手册中找到。

##### 19.16.2.1. 使用 `geli rc.d` 脚本

`geli` 提供了一个 `rc.d` 脚本，它可以用于化 `geli` 的使用。通 [rc.conf\(5\)](#) 配置 `geli` 的方法如下：

```
geli_devices="da2"
geli_da2_flags="-p -k /root/da2.key"
```

将把 `/dev/da2` 配置一个 `geli` provider，其主密钥文件位于 `/root/da2.key`，而 `geli` 在接 provider 将不使用口令字（注意只有在 `geli init` 段使用了 `-P` 才可以做）。系将在之前将 `geli` provider 断。

于如何配置 `rc.d` 的信息可以在使用手册的 [rc.d](#) 一中找到。

## 19.17. 交换区行加密

FreeBSD 提供了易于配置的交换区加密机制。随所用的 FreeBSD 版本，可用的配置可能会有所不同，而配置方法也会有一些差异。可以使用 [gbde\(8\)](#) 和 [geli\(8\)](#) 加密系统来对交换区的加密操作。前面所讨论的加密系统，都用到了 `encswap` 这个 `rc.d` 脚本。

在前面的小节 [如何加密磁盘分区](#) 中，已经就不同的加密系统之间的区别进行了讨论。

### 19.17.1. 为什么需要交换区行加密？

与加密磁盘分区类似，加密交换区有助于保护敏感信息。因此，我们不妨考虑一个需要处理敏感信息的程序，例如，它需要处理口令。如果这些口令一直保持在物理内存中，一切相安无事。然而，如果操作系统始终将内存输出到交换区，以便其他应用程序输出内存，这些口令就可能以未加密的形式写到磁盘上，并被攻击者所轻易得到。加密交换区能有效地解决这个问题。

### 19.17.2. 准备



在本节余下的部分中，我们将使用 `ad0s1b` 作为交换区。

到目前为止，交换区仍是未加密的。很可能其中已存储有明文形式的口令或其他敏感数据。要更正这一问题，首先使用随机数来覆盖交换分区的数据：

```
# dd if=/dev/random of=/dev/ad0s1b bs=1m
```

### 19.17.3. 使用 [gbde\(8\)](#) 来加密交换区

`/etc/fstab` 中与交换区相关的行中，名称后追加 `.bde` 后：

| # Device        | Mountpoint | FStype | Options | Dump | Pass# |
|-----------------|------------|--------|---------|------|-------|
| /dev/ad0s1b.bde | none       | swap   | sw      | 0    | 0     |

### 19.17.4. 使用 [geli\(8\)](#) 来加密分区

一种方法是使用 [geli\(8\)](#) 来达到加密交换区的目的，其过程与使用 [gbde\(8\)](#) 大体相似。此时，在 `/etc/fstab` 中交换区相关的行中，名称后追加 `.eli` 后：

| # Device        | Mountpoint | FStype | Options | Dump | Pass# |
|-----------------|------------|--------|---------|------|-------|
| /dev/ad0s1b.eli | none       | swap   | sw      | 0    | 0     |

[geli\(8\)](#) 默认情况下使用密钥长度 256-位的 AES 加密算法。

当然，这些默认是可以通 `/etc/rc.conf` 中的 `geli_swap_flags` 来修改的。下面的配置表示 `rc.d` 脚本 `encswap` 建立一个 [geli\(8\)](#) 交换区，在其上使用密钥长度 128-位的 Blowfish 加密算法，4 kilobytes 的扇区尺寸，并采用“最后一次挂载卸下”的策略：

```
geli_swap_flags="-e blowfish -l 128 -s 4096 -d"
```

参 [geli\(8\)](#) 手册中关于 [onetime](#) 命令的说明，以了解其他可用的选项。

### 19.17.5. 加密所作的配置能做什么作用

在重启系统之后，就可以使用 [swapinfo](#) 命令来检查加密交换区是否已在正常运行了。

如果使用了 [gbde\(8\)](#)，则：

```
% swapinfo
Device          1K-blocks    Used    Avail Capacity
/dev/ad0s1b.bde  542720         0    542720      0%
```

如果使用了 [geli\(8\)](#)，则：

```
% swapinfo
Device          1K-blocks    Used    Avail Capacity
/dev/ad0s1b.eli  542720         0    542720      0%
```

## 19.18. 高可用性存储 (HAST)

### 19.18.1. 概述

高可用性是担任数据的重要的一主要需求，而高可用存储是系统环境中的一部分。高可用存储 Highly Available Storage，或 HAST，是由 Paweł Jakub Dawidek <[pjd@FreeBSD.org](mailto:pjd@FreeBSD.org)> 开发的一用于提供在多台物理上隔离的系统之间以透明的方式，通过 TCP/IP 网络数据的高可用性框架。HAST 可以看作通过网络运行的 RAID1 (镜像)，类似于 GNU/Linux® 平台上的 DRBD® 存储系统。配合 FreeBSD 提供的其他高可用性基础设施，如 CARP，HAST 可以用来构建可以抵御硬件故障的高可用存储集群。

读完，你将了解：

- 何谓 HAST，它如何工作以及提供哪些功能。
- 如何在 FreeBSD 上配置和使用 HAST。
- 如何与 CARP 及 [devd\(8\)](#) 配合构建可靠的存储系统。

在开始之前，则：

- 了解 UNIX® 和 FreeBSD 的基础知识 ([UNIX 基础](#))。
- 知道如何配置网络接口以及其他核心 FreeBSD 子系统 ([配置和调整](#))。
- 理解 FreeBSD 的网络功能 ([网络通信](#))。
- 使用 FreeBSD 8.1-RELEASE 或更新版本。

HAST 项目是由 FreeBSD 基金会赞助完成的，并得到了来自 [OMCnet Internet Service GmbH](#) 和 [TransIP](#)



BV 的支持。

## 19.18.2. HAST 的功能

HAST 系提供的功能主要包括：

- 可以掩本地硬的 I/O 。
- 文件系无，因而可以配合 FreeBSD 支持的任何文件系使用。
- 高效率的快速重新同机制，令系只同在一点停机修改的。
- 可以在已部署好的境中添加冗余。
- 配合 CARP、Heartbeat 或其他似的工具，可以健壮的可存系。

## 19.18.3. HAST 的行机制

由于 HAST 本上是在多个机器同地行制，因此它需要至少个 (物理的机器) - 其一作主 (也称作 master) 点，一个作从 (slave) 点。台机器会共同成一个集群。



目前 HAST 只能使用最多个集群点。

由于 HAST 是配置成以主从点的方式行，在任何刻都只能有唯一的一个点是主点。主点，也称作活点，理理由 HAST 管理的的全部 I/O 求。而从点会自主点同数据的更操作。

在 HAST 系中的物理包括：

- 本地磁 (在主点上)
- 程磁 (在从点上)

HAST 在的的上同行，使其文件系和用程序透明。HAST 在 /dev/hast/ 目中提供准的 GEOM 供其他工具或用程序使用，因此，在使用上，用程序或文件系而言，HAST 提供的与普通的裸或分区等没有任何区。

到本地磁的次写、除或存刷写操作，都会同通 TCP/IP 到程磁上。操作是由本地磁完成，除非本地磁上的数据不是最新的，或生了 I/O 。在情况下，操作会在从点上完成。

### 19.18.3.1. 同及制模式

HAST 希望提供快速的故障恢能力。基于一考量，少在某个点停机后需要的同就十分重要。了提供快速的同能力，HAST 会一保存在磁上的区段位映射表 (bitmap of dirty extents)，在普通的同模式中，它只同些部分的数据 (初始的同除外)。

理同有多不同的方式，HAST 以下几同方式：

- *memsync*：当本地的写操作已完成，并且程点已收到数据，便数据的写操作已完成，而不是等待程点完成数据的写操作。程点在出回之后，会立即始行写操作。模式的目的是少，但在同仍然保持很好的可性。目前 *memsync* 制模式尚未。
- *fullsync*：只有在本地写操作完成，并且程的写操作也已完成的情况下，才数据的写操作已完成。模式是最保，同也是最慢的一制模式。是目前系的制模式。

- *async*：在本地写操作完成后，即数据已写完。是最快，同时也是最大的制模式，一般而言只有在一点的延迟大时才考虑使用。目前 *async* 制模式尚未。



目前，只支持 *fullsync* 制模式。

#### 19.18.4. HAST 的配置

HAST 需要 *GEOM\_GATE* 支持才能正常工作。系自的 *GENERIC* 内核并不包含 *GEOM\_GATE*，但默的 FreeBSD 安装包包含了 *geom\_gate.ko* 内核模。如果系行了裁剪，个模是否可用。此外，*GEOM\_GATE* 也可以静内核，方法是在内核的配置中添加下面的置：

```
options GEOM_GATE
```

从操作系的角度，HAST 框架包含了下面些部件：

- 行数据同的 *hastd(8)* 服程序，
- 用于行管理操作的 *hastctl(8)* 用管理工具，
- 配置文件 *hast.conf(5)*。

下面的例子将介使用 HAST 在个之间以 *主-从* 模式制数据的方法。个点的名字分是 *hasta* 其 IP，地址 172.16.0.1，以及 *hastb*，其 IP 地址 172.16.0.2。台机器都使用尺寸相同的磁 /dev/ad6 来用于 HAST 的行。HAST 存池 (有也称源，例如位于 /dev/hast/ 的文件) 将命名 test。

HAST 的配置文件是 /etc/hast.conf。在个点上，个文件的内容是完全一的。最配置如下：

```
resource test {
  on hasta {
    local /dev/ad6
    remote 172.16.0.2
  }
  on hastb {
    local /dev/ad6
    remote 172.16.0.1
  }
}
```

如果需要更高的配置，参机手册 *hast.conf(5)*。



在 *remote* 句中也可以使用主机名。情况下需要保些主机名是可以解析的，例如在 /etc/hosts 文件中，或在本地 DNS 中行了定。

在在个点上都有同的配置了，接下来我需要建 HAST 存池。在个点上分行下面的命令来初始化本地此怕，并 *hastd(8)* 服：

```
# hastctl create test
# /etc/rc.d/hastd onestart
```



没有办法使用已经包含文件系统的 GEOM 来创建池（换言之，已经存在的文件系统无法成为 HAST 管理的存储池），这是因为创建池的过程需要保存一些元数据，而已写入文件系统的块不再能提供保存这些元数据所需的空间。

HAST 并不指定节点的角色（主 或 从）。节点的角色是由管理手工，或由类似 Heartbeat 的工具通过 [hastctl\(8\)](#) 来完成配置的。在希望成为主节点的系（[hasta](#)）上运行下面的命令令其成为主节点：

```
# hastctl role primary test
```

类似地，用下面的命令来指明从节点（[hastb](#)）：

```
# hastctl role secondary test
```



有可能会出现两个节点之间无法正常通信，但又都配置为主节点的情况；这称作 **分裂** 的状态是十分危险的。在 [从分裂状态恢复](#) 中介绍了如何从该状态中恢复的方法。

接下来，可以在每个节点上使用 [hastctl\(8\)](#) 工具来检查节点自身是否正确：

```
# hastctl status test
```

其中比较重要的是 **status**（状态）一行，在每个节点上，其输出均为 **complete**（完好）。如果系统输出的输出是 **degraded**（降级），则表示出了问题。正常情况下，节点的同步已就绪。当 **hastctl status** 命令告的 **dirty** 数据块数量 0 字，表示每个节点的数据已完全同步。

最后一项是在 GEOM 块 `/dev/hast/test` 上创建文件系统。这项工作必须在 **主** 节点上运行（因为 `/dev/hast/test` 只在 **主** 节点上输出），随硬件尺寸的不同，可能需要花费数分钟的时间：

```
# newfs -U /dev/hast/test
# mkdir /hast/test
# mount /dev/hast/test /hast/test
```

一旦完成了 HAST 框架的配置，最后一项就是确保 HAST 在系统启动过程中会自动运行了。为了达到这个目的，在 `/etc/rc.conf` 文件中添加一行配置：

```
hastd_enable="YES"
```

#### 19.18.4.1. 故障转移配置

这个例子的目的在于建立一套健壮存储系统，令其能抵御在任何一个节点上发生的故障。其中的任何任务是

集群中的主节点发生故障的情形行及的救理。当生情况，从节点可以无缝地接手主点的工作，文件系统行并挂接，从而行，而不失任何数据。

了成一任，需要使用 FreeBSD 提供的功能 - CARP 所提供的 IP 自故障移能力。CARP 是共用地址冗余 Common Address Redundancy Protocol 的写，它允多个同网段的主机共享同一 IP 地址。根据 [Common Address Redundancy Protocol \(CARP, 共用地址冗余\)](#) 的介在个点上配置 CARP。完成些配置之后，个点都会有自己的 carp0 网接口，共用 IP 地址 172.16.0.254。然，集群中的 HAST 主点也必是 CARP 主点。

前面一中建的 HAST 存池在可以提供网上的其他主机使用了。其上的文件系统可以通 NFS、Samba 等等，以共用 IP 地址 172.16.0.254 来。在余下的唯一是自化主点故障的理。

当 CARP 网接口的路状生化，FreeBSD 操作系会生一个 [devd\(8\)](#) 消息，就可以 CARP 网接口的状了。CARP 接口的状化表示点生故障，或重新回到了网中。些情况下需要行特定的脚本来完成理的。

了截 CARP 网接口的状化，需要在个点的 /etc/devd.conf 文件中添加如下的置：

```
notify 30 {
    match "system" "IFNET";
    match "subsystem" "carp0";
    match "type" "LINK_UP";
    action "/usr/local/sbin/carp-hast-switch master";
};

notify 30 {
    match "system" "IFNET";
    match "subsystem" "carp0";
    match "type" "LINK_DOWN";
    action "/usr/local/sbin/carp-hast-switch slave";
};
```

使的配置生效，需要在个点上行下面的命令：

```
# /etc/rc.d/devd restart
```

当网接口 carp0 的状生化，系会生一个通知消息，允 [devd\(8\)](#) 子系行管理指定的任意脚本，在个例子中是 /usr/local/sbin/carp-hast-switch。个脚本的作用是自化故障移。于前面 [devd\(8\)](#) 配置的具体含，参机手册 [devd.conf\(5\)](#)。

下面是一个脚本的示例：

```
#!/bin/sh

# Original script by Freddie Cash <fjwcash@gmail.com>
# Modified by Michael W. Lucas <mwlucas@BlackHelicopters.org>
# and Viktor Petersson <vpetersson@wireload.net>
```

```

# The names of the HAST resources, as listed in /etc/hast.conf
resources="test"

# delay in mounting HAST resource after becoming master
# make your best guess
delay=3

# logging
log="local0.debug"
name="carp-hast"

# end of user configurable stuff

case "$1" in
    master)
        logger -p $log -t $name "Switching to primary provider for ${resources}."
        sleep ${delay}

        # Wait for any "hastd secondary" processes to stop
        for disk in ${resources}; do
            while $( pgrep -lf "hastd: ${disk} \(\secondary\) " > /dev/null 2>&1 ); do
                sleep 1
            done

            # Switch role for each disk
            hastctl role primary ${disk}
            if [ $? -ne 0 ]; then
                logger -p $log -t $name "Unable to change role to primary for 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 ${resources}; do
        mkdir -p /hast/${disk}
        fsck -p -y -t ufs /dev/hast/${disk}
        mount /dev/hast/${disk} /hast/${disk}
    done

;;

slave)
    logger -p $log -t $name "Switching to secondary provider for ${resources}."

    # Switch roles for the HAST resources
    for disk in ${resources}; do
        if ! mount | grep -q "^/dev/hast/${disk} on "
        then
        else
            umount -f /hast/${disk}
        fi
        sleep $delay
        hastctl role secondary ${disk} 2>&1
        if [ $? -ne 0 ]; then
            logger -p $log -t $name "Unable to switch role to secondary for
resource ${disk}."
            exit 1
        fi
        logger -p $log -t $name "Role switched to secondary for resource ${disk}."
    done
;;
esac

```

而言之，在点成网点的 **master / primary** 点，脚本会行下面的操作：

- 在本点升格 HAST 存池的主点。
- 对 HAST 存池上的文件系。
- 挂接存池中的文件系到当的位置。

当点成 **backup / secondary** 点：

- 卸下 HAST 存池。
- 将本点降格 HAST 存池的从点。



必注意， 上面的脚本只是概念性的介。 它并不能理所有可能生的情况， 因此根据情况行修改， 例如/停止必要的服， 等等。



在前面的例子中， 出于示的目的我使用的是准的 UFS 文件系。 了少恢所需的， 可以使用日志的 UFS 文件系， 或者使用 ZFS 文件系。

更具体的信息和例子参 [HAST Wiki](#) 面。

## 19.18.5. 故障排除

### 19.18.5.1. 一般故障排除提示

HAST 通常都能无故障地运行，不过，和任何其他软件产品一样，有它也可能无法以希望的方式运行。导致问题的可能性有很多，但一般来说，首先要确保集群中所有节点的角色是同等的。

当排除 HAST 故障时，可提高 `hastd(8)` 的优先级。可以通过在 `hastd(8)` 服务指定 `-d` 参数来实现。需要明确的是，可以多次指定同一参数来进一步提高优先级。此外，可以考虑使用 `-F` 参数来启动服务，它会令 `hastd(8)` 服务在前台运行。

### 19.18.5.2. 从分裂状态恢复

当集群中的两个节点之间无法相互通信时，两个节点都会认为自己为主节点，从而导致 **分裂** 的状态。该情形十分危险，因为两个节点会产生互相无法合并的数据。该情形需要系统管理员施手工干预。

从该状态中恢复时，管理员必须决定哪一个节点包含最重要的数据（或者手工合并某些更改）并对 HAST 进行一次完整的同步操作，覆盖有该数据的那个节点的数据。要完成该工作，在有该数据的节点上运行下面的命令：

```
# hastctl role init <resource>
# hastctl create <resource>
# hastctl role secondary <resource>
```

# Chapter 20. GEOM: 模块化磁盘框架

## 20.1. 概述

本章将介绍以 FreeBSD GEOM 框架来使用磁盘。它包括了使用同一框架来配置的主要的 RAID 控制工具。这一章不会深入讨论 GEOM 如何管理或控制 I/O、其下的子系统或驱动。你可以从 [geom\(4\)](#) 手册及其许多 SEE ALSO 参考文献中得到一些信息。这一章也不是 RAID 配置的权威介绍，它只介绍由支持 GEOM 的 RAID 驱动。

读完本章，你将了解：

- 通用 GEOM 支持的 RAID 类型。
- 如何使用基本工具来配置和管理不同的 RAID 驱动。
- 如何通过 GEOM 使用镜像、条带、加密和挂载在程序的磁盘驱动。
- 如何排除挂载在 GEOM 框架上的磁盘驱动的问题。

本章之前，你还：

- 理解 FreeBSD 如何管理磁盘驱动 ([存盘](#))。
- 了解如何配置和安装新的 FreeBSD 内核 ([配置 FreeBSD 的内核](#))。

## 20.2. GEOM 介绍

GEOM 允许管理和控制类 (classes) - 主引导记录、BSD 分区 (label)，等等 - 通用使用 provider，或在 /dev 中的特殊文件。它支持多种 RAID 配置，GEOM 能够向操作系统，以及在其上运行的工具提供透明的驱动方式。

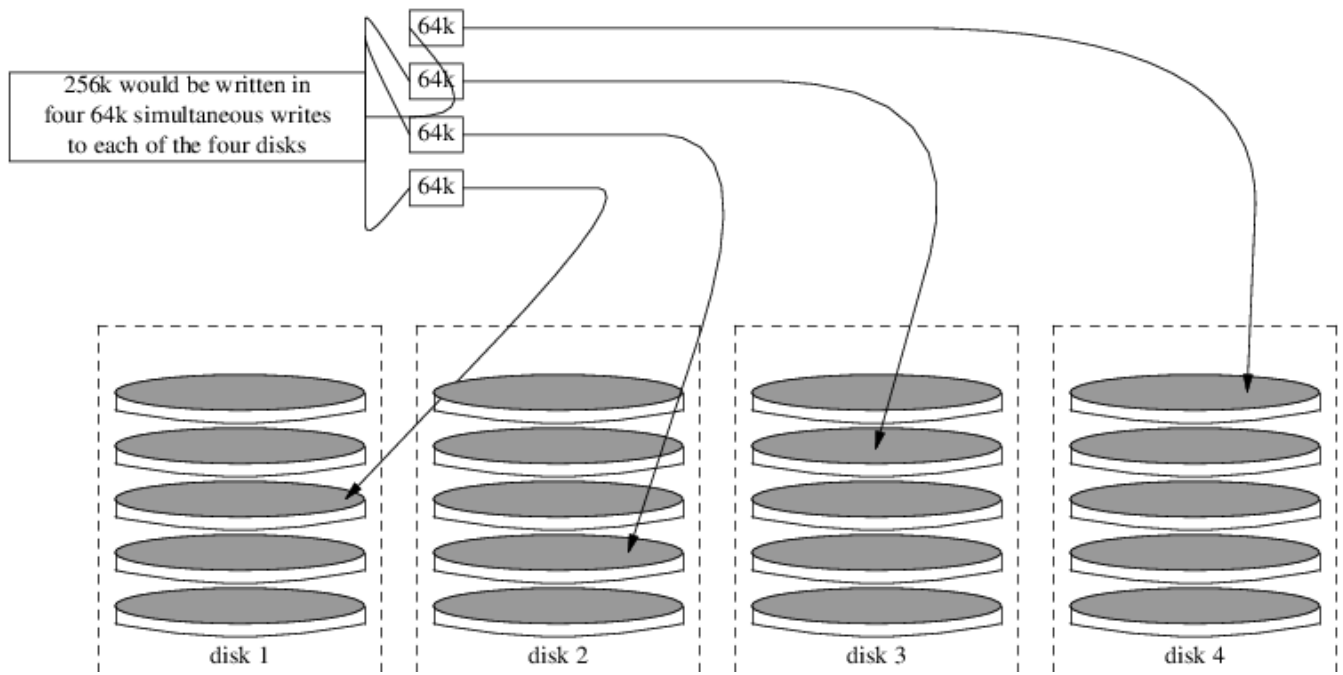
## 20.3. RAID0 - 条带

条带是一种将多个磁盘驱动器合并成一个卷的方法。通常情况下，它是通过硬件控制器来完成的。GEOM 磁盘子系统提供了 RAID0 的驱动支持，它也成为磁盘条带。

在 RAID0 系统中，数据被分成多个块，每个块将分写入排列的所有磁盘。与先前需要等待系统将 256k 数据写到一个磁盘上不同，RAID0 系统，能够同时分块将打碎的 64k 写到四个磁盘上，从而提供更好的 I/O 性能。这一性能提升能够通过使用多个磁盘控制器来进一步改进。

在 RAID0 条带中的一个块的大小必须一致，因为 I/O 请求是分散到多个块上的，以便每个块上的读写并行完成。





## Procedure: 在未格式化的 ATA 磁盘上建立条带

1. 加载 `geom_stripe.ko` 模块：

```
# kldload geom_stripe
```

2. 确信存在合适的挂载点 (mount point)。如果该卷将成根分区，那你可以把它挂接到其他位置，如 `/mnt`：

```
# mkdir /mnt
```

3. 确定将被做成条带卷的磁盘的名称，并创建新的条带。例如而言，要将该未用的、尚未分区的 ATA 磁盘 `/dev/ad2` 和 `/dev/ad3` 做成一个条带：

```
# gstripe label -v st0 /dev/ad2 /dev/ad3
Metadata value stored on /dev/ad2.
Metadata value stored on /dev/ad3.
Done.
```

4. 接着需要写准的 label，也就是通常所指的分区表到新卷上，并安装准的引导代码：

```
# bsdlabel -wB /dev/stripe/st0
```

5. 上述程序将在 `/dev/stripe` 目录中的 `st0` 磁盘上建立一个新磁盘。包括 `st0a` 和 `st0c`。磁盘，就可以在 `st0a` 磁盘上用下述 `newfs` 命令来建立文件系统了：

```
# newfs -U /dev/stripe/st0a
```

在屏幕上将显示一些数字，整个操作可能能在数秒内完成。现在可以挂载做好的卷了。

要挂载新建的条带：

```
# mount /dev/stripe/st0a /mnt
```

要在程序中自动挂载该条带上的文件系统，需要把关于卷的信息放到 `/etc/fstab` 文件中。达到此目的，需要创建一个叫 `stripe` 的永久的挂载点：

```
# mkdir /stripe
# echo "/dev/stripe/st0a /stripe ufs rw 2 2" \
  >> /etc/fstab
```

此外，`geom_stripe.ko` 模块也必须通过 `/boot/loader.conf` 中添加下述配置，以便在系统初始化过程中自动加载：

```
# echo 'geom_stripe_load="YES"' >> /boot/loader.conf
```

## 20.4. RAID1 - 镜像

镜像是许多公司和家庭用户使用的一种无中断的技术。简单地，镜像的概念就是磁盘 B 是磁盘 A 的副本，或者磁盘 C+D 是磁盘 A+B 的同等副本，等等。无论磁盘配置如何，技术的共同特点都是一磁盘或分区的内容会同等复制到别的地方。因此，除了能很容易地恢复信息之外，还能在无中断服务或任何的情况下运行，甚至直接将副本送到数据保安公司异地存储。

在开始做这件事之前，首先准备两个容量相同的磁盘驱动器，下面的例子假定它们都是使用直接访问方式 (Direct Access, [da\(4\)](#)) 的 SCSI 磁盘。

### 20.4.1. 主磁盘运行镜像

假定系统中有 FreeBSD 安装到了第一个，也就是 `da0` 上，那么告诉 `gmirror(8)` 将主要数据保存在哪里。

在开始建镜像卷之前，可以用更多的信息，并开放所有的完全信息。可以通过将 `sysctl(8)` 变量 `kern.geom.debugflags` 设置下面的值来：

```
# sysctl kern.geom.debugflags=17
```

接下来需要建镜像。第一个步骤的第一步是在主磁盘上保存元数据信息，也就是用下面的命令来建 `/dev/mirror/gm` ：



在引导用的基础上新建镜像，有可能会致保存在磁盘上最后一个扇区的数据丢失。在新安装 FreeBSD 之后立即建镜像可以降低此风险。下面的操作与默认的 FreeBSD 9.X 安装过程不兼容，因为它采用了新的 GPT 分区格式。GEOM 会覆盖 GPT 元数据，会导致数据丢失，并有可能致系统无法引导。

```
# gmirror label -vb round-robin gm0 /dev/da0
```

系统输出下面的回：

```
Metadata value stored on /dev/da0.
Done.
```

初始化 GEOM，该操作会加载内核模块 `/boot/kernel/geom_mirror.ko`：

```
# gmirror load
```



当这个命令运行完之后，系统会在 `/dev/mirror` 目录中创建点 `gm0`。

配置在系⌈初始化⌈程中自⌈加⌈ geom\_mirror.ko :

```
# echo 'geom_mirror_load="YES"' >> /boot/loader.conf
```

⌈⌈ /etc/fstab 文件, 将其中先前的 da0 改⌈新的⌈像⌈⌈ gm0。

如果 vi(1) 是⌈喜⌈的⌈器, 以下⌈是完成此⌈任⌈的一个⌈便方法 :



```
# vi /etc/fstab
```

在 vi(1) 中⌈⌈⌈有的 fstab 内容, 具体操作是 :w /etc/fstab.bak。接着, 把所有旧的 da0 替⌈成 gm0, 也就是⌈入命令 :s/da/mirror\gm/g。

修改完后的 fstab 文件⌈⌈是下面的⌈子。磁⌈⌈器是 SCSI 或 ATA 甚至 RAID 都没有⌈系, 最⌈的⌈果都是 gm。

| # Device            | Mountpoint | FStype | Options   | Dump | Pass# |
|---------------------|------------|--------|-----------|------|-------|
| /dev/mirror/gm0s1b  | none       | swap   | sw        | 0 0  |       |
| /dev/mirror/gm0s1a  | /          | ufs    | rw 1 1    |      |       |
| /dev/mirror/gm0s1d  | /usr       | ufs    | rw 0 0    |      |       |
| /dev/mirror/gm0s1f  | /home      | ufs    | rw 2 2    |      |       |
| #/dev/mirror/gm0s2d | /store     | ufs    | rw 2 2    |      |       |
| /dev/mirror/gm0s1e  | /var       | ufs    | rw 2 2    |      |       |
| /dev/acd0           | /cdrom     | cd9660 | ro,noauto | 0 0  |       |

重⌈系⌈ :

```
# shutdown -r now
```

在系⌈初始化⌈程中, 新建的 gm0 会代替 da0 ⌈⌈工作。系⌈完成初始化之后, 可以通⌈⌈⌈ mount 命令的⌈出来⌈看效果 :

| # mount | Filesystem         | 1K-blocks | Used    | Avail    | Capacity | Mounted on     |
|---------|--------------------|-----------|---------|----------|----------|----------------|
|         | /dev/mirror/gm0s1a | 1012974   | 224604  | 707334   | 24%      | /              |
|         | devfs              | 1         | 1       | 0        | 100%     | /dev           |
|         | /dev/mirror/gm0s1f | 45970182  | 28596   | 42263972 | 0%       | /home          |
|         | /dev/mirror/gm0s1d | 6090094   | 1348356 | 4254532  | 24%      | /usr           |
|         | /dev/mirror/gm0s1e | 3045006   | 2241420 | 559986   | 80%      | /var           |
|         | devfs              | 1         | 1       | 0        | 100%     | /var/named/dev |

⌈个⌈出是正常的。最后, 使用下面的命令将 da1 磁⌈加到⌈像卷中, 以⌈始同⌈程 :

```
# gmirror insert gm0 /dev/da1
```

在创建像卷的过程中，可以用下面的命令看状态：

```
# gmirror status
```

一旦像卷的创建操作完成，该命令的输出就会变成：

| Name       | Status   | Components |
|------------|----------|------------|
| mirror/gm0 | COMPLETE | da0<br>da1 |

如果有或者创建仍在进行，输出中的 **COMPLETE** 就会是 **DEGRADED**。

## 20.4.2. 故障排除

### 20.4.2.1. 系统引导问题

如果系统引导出现以下面的提示：

```
ffs_mountroot: can't find rootvp  
Root mount failed: 6  
mountroot>
```

这种情况使用源或按重机器。在引导菜单中，按第六 (6) 个键。将系统入 **loader(8)** 提示符。在此手工加载内核模块：

```
OK? load geom_mirror  
OK? boot
```

如果这样做能解决，表明由于某原因模块没有被正确加载。在 `/boot/loader.conf` 中相关条目是否正确。如果仍然存在，可以在内核配置文件中加入：

```
options GEOM_MIRROR
```

然后重新编译和安装内核来解决这个问题。

### 20.4.3. 从磁盘故障中恢复

磁盘镜像的一大好处是在当其中一个磁盘出现故障，可以很容易地将其替换掉，并且通常不会丢失数据。

考虑前面的 RAID1 配置，假设 `da1` 出了故障并需要替换，要替换它，首先确定哪个磁盘出了故障，并替换它。此，可以用上新的磁盘，并重新配置。之后可以用下面的命令来完成磁盘的替换操作：

```
# gmirror forget gm0
```

```
# gmirror insert gm0 /dev/da1
```

在重建过程中可以用 `gmirror status` 命令来看进度。就是看看。

## 20.5. RAID3 - 使用专用校验的字条

RAID3 是一种将多个磁盘成一个卷的技术，在这个配置中包含一个用于校验的。在 RAID3 系中，数据会以字节位拆分并写入除校验之外的全部磁盘中。这意味着从 RAID3 中读取数据将会所有的磁器。采用多个磁盘控制器可以一改善性能。RAID3 阵列最多可以容忍其中的 1 个磁器出现故障，它可以提供全部磁器容量的  $1 - 1/n$ ，此  $n$  是阵列中的磁盘数量。配置比较适合保存大容量的数据，例如多媒体文件。

在建立 RAID3 阵列，至少需要 3 磁盘。所有的尺寸必须一致，因为 I/O 请求会并分派到不同的上。此外，由于 RAID3 本身的，的数量必须恰好是 3, 5, 9, 17, 等等 ( $2^n + 1$ )。

### 20.5.1. 建立专用的 RAID3 阵列

在 FreeBSD 中，RAID3 是通过 [graid3\(8\)](#) GEOM class 的。在 FreeBSD 中建立专用的 RAID3 阵列需要下述。



当然理论上从 RAID3 阵列 FreeBSD 是可行的，但并不常，也不推荐做。

1. 首先，在引导加载器中用下面的命令加载 `geom_raid3.ko` 内核模块：

```
# graid3 load
```

此外，也可以通过命令行手工加载 `geom_raid3.ko` 模块：

```
# kldload geom_raid3.ko
```

2. 创建用于挂载卷的挂载点目录：

```
# mkdir /multimedia/
```

3. 确定将要加入阵列的磁盘名称，并创建新的 RAID3 阵列。最后，该阵列将代表整个阵列。下面的例子使用三个未分区的 ATA 磁盘：ada1 和 ada2 保存数据，而 ada3 用于校验。

```
# graid3 label -v gr0 /dev/ada1 /dev/ada2 /dev/ada3
Metadata value stored on /dev/ada1.
Metadata value stored on /dev/ada2.
Metadata value stored on /dev/ada3.
Done.
```

4. 新建的 `gr0` 阵列分区，并在其上创建 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
```

屏幕上会输出许多数字，该过程需要一段时间才能完成。此后，就完成了创建卷的全部操作，可以挂载它了。

5. 最后一项是挂载文件系统：

```
# mount /dev/raid3/gr0p1 /multimedia/
```

现在可以使用 RAID3 阵列了。

除了上述配置在系统重启后可用，还需要进行一些额外的配置操作。

1. 在挂载卷之前必须首先加载 `geom_raid3.ko` 模块。将下面的配置添加到 `/boot/loader.conf` 文件中，可以确保在引导过程中自动加载该模块：

```
geom_raid3_load="YES"
```

2. 需要在 `/etc/fstab` 文件中加入下列配置，以便系统引导时自动挂载列上的文件系统：

```
/dev/raid3/gr0p1    /multimedia ufs rw 2    2
```

## 20.6. GEOM Gate 网络

通过 `gate` 工具，GEOM 支持以编程方式使用，例如磁盘、CD-ROM、文件等等。它和 NFS 类似。

在开始工作之前，首先要创建一个导出文件。该文件的作用是指定可以导出的源，以及提供何等的权限。例如，要把第一个 SCSI 盘的第四个 slice 导出，它的 `/etc/gg.exports` 会是类似下面的样子：

```
192.168.1.0/24 RW /dev/da0s4d
```

它表示允许同属私有子网的所有机器在 `da0s4d` 分区上的文件系统。

要导出一个，首先确保它没有被挂载，然后是 `ggated(8)` 服务：

```
# ggated
```

在我将在客户机上 `mount` 它，使用下面的命令：

```
# ggatec create -o rw 192.168.1.1 /dev/da0s4d
ggate0
# mount /dev/ggate0 /mnt
```

到此为止，客户端已经可以通过挂载点 `/mnt` 了。



注意，如果它已被服务器或网络上的任何其他机器挂载，前述操作将会失败。

如果不再需要使用它，就可以使用 `umount(8)` 命令来安全地将其卸下了，它一点和其他磁盘类似。

## 20.7. 磁盘添加卷

在系统初始化的过程中，FreeBSD 内核会扫描到的创建点。这种方式存在一些，例如，在通过 USB 添加磁盘如何处理？很可能有磁盘最初被命名为 `da0` 而在之后，它由 `da0` 变成了 `da1`。而它会在挂载 `/etc/fstab` 中的文件系统造成问题，有些，它可能在系统引导时导致无法正常。



解决这个问题的一个方法是以直接方式式地行 SCSI 命名， 当在 SCSI 上加新磁盘， 磁盘将使用一个未用的号。 但如果 USB 取代了主 SCSI 磁盘的位置？ 由于 USB 通常会在 SCSI 之前到， 因此很可能出现现象。 当然， 可以通过在系引之后再入些磁盘来解决这个问题。 一种解决这个问题的方法， 是只使用 ATA 磁盘， 并避免在 /etc/fstab 中列出 SCSI 磁盘。

有一个更好的解决方法。 通过使用 `glabel` 工具， 管理磁盘或用磁盘可以打上磁标， 并在 /etc/fstab 中使用磁标。 由于 `glabel` 会将磁标保存在磁盘 provider 的最后一个扇区， 在系重启之后， 它仍会持续存在。 因此， 通盘将具体的磁标替换使用磁标表示， 无论磁盘点成什么， 文件系统都能顺利地完成挂接。



磁标并不是磁标一定是永久性的。 `glabel` 工具既可以建永久性磁标， 也可以建非永久性磁标。 在重启， 只有永久性磁标会保持。 参阅机手册 [glabel\(8\)](#) 以了解两者之间的差别。

### 20.7.1. 磁标类型和使用示例

有磁标类型的磁标， 一种是普通磁标， 一种是文件系统磁标。 磁标可以是永久性的或非永久性的。 永久性的磁标可以通过 [tunefs\(8\)](#) 或 [newfs\(8\)](#) 命令。 根据文件系统的类型， 它将在 /dev 下的一个子目中被建。 例如， UFS2 文件系统的磁标会建到 /dev/ufs 目中。 永久性的磁标可以使用 `glabel label` 建。 它不再是文件系统特定的， 而是会在 /dev/label 目中被建。

非永久性的磁标在系下次重启会消失， 有些磁标会建到 /dev/label 目中， 很合用之用。 可以使用 `glabel create` 建非永久性的磁标。 参阅 [glabel\(8\)](#) 手册以获取更多信息。

要为一个 UFS2 文件系统建永久性磁标， 而不破坏其上的数据， 可以使用下面的命令：

```
# tunefs -L home /dev/da3
```



如果文件系统满了， 可能会导致数据损坏； 不， 如果文件系统快满了， 此首先删除一些无用的文件， 而不是加。

在， 磁标可以在 /dev/ufs 目中看到， 并将其加入 /etc/fstab：

|               |       |     |    |   |   |
|---------------|-------|-----|----|---|---|
| /dev/ufs/home | /home | ufs | rw | 2 | 2 |
|---------------|-------|-----|----|---|---|



当行 `tunefs`， 首先卸下文件系统。

在可以像平常一样挂接文件系统了：

```
# mount /home
```

在， 只要在系引通 /boot/loader.conf 配置加了内核模块 `geom_label.ko`， 或在内核指定了 `GEOM_LABEL` 磁标， 磁标点由于磁标而顺序生成， 就不会影响文件系统的挂接了。

通使用 `newfs` 命令的 `-L` 参数， 可以在建文件系统其添加默认的磁标。 参阅机手册 [newfs\(8\)](#) 以了解一的情况。

下列命令可以清除：

```
# glabel destroy home
```

以下的例子展示了如何为一个磁盘打上标签。

### 例 31. 为磁盘打上标签

为磁盘打上永久性标签，系统就能正常工作，即使磁盘被移动到了另外一个控制器或者移动到了一个不同的系统上。此例中我假设使用了一个 ATA 磁盘，当前这个磁盘被系统识别为 `ad0`。假设使用了标准的 FreeBSD 分区方案，`/`、`/var`、`/usr` 和 `/tmp` 文件系统，有一个 swap 分区。

重启系统，在 `loader(8)` 提示符下输入 `4` 进入到单用户模式。然后输入以下的命令：

```
# 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
```

系统加载完成后进入多用户模式。在输入完后，`/etc/fstab` 用各自的标签替换下常有的名称。最后 `/etc/fstab` 看起来差不多是这样的：

| # 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     |

现在可以重启系统了。如果一切顺利的话，系统可以正常工作并且 `mount` 命令显示：

```
# 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)
```

从 FreeBSD 7.2 开始，`glabel(8)` class 新增加了一个用于 UFS 文件系统唯一标识符，`ufsid` 的支持。这些标识符可以在 `/dev/ufsid` 目录中找到，它会在系统引导时自动创建。在 `/etc/fstab` 机制中，也可以使用 `ufsid` 标识符。也可以使用

`glabel status` 命令来取得与文件系统相关的 `ufsid` 列表：

```
% glabel status
      Name  Status  Components
ufsid/486b6fc38d330916      N/A  ad4s1d
ufsid/486b6fc16926168e      N/A  ad4s1f
```

在上面的例子中 `ad4s1d` 代表了 `/var` 文件系统，而 `ad4s1f` 代表了 `/usr` 文件系统。可以使用这些 `ufsid` 来挂载它，在 `/etc/fstab` 中配置类似：

```
/dev/ufsid/486b6fc38d330916      /var      ufs      rw      2      2
/dev/ufsid/486b6fc16926168e      /usr      ufs      rw      2      2
```

所有包含了 `ufsid` 的文件系统都可以用这种方式挂载，从而消除了需要手工创建永久性文件的麻烦，而又能提供与匿名无挂载方式的便利。

## 20.8. 通过 GEOM 使用 UFS 日志

随着 FreeBSD 7.0 的发布，提供了长期人所期待的日志功能的支持。这个支持采用了 GEOM 子系统，可以很容易地使用 [gjournal\(8\)](#) 工具来进行配置。

日志是什么？日志的作用是保存文件系统事务的记录，即言之，完成一次完整的磁盘写入操作所需的记录，这些记录会在元数据以及文件数据写入之前，写入到磁盘中。事务日志可以在随后用于重放并完成文件系统事务，以避免文件系统出现不一致的记录。

这个方法是一阻止文件系统丢失数据并产生不一致的方法。与 Soft Updates 追踪并保证元数据更新顺序的方法不同，它会立即地将日志保存到指定此任保留的磁盘空间上，在某些情况下可全部存放到块外一块磁头上。

与其他文件系统的日志不同，`gjournal` 采用的是基于块，而不是作为文件系统的一部分的方式 - 它只是作为一个 GEOM 扩展。

如果希望用 `gjournal`，FreeBSD 内核需要下列支持 - 这是 FreeBSD 7.0 以及更高版本系统上的默认配置：

```
options UFS_GJOURNAL
```

如果使用日志的卷需要在启动的时候被挂载，需加载 `geom_journal.ko` 内核模块，将以下行加入 `/boot/loader.conf`：

```
geom_journal_load="YES"
```

这个功能也可被包含在一个定制的内核，需在内核配置文件中加入以下行：

```
options GEOM_JOURNAL
```

在，可以清空的文件系统建立日志了。对于新的 SCSI 磁盘 da4，具体的操作如下：

```
# gjournal load
# gjournal label /dev/da4
```

然后，就会出一个与 /dev/da4 同名的 /dev/da4.journal 点。接下来，可以在这个点上建立文件系统：

```
# newfs -O 2 -J /dev/da4.journal
```

这个命令将建立一个包含日志的 UFS2 文件系统。

然后就可以用 `mount` 命令来挂载了：

```
# mount /dev/da4.journal /mnt
```



当磁盘包含多个 slice 时，每个 slice 上都会建立日志。例如，如果有 ad4s1 和 ad4s2 两个 slice，`gjournal` 会建立 ad4s1.journal 和 ad4s2.journal。

出于性能考虑，可能会希望在其他磁盘上保存日志。对于某些情形，可以在启用日志的后面，由日志提供者或存储。在存储的文件系统上，可以用 `tunefs` 来启用日志；不过，在修改文件系统之前，必须先行。多数情况下，如果无法建立的日志，`gjournal` 就会失败，并且不会防止由于不当使用 `tunefs` 而造成的数据丢失。

对于 FreeBSD 系统的磁盘使用日志也是可能的。参看 [Implementing UFS Journaling on a Desktop PC](#) 以获取更多信息。

# Chapter 21. 文件系统 Support

## 21.1. 概述

文件系统对于任何操作系统来说都是一个不可缺的部分。它允许用户在存储文件，提供数据的安全性，当然，是使硬件能具有新的用途。不同的操作系统通常都有一个共同的主要方面，那就是它原生的文件系统。在 FreeBSD 上，一个文件系统通常被称为快速文件系统或者 FFS，它是基于原来的 Unix™ 文件系统，通常也被称为 UFS。它是 FreeBSD 用于在磁盘上存储数据的原生的文件系统。

FreeBSD 也支持数量繁多的不同的文件系统，用于提供本地从其他操作系统上数据的支持，那些就是指存放在本地挂的 USB 存储，闪存和硬盘上的数据。它支持一些非原生的文件系统。有些文件系统是在其他的操作系统上运行的，像 Linux® 的扩展文件系统（EXT），和 Sun™ 的 Z 文件系统（ZFS）。

FreeBSD 上对于各文件系统的支持分成不同的层次。一些要求加入内核模块，另外的可能要求安装一系列的工具。这一章旨在帮助 FreeBSD 用户在他自己的系统上安装其他的文件系统，由 Sun™ 的 Z 文件系统开始。

在读了这一章之后，你将了解：

- 原生与被支持的文件系统之间的区别。
- FreeBSD 支持哪些文件系统。
- 如何启用，配置，安装和使用非原生的文件系统。

在开始本章以前，请：

- 了解 UNIX® 和 FreeBSD 基本知识 ([UNIX 基础](#))。
- 熟悉基本的内核配置/安装方法 ([配置 FreeBSD 的内核](#))。
- 熟悉在 FreeBSD 上安装第三方软件 ([安装应用程序](#), [Packages](#) 和 [Ports](#))。
- 熟悉 FreeBSD 上的磁盘，分区和命名 ([存储](#))。

## 21.2. Z 文件系统 (ZFS)

Z 文件系统是由 Sun™ 使用存储池方法的新技术。它就是只有在需要存储数据的空闲空间才会被使用。它也保证数据最大完整性而冗余的，支持数据快照，多拷贝和数据校验。它加了被称为 RAID-Z 的新的数据控制类型。RAID-Z 是类似于 RAID5 型，但被设计成防止写入漏洞。

### 21.2.1. 安装 ZFS

ZFS 子系统需利用到大量的系统资源，所以可能需要一些调整来日常应用提供最大化的效能。作为 FreeBSD 的一重要特性的特性，它可能在不久的将来有所变化；无论如何，下面的内容是我想推荐的：

#### 21.2.1.1. 内存

总的系统内存至少要有 1GB，推荐 2GB 或者更多。在此所有的例子中，我使用了 1GB 内存的系统并配合了一些恰当的调校。

有些人在少于 1GB 内存的环境下有幸正常使用，但是在内存有限的物理内存的条件下，当系统的内存很高时，FreeBSD 有可能因于内存耗尽而崩溃。

#### 21.2.1.2. 内核配置

我们建议把未使用的选项和参数从内核配置文件中去除。既然大部分的选项都有以模块的形式存在，它就可以很容易的通过 `/boot/loader.conf` 加载。

i386™ 架构的用户可以在内核配置文件中加入以下的选项，重新编译内核并重启机器：

```
options      KVA_PAGES=512
```

这个选项将扩展内核的地址空间，因而允许 `vm.kvm_size` 能超越 1 GB 的限制(PAE 2 GB)。为了输出这个选项最合适的值，把以兆(MB)单位所需的地址空间除以 4 得到。在这个例子中，512 约 2 GB。

#### 21.2.1.3. Loader 可选项参数

所有架构上 FreeBSD 都建议加大 `kmem` 地址空间。在有 1GB 物理内存的系统上，在 `/boot/loader.conf` 中加入如下的参数并且重启后通过了。

```
vm.kmem_size="330M"
vm.kmem_size_max="330M"
vfs.zfs.arc_max="40M"
vfs.zfs.vdev.cache.size="5M"
```

更多 ZFS 相关推荐学校的文档参看 <http://wiki.freebsd.org/ZFSTuningGuide>。

### 21.2.2. 使用 ZFS

FreeBSD 有一套机制能在系统初始化时挂载 ZFS 存储池。可以通过以下的命令设置：

```
# echo 'zfs_enable="YES"' >> /etc/rc.conf
# /etc/rc.d/zfs start
```

本文剩余的部分假定系统中有 3 个 SCSI 磁盘可用，它们的名称分别为 `da0`，`da1` 和 `da2`。IDE 硬件的用户可以使用 `ad` 代替 SCSI。

#### 21.2.2.1. 单个磁盘存储池

在单个磁盘上建立一个池，非冗余的 ZFS，使用 `zpool` 命令：

```
# zpool create example /dev/da0
```

可以通过 `df` 的输出查看新的存储池：

```
# 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
```

清楚地表明了 `example` 存池不能成功建立而且被挂掉了。我不能像普通文件系统那样使用它，就像以下例子中演示的那样，我不能在上面建立文件并：

```
# 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
```

遗憾的是这个存池并没有利用到 ZFS 的任何特性。在这个存池上建立一个文件系统，并用：

```
# zfs create example/compressed
# zfs set compression=gzip example/compressed
```

在 `example/compressed` 是一个用了的 ZFS 文件系统了。可以限制一些大的文件到 `/example/compressed`。

使用这个命令可以禁用：

```
# zfs set compression=off example/compressed
```

使用如下的命令卸载这个文件系统，并用 `df` 工具：

```
# 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
```

重新挂在这个文件系统使之能被，并用 `df` ：

```
# zfs mount example/compressed
# df
Filesystem      1K-blocks    Used   Avail Capacity  Mounted on
/dev/ad0s1a      2026030  235234  1628714     13      /
devfs              1         1         0    100    /dev
/dev/ad0s1d      54098308 1032864 48737580      2    /usr
example          17547008      0 17547008      0    /example
example/compressed 17547008      0 17547008      0    /example/compressed
```

存池与文件系统也可通过 `mount` 的输出看：

```
# mount
/dev/ad0s1a on / (ufs, local)
devfs on /dev (devfs, local)
/dev/ad0s1d on /usr (ufs, local, soft-updates)
example on /example (zfs, local)
example/data on /example/data (zfs, local)
example/compressed on /example/compressed (zfs, local)
```

正如前面所提到的，ZFS 文件系统，在创建之后就能像普通的文件系统那样使用。然而，它有很多其他的特性是可用的。在下面的例子中，我将创建一个新的文件系统，`data`。并要在上面存一些重要的文件，所以文件系统需要被置成把每一个数据都保存两拷：

```
# zfs create example/data
# zfs set copies=2 example/data
```

现在可以再次使用 `df` 看数据和空的使用状况：

```
# 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
```

注意存池上的每一个文件系统都有着相同数量的可用空间。这就是我在一些例子中使用 `df` 的原因，是了文件系统都是从相同的存池取得它所需的空。ZFS 去掉了如卷和分区此的概念，并允许多个文件系统占用同一个存池。不再需要文件系统与存池的时候能像它：

```
# zfs destroy example/compressed
# zfs destroy example/data
# zpool destroy example
```



磁盘无法避免的会坏掉和停止。当磁盘坏掉的时候，上面的数据都将丢失。一个避免因磁盘坏而丢失数据的方法是使用 RAID。ZFS 在它的存储池中支持的特性，便是下一节将探讨的。

#### 21.2.2.2. ZFS RAID-Z

正如前文中所提到的，这一章将假设存在 3 个 SCSI 盘，da0，da1 和 da2 (或者 ad0 和超出此例使用了 IDE 磁盘)。使用如下的命令建立一个 RAID-Z 存储池：

```
# zpool create storage raidz da0 da1 da2
```



Sun™ 推荐在一个 RAID-Z 配置中使用的磁盘数量 3 至 9 块。如果要求在单独的一个存储池中使用 10 块或更多的磁盘，考虑分拆成更小 RAID-z 池。如果只有 2 块磁盘，并仍然需要冗余，考虑使用 ZFS 的 mirror 特性。更多请参考 [zpool\(8\)](#) 手册。

`zpool storage` 至此就创建好了。可以如前文提到的那样使用 `mount(8)` 和 `df(1)` 命令。如需配置更多的磁盘，把它加到列表的后面。在存储池上创建一个叫 `home` 的文件系统，用它的文件最都将被保存在上面：

```
# zfs create storage/home
```

像前文中提到的那样，用它的目录与文件也可用并保存多拷贝，可通过如下的命令完成：

```
# zfs set copies=2 storage/home
# zfs set compression=gzip storage/home
```

把用的数据都拷贝来并创建一个符号链接，让他开始使用个新的目录：

```
# cp -rp /home/* /storage/home
# rm -rf /home /usr/home
# ln -s /storage/home /home
# ln -s /storage/home /usr/home
```

在用过的数据都保存在新创建的 `/storage/home` 上了。添加一个新用户并以它自身登录。

创建一个可日后用来回退的快照：

```
# zfs snapshot storage/home@08-30-08
```

注意快照将只会取一个真的文件系统，而不是某个用目录或文件。`@` 字符是文件系统名或卷名的分隔符。当用目录被破坏，可用如下命令恢复：

```
# zfs rollback storage/home@08-30-08
```

得到所有可用快照的列表，可使用 `ls` 命令看文件系统的 `.zfs/snapshot` 目录。例如，运行如下命令来看之前

取的快照：

```
# ls /storage/home/.zfs/snapshot
```

可以写一个脚本来定期取用数据的快照，久而久之，快照可能消耗掉大量的磁盘空间。之前建的快照可用以下命令除：

```
# zfs destroy storage/home@08-30-08
```

在所有这些之后，我没有理由再把 /store/home 放置了。它称真正的 /home 文件系统：

```
# zfs set mountpoint=/home storage/home
```

使用 `df` 和 `mount` 命令将示在系把我文件系真正当作了 /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
```

就基本完成了 RAID-Z 的配置了。使用夜 `periodic(8)` 取有文件系统建之的状态更新，行如下的命令：

```
# echo 'daily_status_zfs_enable="YES"' >> /etc/periodic.conf
```

### 21.2.2.3. 修 RAID-Z

一 RAID 都有它状的方法。ZFS 也不例外。可以使用如下的命令看 RAID-Z：

```
# zpool status -x
```

如果所有的存池于健康状并且一切正常的，将返回如下信息：

```
all pools are healthy
```

如果存在，可能是一个磁下，那返回的存池的状态将看上去是似个子的：

```
pool: storage
state: DEGRADED
status: One or more devices has been taken offline by the administrator.
        Sufficient replicas exist for the pool to continue functioning in a
        degraded state.
action: Online the device using 'zpool online' or replace the device with
        'zpool replace'.
scrub: none requested
config:

    NAME      STATE    READ WRITE CKSUM
    storage   DEGRADED    0     0     0
      raidz1  DEGRADED    0     0     0
        da0   ONLINE      0     0     0
        da1   OFFLINE      0     0     0
        da2   ONLINE      0     0     0

errors: No known data errors
```

在这个例子中，是由管理把此下后的状态。可以使用如下的命令将磁下：

```
# zpool offline storage da1
```

在切断系源之后就可以替下 da1 了。当系再次上，使用如下的命令替磁：

```
# zpool replace storage da1
```

至此可用不 -x 志的命令再次状态：

```
# 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，一切都显示正常。

#### 21.2.2.4. 数据校验

正如前面所提到的，ZFS 使用 **校验和(checksum)** 来存储数据的完整性。在文件系统建立自用的，可使用以下的命令禁用：

```
# zfs set checksum=off storage/home
```

这不是个明智的，因校验和 不仅非常有用而且只需占用少量的存储空间。并且用它也不会明显的消耗多资源。用后就可以 ZFS 使用校验和校验来数据的完整。这个程序通常称 "scrubbing"。可以使用以下的命令 **storage** 存储池里数据的完整性：

```
# zpool scrub storage
```

这个程序需花相当的时间，取决于存储的数据量。而且 I/O 非常密集，所以在任何时间只能进行一个的操作。在 scrub 完成之后，状态就会被更新，可使用如下的命令看：

```
# zpool status storage
pool: storage
state: ONLINE
scrub: scrub completed with 0 errors on Sat Aug 30 19:57:37 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
```

这个例子中完成得非常清楚。这个特性可以帮助在很短的一段时间内保证数据的完整。

Z 文件系统有更多的，参看 [zfs\(8\)](#) 和 [zpool\(8\)](#) 手册。

# Chapter 22. Vinum 卷管理程序

## 22.1. 概述

无论有什么样的磁盘，它都会有一些潜在问题：

- 它可能容量太小。
- 它可能速度太慢。
- 它可能也太不可靠。

为了解决这些问题，人们提出并实现了多种不同的解决方案。为了解决这些问题，一些用户采用了多个，有些甚至是冗余的磁盘方法。除了支持多种不同的硬件 RAID 控制器之外，FreeBSD 的基本系统中包括了 Vinum 卷管理器，它是一个用以虚拟磁盘的驱动程序。Vinum 是一种称做卷管理器，或者用于解决前面提到的三个问题的虚拟磁盘驱动程序。Vinum 能够提供比磁盘系统更好的活性、性能和可靠性，并能够单独或配合使用 RAID-0、RAID-1 和 RAID-5 模型。

这一章描述了磁盘存储的潜在问题，并介绍了 Vinum 卷管理器。



从 FreeBSD 5 开始，对 Vinum 进行了重写，以便使其符合 GEOM 架构 (GEOM 模块化磁盘框架)，同时保留其原有的意图、功能，以及保存在磁盘上的元数据格式。这一重写的版本称为 `gvinum` (表示 *GEOM vinum*)。接下来的文字中 *Vinum* 是一个抽象的名字，通常并不具体指某一特定的磁盘。新版本中所有的指令都通过 `gvinum` 命令来操作，而磁盘的内核模块的名字，也由 `vinum.ko` 改为了 `geom_vinum.ko`，而在 `/dev/vinum` 中的所有设备点，也改放到了 `/dev/gvinum`。从 FreeBSD 6 开始，旧版的 Vinum 已不再提供。

## 22.2. 磁盘容量太小

磁盘越大，存储的数据也就越多。常常会遇到需要一个比可用磁盘大得多的文件系统。无可否认，这个问题已没有十年前那么严峻了，但它仍然存在。通常建立一个在多个磁盘上存储数据的抽象，一些系统可以解决这个问题。

## 22.3. 瓶颈

现代系统常常需要一个高度并行的方式来访问数据。例如，巨大的 FTP 或 HTTP 服务器可以支持数以千计的并行会话，可以有多个达到外部世界的 100 Mbit/s，远远地超过了大多数磁盘的数据传输速率。

当前的磁盘驱动器最高可以以 70 MB/s 的速度传输数据，但这个问题在一个有多不受约束的进程在一个驱动器的环境中并不重要，它可能只完成了某些的一小部分。通常情况下，从磁盘子系统角度来看就更加有趣：重要的参数是在子系统上的负荷，这句话是占用了驱动器多少。

在任何磁盘中，驱动器必须先寻道，等待磁盘第一个扇区，然后进行读写。这些操作看起来可能很小：我们不会感到任何中断。

假设 10 kB 数据，在的高性能磁盘平均寻道时间是 3.5ms。最快的驱动器可以旋转在 15,000 rpm，所以平均地址寻道 2ms。在 70 MB/s 的速度下，数据的传输大约 150 μs，几乎无法和寻址相比。在一般情况下，高效的磁盘也会降低到 1 MB/s 虽然它的快慢依赖于所存储数据的大小。

由于多个磁盘的一般和明确的解决方法是采用 "多个磁盘":而不是只使用一个大磁盘, 它使用几个比小的磁盘合起来形成一个大的磁盘. 每个磁盘都可以独立地运行数据, 所以通过使用多个磁盘大大提高了数据吞吐量。

当然, 所要求的吞吐量的提高要比磁盘的数量小得多。 尽管多个磁盘并行运行数据, 但没有办法保证能平均分配到每个磁盘上。不可避免一个磁盘的访问可能比另一个要高得多。

磁盘的访问平衡很大程度依赖于磁盘上数据的共享方式。 在下面的图中, 将磁盘存储想象成一个巨大的数据扇区, 像一本书的页 那用页号来指定地址. 最明显的方法是把虚拟磁盘分成多个小的扇区, 每个扇区大小就是独立的磁盘大小, 用这种方法来存储数据, 就像把一本厚厚的书分成很多小的章节。 这个方法叫做 串盘 它有一个特点就是磁盘不需要有任何特定的大小关系。 当访问到的虚拟磁盘根据它的地址空间来分布的时候, 它能工作得很好。 当访问集中在一个比小的区域的的时候, 性能的提高没有显著的改变。 串盘图例说明了用串盘的方式来分配存储单元的次序。

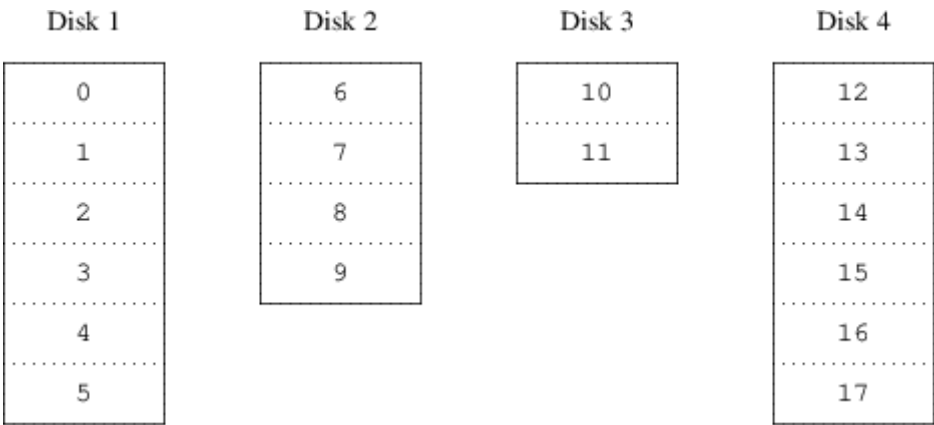


图 103. 串盘图

另外一种映射方法是把地址空间分布在比小的容量相同的磁盘上, 从而能在不同的磁盘上存储它。例如, 前256个扇区可能存储在第一个磁盘上, 接着的256个扇区存储在第二个磁盘上等等。 写完最后一个磁盘后, 程序会重复以前的工作, 直到所有的磁盘被写满。 这个映射叫做 分段(striping) 或者 RAID-0. 分段要求很精确地地址, 通过多个磁盘运行数据的时候, 它 可能会引起额外的I/O 开销, 但它也可能提供更多的性能。 分段图例显示了用分段形式分配的存储单元的次序。

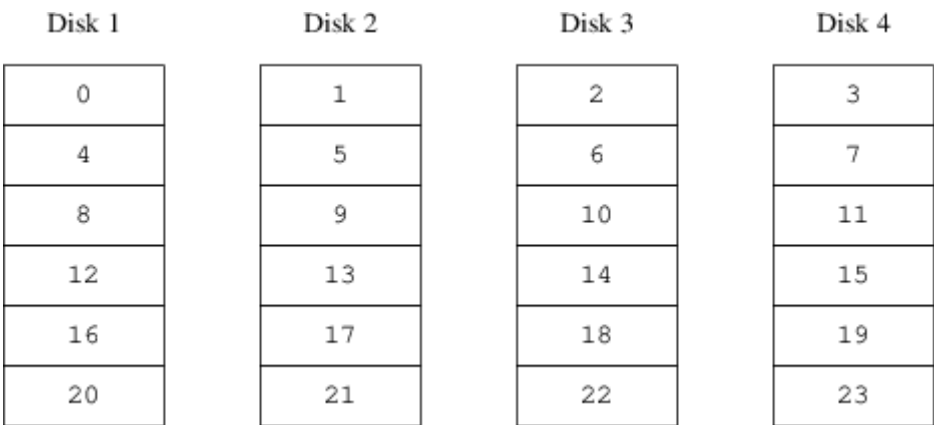


图 104. 分段图

## 22.4. 数据的完整性

磁盘的最后一个缺点是它不太可靠。 虽然磁盘驱动器的可靠性在过去几年有了很大的提高, 但它仍然是服务器中最容易损坏的核心部件。 当它产生故障的时候, 后果可能是毁灭性的: 替换坏的磁盘驱动器并恢复数据可能要花几天时间。

解决这个问题的方法是建立镜像，在不同的物理硬件上对数据做多个副本。根据 RAID 的输出顺序，这个技术也被叫做 RAID 0 或者 RAID-1。任何写到卷的数据也会被写到镜像上，所以可以从任何一个副本读取数据，如果其中有一个出现故障，数据也可以从其他磁盘上读到。

镜像有个问题：

- 价格. 它需要两倍的存储空间。
- 性能影响。写入操作必须在两个磁盘上进行，所以它要花两倍的IO。读取数据并不会影响性能：它甚至看起来会更快。

一个可行的方案采用 奇偶校验 的方式，用以实现 RAID 2、3、4 和 5。其中，RAID-5 是我所最感兴趣的。在 Vinum 的实现中，它是一个条带化的身体，其中，一个条带中都以一个字节的方式，来保存其它条带的奇偶校验。所以，RAID-5 plex 除了在条带中都包含了一个奇偶校验之外，每个 RAID-5 也就和普通的条带 plex 一样了。作为 RAID-5 的一个要求，奇偶校验在一个条带中的顺序都是不同的。数据块的编号，决定了它的相对位置。

| Disk 1 | Disk 2 | Disk 3 | Disk 4 |
|--------|--------|--------|--------|
| 0      | 1      | 2      | Parity |
| 3      | 4      | Parity | 5      |
| 6      | Parity | 7      | 8      |
| Parity | 9      | 10     | 11     |
| 12     | 13     | 14     | Parity |
| 15     | 16     | Parity | 17     |

图 105. RAID-5 的布局

与镜像相比，RAID-5 最显著的特点在于只需使用少得多的存储空间。读取类似于条带式存储的，但写入会慢得多，大约相当于性能的 25%。如果一个磁盘失效，整个阵列仍然可以在降级模式运行：读取来自正常的磁盘数据操作照常进行，但读取失效的磁盘的数据，来自于余下磁盘上相应的计算结果。

## 22.5. Vinum 目录

为了解决这些问题，Vinum 提出了一个四层的目录：

- 最显著的目录是虚拟磁盘，叫做 卷(volume)。卷本质上与一个UNIX 磁盘 设备有同样的属性，虽然它并不是完全一样。它没有大小的限制。
- 卷下面是 *plexes*，它是一个表示卷的所有地址空间。在每次IO中的每个水平都能提供冗余功能。可以把plex 想象成用一个对象排列的方式组织起来的独立磁盘，每个都包含同样的数据。
- 由于Vinum 存在于UNIX 磁盘存储框架中,所以它也可能使用UNIX 分区作为多个磁盘plex 的组成部分,但事实上并非不可:UNIX 磁盘只能有有限数量的分区。取而代之，Vinum 把一个物理的UNIX 分区 (the drive) 分解成叫做subdisks的相应区域,它可以使用一个来plex 建立。
- Subdisks 位于 Vinum 设备上,当前的UNIX 分区。Vinum 设备可以包含很多的subdisks。除了设备起始的一小块区域用来存储配置和描述信息以外，整个设备都可以用于存储数据。

下面的章节描述了一些目录提供了Vinum 所要求的功能的方法。



### 22.5.1. 卷的大小要求

在Vinum的配置中，Plex可以把多个subdisk 分布在所有的上。 果， 个独立的器的大小都不会限制plex 的大小，从而不会限制卷的大小

### 22.5.2. 多余的数据存

Vinum 通一个卷上多个plex 来完成象的功能。 个plex 是一个在一个卷中的数据的描述。一个卷可以包含一个 到八个plex。

然一个plex 描述了一个卷的所有数据，， 但可能描述的部分被物理地失了。可能是的 （没有plex 部分定一个subdisk）也可能是意外的故障 （由于器的故障致）。只要至少有一个plex 能 卷的完全地址提供数据，卷就能正常工作。

### 22.5.3. 性能

Vinum 在plex 水平既行串也行分段：

- 一个串的plex流使用 个subdisk 的地址空。
- 一个 分段的plex 在 个subdisk 上 分数据. Subdisk 必是大小一的，了从一个接的plex 中 区分它，必至少有 个subdisk。

### 22.5.4. plex 更有效？

FreeBSD 12.0提供的Vinum 版本能plex:

- 串的plex 更加活：它可以包含任何数量的subdisk， subdisk 也可能有不同的度。Plex 可以通过添加外的subdisk 来得到展。与分段 plex 不同，它需要的 CPU 更少，尽管 CPU 上的差是不可量的。一方面，它的可能不平衡，一个磁可能很重，而其他的可能很空。
- 分段(RAID-0) plexes 的最大点是它少了不平衡的情况: 通一个最合大小的分段 (大是256 kB), 甚至可以在各个成的器上降低。方法的缺点是在subdisk 上受到非常限制：它必是同大小, 通添加新的subdisk 来展一个plex 是非常的,以至Vinum 当前没有它. Vinum 利用一个外的, 代价不高的限制：一个分段的plex 必有至少 个subdisk，否， 它就无法区分接的 plex 了。

Vinum Plex 一下 个plex 的点和缺点.

表 8. Vinum Plex

| Plex 型 | 最少subdisks | 可否添加subdisks | 尺寸相同 | 用               |
|--------|------------|--------------|------|-----------------|
| 串      | 1          | 可以           | 不必   | 有很大性和中性能的大数据量存。 |
| 分段     | 2          | 不可以          | 必    | 大量并,具有高性能。      |

## 22.6. 一些例子

Vinum 着一个描述本系中象的配置数据。 始， 用可以在 [gvinum\(8\)](#) 工具来从若干配置文件生成配置数据。 Vinum 在其控制的 个磁分区 (在 Vinum 中称 device)

上都保存配置数据的副本。一旦数据在状态变化时都会更新，因而每个 Vinum 对象，都能恢复其状态。

### 22.6.1. 配置文件

配置文件描述了独立的 Vinum. 一个卷的定义可能是这样的：

```
drive a device /dev/da3h
volume myvol
plex org concat
sd length 512m drive a
```

这个文件描述了四个 Vinum 目：

- *drive* 行描述了一个磁盘分区（设备）和与下面的硬件相关的它的位置。它给出了一个符号名 *a*。这个与名称分开的符号名允许磁盘从一个位置移到另一个位置而不会混乱。
- *volume* 行描述了一个卷。唯一的必需属性是名称，在这个例子中是 *myvol*。
- *plex* 行定义了一个 plex。唯一需要的参数是名称，在这个例子中是 *concat*。没有名称是必然的：系统自动添加 suffix *.px* 来从卷名称生成一个名字，这里的 *x* 是在卷中的 plex 的序号。而每个 plex 将被叫做 *myvol.p0*。
- *sd* 行描述了一个 subdisk。最小的参数是存 subdisk 的设备名称，和 subdisk 的容量。对于 plex，没有名称也是必然的：系统自动添加 suffix *.sx* 来分配源自 plex 的名称，这里 *x* 是 plex 中 subdisk 的序号。Vinum 每个 subdisk 命名 *myvol.p0.s0*。

处理完这个文件后，[gvinum\(8\)](#) 会生成下面的输出：

```
# gvinum -> create config1
Configuration summary
Drives:      1 (4 configured)
Volumes:     1 (4 configured)
Plexes:      1 (8 configured)
Subdisks:    1 (16 configured)
```

|               |             |                  |                  |
|---------------|-------------|------------------|------------------|
| D a           | State: up   | Device /dev/da3h | Avail: 2061/2573 |
| MB (80%)      |             |                  |                  |
| V myvol       | State: up   | Plexes: 1        | Size: 512 MB     |
| P myvol.p0    | C State: up | Subdisks: 1      | Size: 512 MB     |
| S myvol.p0.s0 | State: up   | P0: 0 B          | Size: 512 MB     |

这些输出内容展示了 [gvinum\(8\)](#) 的主要列表格式。在一个新的 Vinum 卷 中用图形展示了这个配置。

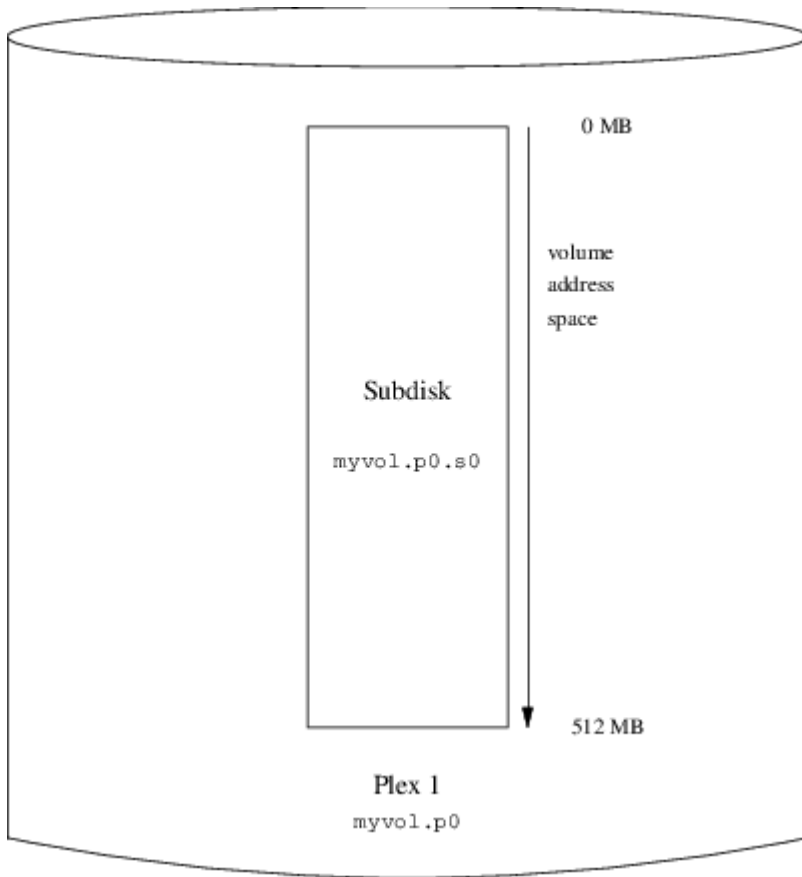


图 106. 一个典型的 Vinum 卷

下面这个图显示了一个由按顺序排列的 subdisk 组成的 plex。 在这个小小的例子中，卷包含一个 plex，plex 包含一个 subdisk。

这个卷本身和普通的磁盘分区相比并没有什么特别的优越性，它包含了一个 plex，因此不是冗余的。这个 plex 中包括了一个子磁盘，因此它和从磁盘分区分配内存没什么区别。接下来的几节，将介绍一些更有用的配置方法。

### 22.6.2. 提高可靠性：镜像

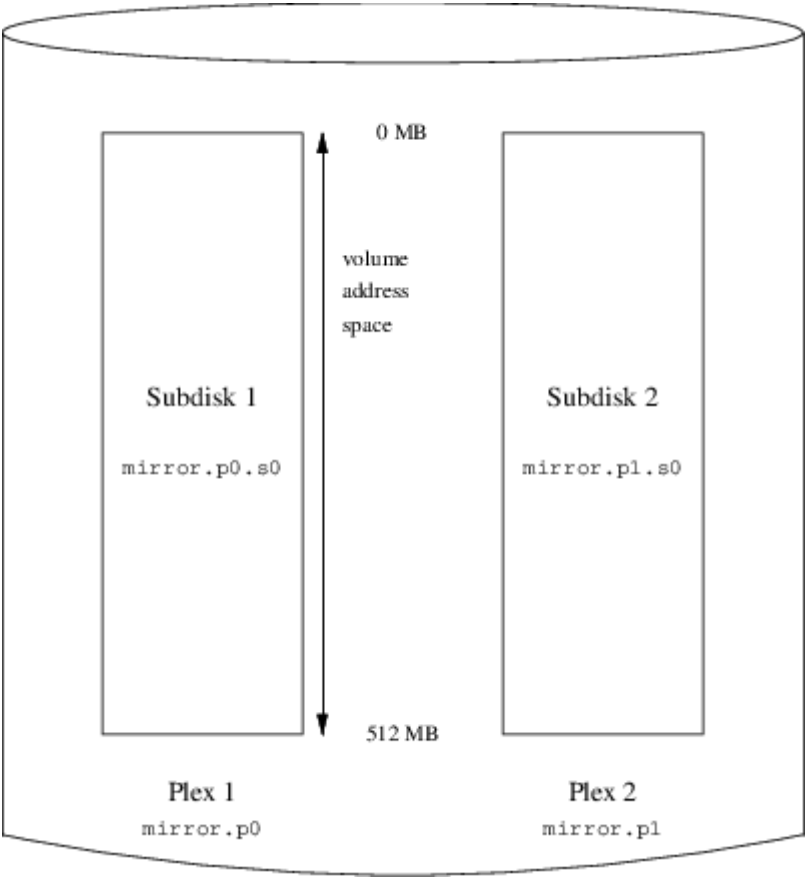
卷的可靠性可以通过镜像来提高。在配置镜像卷时，确保 plex 分布在不同的磁盘上十分重要，如果一个磁盘坏掉了，就不会影响到这个 plex。下面的配置将映射卷：

```
drive b device /dev/da4h
volume mirror
  plex org concat
    sd length 512m drive a
  plex org concat
    sd length 512m drive b
```

上面的例子中，并不需要再次指定磁盘 *a*，因为 Vinum 控制所有其配置数据的镜像。完成定义之后，配置如下所示：

|                |                       |                  |                  |  |
|----------------|-----------------------|------------------|------------------|--|
| Drives:        | 2 (4 configured)      |                  |                  |  |
| Volumes:       | 2 (4 configured)      |                  |                  |  |
| Plexes:        | 3 (8 configured)      |                  |                  |  |
| Subdisks:      | 3 (16 configured)     |                  |                  |  |
|                |                       |                  |                  |  |
| D a            | State: up             | Device /dev/da3h | Avail: 1549/2573 |  |
| MB (60%)       |                       |                  |                  |  |
| D b            | State: up             | Device /dev/da4h | Avail: 2061/2573 |  |
| MB (80%)       |                       |                  |                  |  |
|                |                       |                  |                  |  |
| V myvol        | State: up             | Plexes: 1        | Size: 512 MB     |  |
| V mirror       | State: up             | Plexes: 2        | Size: 512 MB     |  |
|                |                       |                  |                  |  |
| P myvol.p0     | C State: up           | Subdisks: 1      | Size: 512 MB     |  |
| P mirror.p0    | C State: up           | Subdisks: 1      | Size: 512 MB     |  |
| P mirror.p1    | C State: initializing | Subdisks: 1      | Size: 512 MB     |  |
| MB             |                       |                  |                  |  |
|                |                       |                  |                  |  |
| S myvol.p0.s0  | State: up             | P0: 0            | B Size: 512 MB   |  |
| S mirror.p0.s0 | State: up             | P0: 0            | B Size: 512 MB   |  |
| S mirror.p1.s0 | State: empty          | P0: 0            | B Size: 512 MB   |  |

像 Vinum 卷 以形方式展示了其。



107. 像 Vinum 卷

个例子中， 一个 plex 包含了完整的 512 MB 地址空。在前面的例子中， plex 只包括一个子。

### 22.6.3. 并行性能

前面例子中的镜像卷要比没有镜像的卷具有更好的容量能力， 但它的性能要差一些： 一次写入卷， 需要同时写到多个磁头上， 因而也就需要更大的磁盘空间。 如果希望非常好的性能， 需要另外一种方式： 不做镜像， 而将数据分成条放到尽可能多的、不同的磁头上。下面给出了一个跨越四个磁盘的 plex 卷：

```
drive c device /dev/da5h
drive d device /dev/da6h
volume stripe
plex org striped 512k
  sd length 128m drive a
  sd length 128m drive b
  sd length 128m drive c
  sd length 128m drive d
```

和之前类似， 并不需要定义 Vinum 已知的磁头。在完成定义之后， 将得到如下配置：

|                 |                       |                  |                  |  |
|-----------------|-----------------------|------------------|------------------|--|
| Drives:         | 4 (4 configured)      |                  |                  |  |
| Volumes:        | 3 (4 configured)      |                  |                  |  |
| Plexes:         | 4 (8 configured)      |                  |                  |  |
| Subdisks:       | 7 (16 configured)     |                  |                  |  |
|                 |                       |                  |                  |  |
| D a             | State: up             | Device /dev/da3h | Avail: 1421/2573 |  |
| MB (55%)        |                       |                  |                  |  |
| D b             | State: up             | Device /dev/da4h | Avail: 1933/2573 |  |
| MB (75%)        |                       |                  |                  |  |
| D c             | State: up             | Device /dev/da5h | Avail: 2445/2573 |  |
| MB (95%)        |                       |                  |                  |  |
| D d             | State: up             | Device /dev/da6h | Avail: 2445/2573 |  |
| MB (95%)        |                       |                  |                  |  |
|                 |                       |                  |                  |  |
| V myvol         | State: up             | Plexes: 1        | Size: 512 MB     |  |
| V mirror        | State: up             | Plexes: 2        | Size: 512 MB     |  |
| V striped       | State: up             | Plexes: 1        | Size: 512 MB     |  |
|                 |                       |                  |                  |  |
| P myvol.p0      | C State: up           | Subdisks: 1      | Size: 512 MB     |  |
| P mirror.p0     | C State: up           | Subdisks: 1      | Size: 512 MB     |  |
| P mirror.p1     | C State: initializing | Subdisks: 1      | Size: 512 MB     |  |
| MB              |                       |                  |                  |  |
| P striped.p1    | State: up             | Subdisks: 1      | Size: 512 MB     |  |
|                 |                       |                  |                  |  |
| S myvol.p0.s0   | State: up             | P0: 0 B          | Size: 512 MB     |  |
| S mirror.p0.s0  | State: up             | P0: 0 B          | Size: 512 MB     |  |
| S mirror.p1.s0  | State: empty          | P0: 0 B          | Size: 512 MB     |  |
| S striped.p0.s0 | State: up             | P0: 0 B          | Size: 128 MB     |  |
| S striped.p0.s1 | State: up             | P0: 512 kB       | Size: 128 MB     |  |
| S striped.p0.s2 | State: up             | P0: 1024 kB      | Size: 128 MB     |  |
| S striped.p0.s3 | State: up             | P0: 1536 kB      | Size: 128 MB     |  |

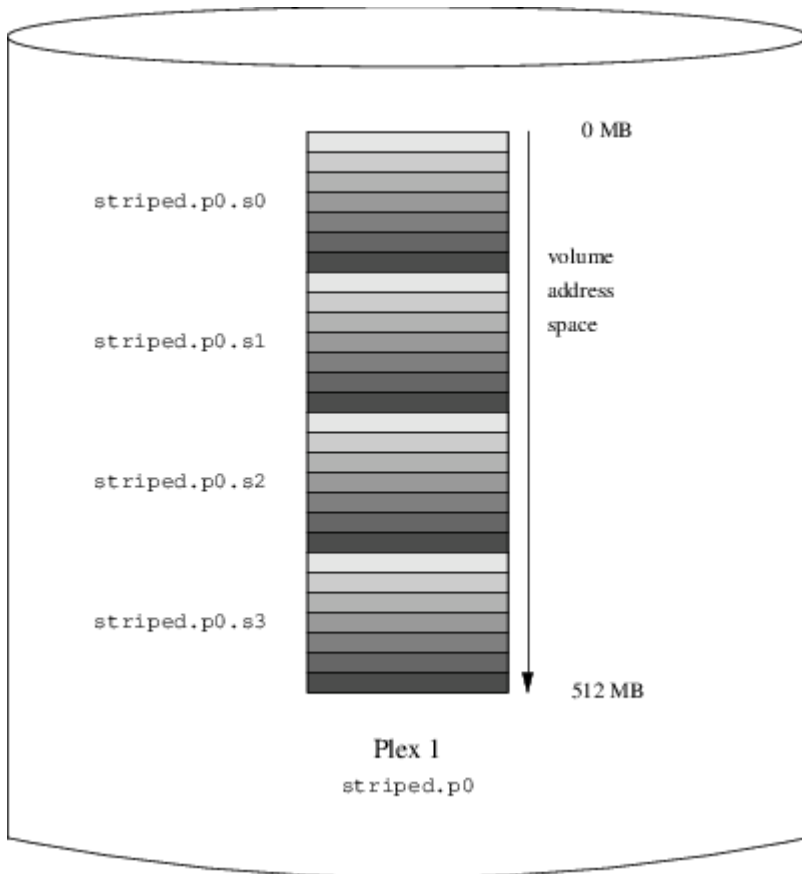


图 108. 条带化的 Vinum 卷

每个卷在条带化的 Vinum 卷中映射。条带的阴影部分，表示在 plex 地址空间中的位置：颜色最浅的在最前面，而最深的在最后。

## 22.6.4. 高性能容在

如果硬件足够多，也能构建比标准 UNIX® 分区同时提高了容量和性能的卷。典型的配置文件类似：

```
volume raid10
plex org striped 512k
  sd length 102480k drive a
  sd length 102480k drive b
  sd length 102480k drive c
  sd length 102480k drive d
  sd length 102480k drive e
plex org striped 512k
  sd length 102480k drive c
  sd length 102480k drive d
  sd length 102480k drive e
  sd length 102480k drive a
  sd length 102480k drive b
```

第二个 plex 中的子卷和第一个 plex 中的子卷了不同的驱动器：这能帮助确保即使同时使用不同的驱动器，写操作也不会同时发生在同一个驱动器上。

像并条带化的 Vinum 卷映射出了卷的映射。

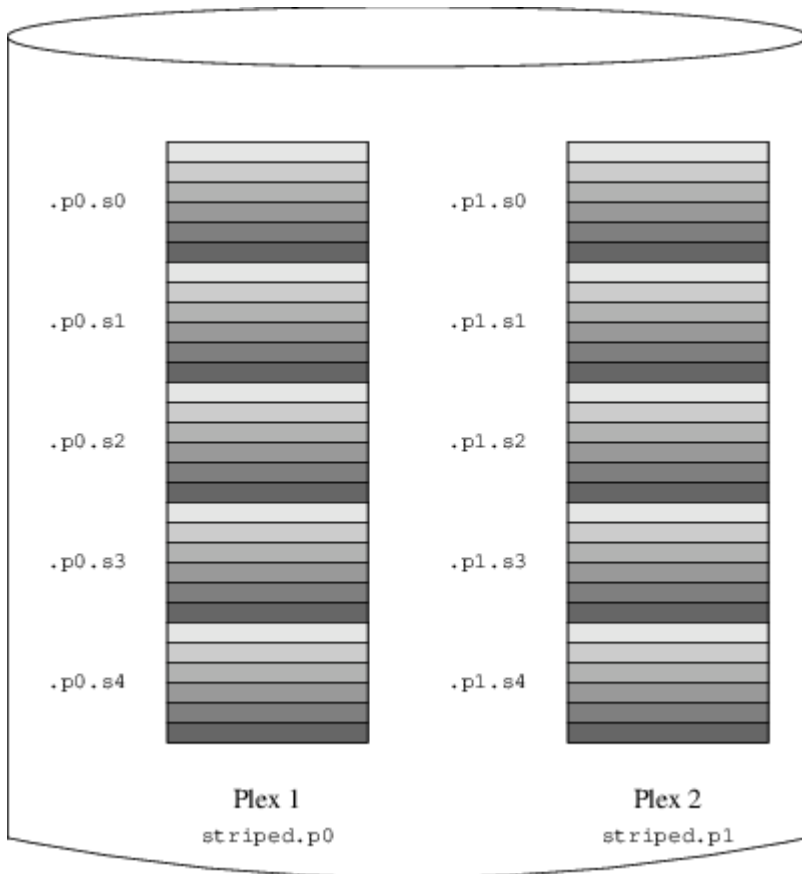


图 109. 像并条化的 Vinum 卷

## 22.7. 对象命名

如前面所描述的那， Vinum 会 plex 和子指定默认的名字， 而些名字也是可以定制的。 不推修改默认的名字： 使用允对象任意命名的 VERITAS 卷管理器的明， 一活性并没有来太多的好， 相反， 它很容易致象的混。

名字中可以包括任何非空白的字符， 但一般来， 建只使用字母、 数字和下。 卷、 plex， 以及子的名字， 可以包含最多 64 个字符， 而器的名字， 最可以使用 32 个字符。

Vinum 象会在 /dev/gvinum 之下生成点。 前述的配置将使 Vinum 建以下点：

- 个卷的点。 些是 Vinum 使用的主要。 因此， 前述配置包括下列： /dev/gvinum/myvol、 /dev/gvinum/mirror、 /dev/gvinum/striped、 /dev/gvinum/raid5 以及 /dev/gvinum/raid10。
- 所有卷的直接都存放在 /dev/gvinum/ 中。
- 目 /dev/gvinum/plex， 以及 /dev/gvinum/sd 中相地存放了个 plex 以及子的点。

例如， 考下面的配置文件：

```
drive drive1 device /dev/sd1h
drive drive2 device /dev/sd2h
drive drive3 device /dev/sd3h
drive drive4 device /dev/sd4h
volume s64 setupstate
    plex org striped 64k
        sd length 100m drive drive1
        sd length 100m drive drive2
        sd length 100m drive drive3
        sd length 100m drive drive4
```

理解完文件之后，[gvinum\(8\)](#) 将在 `/dev/gvinum` 中建立下面的内容：

```
drwxr-xr-x  2 root  wheel      512 Apr 13 16:46 plex
crwxr-xr--  1 root  wheel    91,  2 Apr 13 16:46 s64
drwxr-xr-x  2 root  wheel      512 Apr 13 16:46 sd

/dev/vinum/plex:
total 0
crwxr-xr--  1 root  wheel    25, 0x10000002 Apr 13 16:46 s64.p0

/dev/vinum/sd:
total 0
crwxr-xr--  1 root  wheel    91, 0x20000002 Apr 13 16:46 s64.p0.s0
crwxr-xr--  1 root  wheel    91, 0x20100002 Apr 13 16:46 s64.p0.s1
crwxr-xr--  1 root  wheel    91, 0x20200002 Apr 13 16:46 s64.p0.s2
crwxr-xr--  1 root  wheel    91, 0x20300002 Apr 13 16:46 s64.p0.s3
```

虽然 `plex` 和子项一般并不推荐指定名字，但是必须给 `Vinum` 项器命名。而且，当把项器移到不同的地方，它仍然能被自动地找出来。项器名最多可以包含 32 个字符。

### 22.7.1. 建立文件系统

对于系统而言，卷和磁盘是一样的。唯一的例外是，与 `UNIX®` 项器不同，`Vinum` 并不对卷进行分区，因而它也就不包含分区表。要求修改某些磁盘工具，特别是 [newfs\(8\)](#)，它会将对 `Vinum` 卷名当作分区。例如，磁盘项器的名字可能是 `/dev/ad0a` 或 `/dev/da2h`。有些名字分表示在第一个 (0) IDE (ad) 磁盘上的第一个分区 (a)，以及第三个 (2) SCSI 磁盘 (da) 上的第八个分区 (h)。而相比而言，`Vinum` 卷可能叫做 `/dev/gvinum/concat`，这个名字和分区名没有什么关系。

要在一个卷上建立文件系统，需要使用 [newfs\(8\)](#)：

```
# newfs /dev/gvinum/concat
```

## 22.8. 配置 Vinum

在 `GENERIC` 内核中，并不包含 `Vinum`。可以创建一个定制的包含 `Vinum` 的内核，然而并不推荐这样做。而且



Vinum 的加载方法，是使用内核模块 (kld)。甚至不需要使用 `kldload(8)` 来加载 Vinum：在 `gvinum(8)` 中，它会检查模块是否已加载，如果没有，会自动地加载它。

### 22.8.1. 磁盘

Vinum 将配置信息，采用与配置文件一样的形式来存放到磁盘分区上。当从配置数据中读取，Vinum 会看到一系列在配置文件中不可用的磁盘。例如，磁盘配置文件可能包含下面的文字：

```
volume myvol state up
volume bigraid state down
plex name myvol.p0 state up org concat vol myvol
plex name myvol.p1 state up org concat vol myvol
plex name myvol.p2 state init org striped 512b vol myvol
plex name bigraid.p0 state initializing org raid5 512b vol bigraid
sd name myvol.p0.s0 drive a plex myvol.p0 state up len 1048576b driveoffset 265b
plexoffset 0b
sd name myvol.p0.s1 drive b plex myvol.p0 state up len 1048576b driveoffset 265b
plexoffset 1048576b
sd name myvol.p1.s0 drive c plex myvol.p1 state up len 1048576b driveoffset 265b
plexoffset 0b
sd name myvol.p1.s1 drive d plex myvol.p1 state up len 1048576b driveoffset 265b
plexoffset 1048576b
sd name myvol.p2.s0 drive a plex myvol.p2 state init len 524288b driveoffset 1048841b
plexoffset 0b
sd name myvol.p2.s1 drive b plex myvol.p2 state init len 524288b driveoffset 1048841b
plexoffset 524288b
sd name myvol.p2.s2 drive c plex myvol.p2 state init len 524288b driveoffset 1048841b
plexoffset 1048576b
sd name myvol.p2.s3 drive d plex myvol.p2 state init len 524288b driveoffset 1048841b
plexoffset 1572864b
sd name bigraid.p0.s0 drive a plex bigraid.p0 state initializing len 4194304b driveoffset
1573129b plexoffset 0b
sd name bigraid.p0.s1 drive b plex bigraid.p0 state initializing len 4194304b driveoffset
1573129b plexoffset 4194304b
sd name bigraid.p0.s2 drive c plex bigraid.p0 state initializing len 4194304b driveoffset
1573129b plexoffset 8388608b
sd name bigraid.p0.s3 drive d plex bigraid.p0 state initializing len 4194304b driveoffset
1573129b plexoffset 12582912b
sd name bigraid.p0.s4 drive e plex bigraid.p0 state initializing len 4194304b driveoffset
1573129b plexoffset 16777216b
```

其中最明显的区别是，指定了配置的位置信息、名称（有些在配置文件中是不可用的，但不鼓励用自行指定）以及状态信息（是用不能指定的）。Vinum 并不在配置信息中保存关于磁盘器的信息：它会扫描已配置的磁盘中包含 Vinum 的分区。这使得 Vinum 能在 UNIX® 磁盘器被指定了不同的 ID 也能正确识别它。

#### 22.8.1.1. 自启动

`Gvinum` 在通过 `loader.conf(5)` 加载了内核模块之后就能自启动。在添加 `Gvinum` 模块，需在 `/boot/loader.conf` 中加入 `geom_vinum_load="YES"`。

当使用 `gvinum start` 命令来启动 Vinum 时，Vinum 会从某一个 Vinum 设备中读取配置数据。正常情况下，每个设备上都包含了同设备的配置数据副本，因此从每个设备上读取是无所问题的。但是，在系统崩溃之后，Vinum 就必须从一个设备上的配置数据是最新的，并从上面读取配置。如果需要，它会更新其它设备上的配置。

## 22.9. 使用 Vinum 作根文件系统

如果文件系统使用完全像的 Vinum 配置，有人也会希望根文件系统也作了像。像配置要比像其它文件系统麻烦一些，因为：

- 根文件系统在引导过程中很早的时候就必须处于可用状态，因此 Vinum 的基施在第一时间就可用了。
- 包含根文件系统的卷，同时也保存了系统的引导程序和内核，因此它必须能被宿主系统的内建工具（例如 PC 机的 BIOS）访问，而通常是没有办法让它了解 Vinum 的结构的。

下面几章中，我们“根卷”包含根文件系统的 Vinum 卷。把该卷命名“`root`”可能是个不好的主意，不仅从技术上讲，并不严格地要求这么做。不过，接下来的命令例子都使用该名字。

### 22.9.1. 及早启动 Vinum 以满足根文件系统的要求

有多依赖于它的尺度：

- Vinum 必须在启动时可以被内核使用。因此，在自举中所介绍的方法，也就无法满足一任性的需要了。在接下来的配置中，也不能设置 `start_vinum` 参数。第一方法是通将 Vinum 静置到内核中来，即，它就在任何时候都可用了，当然一般并不需要。第二方法是通 `/boot/loader`（[第三段](#)，`/boot/loader`）来尽早加载 vinum 内核模块，即操作发生在内核加载之前。可以通将下面的配置：

```
geom_vinum_load="YES"
```

加入到 `/boot/loader.conf` 文件中来。

- 对于 `Gvinum` 而言，所有的操作都是在内核模块加载时自行执行的，因此上面的操作，也就是所要执行的全部工作了。

### 22.9.2. 基于 Vinum 的卷在引导时可以启动

因目前的 FreeBSD 引导程序只有 7.5 KB 的代码，并且已承担了从 UFS 文件系统中读取文件（例如 `/boot/loader`）的重任，因此完全没有办法再让它去分析 Vinum 配置数据中的 Vinum 结构，并得到引导卷本身的信息。因此，需要一些技巧来引导代码提供标准的“`a`”分区，而它包含了根文件系统。

要有些得以实现，根卷需要满足下面的条件：

- 根卷不能是条卷或 RAID-5 卷。
- 根卷 plex 不能包含间接的子。

需要明确的是，使用多个 plex，每个 plex 都制作根文件系统的副本，是需要而且是可行的。然而，引导程序只能使用这些副本中的一个来引导系统，直到内核自行挂接根文件系统为止。这些 plex 中的每个子，在之后会有它自己的“`a`”分区，以表示一个可以引导的。一个“`a`”分区，尽管并不需要和其它包含根卷的 plex 位于各自设备的同一位置。但是，创建 Vinum 卷使得像卷相互称，从而能避免了混。

为了建立一个根卷的 "a" 分区，需要完成下面的操作：

1. 使用下面的命令来了解根卷成子的位置 (从开始的偏移量) 和尺寸：

```
# gvinum l -rv root
```

需要注意的是，Vinum 偏移量和尺寸的位是字。它必须是 512 的整数倍，才能得到 `bsdlabel` 命令所需的号。

2. 在一个根卷成上，行命令：

```
# bsdlabel -e devname
```

其中，于没有 slice (也就是 fdisk) 表的磁，*devname* 必须是磁的名字 (例如 da0)，或者是 slice 的名字 (例如 ad0s1)。

如果上已经有了 "a" 分区 (比如，包含 Vinum 之前的根文件系统)，改其它的名字，以便 (如果需要的)，但它并不会用于系。注意，活的分区 (似正挂接的根文件系统) 不能被改名，因此，要完成工作，必须从 "Fixit" 或者分操作，并 (在像情形中) 首先操作那些非引。

然后，上 Vinum 分区的偏移 (如果有的) 必须加到个上根卷的子。其果，将成新的 "a" 分区的 "offset" 。个分区的 "size" ，可以根据前面的配置算得出。"fstype" 是 4.2BSD。"fsize"、"bsize"，以及 "cp9" ，与文件系的匹配，尽管在配置 Vinum 并不重要。

，新的 "a" 分区，将建并覆一上的 Vinum 分区的。注意，`bsdlabel` 只有在 Vinum 分区的 fstype 被 "vinum" ，才允许做。

3. 就成了！所有的 "a" 分区在都已存在，而且是根卷的一副本。烈建再次其果，方法是：

```
# fsck -n /dev/devnamea
```

必须注意，所有包含控制信息的文件，都必须放到 Vinum 卷上的根文件系统。在新的 Vinum 根卷，它可能和在用的根文件系统不匹配。因此，`/etc/fstab` 和 `/boot/loader.conf` 个文件需要特地注意。

在下次重，引程序需要从新的基于 Vinum 的根文件系统中取当的控制信息，并据此工作。在内核初始化程的尾部分，在所有的都被宣示之后，如果示了下面的信息，表示配置成功：

```
Mounting root from ufs:/dev/gvinum/root
```

### 22.9.3. 基于 Vinum 的根文件系统配置示例

在 Vinum 根卷配置好之后，`gvinum l -rv root` 的输出可能如下面的样子：

```
...
Subdisk root.p0.s0:
    Size:          125829120 bytes (120 MB)
    State: up
    Plex root.p0 at offset 0 (0 B)
    Drive disk0 (/dev/da0h) at offset 135680 (132 kB)

Subdisk root.p1.s0:
    Size:          125829120 bytes (120 MB)
    State: up
    Plex root.p1 at offset 0 (0 B)
    Drive disk1 (/dev/da1h) at offset 135680 (132 kB)
```

需要注意的是 **135680**，也就是偏移量（相当于 `/dev/da0h` 分区）。它相当于 `bsdlabel` 方法中的 265 个 512-字节的磁道。类似地，根卷的尺寸是 245760 个 512-字节的磁道。`/dev/da1h` 中，包含了根卷的第二个副本，采用了同样的配置。

下面的 `bsdlabel` 输出如下面的样子：

```
...
8 partitions:
#      size  offset  fstype  [fsize bsize bps/cpg]
a:   245760    281   4.2BSD    2048 16384    0 # (Cyl. 0*- 15*)
c: 71771688     0  unused      0     0    0 # (Cyl. 0 - 4467*)
h: 71771672    16   vinum             # (Cyl. 0*- 4467*)
```

可以看到，装的 "a" 分区的 "size" 参数和前面的一致，而 "offset" 参数是 Vinum 分区 "h"，以及其中一分区（或 slice）的偏移量之和。这是一种典型的配置，它能避免在无法引导程序产生 panic 中介断的崩溃。此外，我也看到整个 "a" 分区完全位于其上包含了 Vinum 数据的 "h" 分区之中。

注意，在上面的配置中，整个磁盘都是 Vinum 使用的，而且没有留下 Vinum 之前的根分区，因为它永久性地成为了新建的 Vinum 配置中的一个子项。

### 22.9.4. 故障排除

如果遇到了问题，你需要从中恢复的方法。下面列出了一些常见缺陷，及其解决方法。

#### 22.9.4.1. 系统的引导程序加载了，但无法启动

如果由于某些原因系统不再启动，引导程序可以在 10-秒 倒计时的时候，按 `space` 键来停止。加载器变量（例如 `vinum.autostart`）可以通过使用 `show` 命令来看，并使用 `set` 和 `unset` 命令来设置。

如果遇到的问题是由于 Vinum 的内核模块没有列入加载的列表，而没有正确加载，那么使用 `load geom_vinum` 会有所帮助。

此后，可以使用 `boot -as` 来重启程序。 `-as` 会要求内核所挂载的根文件系统 (`-a`)，并使引导程序在单用户模式停止 (`-s`)，此根文件系统是以只读方式挂载的。因此，即使只挂载了多 plex 卷中的一个 plex，也不会引致 plex 之数据不一致的问题。

当提示输入要挂载的根文件系统时，可以输入任何一个包含根文件系统的名称。如果正确地配置了 `/etc/fstab`，默认的名称是类似 `ufs:/dev/gvinum/root`。一般可以使用类似 `ufs:da0d` 的名称来代替它，因为它通常包括了 Vinum 之前的根文件系统。需要注意的是，如果在里面输入了 "a" 分区，它可能表示的名称上是 Vinum 根卷的一个子卷，而在镜像式配置中，它只会挂载镜像的根卷中的一个。如果之后将这个文件系统以读写方式挂载，则需要从 Vinum 根卷中去其他的 plex，否则这些卷中可能会包含不一致的数据。

#### 22.9.4.2. 只加载了主引导程序

如果 `/boot/loader` 加载失败，而主引导程序加载正常（在 BIOS，屏幕最左端有一列有一个旋转的 D），可以在此处中断主引导程序的进程，方法是按 `space` 键。它将在引导的第二阶段处停，具体可以参见 [第一段](#)，[/boot/boot1](#)，和 [第二段](#)，[/boot/boot2](#)。此时，可以从一个分区，例如原先包含根文件系统，并不再叫作 "a" 的那个分区，启动。

#### 22.9.4.3. 无法启动，引导程序产生 panic

这种情况一般是由于 Vinum 安装过程中破坏了引导程序造成的。不幸的是，Vinum 目前只在分区起始的地方保留了 4 KB 的空闲，之后就起始写 Vinum 的信息了。然而，目前第一阶段的引导程序，加上 `bsdlabeled` 嵌入的内容需要 8 KB。因此，如果 Vinum 分区从偏移量 0 开始，而该 slice 或磁盘能容纳，则 Vinum 的安装将覆盖掉引导程序。

类似地，如果从上述情形中恢复，例如，从 "Fixit" 启动，并通 `bsdlabeled -B` 按照 [第一段](#)，[/boot/boot1](#)，和 [第二段](#)，[/boot/boot2](#) 中介的方法来恢复引导程序，则引导程序会覆盖掉 Vinum，因此 Vinum 也就得不到它的磁盘了。尽管它并不会真的覆盖掉 Vinum 的配置数据，或者 Vinum 卷上的数据，并且可以通过输入一模一样的 Vinum 配置数据来恢复，但从该状况中完全恢复是非常困难的。要真正解决问题，必须将整个 Vinum 分区向后移动至少 4 KB，以便使 Vinum 和系统的引导程序不再冲突。

# Chapter 23. 虚拟化

## 23.1. 概述

虚拟化软件能`在`同一台机器上`同时`运行多个操作系统。在 PC 上，`系统`通常由一个`运行`虚拟化软件的宿主操作系统，以及一系列`客户`操作系统`组成`。

读完本章，`你将`了解：

- 宿主操作系统与`客户`操作系统的区别。
- 如何在采用 Intel® 处理器的 Apple® Macintosh® 计算机上安装 FreeBSD。
- 如何在 Microsoft® Windows® 以 Virtual PC 安装 FreeBSD。
- 如何在`虚拟化`环境`中` FreeBSD 系统`运行`性能`如何`。

在`本章`之前，`你将`：

- 理解 UNIX® 和 FreeBSD 的`基础知识` ([UNIX 基础](#))。
- 了解如何安装 FreeBSD ([安装 FreeBSD](#))。
- 了解如何配置网络`连接` ([高级网络](#))。
- 了解如何安装第三方`软件` ([安装应用程序](#), [Packages](#) 和 [Ports](#))。

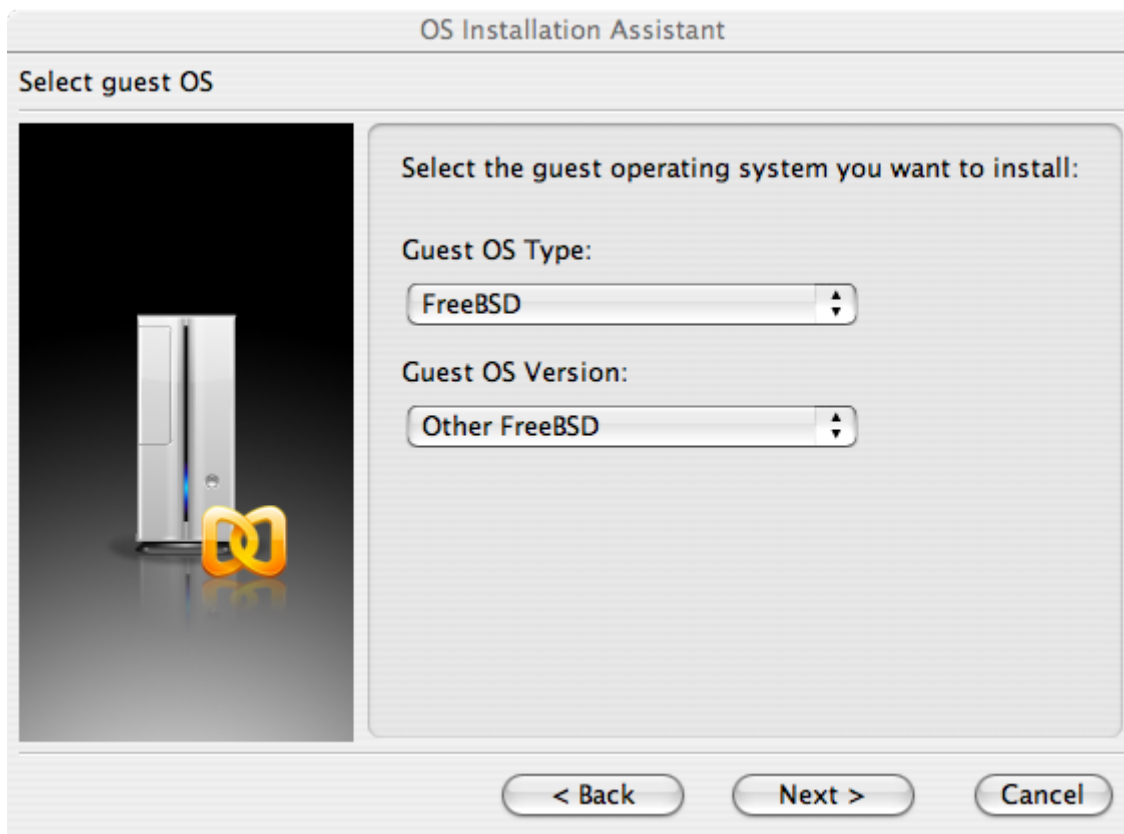
## 23.2. 作为`客户` OS 的 FreeBSD

### 23.2.1. MacOS 上的 Parallels

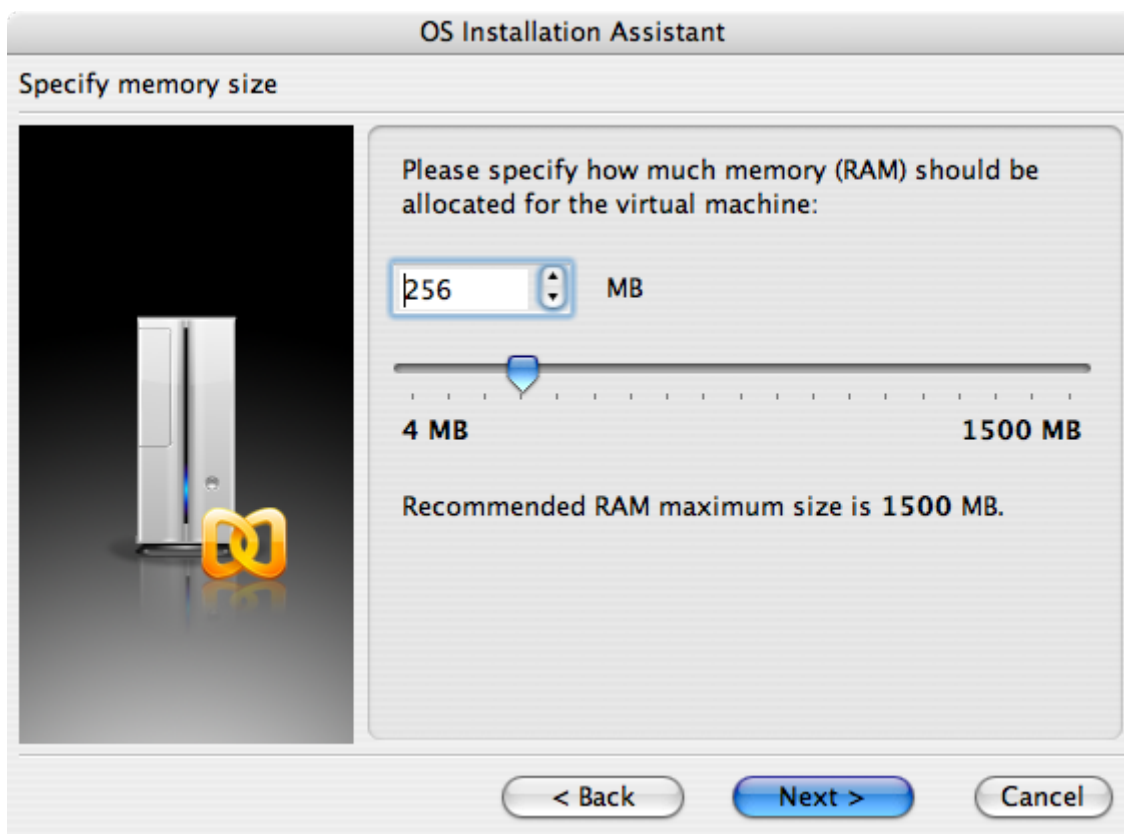
`Mac` 上的 Parallels Desktop 是一个`可用于`采用 Intel® 处理器，并`运行` Mac OS® 10.4.6 或更高版本的 Apple® Mac® 计算机的`商业软件`。它`为` FreeBSD 系统`提供了`完整的支持。在 Mac OS® X 上安装了`它`之后，`用户`需要配置`虚拟机`并安装所需的`客户`操作系统。

#### 23.2.1.1. 在 Parallels/Mac OS® X 上安装 FreeBSD

在 Mac OS® X/Parallels 上安装 FreeBSD 的第一步是`创建`一个新的`虚拟机`。在`系统`提示`选择`客户 OS 类型 (Guest OS Type) `时`选择 FreeBSD，并根据`使用` FreeBSD `虚拟机`例的需要分配`磁盘`和内存：



□多数在 Parallels 上使用 FreeBSD 的情形而言，4GB 磁□空□和 512MB 的 RAM 就□用了：





## OS Installation Assistant

### Select action type



Please specify what kind of hard disk you want to install to the virtual machine. If you do not want to add a hard disk now, select "Do not add hard disk" option. You will be able to add it later using a Configuration Editor.

- ☒ Create a new virtual hard disk
- ☐ Use an existing hard disk image
- ☐ Do not add hard disk

< Back

Next >

Cancel

## OS Installation Assistant

### Specify hard disk options



Please specify a preferred virtual disk size:

8000 MB

Select disk format:

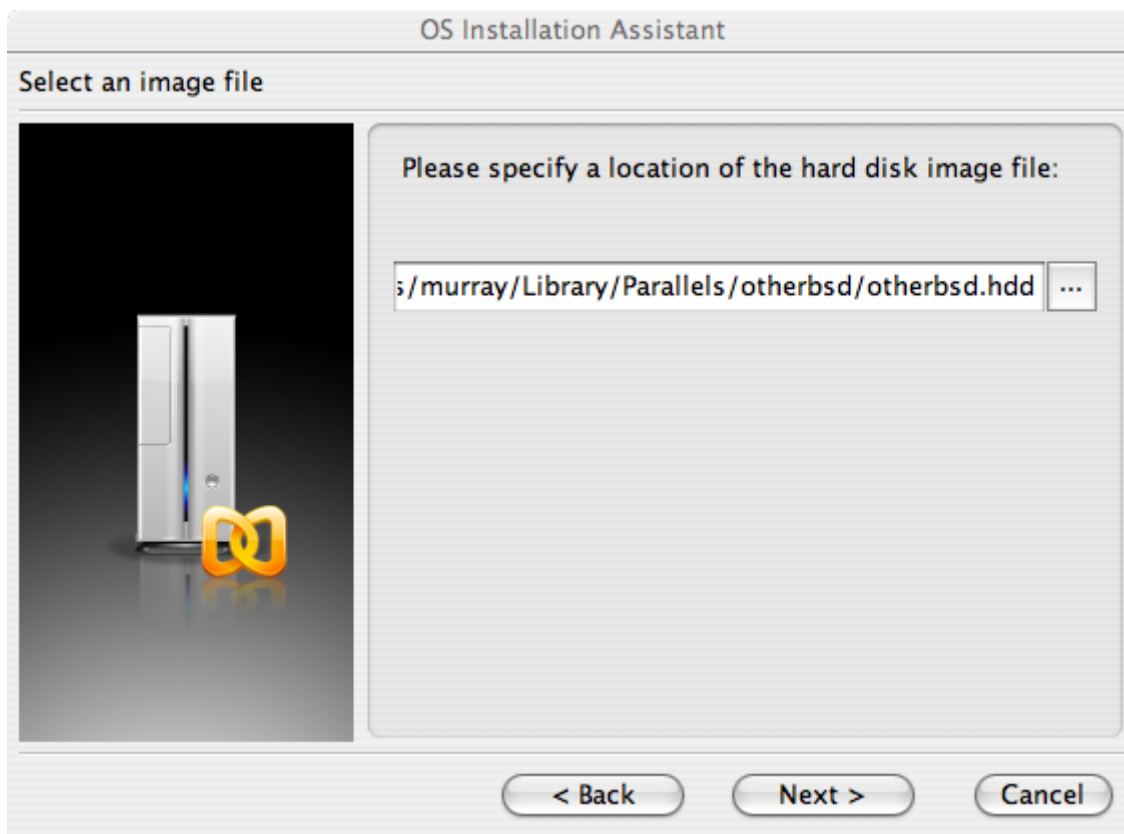
- ☐ Expanding (recommended).  
Disk image file is small initially and grows as you add more data to the virtual machine. This disk format takes less time to create and saves disk space on the host.
- ☒ Plain.  
Disk image file consumes all the allocated space right from the start. It takes more time to create but allows guest OS to operate faster.

< Back

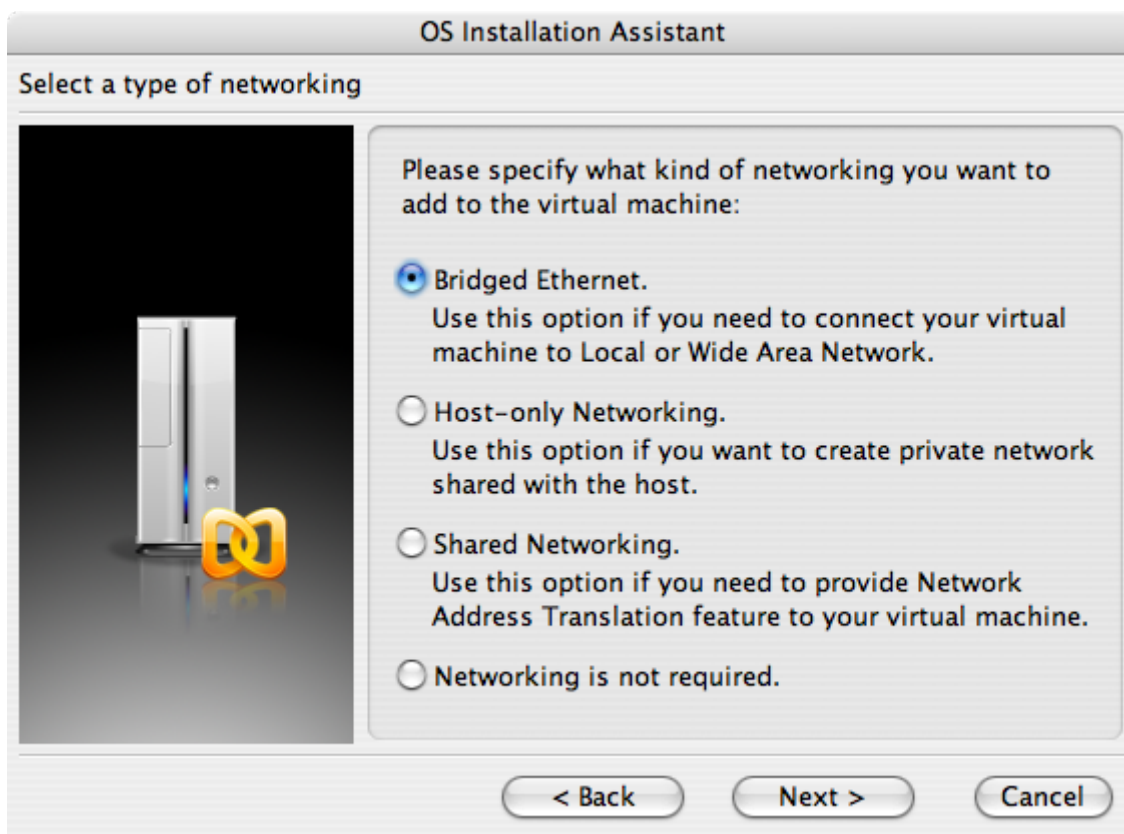
Next >

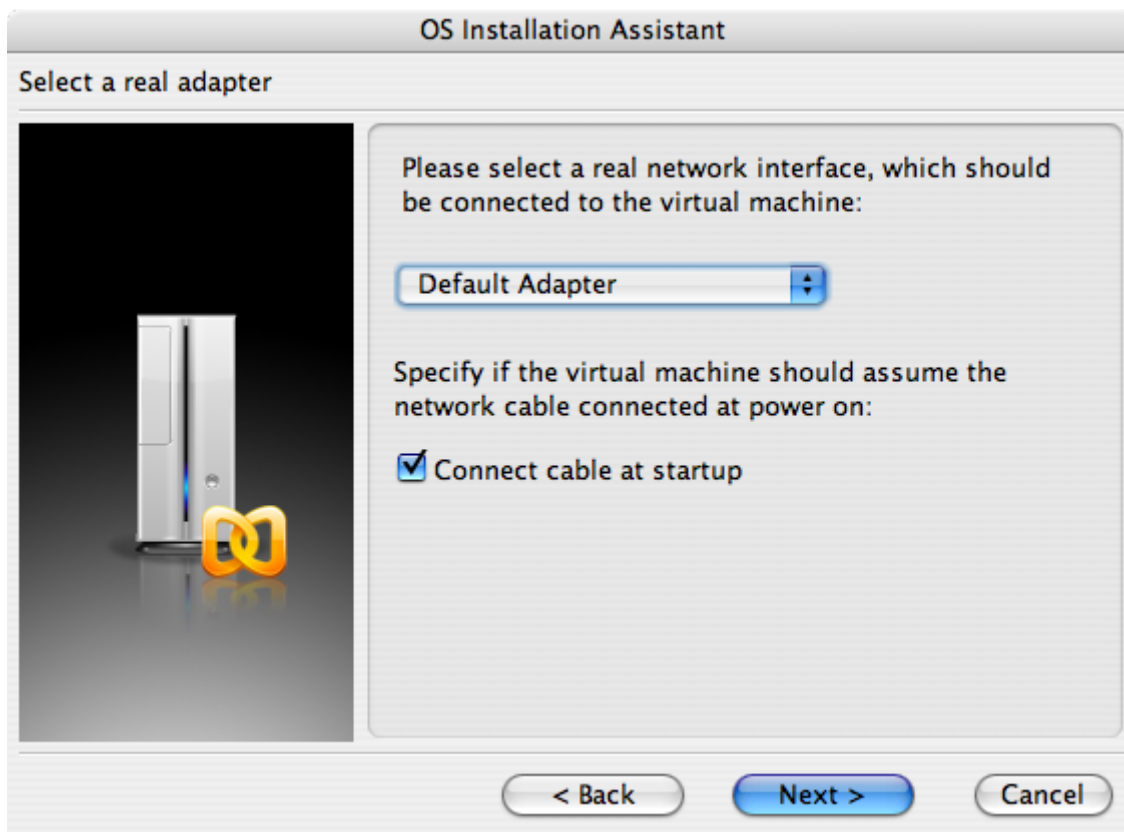
Cancel



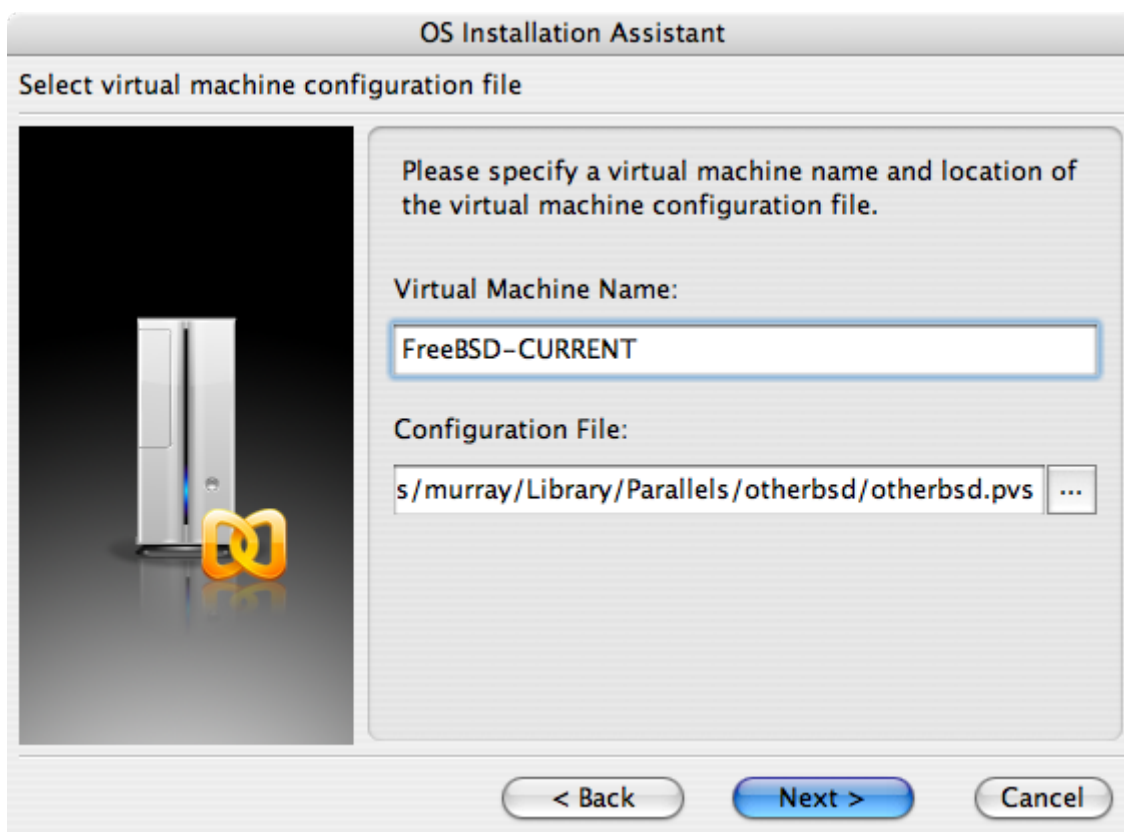


要使用的网络和网桥类型：



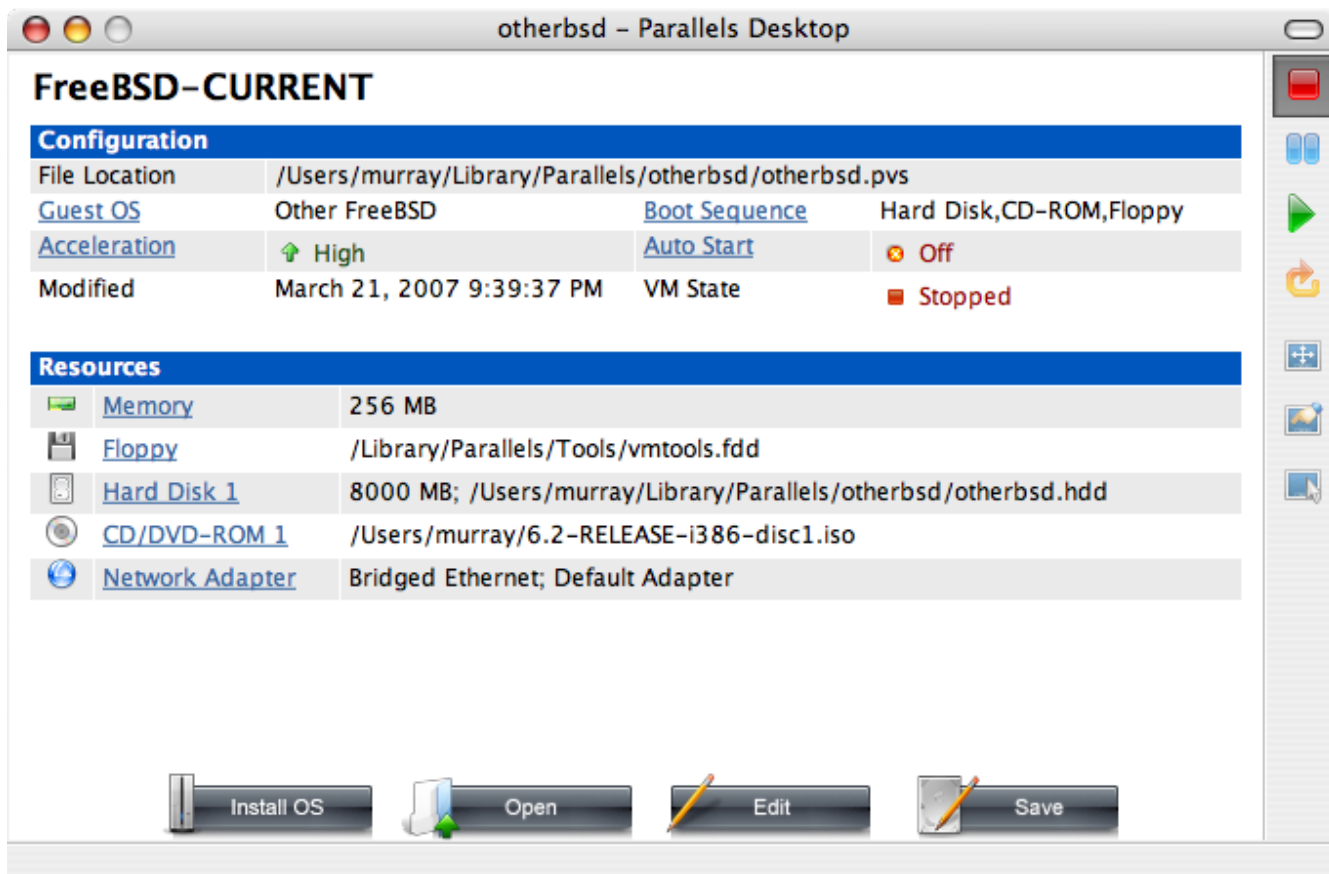


保存并完成配置：



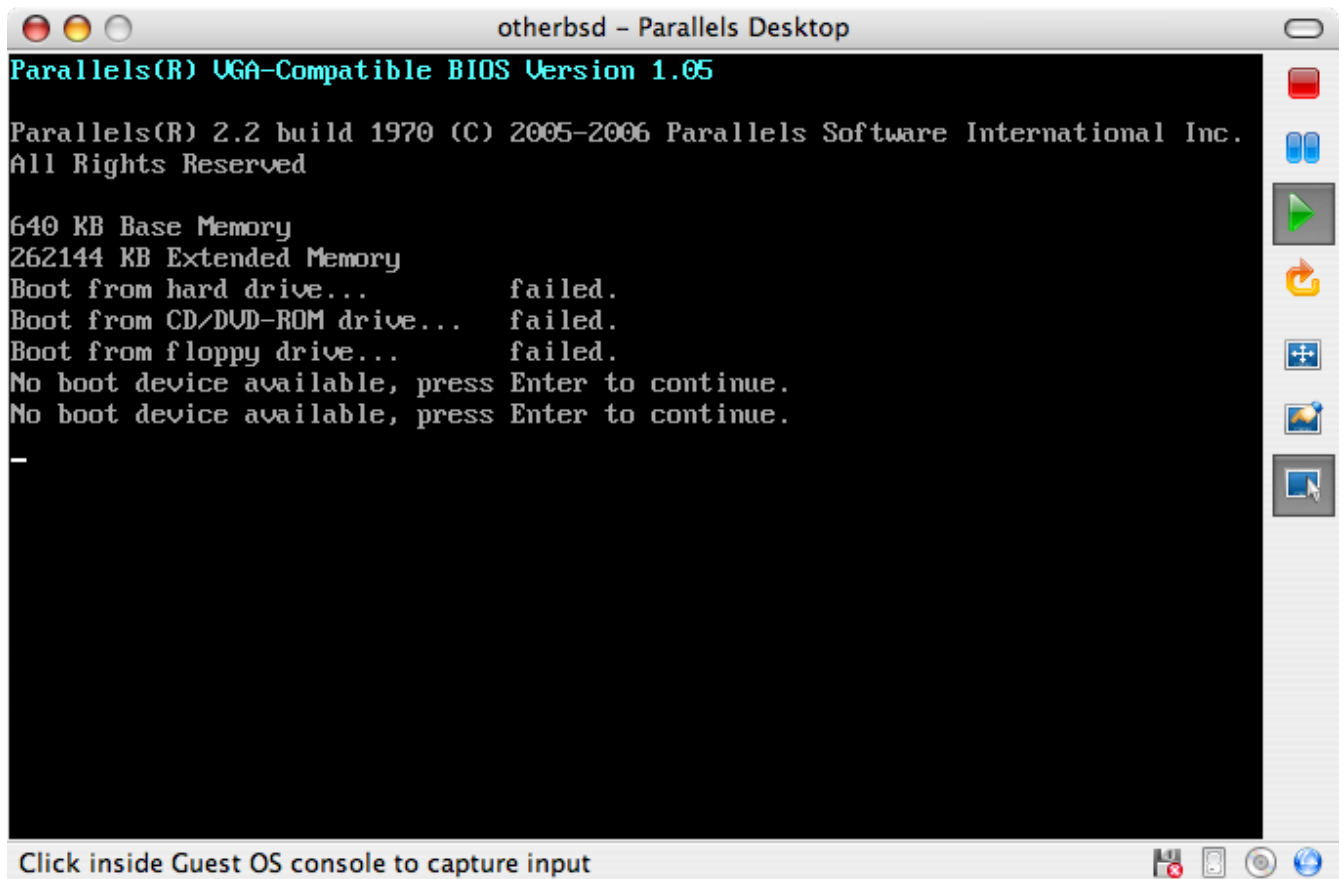


在创建了 FreeBSD 虚拟机之后，需要在其中安装 FreeBSD。最好的做法是使用官方的 FreeBSD CDROM 或从官方 FTP 站点下载的 ISO 映像来完成这个任务。如果您的本地 Mac® 文件系统中有 ISO 映像文件，或您的 Mac® 的 CD 驱动器中有 CDROM，就可以在 FreeBSD Parallels 窗口的右下角点光碟图标。之后，系统将弹出一个窗口，供您完成将虚拟机中的 CDROM 驱动器连接到本地的 ISO 文件或真正的 CDROM 驱动器上。

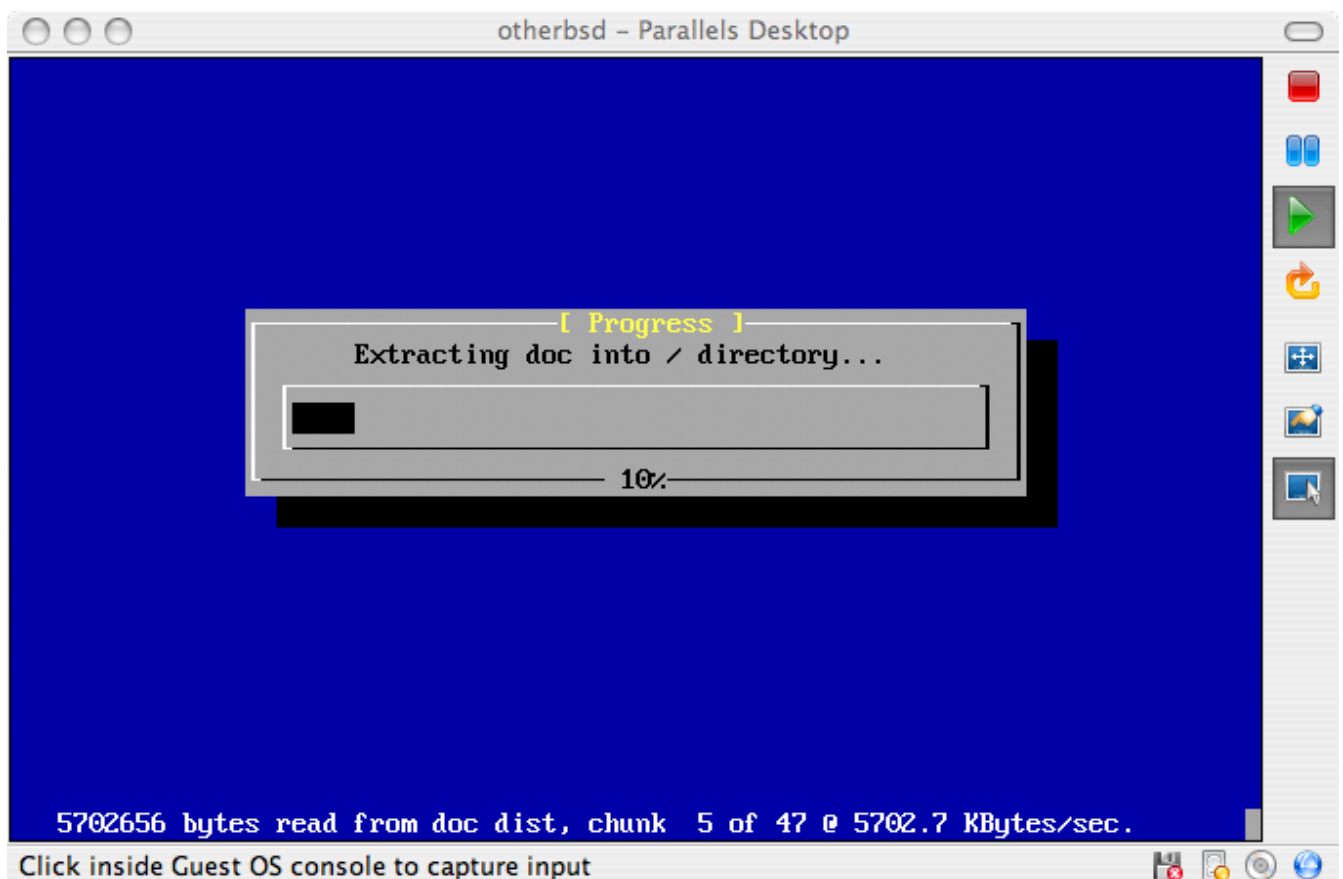


在完成了将 CDROM 与您的安装源完成连接之后，就可以按重启 (reboot) 按钮来重启 FreeBSD 虚拟机了。

Parallels 将配合一个特殊的 BIOS 程序，后者能像普通的 BIOS 一样检查系统中是否有 CDROM 设备。



此时，它就能进入到 FreeBSD 安装介绍并开始 [安装 FreeBSD](#) 中所介绍的标准的基于 sysinstall 安装的过程。



此时可以安装 X11，但不要对它进行配置。在完成安装之后，重新并入新安装的 FreeBSD 虚拟机。

### 23.2.1.2. 在 Mac OS® X/Parallels 上配置 FreeBSD

在将 FreeBSD 安装到 Mac OS® X 的 Parallels 上之后，需要行一系列的配置，以便系的虚拟化操作行化。

#### 1. 配置引加器量

最重要的一是通低 `kern.hz` 量来降低 Parallels 境中的 FreeBSD 的 CPU 的使用。可以通在 `/boot/loader.conf` 中加下述配置来完成：

```
kern.hz=100
```

如果不使用个配置，置的 FreeBSD Parallels 客 OS 会在理器的 iMac® 上使用大 15% 的 CPU。如此修改之后，空的使用量就少到大 5% 了。

#### 2. 建新的内核配置文件

可以去全部 SCSI、FireWire，以及 USB 程序。Parallels 提供了一个由 `ed(4)` 的虚网，因此，除了 `ed(4)` 和 `miibus(4)` 之外的其他网接口都可以从内核中去。

#### 3. 配置网

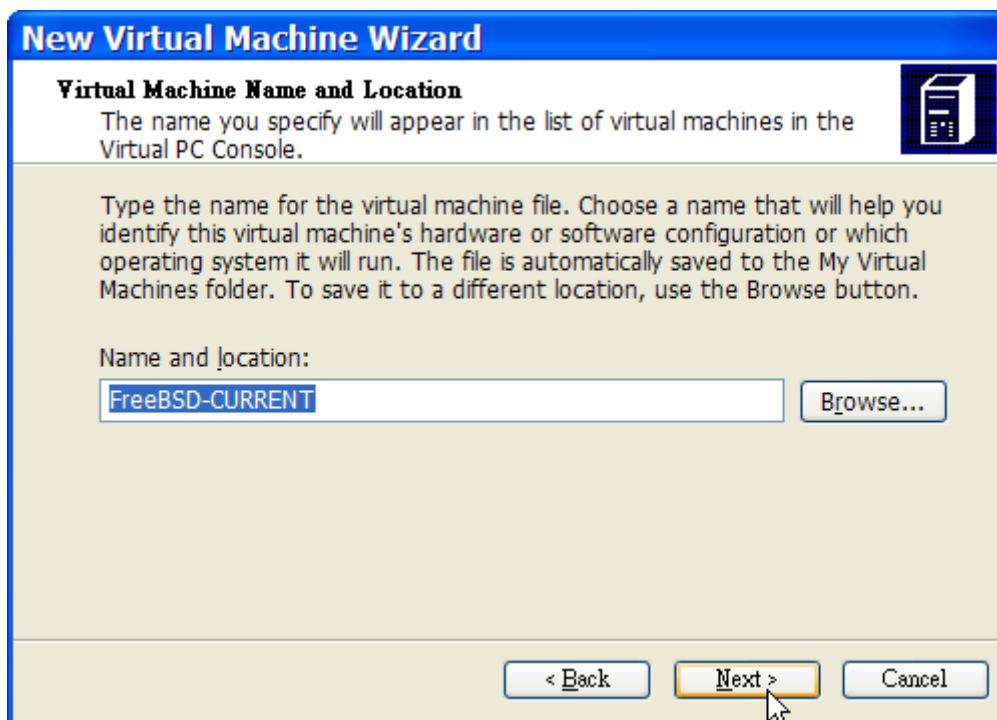
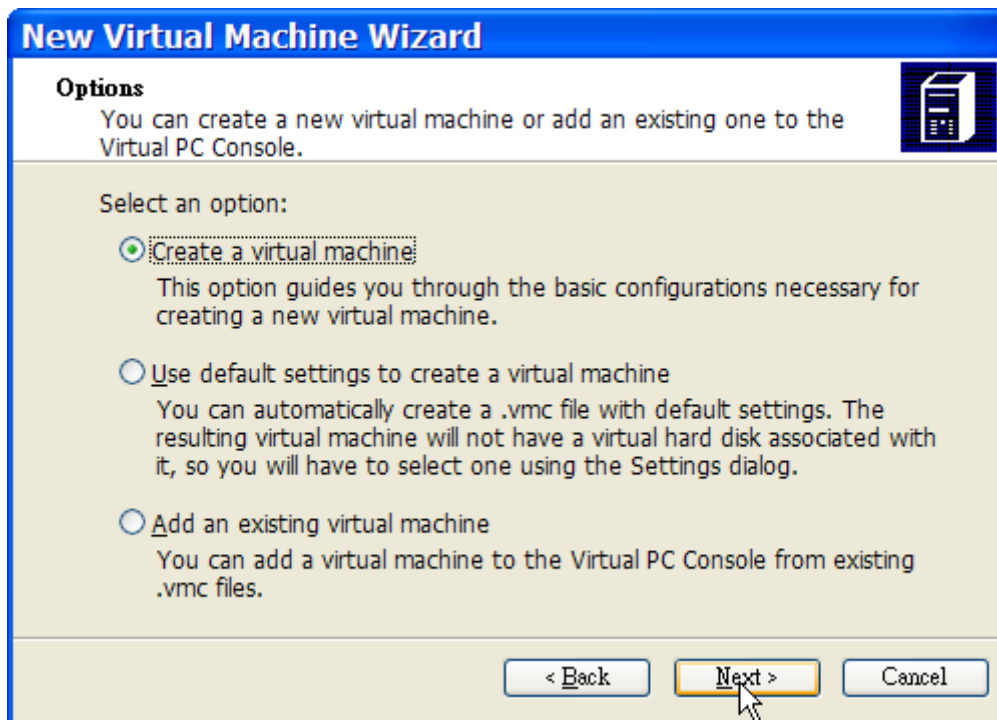
最基本的网配置，是通使用 DHCP 来将的虚机与宿主 Mac® 接入同一个局域网。可以通在 `/etc/rc.conf` 中加入 `ifconfig_ed0="DHCP"` 来完成。更高一些的配置方法，参 [高网](#) 中的介。

### 23.2.2. Windows® 上的 Virtual PC

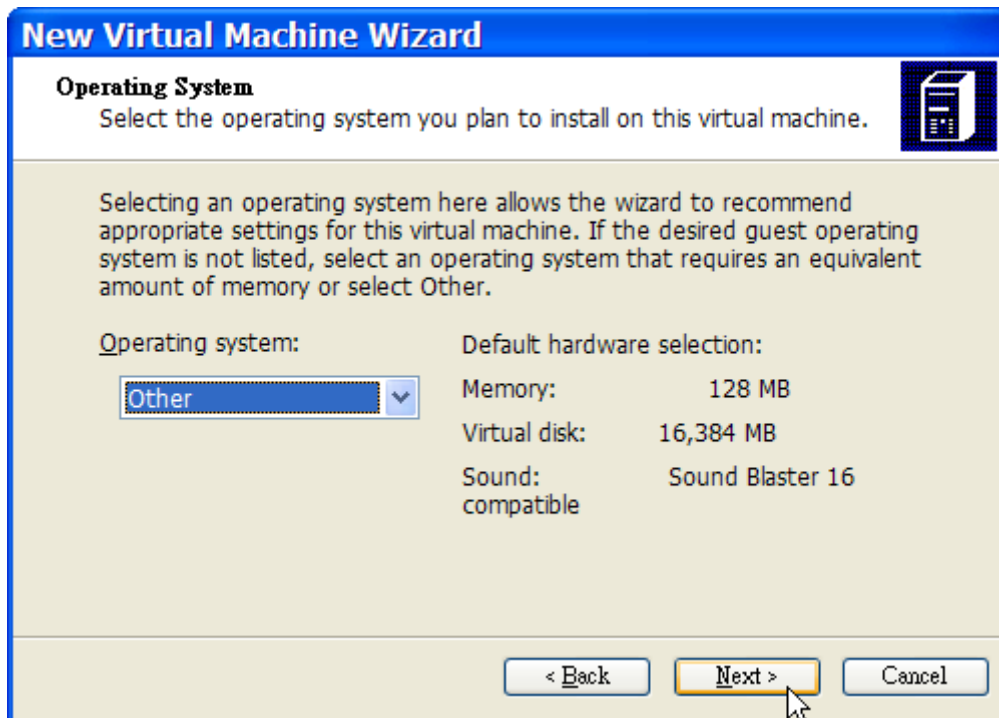
Virtual PC 是 Microsoft® 上的 Windows® 软件产品，可以免费下载使用。相关系统要求，请参考 [system requirements](#) 说明。在 Microsoft® Windows® 装完 Virtual PC 之后，必须用所安装的虚拟机来做相应配置。

#### 23.2.2.1. 在 Virtual PC/Microsoft® Windows® 上安装 FreeBSD

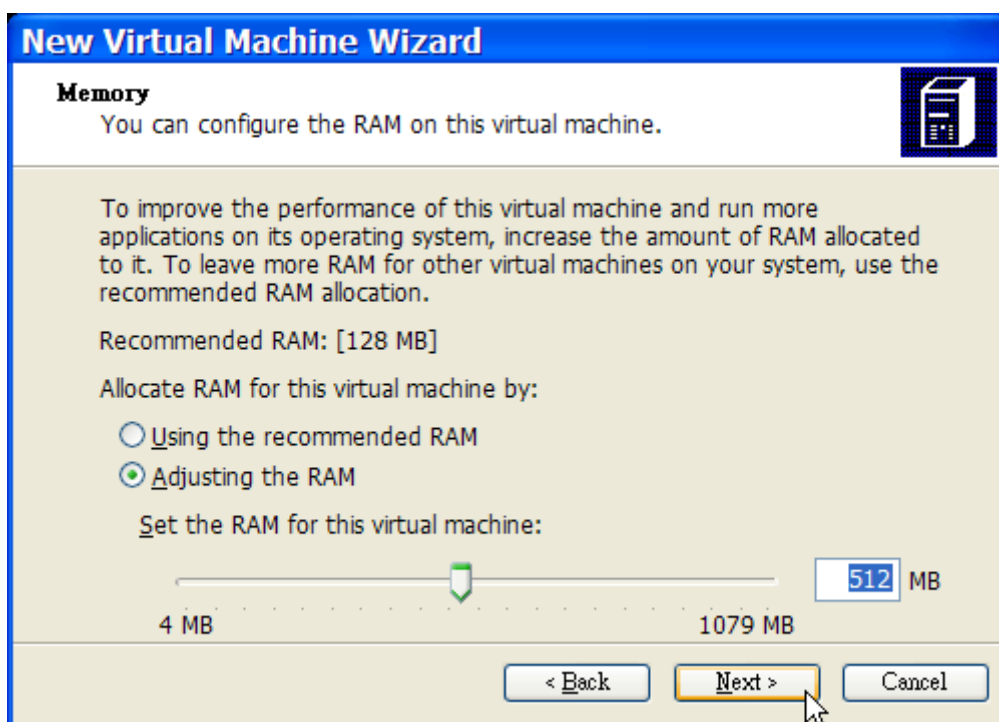
在 Microsoft® Windows®/Virtual PC 上安装 FreeBSD 的第一步是新虚拟机。如下所示，在提示窗口中点击 Create a virtual machine：



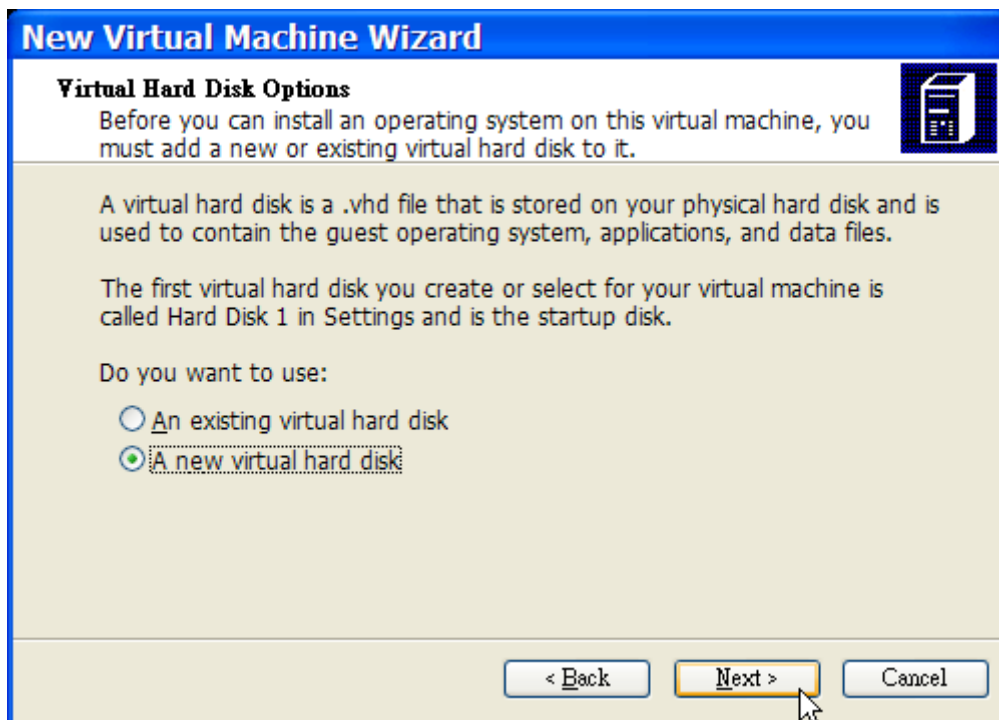
然后在 Operating system 选择 Other：



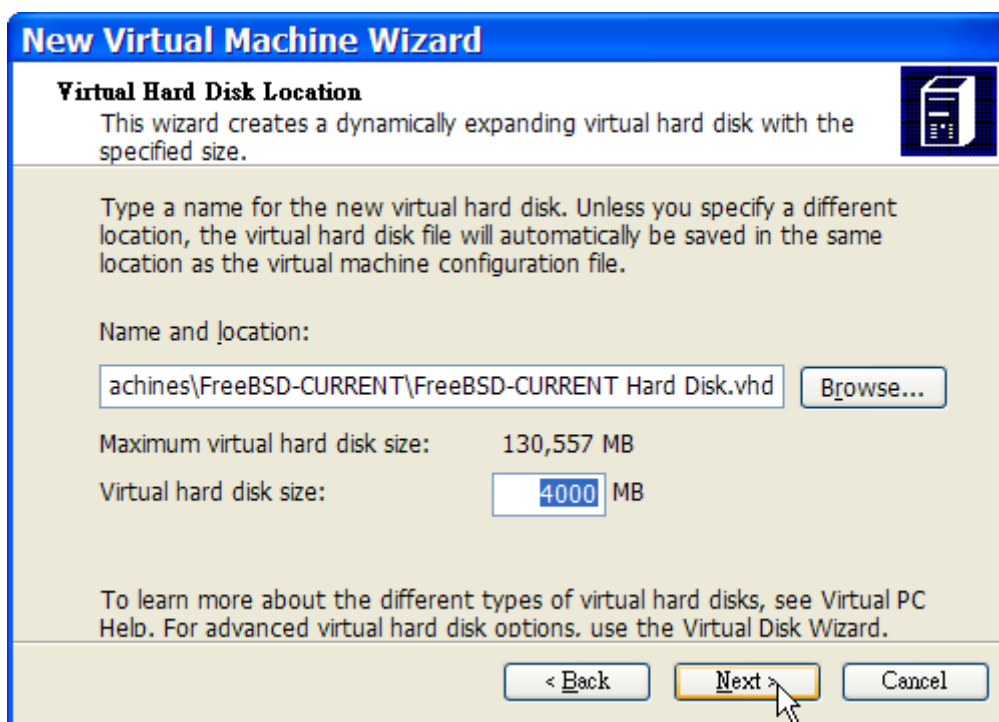
并依据自身需求来调整硬盘容量和内存的分配。大多数在 Virtual PC 使用 FreeBSD 的情况而言，大约 4GB 的硬盘空间以及 512MB 的内存就足够了。





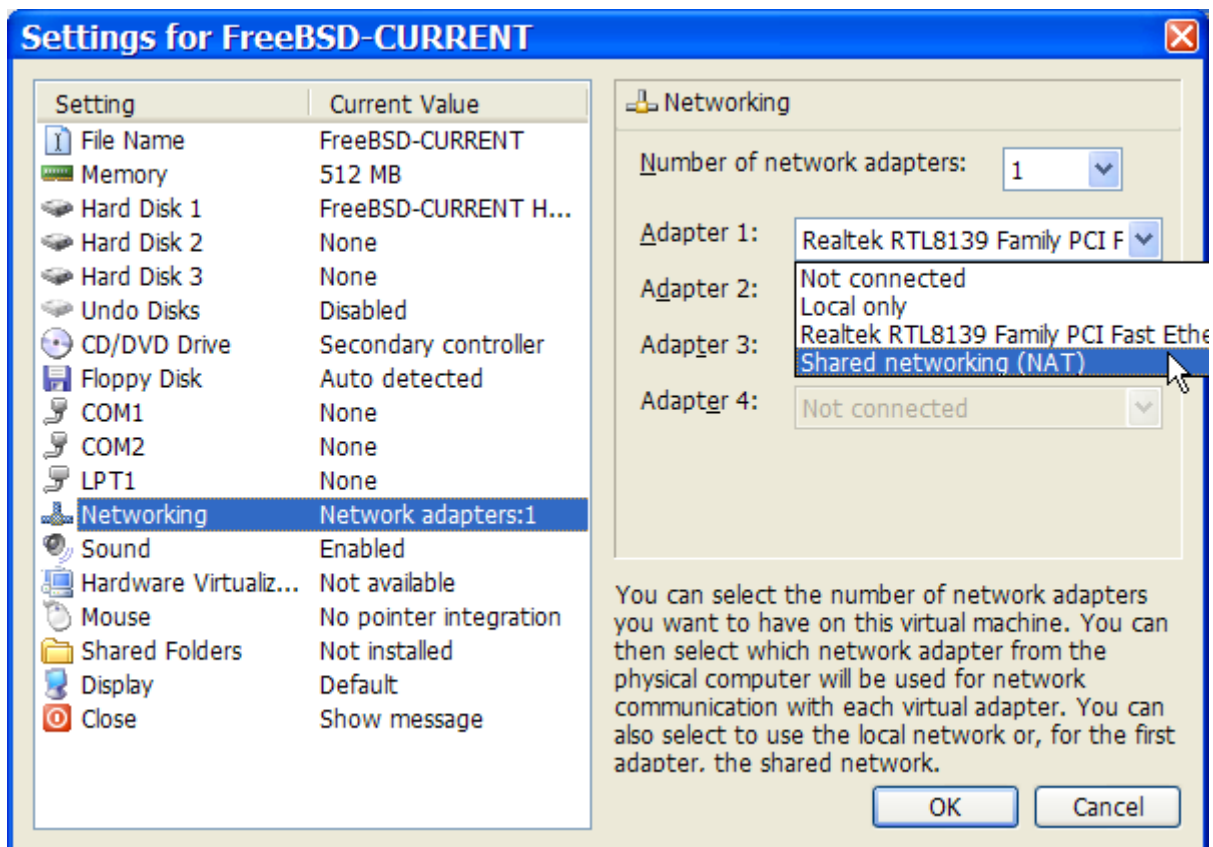
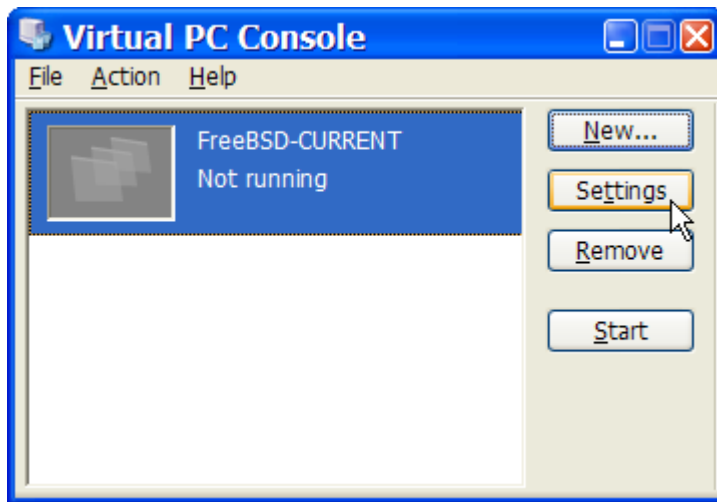


保存并完成配置：

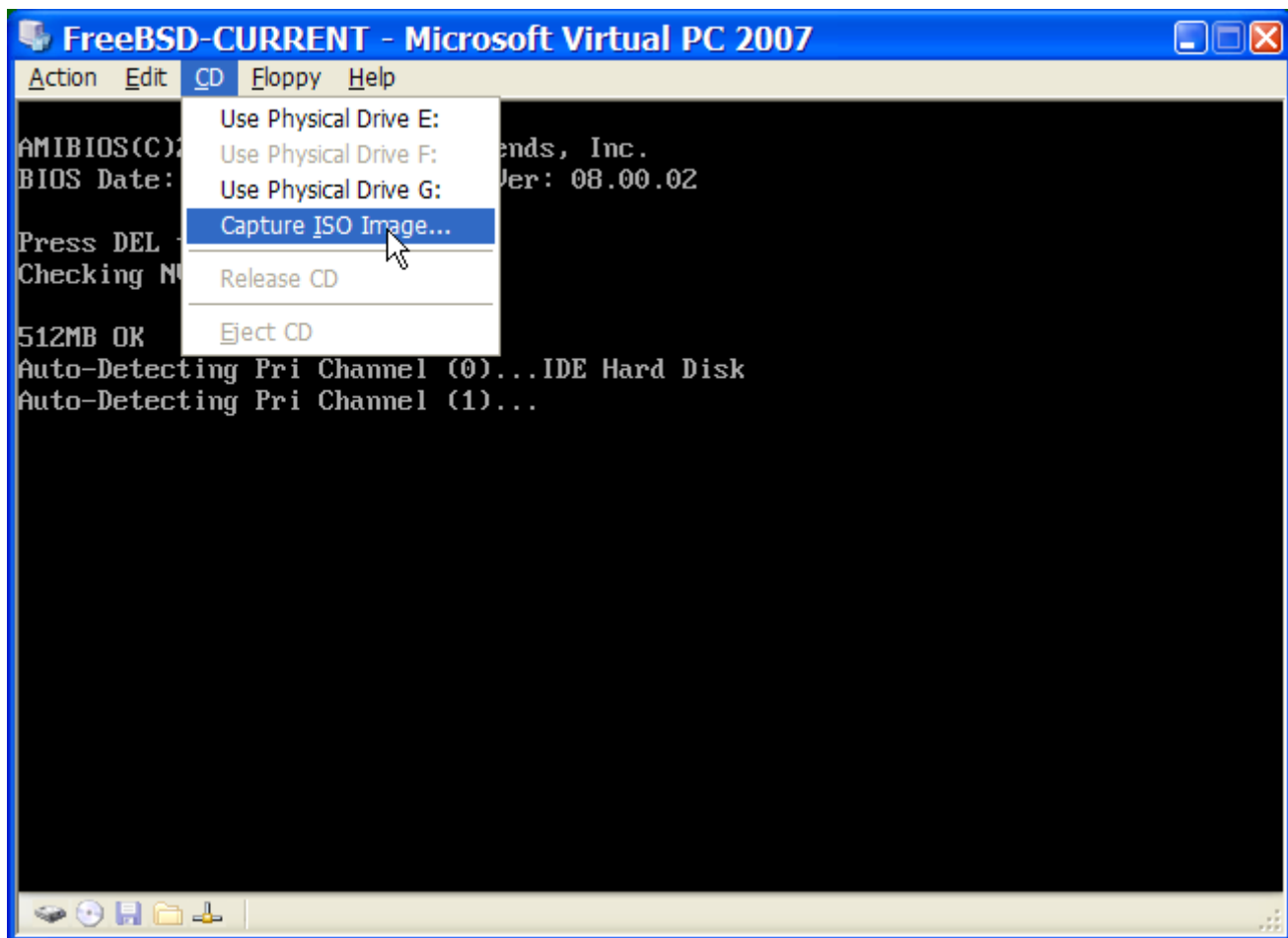
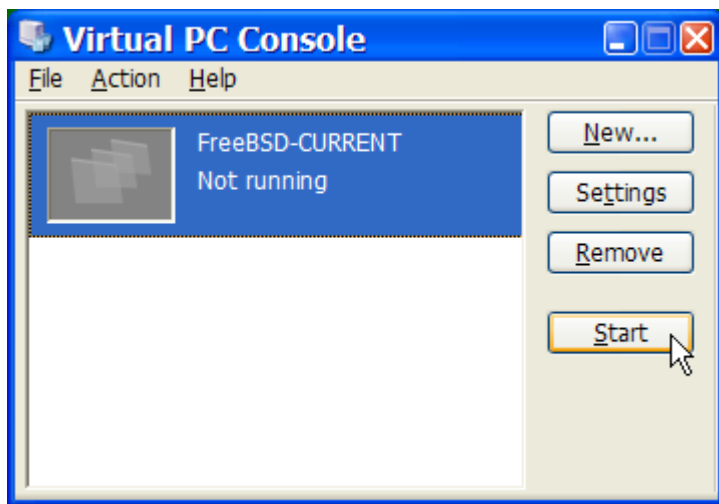


接下来新建的 FreeBSD 虚拟机，并在 **Settings**，以指定网络以及网络：

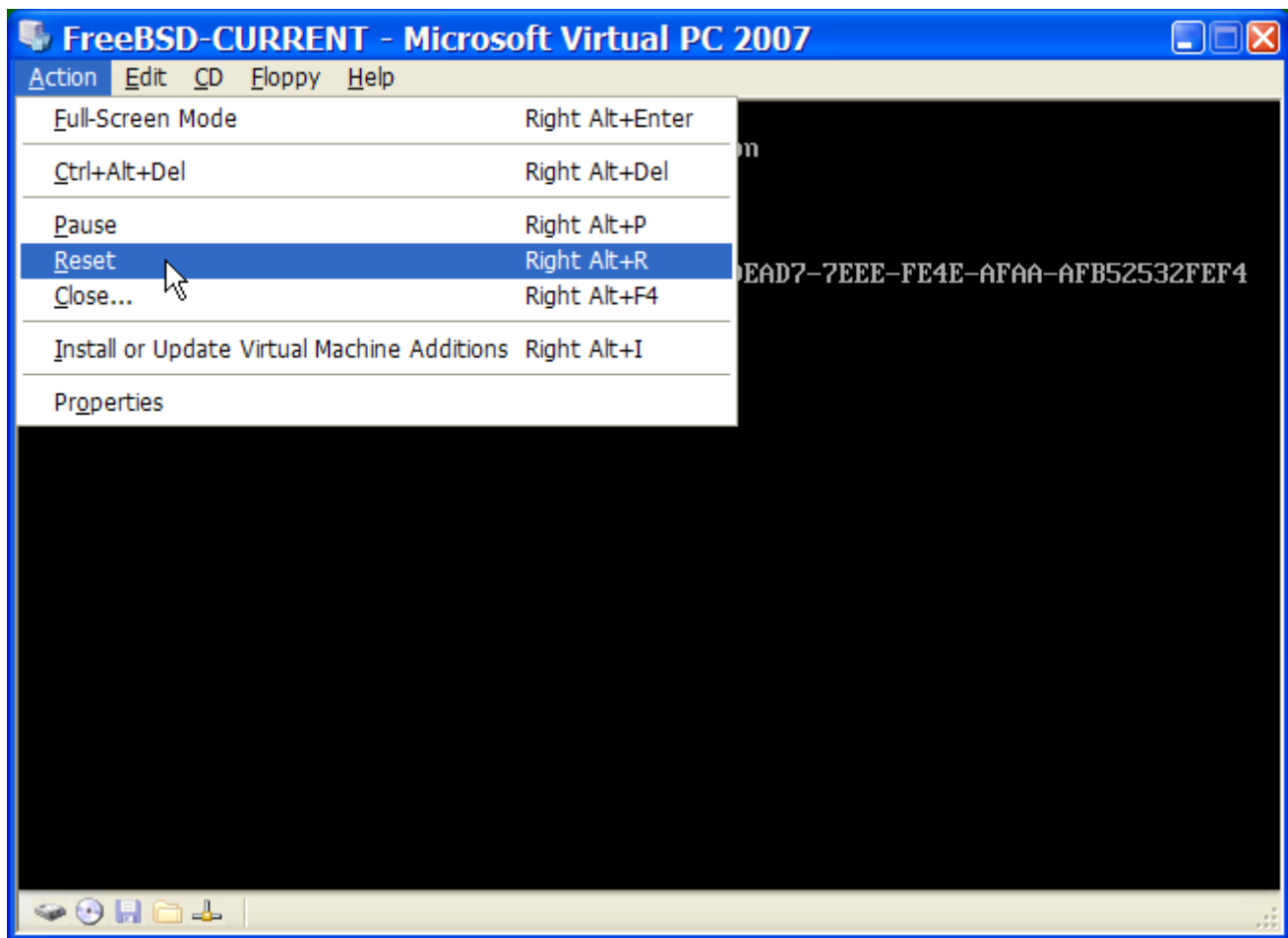




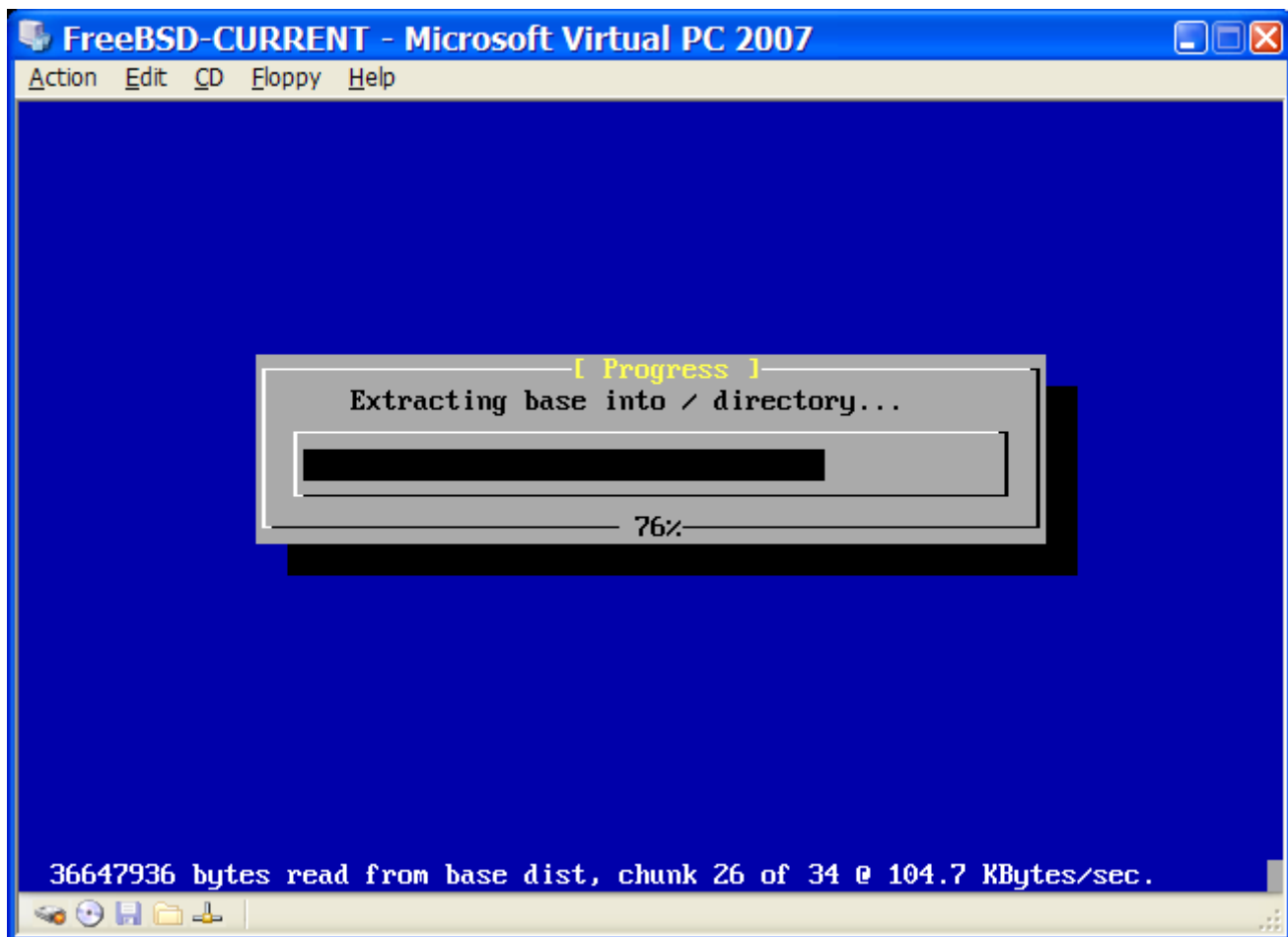
在新建 FreeBSD 虚拟机以后，就可以向其安装 FreeBSD。安装方面，比较好的作法是使用官方的 FreeBSD 光盘或从官方 FTP 站下载 ISO 镜像。若你的 Windows® 系统内已有 ISO 镜像，那就可以在 FreeBSD 虚拟机上双击，以开始。接着在 Virtual PC 窗口内按 CD 再按 **Capture ISO Image...**。接着出现一个对话框，可以把虚拟机内的光盘绑定到 ISO 镜像，或者是真的光盘。



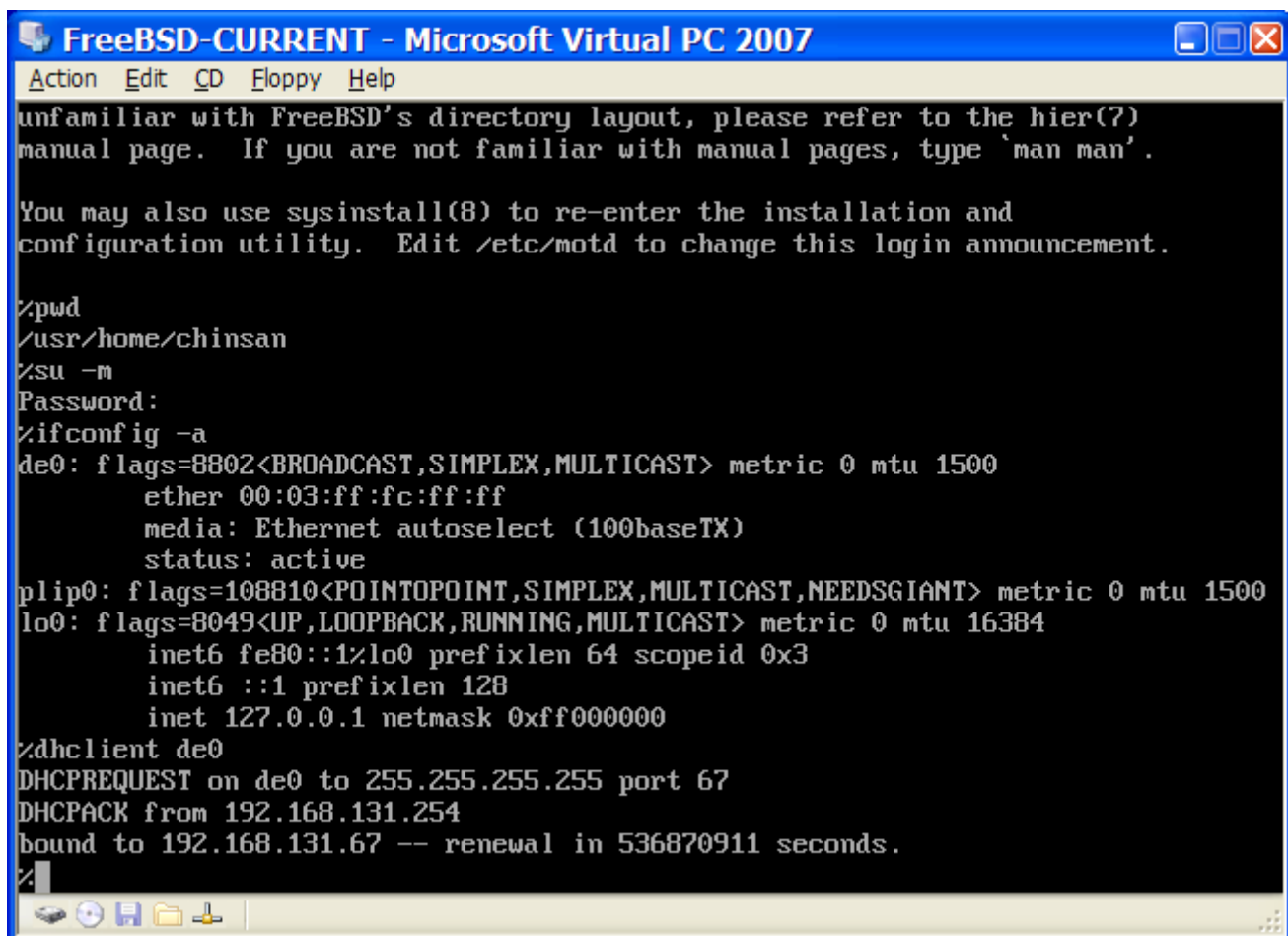
选好光碟来源之后，就可以重新开机，也就是先按 **Action** 再按 **Reset** 即可。Virtual PC 会以特殊 BIOS 开机，并与普通 BIOS 一样会先检测是否有光碟驱动器。



此时，它会加载 FreeBSD 安装光盘，并开始运行在 [安装 FreeBSD](#) 中所介绍的 sysinstall 安装程序。此时也可以顺便安装 X11，但不要进行相关设定。



完成安装之后，得把安装光盘或者 ISO 镜像退出。最后，把装好的 FreeBSD 虚拟机重新开机即可。



### 23.2.2.2. 调整 Microsoft® Windows®/Virtual PC 上的 FreeBSD

在 Microsoft® Windows® 上以 Virtual PC 装好 FreeBSD 后，需要做一些调整，以便将虚拟机内的 FreeBSD 最佳化。

#### 1. 调整 boot loader 参数

最重要的调整乃是藉由降低 `kern.hz` 来降低 Virtual PC 境内 FreeBSD 的 CPU 占用率。在 `/boot/loader.conf` 内加上下列调整即可：

```
kern.hz=100
```

若不作调整，那光是 idle 状态的 FreeBSD Virtual PC guest OS 就会在单一处理器的系统上大抵有 40% 的 CPU 占用率。作了上述修改之后，占用率大约会降至 3%。

#### 2. 建立一个新的内核配置文件

可以放心把所有的 SCSI，FireWire 和 USB 设备都移除。Virtual PC 有提供 `de(4)` 的虚拟网卡，因此除了 `de(4)` 以及 `miibus(4)` 以外其他的网卡也都可以从内核的配置文件中移除。

#### 3. 调整网络

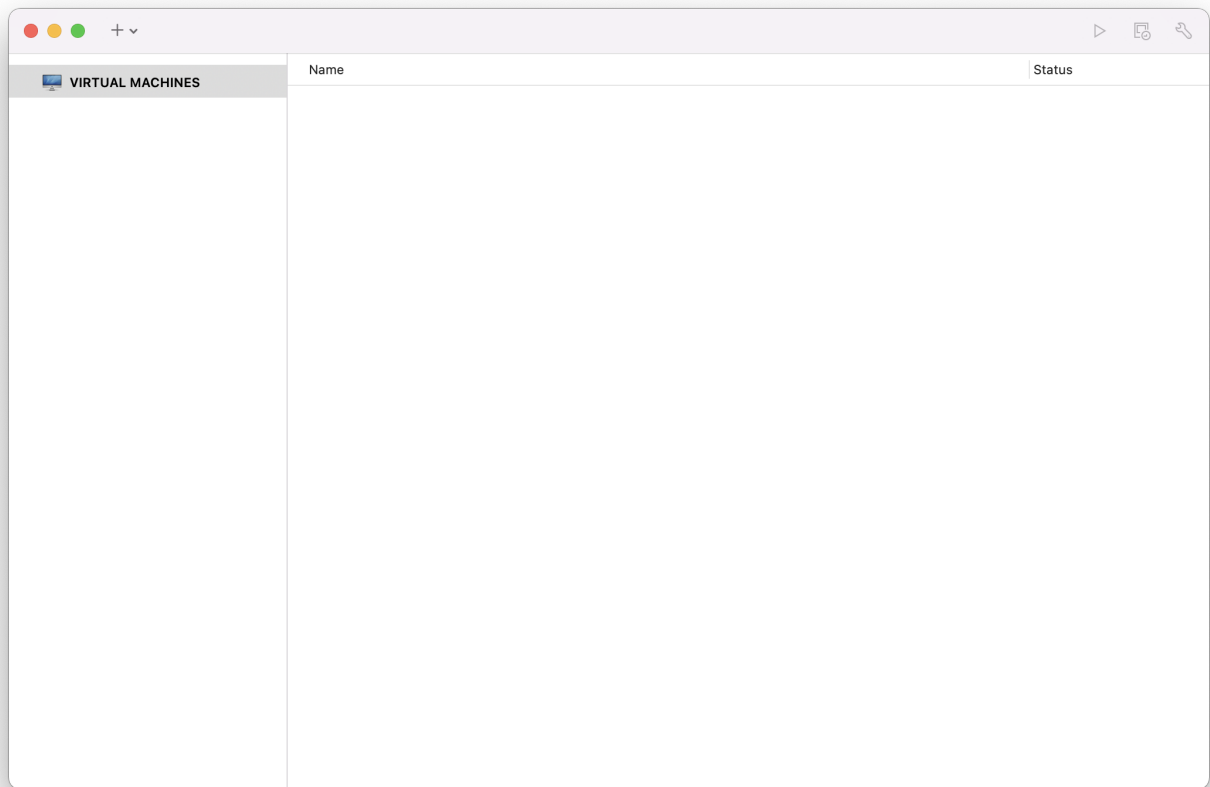
可以让虚拟机使用 DHCP 来调整与 host (Microsoft® Windows®) 相同的本地网络环境，只要在 `/etc/rc.conf` 加上 `ifconfig_de0="DHCP"` 即可完成。其他的高网速设置，可参看[高网速](#)。

### 23.2.3. 运行于 MacOS 的 VMware

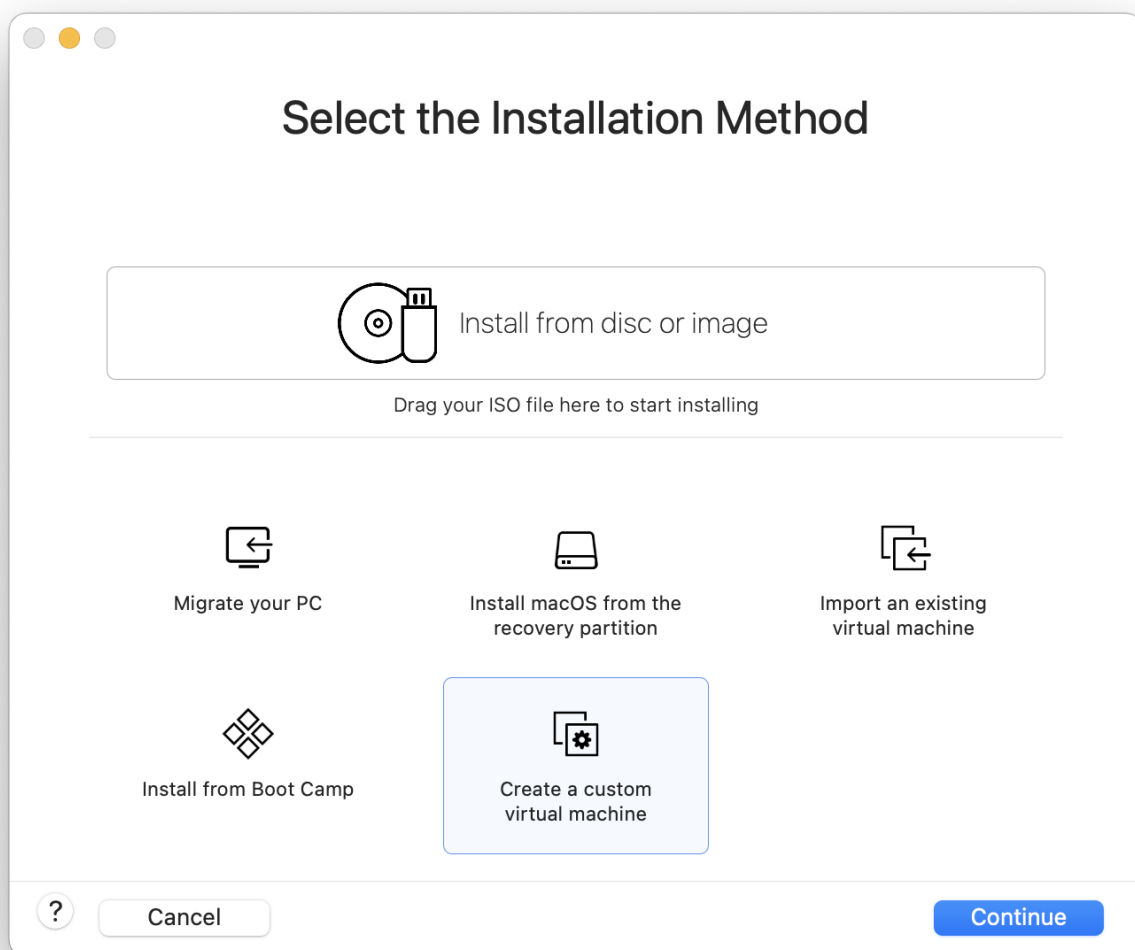
Mac® 版本的 VMware Fusion 是一个商业软件，运行在基于 Intel® 的 Apple® Mac® 计算机的 Mac OS® 10.4.9 或更版本的操作系统上。FreeBSD 是一个完全被支持的客操作系统。在 Mac OS® X 上安装了 VMware Fusion 之后，用户就可以着手配置一个虚拟机并安装客操作系统。

#### 23.2.3.1. 在 VMware/Mac OS® X 上安装 FreeBSD

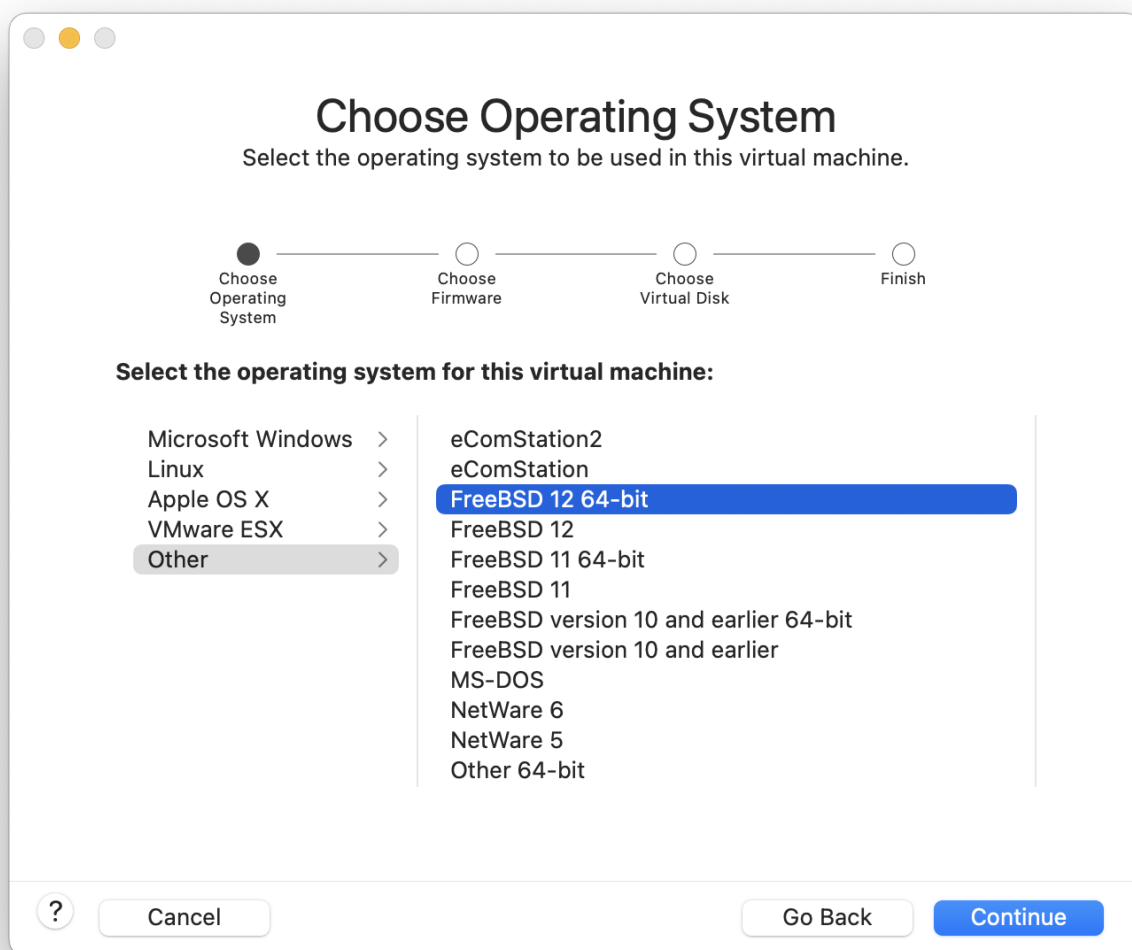
第一步是运行 VMware Fusion，虚拟机将被装填。点击 "New" 新建 VM：



New Virtual Machine Assistant 将被用来帮助创建 VM, 点击 Continue :

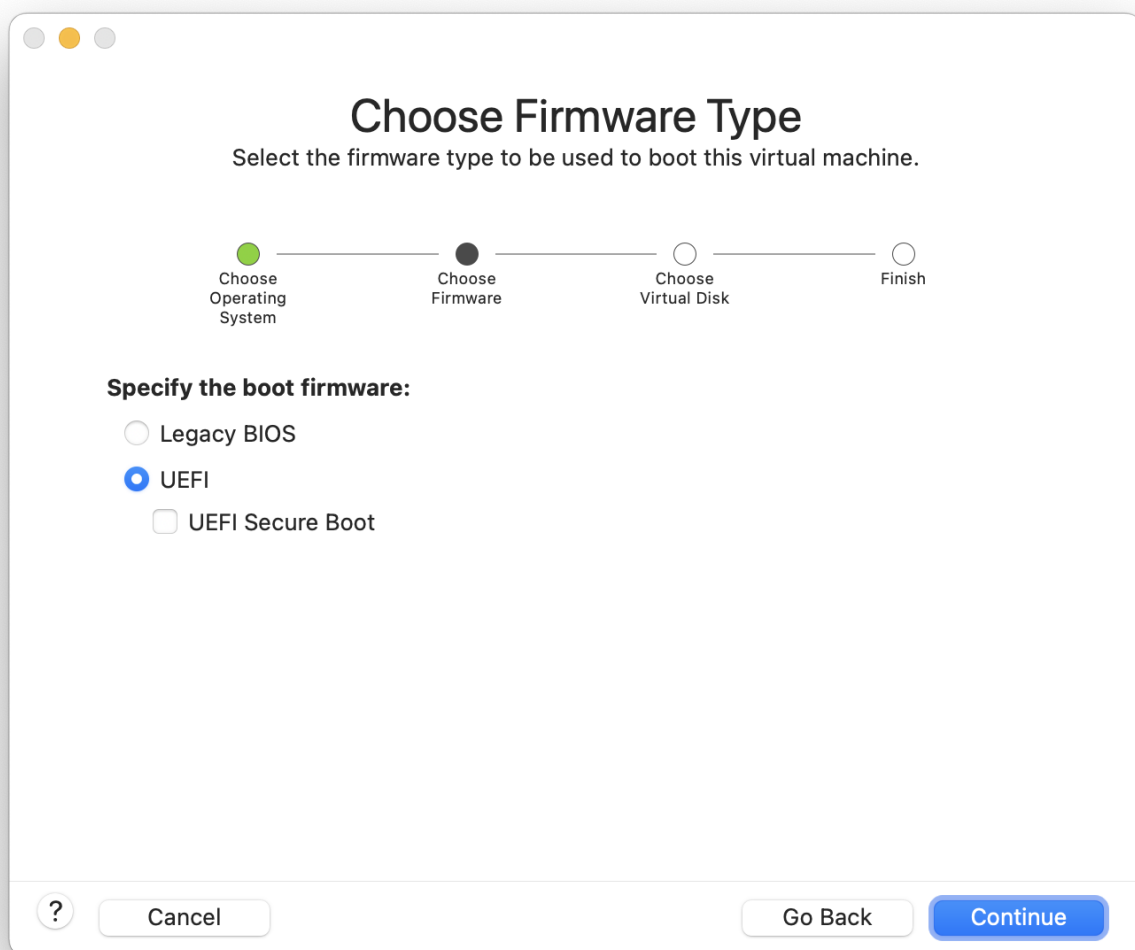


在 Operating System 欄位 Other, Version 欄位 FreeBSD 或 FreeBSD 64-bit。

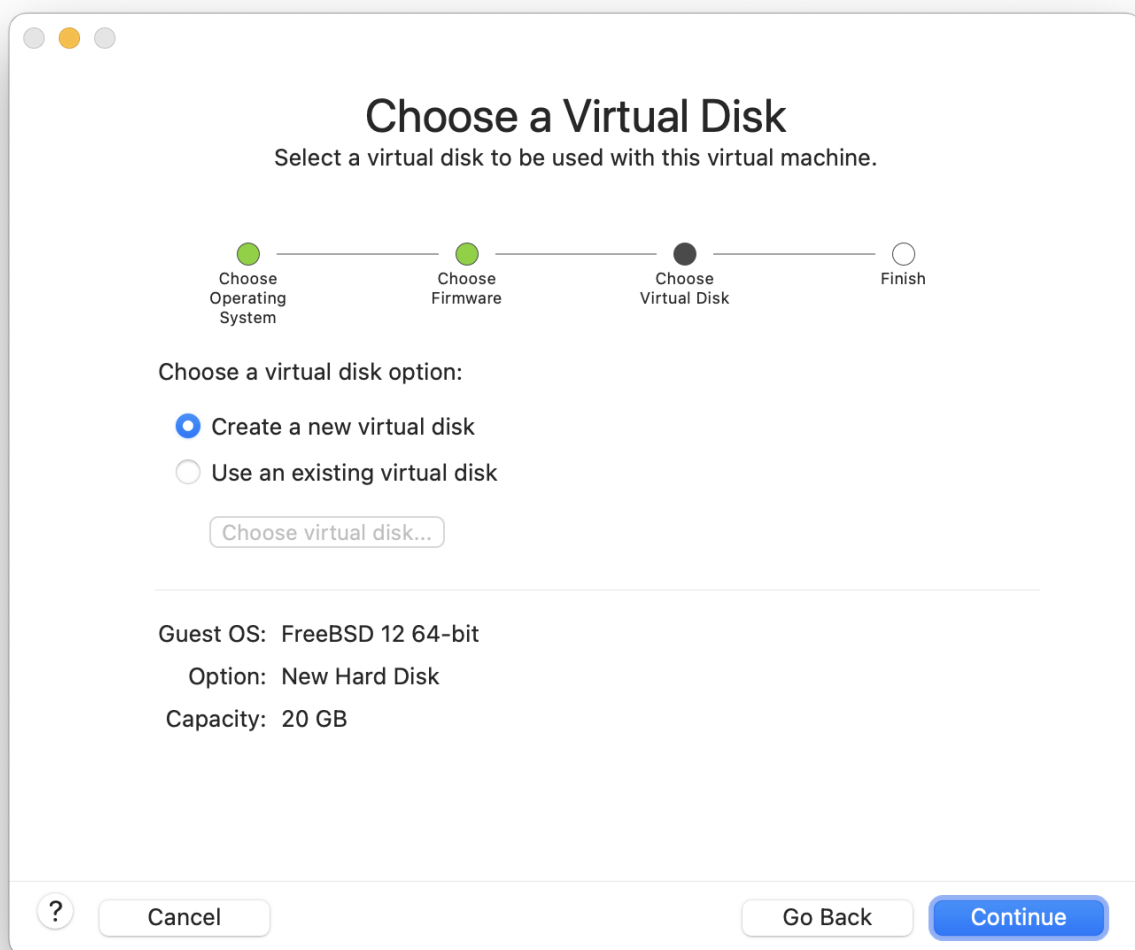


□一个□想要的 VM □像名字和存□的目□位置。

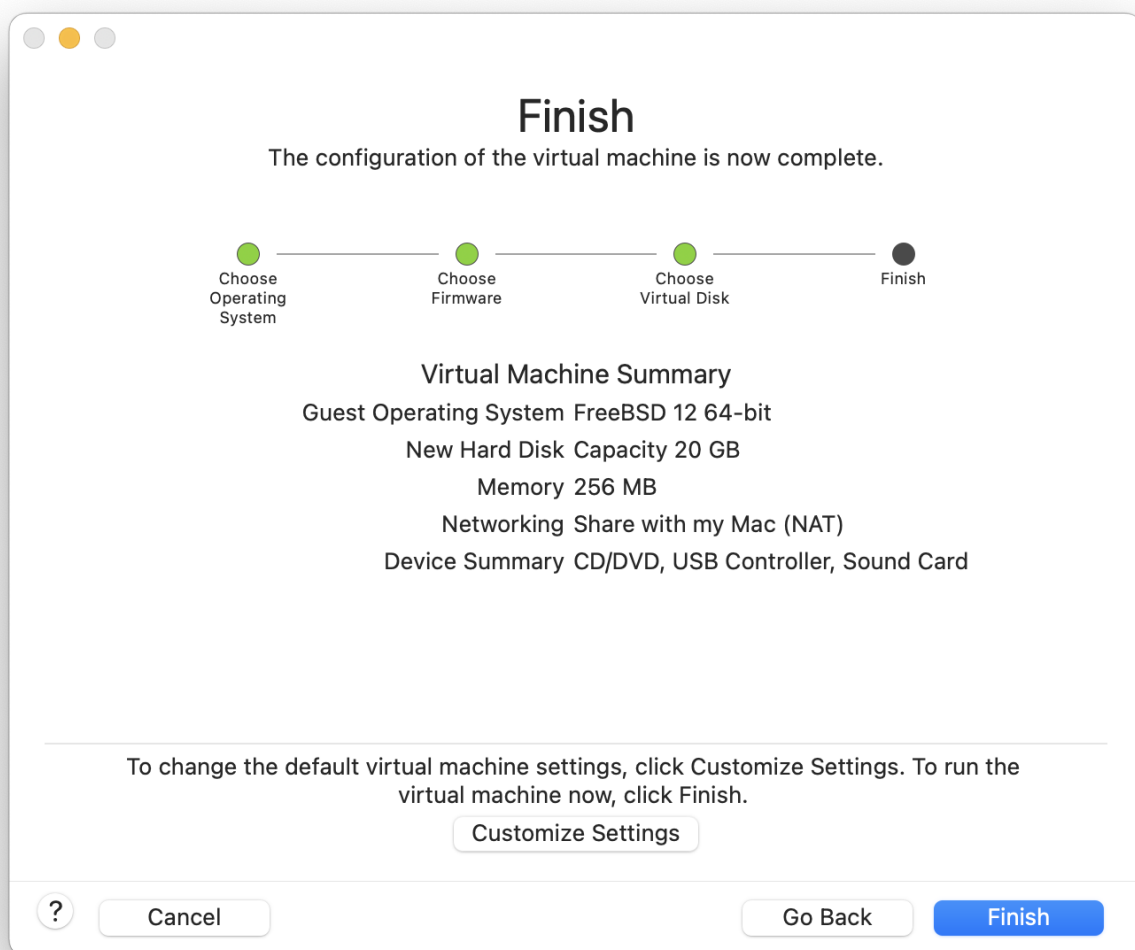




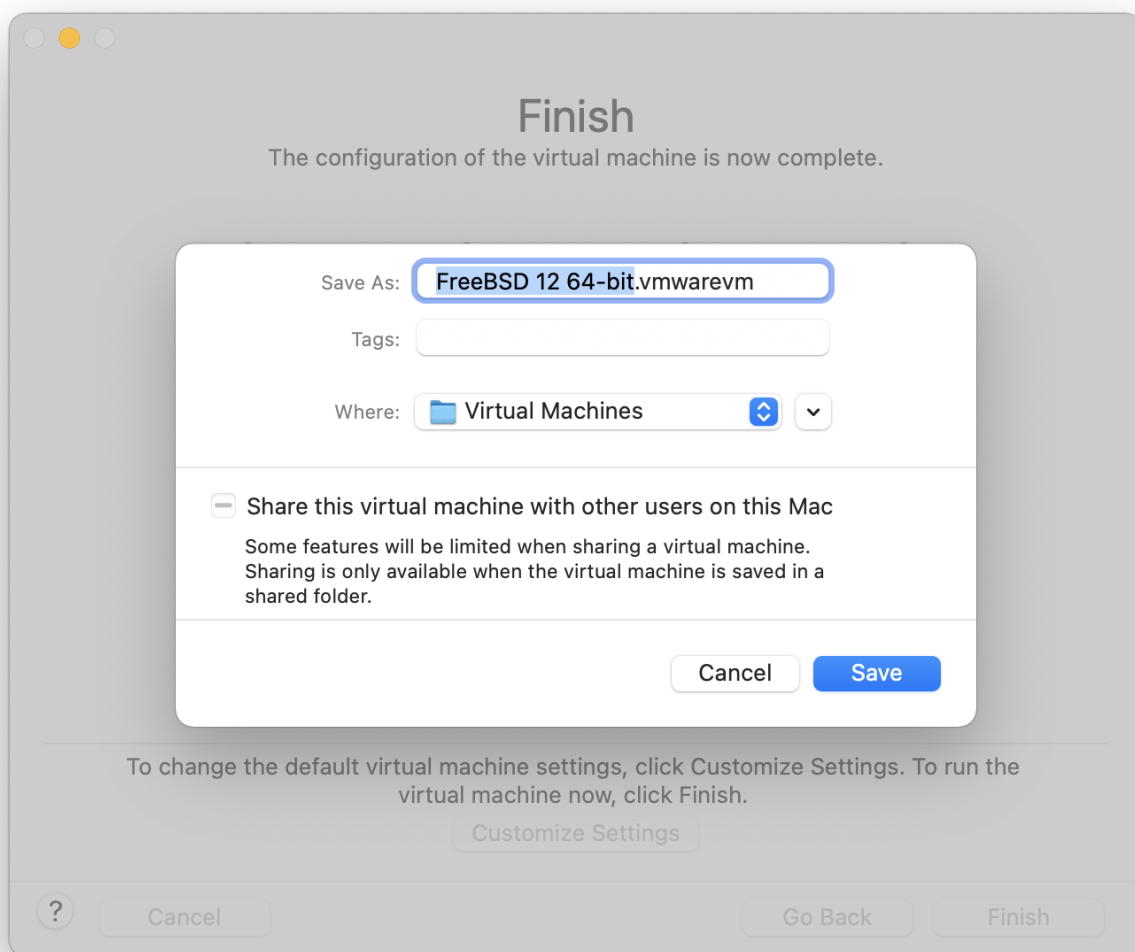
□ VM 虚□硬□的大小：



安装 VM 的方式， 从一个 ISO 镜像或一张 CD 安装：



一旦点了 Finish， VM 就会开了：



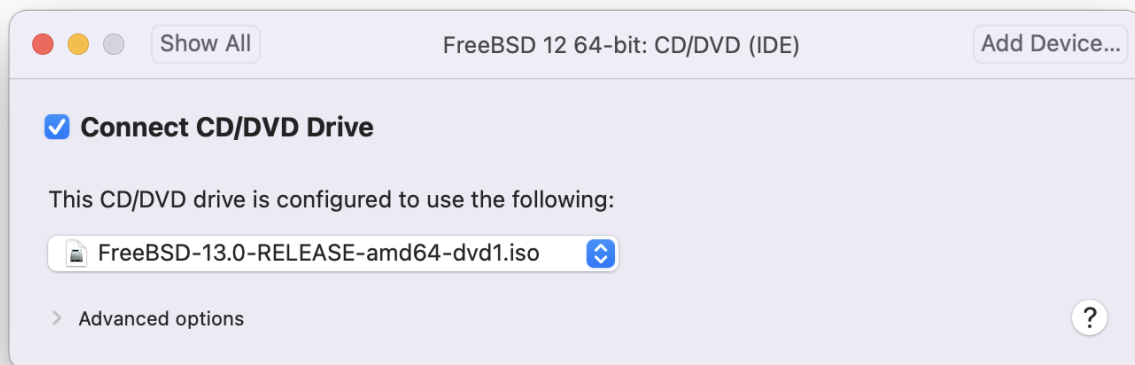
以通常的方式安装 FreeBSD 或者参照 [安装 FreeBSD](#) 中的：



安装完成之后，您就可以修改一些 VM 的配置，比如内存大小：



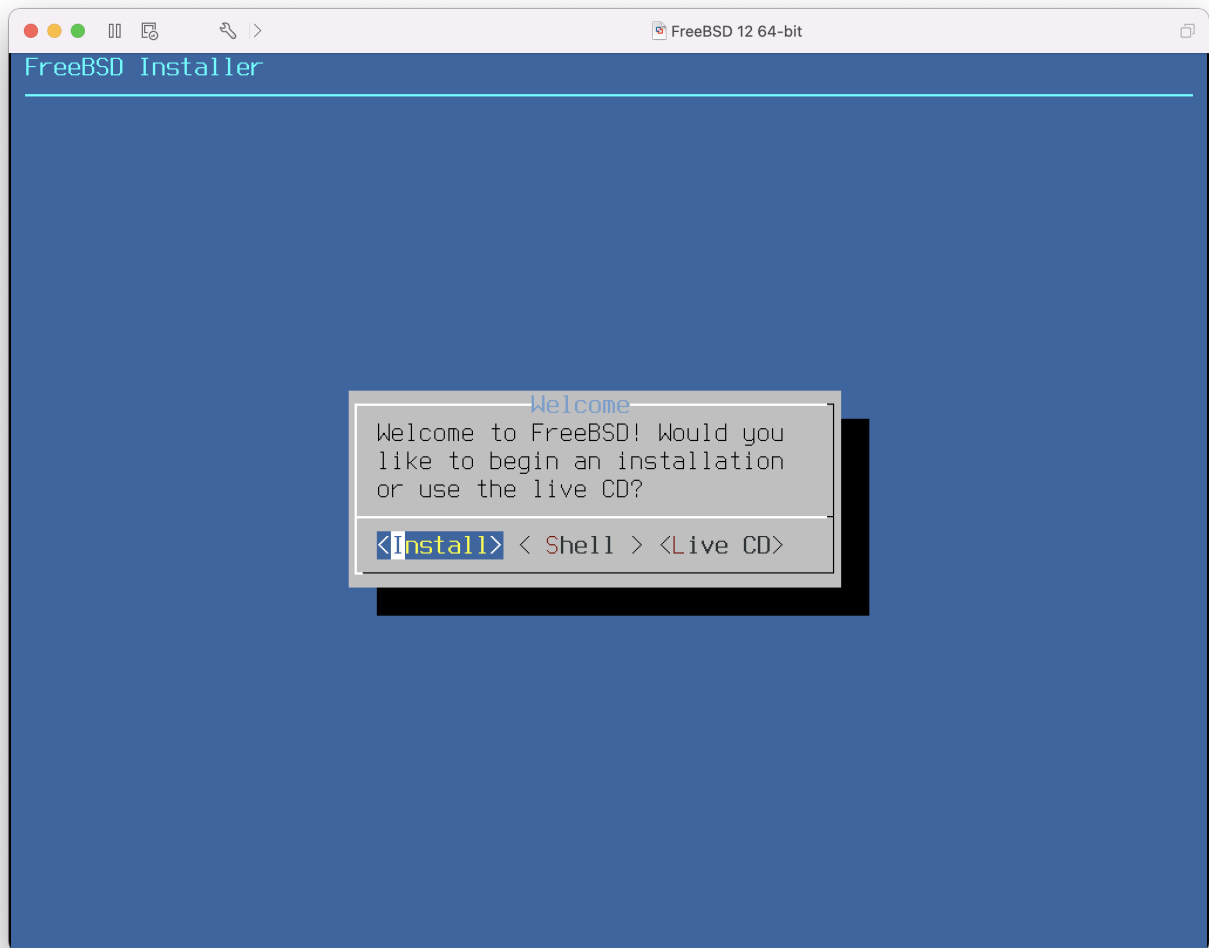
在 VM 运行的时候，VM 系硬件的配置是无法修改的。



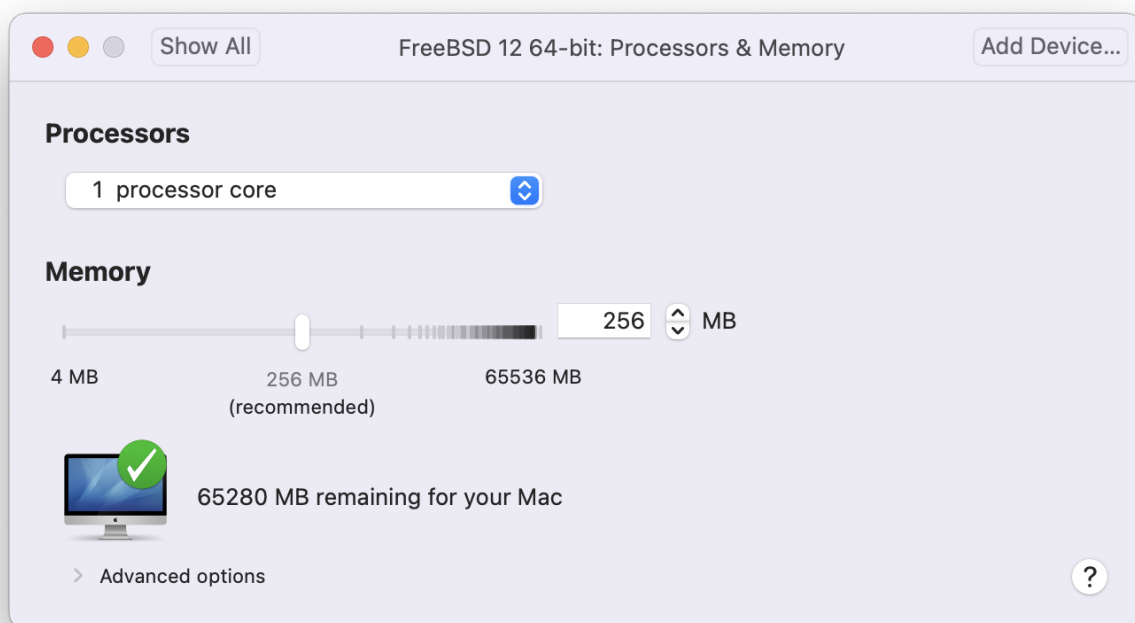
配置 VM 的 CPU 数量：



CD-ROM的状态。通常当它不在需要 CDROM/ISO 的时候可以切断它与 VM 的连接。



最后一项需要修改的是 VM 与网连接的方式。如果你希望除了宿主以外的机器也能连接到 VM，你可以选择 Connect directly to the physical network (Bridged)。如果你选择 Share the host's internet connection (NAT) 的，VM 可以连接上网，但是不能从外面访问。



在修改完配置之后，就可以从新安装的 FreeBSD 虚拟机了。

### 23.2.3.2. 配置运行于 Mac OS® X/VMware 上的 FreeBSD

在 Mac OS® X 上的 VMware 上安装完 FreeBSD 之后，有些配置的值可用来优化虚拟机。

#### 1. 设置 boot loader 值

最重要的是降低 `kern.hz` 来减少 VMware 上 FreeBSD 的 CPU 使用率。需要在 `/boot/loader.conf` 里加入以下行：

```
kern.hz=100
```

如果没有配置，VMware 上的 FreeBSD 客户 OS 空间将占用 iMac® 上一个 CPU 大约 15% 的资源。在修改此配置之后约 5%。

#### 2. 建立一个新的内核配置文件

可以去掉所有的 FireWire, USB 等的程序。VMware 提供了一个 `em(4)` 支持的虚拟网络适配器，所以除了 `em(4)` 之外的网络都可以被剔除。

#### 3. 设置网络

最基本的网络配置包括使用 DHCP 把你的虚拟机连接到宿主 Mac® 相同的本地网络上。在 `/etc/rc.conf` 中加入：`ifconfig_em0="DHCP"`。更多有关网络的配置可以参看 [高网络](#)。



## 23.3. 作为宿主 OS 的 FreeBSD

在过去的几年中 FreeBSD 并没有任何可用的并被官方支持的虚拟化解决方案。一些用户曾使用利用 Linux® 二进制兼容运行的 VMware 旧版并多半已过的版本 (比如 [emulators/vmware3](#))。在 FreeBSD 7.2 发布不久, Sun 开源版本 (Open Source Edition OSE) 的 VirtualBox™ 作为一个 FreeBSD 原生的程序出现在了 Ports Collection 中。

VirtualBox™ 是一个非常灵活, 完全虚拟化的软件, 并且可在大多数的操作系统上使用, 包括 Windows®, Mac OS®, Linux® 和 FreeBSD。同样也能把 Windows® 或 UNIX® 作为客户系统运行。它有一个开源和一个私有版本。从用户角度来看, OSE 版本最主要的限制也是缺乏 USB 的支持。其他更多的差异可以通过接 <http://www.virtualbox.org/wiki/Editions> 看 "Editions" 页面。目前, FreeBSD 上只有 OSE 版本可用。

### 23.3.1. 安装 VirtualBox™

VirtualBox™ 已作为一个 FreeBSD port 提供, 位于 [emulators/virtualbox-ose](#), 可使用如下的命令安装:

```
# cd /usr/ports/emulators/virtualbox-ose
# make install clean
```

在配置框中的一个有用的组件是 **GusetAdditions** 程序套件。它在客户操作系统中提供了一些有用的特性, 比如集成鼠标指针 (允许在宿主和客户系统上使用鼠标, 而不用事先按下某个特定的快捷键来切换) 和更快的渲染, 特别是在 Windows® 客户系统中。在安装客户操作系统之后, 客户附加组件可在 **Devices** 菜单中得到。

在第一次运行 VirtualBox™ 之前需要做一些配置上的修改。port 会安装一个内核模块至 `/boot/modules` 目录, 此模块需要事先加载:

```
# kldload vboxdrv
```

可以在 `/boot/loader.conf` 中加入以下的配置使此模块在机器重启之后能自动加载:

```
vboxdrv_load="YES"
```

在 3.1.2 之前版本的 VirtualBox™ 需要挂接 `proc` 文件系统。在新版本中不再有此要求, 因为它使用了由 [sysctl\(3\)](#) 提供的功能。

当使用旧版本的 port 时, 需要使用下面的命令来挂载 `proc`:

```
# mount -t procfs proc /proc
```

为了使配置能在重启后开始生效, 需要在 `/etc/fstab` 中加入以下行:

```
proc    /proc    procfs  rw  0    0
```

如果在运行 VirtualBox™ 的终端中出现了类似如下的消息：



```
VirtualBox: supR3HardenedExecDir: couldn't read "", errno=2 cchLink=-1
```

此故障可能是由 proc 文件系统导致的。使用 `mount` 命令检查文件系统是否正挂载。

在安装 VirtualBox™ 时会自建 `vboxusers` 组。所有需要使用 VirtualBox™ 的用户必须被添加到此组中的成员。可以使用 `pw` 命令添加新的成员：

```
# pw groupmod vboxusers -m yourusername
```

运行 VirtualBox™，可以通过当前图形环境中的 Sun VirtualBox，也可以在虚拟机终端中输入以下的命令：

```
% VirtualBox
```

获得更多有关配置和使用 VirtualBox™ 的信息，访问官方网站 <http://www.virtualbox.org>。鉴于 FreeBSD port 非常新，并仍处于测试状态。查看 FreeBSD wiki 上的相关页面 <http://wiki.FreeBSD.org/VirtualBox> 以获取最新的信息和故障排除。

# Chapter 24. 本地化－I18N/L10N使用和配置

## 24.1. 概述

FreeBSD是一个由分布于全世界的用户和贡献者支持的项目。本章将介绍FreeBSD的国际化和本地化的情况,允许非英语用户也能使用FreeBSD很好地工作。在系统和应用水平上,主要是通过行i18N标准来做的,所以这里我将向读者提供简单的介绍。

读完一章,你将了解:

- 不同的语言和地域是如何在代操作系统上运行的。
- 如何简单的登入shell配置本地化。
- 如何配置的控制台非英语。languages.
- 如何使用不同的语言来有效地使用X Windows。
- 在里可以查到更多有符合i18N标准的程序的信息。

本章之前,应当了解:

- 安装外的第三程序 ([安装程序. Packages 和 Ports](#))。

## 24.2. 基础知识

### 24.2.1. I18N/L10N 是什么?

人们把internationalization写成I18N,中的数字是前后两个字母的字母个数。L10N依据"localization"使用同样的命名。I18N/L10N方法、系统和应用合在一起,允许使用他自己所的语言。

I18N应用程序使用I18N工具来编程。它允许人们写一个的文件,就可以将显示的菜单和文本翻成本地语言。我非常鼓励程序遵循。

### 24.2.2. 为什么要使用I18N/L10N?

I18N/L10N标准能很好地支持看、入或理非英语。

### 24.2.3. I18N支持哪些语言?

I18N和L10N不是FreeBSD特有的。当前,它能支持世界上大部分主力语言,包括但不限于:中文, 德文, 日文, 朝文, 法文, 俄文, 越南文等等。

## 24.3. 使用本地化语言

I18N不是FreeBSD特有的,它是一个。我鼓励帮助FreeBSD完善。

本地化配置需要具三个条件:语言代 (Language Code)、国家代 (Country Code) 和(Encoding)。本地名字可以用下面些部分来造:

### 24.3.1. 语言和国家代码

为了用特殊的语言来FreeBSD系进行本地化（或其他UNIX®系），用者必要知道相应的国家和语言代码（国家代码告诉应用程序使用哪一种语言）。此外，WEB服务器，SMTP/POP服务器，web服务器等都是以个为基础的。下面就是一个国家和语言代码的例子：

| 语言/国家代码 | 描述   |
|---------|------|
| en_US   | 美国英语 |
| ru_RU   | 俄语   |
| zh_CN   | 简体中文 |

### 24.3.2. 多字节

一些语言不使用 ASCII 字符，它们使用8位、16位或多字节的字符，更多的信息参考 [multibyte\(3\)](#)。比老的程序可能会无法处理它们，并可能将控制字符。比新的程序通常会输出 8-位字符。随机的不同，用者可能不得不将多字节字符支持输入程序，或进行一些额外的配置，才能正常使用它们。要输入和处理多字节字符，[FreeBSD Ports Collection](#) 已为许多语言提供了不同的程序。参考各个 FreeBSD Port 中的 I18N 文。

特别需要指出的是，用者可能需要查看程序的文档，以确定如何正确地配置它，或需要配置 configure/Makefile/编辑器 指定什么样的参数。

记住下面些：

- 特定语言的C字符集（参 [multibyte\(3\)](#)），例如 ISO8859-1, ISO8859-15, KOI8-R, CP437。
- 字节或多字节编码，如EUC, Big5。

可以在[IANA Registry](#)看一下流行的字符集列表。



与此不同的是，FreeBSD 使用与 X11-兼容的本地化模式。

### 24.3.3. I18N应用程序

在FreeBSD Ports和Package系里面，I18N应用程序已使用 **I18N** 来命名。然而它不是支持需要的语言。

### 24.3.4. 本地化设置

通常只要在登入shell里面设置 **LANG** 本地化，一般通设置用的 `~/login_conf` 或用shell的配置文件（`~/profile`, `~/bashrc`, `~/cshrc`）。没有必要设置 **LC\_CTYPE**, **LC\_TIME**。更多的信息参考特定语言的FreeBSD文。

当在的配置文件中设置下面个量：

- **LANG** POSIX® 置本地化语言功能。

- `MM_CHARSET` 用程序的MIME字符集。

包括用的shell配置，特定的用配置和X11配置。

#### 24.3.4.1. 置本地化的方法

有方法来置本地化，接下来都会描述。第一（推）就是在 [登入分](#) 里面指定境。第二方法是把境加到shell的 [文件](#) 里面。

##### 24.3.4.1.1. 登入分方法

方法允把本地化名称和MIME字符集的境量可能的shell，而不是加到个特定shell的 [文件](#) 里面。[用置 Level Setup](#) 允普通用自己完成个置，而[管理置](#)需要超用限。

##### 24.3.4.1.1.1. 用置

有一个置用根目文件login\_conf的小例子，它上述个量置了Latin-1。

```
me:\
:charset=ISO-8859-1:\
:lang=de_DE.ISO8859-1:
```

是一个login\_conf置繁体中文的BIG-5的例子。置下面的大部分量，[因很多件都没有](#)中文，日文和文置正的本地化量。

```
#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:\
:setenv=LC_COLLATE=zh_TW.Big5:\
:setenv=LC_CTYPE=zh_TW.Big5:\
:setenv=LC_MESSAGES=zh_TW.Big5:\
:setenv=LC_MONETARY=zh_TW.Big5:\
:setenv=LC_NUMERIC=zh_TW.Big5:\
:setenv=LC_TIME=zh_TW.Big5:\
:charset=big5:\
:xmodifiers="@im=gcin": #Set gcin as the XIM Input Server
```

更多的信息参考[管理置](#)和[login.conf\(5\)](#)

##### 24.3.4.1.2. 管理置

用的登入分在 /etc/login.conf里面是否置了正的言。主要定下面的几个置：

```
language_name|Account Type Description:\n:charset=MIME_charset:\n:lang=locale_name:\n:tc=default:
```

再次使用前面的Latin-1的例子：

```
german|German Users Accounts:\n:charset=ISO-8859-1:\n:lang=de_DE.ISO8859-1:\n:tc=default:
```

在修改用的登入型之前， 首先行下面的命令：

```
# cap_mkdb /etc/login.conf
```

以便使在 /etc/login.conf 中新配置生效。

**24.3.4.1.3.** 使用 **vipw(8)** 改登入型。

使用**vipw**添加新用户， 看起来像下面：

```
user:password:1111:11:language:0:0:User Name:/home/user:/bin/sh
```

**24.3.4.1.4.** 用**adduser(8)**改登入型。

用**adduser**添加新用户看起来像下面：

- 在/etc/adduser.conf里面置**defaultclass = 语言**。记住， 必须使用其它语言的所有用置 **缺省**。
- 一次使用**adduser(8)**的时候， 一个特定语言的可性回答会像下面出：

```
Enter login class: default []:
```

- 如果打算一个用使用外一语言， ：

```
# adduser -class language
```

**24.3.4.1.5.** 使用**pw(8)**改登入型。

如果使用**pw(8)**来添加新用户， 使用：

```
# pw useradd user_name -L language
```



不推荐使用这个方法，因为它需要一个可能的shell程序一个不同的文件。  
方法来代替这个方法。

用[登入分](#)

了置本地化名称和MIME字符集，只要在/etc/profile或 /etc/csh.login文件里面置个量。下面我使用德做例子：

在/etc/profile里面：

```
LANG=de_DE.ISO8859-1; export LANG
MM_CHARSET=ISO-8859-1; export MM_CHARSET
```

或在/etc/csh.login里面：

```
setenv LANG de_DE.ISO8859-1
setenv MM_CHARSET ISO-8859-1
```

外，可以把上面的置添加到/usr/shared/skel/dot.profile（和前面的/etc/profile一），或者/usr/shared/skel/dot.login（和前面的/etc/csh.login一）。

于X11：

在\$HOME/.xinitrc里面：

```
LANG=de_DE.ISO8859-1; export LANG
```

或者：

```
setenv LANG de_DE.ISO8859-1
```

依的shell(看上面)。

## 24.3.5. 控制台置

于所有的C字符集，在/etc/rc.conf中用正在的言置正的控制台字符：

```
font8x16=font_name
font8x14=font_name
font8x8=font_name
```

儿的font\_name来自于/usr/shared/syscons/fonts目，不.fnt后。

如果需要的，通 **sysinstall** 来配置与字 C 字符集的 keymap 和 screenmap。在 sysinstall 中，Configure 之后 Console 即可行配置。除此之外，也可以在 /etc/rc.conf 中加入似下面的配置：

```
scrnmap=screenmap_name
keymap=keymap_name
keychange="fkey_number sequence"
```

的screenmap\_name是来自/usr/shared/syscons/scrnmaps目，不.scm后。一个映射字体的屏幕布局通常被作一个工作区，用来在VGA适配器字体矩上展8位到9位。如果屏幕字体是使用一个8位的排列，要移些字母些区域。

如果在/etc/rc.conf里面用了moused daemon：

```
moused_enable="YES"
```

那需要在下一段鼠指信息。

默情况下，syscons(4)程序的鼠指在字符集中占用0xd0-0xd3的。如果的言使用个，必把指移出个。要个，需要在/etc/rc.conf 中加入：

```
mousechar_start=3
```

里，keymap\_name 来自于 /usr/shared/syscons/keymaps 目，但去掉了 .kbd 后。如果不定使用一个布局，可以使用 kbdmap(1) 来，而无需反重。

通常，keychange 是定功能，匹配定的端型来是必需的，因功能序列无法在布局中定。

此外并在/etc/ttys 中已所有的ttyv\* 配置了正的端型。目前，相的默定是：

| 字符集置                    | 端型       |
|-------------------------|----------|
| ISO8859-1 or ISO8859-15 | cons25l1 |
| ISO8859-2               | cons25l2 |
| ISO8859-7               | cons25l7 |
| KOI8-R                  | cons25r  |
| KOI8-U                  | cons25u  |
| CP437 (VGA default)     | cons25   |
| US-ASCII                | cons25w  |

于多字字符言，可以在 /usr/ports/language 目中使用正的FreeBSD port。一些port以控制台出，而系把它作串行vty端，因此，必X11和串行控制台准足的vty端。下面是在控制台中使用其他言的程的部分列表：

| 言                           | 特定区域   |
|-----------------------------|--|
| Traditional Chinese (BIG-5) | chinese/big5con                              |
| Japanese                    | japanese/kon2-16dot or japanese/mule-freewnn |



| 语言     | 特定区域                       |
|--------|----------------------------|
| Korean | <a href="#">korean/han</a> |

### 24.3.6. X11设置

虽然X11不是FreeBSD的一部分，但我已为FreeBSD用包含了一些信息。具体可以参考[Xorg Web 站点](#)或是使用的X11 Server的网站。

在~/Xresources里面，可以当调整特定应用程序的I18N设置（如字体，菜单等）。

#### 24.3.6.1. 显示字体

安装 Xorg 服务器 ([x11-servers/xorg-server](#))，然后安装语言的 TrueType® 字体。设置正的地区信息，将能在菜单和其它地方看到所的语言。

#### 24.3.6.2. 输入非英语字符

X11输入方法（XIM）是所有X11客户端的一个新标准。所有将作为XIM客户端来写的X11应用程序从XIM输入服务器输入。不同的语言有几XIM服务器可用。

### 24.3.7. 打印机设置

一些语言的C字符集通常是用硬编码来打印机的。更或多位的字符集需要特定的设置，我推荐使用apsfilter。也可以使用特定语言设备把文本转换为 PostScript®或PDF格式。

### 24.3.8. 内核和文件系统

FreeBSD 的快速文件系统（FFS）是完全支持 8-位 字符的，因此它可以被用于任何语言的 C 字符集（参[multibyte\(3\)](#)），但在文件系统中不会保存字符集的名字；也就是，它不加修改地保存 8-位信息，而并不知道如何。正式来，FFS 目前不支持任何形式的或多字字符集。不，某些或多字符集提供了独立的 FFS 的丁来帮助用于它的支持。目前些要是无法移植的，要于粗，因此我不打算把它加入到源代码中。参考相语言的 Web 站点，以了解于些丁的一情况。

FreeBSD MS-DOS®已能配置成用在MS-DOS®上，Unicode字符集和可的FreeBSD文件系统字符集的更多信息，参考 [mount\\_msdosfs\(8\)](#) 机手册。

## 24.4. 为I18N程序

多FreeBSD Ports已支持I18N了。他中的一些都用-I18N作。些和其他很多程序已内建I18N的支持，不需要考其他的事了。

然而一些像MySQL的语言程序需要重新配置字符集，可在 Makefile里面置，或者直接把参数配置。

## 24.5. 本地化FreeBSD

### 24.5.1. 俄语 (KOI8-R)

关于KOI8-R的更多信息请参见[KOI8-R参考 \(Russian Net Character Set\)](#)。

#### 24.5.1.1. 本地设置

把下面的行加入到您的 ~/.login\_conf 文件：

```
me:My Account:\
    :charset=KOI8-R:\
    :lang=ru_RU.KOI8-R:
```

参看前面的设置[本地化](#)的例子。

#### 24.5.1.2. 控制台设置

- 把下面一行加到 /etc/rc.conf：

```
mousechar_start=3
```

- 并在 /etc/rc.conf 里面添加如下设置：

```
keymap="ru.utf-8"
scrnmap="utf-82cp866"
font8x16="cp866b-8x16"
font8x14="cp866-8x14"
font8x8="cp866-8x8"
```

- 关于/etc/ttys里面的ttyv\*，要使用 **cons25r**作终端类型。

参看前面的设置[控制台](#)的例子。

#### 24.5.1.3. 打印机设置

既然大多数俄语字符的打印机遵循CP866的标准，那您需要一个从KOI8-R到CP866的特定输出器。  
的一个输出器默认地安装在 /usr/libexec/lpr/ru/koi2alt。 一个支持俄语的打印机的/etc/printcap  
看起来是这样的：

```
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:
```

更多信息参考[printcap\(5\)](#)手册。

#### 24.5.1.4. MS-DOS® 文件系统 和 俄 文件名

下面的例子是在挂上MS-DOS® 文件系统后， 用 俄 文件名支持的fstab(5)：

```
/dev/ad0s2      /dos/c  msdos   rw,-Wkoi2dos,-Lru_RU.KOI8-R 0 0
```

-L 用于 地区名称， 而 -W 用于 置字符 表。要使用 -W ， 一定要首先挂接 /usr， 然后再挂接 MS-DOS® 分区， 因 表是放在 /usr/libdata/msdosfs 的。要了解 一的 ， 参考 mount\_msdosfs(8) 手册。

#### 24.5.1.5. X11 置

1. 首先 行前面介的 非-X 的本地化 置。
2. 如果 正使用 Xorg， 安装 x11-fonts/xorg-fonts-cyrillic package。

在 /etc/X11/xorg.conf 文件中的 "Files" 小 。 下面的行， 加到任何其它 FontPath 之前：

```
FontPath      "/usr/local/lib/X11/fonts/cyrillic"
```



看 ports 中的其它西里 字体。

3. 要激活俄 ， 需要在 xorg.conf 文件的 "Keyboard" 小 中加入下列内容：

```
Option "XkbLayout"      "us,ru"  
Option "XkbOptions"     "grp:toggle"
```

要 信 XkbDisable 已 (注 掉) 了。

RUS/LAT 的切 用 CapsLock。老的 CapsLock 功能可以通 Shift + CapsLock 来模 (只有在LAT模式的 候)。

使用 grp:toggle ， RUS/LAT 切 将是 右 Alt， 而使用 grp:ctrl\_shift\_toggle 表示切 是 Ctrl + Shift。使用 grp:caps\_toggle ， RUS/LAT 切 是 CapsLock。旧的 CapsLock 功能仍可通 Shift + CapsLock (只 LAT 模式有效)。由于不明原因， grp:caps\_toggle 在 Xorg 中无法使用。

如果 的 上有 "Windows®" ， 但 RUS 模式下， 某些非字母 映射不正常， 在 的 xorg.conf 文件中加入下面 行：

```
Option "XkbVariant"     ",winkeys"
```



俄 的 XKB 可能并不 某些不具 本地化功能的 用程序所支持。



本地化程序最低限度在程序中使用 `XtSetLanguageProc (NULL, NULL, NULL);` 函数。

参 [KOI8-R for X Window](#) 以得于 X11 用行本地化的指。

### 24.5.2. 置繁体中文

FreeBSD-Taiwan 有一个使用很多中文ports的中文化指南在

<http://netlab.cse.yzu.edu.tw/~statue/freebsd/zh-tut/>。目前, [FreeBSD 中文化指南](#) 的人 是 沈俊 [statue@freebsd.sinica.edu.tw](mailto:statue@freebsd.sinica.edu.tw)。

沈俊 [statue@freebsd.sinica.edu.tw](mailto:statue@freebsd.sinica.edu.tw) 利用 FreeBSD-Taiwan 的 [zh-L10N-tut](#) 建立了 [Chinese FreeBSD Collection \(CFC\)](#)。相的 packages 和脚本等可以在 <ftp://freebsd.csie.nctu.edu.tw/pub/taiwan/CFC/> 到。

### 24.5.3. 德本地化 (合所有的ISO 8859-1言)

Slaven Rezic [eserte@cs.tu-berlin.de](mailto:eserte@cs.tu-berlin.de) 写了一个在 FreeBSD 机器下如何使用日曼言的德指南。德教程可以在 <http://user.cs.tu-berlin.de/~eserte/FreeBSD/doc/umlaute/umlaute.html> 到。

### 24.5.4. 希本地化

Nikos Kokkalis [nickkokkalis@gmail.com](mailto:nickkokkalis@gmail.com) 撰写了于在 FreeBSD 上支持希的完整文章, 在 <http://www.freebsd.org/doc/el/articles/greek-language-support/>。注意篇文章 只有 希的版本。

### 24.5.5. 日和本地化

日本地化参考<http://www.jp.FreeBSD.org/>, 参考 <http://www.kr.FreeBSD.org/>。

### 24.5.6. 非英的FreeBSD文

一些 FreeBSD 的献者已将部分 FreeBSD 文翻成了其他言。可在 [主站](#) 以及 `/usr/shared/doc` 到。

# Chapter 25. 更新与升级 FreeBSD

## 25.1. 概述

FreeBSD 在发行版之始是持续更新的。一些人喜欢使用官方发行的版本，而一些人喜欢与最新的版本保持同步。然而，即使是官方的发行版本也常常需要安全补丁和重大修正方面的更新。不管使用了哪个版本，FreeBSD 都提供了所有更新系统所需的工具，可以轻松地在不同版本间升级。这一章将帮助你决定是跟踪哪个系统是持续使用某个发行的版本。同时列出了一些保持系统更新所需的基本工具。

读了本章后，你将了解到：

- 使用哪些工具来更新系统与 Ports Collection。
- 如何使用 `freebsd-update`, `CVSup`, `CVS`, or `CTM` 哪个系统保持更新。
- 如何比较已安装的系统与原来已知状态。
- 如何使用 `CVSup` 或者文本 `ports` 来更新本地的文本。
- 哪个分支 `FreeBSD-STABLE` 和 `FreeBSD-CURRENT` 的区别。
- 如何通过 `make buildworld` 重新安装整个基本系统(等等)。

在本章之前，你应该了解的：

- 正确配置网络接口(高网络)。
- 知道如何安装附加的第三方软件(安装应用程序、[Packages](#) 和 [Ports](#))。



整个这一章中，`cvsup` 命令都被用来获取 FreeBSD 源代码的更新。你需要安装 `net/cvsup` port 或者二进制包(如果你不想要安装图形界面的 `cvsup` 客户端的，你可以安装 `net/cvsup-without-gui` port)。你也可以使用 `csup(1)` 代替，它已经在基本系统的一部分了。

## 25.2. FreeBSD 更新

打安全补丁是对于计算机系统的一个重要部分，特别是对于操作系统。对于 FreeBSD 来说，很的一段时期以来都不是一件容易的事情。补丁打在源代码上，代码需要被重新编译，然后再重新安装后的程序。

FreeBSD 引入了 `freebsd-update` 工具之后便不再是这样了。这个工具提供了 2 个功能。第一，它可以把二编译的安全和勘误更新直接用于 FreeBSD 的基本系统，而不需要重新编译和安装。第二，这个工具支持主要跟次要的发行版的升级。



由安全小组支持的各个体系和发行版都可使用二进制更新。在升级到一个新的发行版本之前，请先看一下当前发行版的声明，因为它可能包含有关于期望升级版本的重要消息。这些发行声明可以通过以下链接：<http://www.FreeBSD.org/releases/>。

如果 `crontab` 中存在有用到 `freebsd-update` 特性的部分，那么这些在开始以下操作前必须先被禁止。

### 25.2.1. 配置文件

有些用户可能希望调整配置文件 `/etc/freebsd-update.conf` 中的默认配置来更好地控制升级的过程。

可用的参数在文中介绍的很细，但下面的这些可能需要进一步的解释：

```
# Components of the base system which should be kept updated.
Components src world kernel
```

这个参数是控制 FreeBSD 的一部分将被保持更新。默认的是更新源代码，整个基本系统有内核。这些部件跟安装的那些相同，例如，在里加入 `world/games` 就会允许打入游戏相关的补丁。使用 `src/bin` 是允许更新 `src/bin` 目录中的源代码。

最好的做法是把这个选项保留默认，因为如果要修改它去包含一些指定的目录，就需要用列出每一个想要更新的目录。这可能会引起可怕的后果，因为部分的源代码和二进制程序得不到同步。

```
# Paths which start with anything matching an entry in an IgnorePaths
# statement will be ignored.
IgnorePaths
```

添加路径，比如 `/bin` 或者 `/sbin` 这些指定的目录在更新过程中不被修改。这个选项能防止本地的修改被 `freebsd-update` 覆盖。

```
# 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
```

更新指定目录中的未被修改的配置文件。用过的任何修改都会使这些文件的自更新失效。还有一个选项，`KeepModifiedMetadata`，这个能让 `freebsd-update` 在合并时保存修改。

```
# 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/
```

一个 `freebsd-update` 选项是合并的配置文件的列表。文件合并的过程是一系列的 `diff(1)` 补丁类似于更少的 `mergemaster(8)` 合并的选项是接受，打一个文本编辑器，或者 `freebsd-update` 会被中止。在不能确定的时候，先 `/etc` 然后接受合并。更多关于 `mergemaster` 的信息参考 `mergemaster`。

```
# 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补丁和临时文件的。用做一个版本升级的，因此至少有 1 GB 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
```

当置成 `yes` 时，`freebsd-update` 将假每个 `Components` 列表完整的，并且此列表以外的项目不会修改。也就是 `freebsd-update` 会更新 `Components` 列表里的一个文件。

### 25.2.2. 安全补丁

安全补丁存储在程序的机器上，可以使用如下的命令下载并安装：

```
# freebsd-update fetch
# freebsd-update install
```

如果内核打了补丁，那系统需要重新安装。如果一切都顺利，系统就将被打好了补丁而且 `freebsd-update` 可由夜 `cron(8)` 运行。在 `/etc/crontab` 中加入以下条目足以完成任何：

```
@daily                                root    freebsd-update cron
```

这条是每天运行一次 `freebsd-update` 工具。用这个方法，使用了 `cron` 参数，`freebsd-update` 检查是否存在更新。如果有了新的补丁，就会自动下载到本地的磁盘，但不会自动系统打上。`root` 会收到一封邮件告知需手工安装补丁。

如果出了错，可以使用下面的 `freebsd-update` 命令回退到上一次的修改：

```
# freebsd-update rollback
```

完成以后如果内核或任何的内核模块被修改的，就需要重新安装系统。这将使 `FreeBSD` 安装新的二进制程序到内存。

`freebsd-update` 工具只能自动更新 `GENERIC` 内核。如果使用自行编译的内核，在 `freebsd-update` 安装完更新的其余部分之后需要手工重新编译和安装内核。不过，`freebsd-update` 会并更新位于 `/boot/GENERIC` (如果存在) 中的 `GENERIC` 内核，即使它不是当前 (正在运行的) 系统的内核。



保存一个 `GENERIC` 内核的副本到 `/boot/GENERIC` 是一个明智的主意。在断电多时，以及在 [重大和次要的更新](#) 中介入的使用 `freebsd-update` 更新系统会很有用。

除非修改位于 `/etc/freebsd-update.conf` 中的配置，`freebsd-update` 会随其他安装一起内核的源代码运行更新。重新编译并安装定制的内核可以以通常的方式来运行。



通过 `freebsd-update` 分布的更新有并不会涉及内核。如果在运行 `freebsd-update install` 的过程中内核代没有运行，就没有必要重新内核了。不过，由于 `freebsd-update` 每次都会更新 `/usr/src/sys/conf/newvers.sh` 文件，而修版本 (uname -r 报告的 -p 数字) 来自一个文件，因此，即使内核没有生，重新内核也可以 `uname(1)` 报告准的修版本。在大多系统上做会比有帮助，因一信息可以迅速反映机器上安装的文件更新情况。

### 25.2.3. 重大和次要的更新

个过程会除旧的目文件和，将使大部分的第三方用程序无法除。建将所有安装的 ports 先除然后重新安装，或者后使用 `ports-mgmt/portupgrade` 工具升。大多数用将会使用如下命令：

```
# portupgrade -af
```

将保所有的西都会被正的重新安装。注意境量 `BATCH` 置成 `yes` 的将在整个过程中所有回答 `yes`，会助在过程中免去人工的介入。

如果正在使用的是定制的内核，升操作会一些。会需要将一 `GENERIC` 内核的副本放到 `/boot/GENERIC`。如果系中没有 `GENERIC` 内核，可以用以下方法之一来安装：

- 如果只一次内核，位于 `/boot/kernel.old` 中的内核，就是 `GENERIC` 的那一个。只需将个目改名 `/boot/GENERIC` 即可。
- 假如能直接接触机器，可以通过 `CD-ROM` 介来安装 `GENERIC` 内核。将安装入光，并运行下列命令：

```
# mount /cdrom
# cd /cdrom/X.Y-RELEASE/kernels
# ./install.sh GENERIC
```

需要将 `X.Y-RELEASE` 替正在使用的版本。 `GENERIC` 内核默情况下会安装到 `/boot/GENERIC`。

- 如果前面的方法都不可用，可以使用源代来重新和安装 `GENERIC` 内核：

```
# cd /usr/src
# env DESTDIR=/boot/GENERIC make kernel
# mv /boot/GENERIC/boot/kernel/* /boot/GENERIC
# rm -rf /boot/GENERIC/boot
```

如果希望 `freebsd-update` 能正地将内核 `GENERIC`，必保没有 `GENERIC` 配置文件行任何。此外，建取消任何其他特殊的 (例如使用空的 `/etc/make.conf`)。

上述并不需要使用个 `GENERIC` 内核来引系。

重大和次要的更新可以由 `freebsd-update` 命令后指定一个行版本来行，例来，下面的命令将助升到 FreeBSD 8.1：



```
# freebsd-update -r 8.1-RELEASE upgrade
```

在执行这个命令之后，`freebsd-update` 将会先解析配置文件和评估当前的系统以取得更新系统所需的必要信息。然后便会显示出一个包含了已安装到与未安装到的文件列表。例如：

```
Looking up update.FreeBSD.org mirrors... 1 mirrors found.
Fetching metadata signature for 8.0-RELEASE from update1.FreeBSD.org... done.
Fetching metadata index... done.
Inspecting system... done.

The following components of FreeBSD seem to be installed:
kernel/smp src/base src/bin src/contrib src/crypto src/etc src/games
src/gnu src/include src/krb5 src/lib src/libexec src/release src/rescue
src/sbin src/secure src/share src/sys src/tools src/ubin src/usbin
world/base world/info world/lib32 world/manpages

The following components of FreeBSD do not seem to be installed:
kernel/generic world/catpages world/dict world/doc world/games
world/proflibs

Does this look reasonable (y/n)? y
```

此时，`freebsd-update` 将会下载所有升级所需的文件。在某些情况下，用户可能被问及需安装些什么和如何行之的。

当使用定制内核时，前面的输出会产生以下面的警告：

```
WARNING: This system is running a "MYKERNEL" kernel, which is not a
kernel configuration distributed as part of FreeBSD 8.0-RELEASE.
This kernel will not be updated: you MUST update the kernel manually
before running "/usr/sbin/freebsd-update install"
```

此输出可以安全地忽略这个警告。更新的 GENERIC 内核将在升级过程的中使用。

在下载完本地系统的补丁之后，一些补丁会被应用到系统上。这个过程需要消耗的时间取决于机器的速度和其配置。在这个过程中将会配置文件所做的行合并 - 一部分需要用户的参与，文件可能会自动合并，屏幕上也可能输出一个编辑器，用于手工完成合并操作。在理想过程中，合并成功的结果会显示应用。失败或被忽略的合并，会导致一个过程的停止。用户可能会希望编辑 `/etc` 并在之后手工合并重要的文件，例如 `master.passwd` 和 `group`。



系统至此没有被修改，所有的补丁和合并都在一个外部目录中进行。当所有的补丁都被成功的打上了以后，所有的配置文件都被合并后，我部就已完成整个升级过程中最困难的部分，下面就需要用来安装一些更新了。

当所有的我部就已完成整个升级过程中最困难的部分，下面就需要用来安装一些更新了。

一旦这个过程完成后，使用如下的命令将升级后的文件安装到磁盘上。

```
# freebsd-update install
```

内核和内核模块会首先被打上补丁。此必须重新引导计算机。如果您使用的是定制的内核，请使用 [nextboot\(8\)](#) 命令来将下一次用于引导系统的内核 `/boot/GENERIC` (它会被更新)：

```
# nextboot -k GENERIC
```



在使用 `GENERIC` 内核之前，确信它包含了用于引导系统所需的全部程序（如果您是在程序运行升级操作，确信网络也是存在的）。特别要注意的情形是，如果之前的内核中静默了通常以内核模块形式存在的程序，一定要通过 `/boot/loader.conf` 机制来将这些模块添加到 `GENERIC` 内核的基础上。此外，您可能也希望取消不重要的服务、磁盘和网挂等等，直到升级过程完成为止。

现在可以用更新后的内核引导系统了：

```
# shutdown -r now
```

在系统重新启动后，需要再次运行 `freebsd-update`。升级的状态被保存着，因此 `freebsd-update` 就无需重新开始，但是会删除所有旧的共享库和目录文件。运行如下命令开始下一个阶段的升级：

```
# freebsd-update install
```



取决于是否有您的版本更新，通常只有 2 个而不是 3 个安装阶段。

您可能需要重新安装第三方软件。这样做的原因是某些已安装的软件可能依赖于在升级过程中已删除的。可使用 [ports-mgmt/portupgrade](#) 自动化这个过程，以如下的命令开始：

```
# portupgrade -f ruby
# rm /var/db/pkg/pkgdb.db
# portupgrade -f ruby18-bdb
# rm /var/db/pkg/pkgdb.db /usr/ports/INDEX-*.db
# portupgrade -af
```

一旦这个阶段完成了以后，再最后一次运行 `freebsd-update` 来结束升级过程。运行如下命令处理升级中的所有：

```
# freebsd-update install
```

如果您使用 `GENERIC` 内核来引导系统，现在是按照通常的方法重新安装并安装新的定制内核的时候了。

重新安装机器上入新版本的 FreeBSD 升级过程至此就完成了。

## 25.2.4. 系统快照

`freebsd-update` 工具也可被用来备份一个已知完好的 FreeBSD 系统当前的版本。它可备份当前的系统工具，和配置文件。使用以下的命令开始快照：

```
# freebsd-update IDS >> outfile.ids
```



这个命令的名称是 IDS，它并不是一个像 `security/snort` 那样的入侵检测系统的替代品。因为 `freebsd-update` 在磁盘上存数据，很自然它也有被篡改的可能。当然也可以使用一些方法来降低被篡改的可能性，比如设置 `kern.securelevel` 和不使用把 `freebsd-update` 数据放在只读文件系统上，例如 DVD 或安全存放的外置 USB 磁盘上。

在系统将会被快照，生成一个包含了文件和它的 `sha256(1)` 哈希值的清单，已知发行版中的与当前系统中安装的文件将会被打印到屏幕上。这就是什么输出被送到了 `outfile.ids` 文件。它的输出太无法用肉眼对照，而且会很快填满控制台的缓冲区。

这个文件中有非常多的行，但输出的格式很容易分析。例如来，要得到一行发行版中不同哈希值的文件列表，已可使用如下的命令：

```
# cat outfile.ids | awk '{ print $1 }' | more
/etc/master.passwd
/etc/motd
/etc/passwd
/etc/pf.conf
```

输出清单短后的，其是有更多的文件。其中有些文件并非人工修改，比如 `/etc/passwd` 被修改是因为添加了用户系统。在某些情况下，有额外的一些文件，如内核模块与 `freebsd-update` 的不同是因为它被更新了。除了指定的文件或目录排除在外，把它添加到 `/etc/freebsd-update.conf` 的 `IDSIgnorePaths` 中。

除了前面清单的部分之外，它也能被当作是升级方法的补充。

## 25.3. Portsnap：一个 Ports Collection 更新工具

FreeBSD 基本系统也包括了一个更新 Ports Collection 的工具：`portsnap(8)`。在运行之后，它会下一个工程网站，校验安全密钥，然后下一个 Ports Collection 的拷贝。密钥是用来校验所有下载文件的完整性，保证它在安装是未被修改。使用以下的命令下载最新的 Ports Collection：

```
# portsnap fetch
Looking up portsnap.FreeBSD.org mirrors... 3 mirrors found.
Fetching snapshot tag from portsnap1.FreeBSD.org... done.
Fetching snapshot metadata... done.
Updating from Wed Aug 6 18:00:22 EDT 2008 to Sat Aug 30 20:24:11 EDT 2008.
Fetching 3 metadata patches.. done.
Applying metadata patches... done.
Fetching 3 metadata files... done.
Fetching 90 patches....10....20....30....40....50....60....70....80....90. done.
Applying patches... done.
Fetching 133 new ports or files... done.
```

这个例子展示的是 `portsnap(8)` 并校验了几个用于当前 ports 的补丁。这表明以前运行，如果是第一次运行，那只会下载 Ports Collection。

在 `portsnap(8)` 成功地完成一次 `fetch` 操作之后，会将校验的 Ports 套件和后面的补丁保存在本地。首次运行 `portsnap` 之后，必须使用 `extract` 安装下载的文件：

```
# portsnap extract
/usr/ports/.cvsignore
/usr/ports/CHANGES
/usr/ports/COPYRIGHT
/usr/ports/GIDs
/usr/ports/KNOBS
/usr/ports/LEGAL
/usr/ports/MOVED
/usr/ports/Makefile
/usr/ports/Mk/bsd.apache.mk
/usr/ports/Mk/bsd.autotools.mk
/usr/ports/Mk/bsd.cmake.mk
...
```

使用 `portsnap update` 命令更新已安装的 Ports：

```
# portsnap update
```

至此更新就完成了，然后便可以使用更新后的 Ports Collection 来安装或升级应用程序。

`fetch` 和 `extract` 或 `update` 可以作组合的操作行，如下例所示：

```
# portsnap fetch update
```

这个命令将会下载最新版本的 Ports 并更新本地位于 `/usr/ports` 的拷贝。

## 25.4. 更新系附的文

除了基本系和 Ports 套件之外，文也是 FreeBSD 操作系统的一个组成部分。尽管是可以通 [FreeBSD 网站](#) 来最新的 FreeBSD 文，一些用的网接可能很慢，甚至完全没有网接。幸运的是，有很多方法可以用来更新随行版本附的 FreeBSD 文的本地副本。

### 25.4.1. 使用 CVSup 来更新文

FreeBSD 文的源代和安装版本都可以通 CVSup 来以与基本系 (参考 [重新 "world"](#)) 似的方法来升。一中将会介：

- 如何安装文所需的工具集，用于从源代来 FreeBSD 文所需的那些工具。
- 如何使用 CVSup 将文下载到 /usr/doc。
- 如何从源代 FreeBSD 文，并将其安装到 /usr/shared/doc。
- 文的程中支持的一些，例如只某些言的版本，或只特定的出格式。

### 25.4.2. 安装 CVSup 和文工具集

从源代 FreeBSD 文需要大量的工具。些工具并不是 FreeBSD 基本系的一部分，因些工具需要占用大量的磁空，而且并不是所有 FreeBSD 用都有用；只有活地撰写 FreeBSD 新文，或常从源代更新文的用才需要些工具。

全部所需的工具，均可通 Ports 套件来安装。 [textproc/docproj](#) port 是由 FreeBSD 文提供的方便安装和更新些工具的主 port。



如果不需要 PostScript® 或 PDF 文的，也可以考安装 [textproc/docproj-nojadetex](#) port。套文工具集包含除了 teTeX typesetting 引之外的其他全部工具。teTeX 是一个很大的工具集，因此如果不需要 PDF 出的，排除它会省很多和磁空。

如欲了解于安装和使用 CVSup 的一信息，参 [使用 CVSup](#)。

### 25.4.3. 更新文源代

CVSup 工具能下文的源代的原始副本，可使用 /usr/shared/examples/cvsup/doc-supfile 文件作配置模板来修改。在 doc-supfile 中的默认主机名是一个无效的占位主机名，但 [cvsup\(1\)](#) 能通命令行来指定主机名，因此文源代可以使用下面的命令从 CVSup 服务器得：

```
# cvsup -h cvsup.FreeBSD.org -g -L 2 /usr/shared/examples/cvsup/doc-supfile
```

将 [cvsup.FreeBSD.org](#) 改最近 CVSup 服务器。参 [CVSup 站点](#) 于像站点的完整列表。

初始的文源代下需要一些，需要耐心等待它完成。

后的更新可以用同的命令来行。由于 CVSup 工具只下次行之后所生的更新，因此在首次行之后再行 CVSup 是很快的。

在出源代之后，可以使用一由 /usr/doc 目中的 Makefile 支持的方法来更新它。通在

/etc/make.conf 中配置 **SUP\_UPDATE**、**SUPHOST** 和 **DOCSUPFILE**，可以通过行：

```
# cd /usr/doc
# make update
```

来完成更新。典型的 /etc/make.conf 中的 **make(1)** 是：

```
SUP_UPDATE= yes
SUPHOST?= cvsup.freebsd.org
DOCSUPFILE?= /usr/shared/examples/cvsup/doc-supfile
```



将 **SUPHOST** 和 **DOCSUPFILE** 的使用 **?=** 来指定的好，是使 **make** 命令行能覆盖这些。在向 **make.conf** 中添加时推时做，以避免在时反时修改个文件。

#### 25.4.4. 文档源代码中可的

FreeBSD 文档的更新和系统支持一些方便只更新一部分文档，或只特定格式及文档的。这些可以在 /etc/make.conf 文件中配置，也可以通过 **make(1)** 工具来指定。

这些包括：

##### **DOC\_LANG**

准和安装的言列表。例如，指定 **en\_US.ISO8859-1** 表示只英文版的文档。

##### **FORMATS**

准出的格式列表。目前，系统支持 **html**、**html-split**、**txt**、**ps**、**pdf**、和 **rtf**。

##### **SUPHOST**

用于用来更新的 CVSup 服务器的主机名。

##### **DOCDIR**

用于安装文档的目录。默认 /usr/shared/doc。

如欲了解 FreeBSD 中其他可供配置的全局 **make** 量，参 **make.conf(5)**。

于 FreeBSD 文档系统的其他情，参 **FreeBSD 文档入门之新手必部分**。

#### 25.4.5. 从源代码安装 FreeBSD 文档

在 /usr/doc 中下了最新的文档源代码快照之后，就可以开始动手文档了。

要更新全部 **DOC\_LANG** 中定义的言的文档，需要行下面的命令：

```
# cd /usr/doc
# make install clean
```

如果在 `make.conf` 中配置了正确的 `DOCSUPFILE`、`SUPHOST` 和 `SUP_UPDATE` 项， 你可以将更新源代码和安装一起完成：

```
# cd /usr/doc
# make update install clean
```

如果只需要更新某个特定语言的文档， 可以在 `/usr/doc` 中与之同名的目录中运行 `make(1)`：

```
# cd /usr/doc/en_US.ISO8859-1
# make update install clean
```

此外， 你可以透过 `make` 变量 `FORMATS` 来控制输出格式， 例如：

```
# cd /usr/doc
# make FORMATS='html html-split' install clean
```

## 25.4.6. 使用文档 Ports

在之前的章节中， 我已展示了从源代码更新 FreeBSD 文档的方法。 基于源代码的更新的方法可能并不是适用于所有的 FreeBSD 系统都可行有效。 文档源代码需要一大堆的工具， 文档工具， 对于 CVS 的一定了解和从其中取出源代码， 还有一些你已经出过代的手续。 这一章我将介绍如何使用 Ports 来更新已安装的 FreeBSD 文档：

- 下载并安装一个好的文档快照， 而不用在本地系统任何部分 (即使不再需要安装整个文档工具了)。
- 下载文档的源代码并使用 ports 框架 (使得取出和安装的过程更容易些)。

更新 FreeBSD 文档的方法都由一个文档工程 <[doceng@FreeBSD.org](mailto:doceng@FreeBSD.org)> 每月更新的文档 ports 提供支持。 这些都列在了 FreeBSD Ports docs 虚拟分下面。

### 25.4.6.1. 安装和安装文档 Ports

文档 ports 使用 ports 的构建框架使得文档的安装得更加容易。 自动化了取出文档源代码， 配以合适的配置和命令行参数运行 `make(1)`， 它使得安装或卸载文档得就像安装 FreeBSD 其他 port 或二进制包那样容易。



一个特性便是当在本地系统文档 ports 时， 文档工具 ports 会被列入依赖关系， 并自行安装。

文档 ports 按以下方式：

- 一个 "主 port"， 在 `misc/freebsd-doc-en` 下可以找到个文档 port。 它是所有文档 ports 的基础。 在默认的情况下， 它只安装英文版文档。
- 一个 "合集 port"， `misc/freebsd-doc-all`， 它将构建并安装所有语言版本的所有文档。
- 最后是各个翻译的 "从属 port"， 比如：`misc/freebsd-doc-hu` 是匈牙利文版的文档。 所有这些都基于主 port 并会安装上相应语言的翻译文档。

以 `root` 用户身份运行如下的命令安装文档：

```
# cd /usr/ports/misc/freebsd-doc-en
# make install clean
```

它将会安装分章的英文版本 HTML 格式文档（与<http://www.FreeBSD.org> 上的相同）到 `/usr/local/shared/doc/freebsd` 目录。

#### 25.4.6.1.1. 文档的目录

文档 `ports` 有多用来修改默认行为的。以下是一段需要列表：

##### WITH\_HTML

允许构建 HTML 格式：每个文档一个单独的 HTML 文件。此文档的文件名情况而定通常是 `article.html`，或 `book.html`，另外附加一些图片。

##### WITH\_PDF

允许构建 Adobe® Portable Document Format，可使用 Adobe® Acrobat Reader®, Ghostscript 或者其他的 PDF 阅读器。此文档的文件名情况而定通常是 `article.pdf` 或 `book.pdf`。

##### DOCBASE

文档将被安装到的目录。默认为 `/usr/local/shared/doc/freebsd`。



注意默认的目录与 `CVSup` 方法所使用的目录不同。这是因为我正在安装的是一个 `port`，而 `ports` 通常会被安装到 `/usr/local` 目录。可以指定 `PREFIX` 变量覆盖默认。

这是一简短的关于如何使用以上提到变量来安装 PDF 格式的匈牙利文档：

```
# cd /usr/ports/misc/freebsd-doc-hu
# make -DWITH_PDF DOCBASE=share/doc/freebsd/hu install clean
```

#### 25.4.6.2. 使用文档 Packages

正如上文所述，从 `ports` 构建文档需要在本地安装一个文档工具和一些所需的磁盘空间。当不直接安装文档工具，或者从源代码需要太多的磁盘空间，我仍然可以安装很好的文档快照的 `ports`。

文档工程 <[doceng@FreeBSD.org](mailto:doceng@FreeBSD.org)> 每个月都会制作 FreeBSD 文档快照的包。这些二进制包可以通过包工具来操作，比如 `pkg_add(1)`，`pkg_delete(1)`，等等。



当使用二进制包，将安装所指定语言相关的 FreeBSD 文档的所有可用格式。

例如，以下的命令将安装最新的匈牙利文档：

```
# pkg_add -r hu-freebsd-doc
```



二进制包使用了以下与 `ports` 名称不同的命名格式：`lang-freebsd-doc`。这里的 `lang` 是语言代码的简短形式，比如 `hu` 表示匈牙利，或者 `zh_cn` 表示简体中文。



### 25.4.6.3. 更新文ports Ports

任何用于更新 ports 的工具都可以被用来更新已安装的文port。 例如， 下面的命令通 ports-mgmt/portupgrade 工具来更新已安装的匈牙利文二制包。

```
# portupgrade -PP hu-freebsd-doc
```

## 25.5. 追踪分支

FreeBSD 有个分支： FreeBSD-CURRENT 和 FreeBSD-STABLE。 一章将两个分支作相与如何保持的系更新。 我将先介 FreeBSD-CURRENT 然后是 FreeBSD-STABLE。

### 25.5.1. 使用最新的 FreeBSD CURRENT

里再次， FreeBSD-CURRENT 是 FreeBSD 的 "最前沿"。 FreeBSD-CURRENT 用要有高的技能， 并且有能力自己解决困的系。 如果是个 FreeBSD 新手， 那在安装之前最好三思。

#### 25.5.1.1. FreeBSD-CURRENT 是什么？

FreeBSD-CURRENT 是 FreeBSD 的展前沿。 包括了在下一个官方行的件中可能存在， 也可能不存在的展、 性改、 以及渡性的机制。 尽管多 FreeBSD 者天都会 FreeBSD-CURRENT 源代， 但有些代仍然会是不能的。 然些会很快解决， 但 FreeBSD-CURRENT 是来破坏是正希望的功能性改善， 很可能完全取决于取源代的机！

#### 25.5.1.2. 需要 FreeBSD-CURRENT ？

FreeBSD-CURRENT 合下三主要趣体：

1. FreeBSD 社区的成： 工作在源的某部分的人和保持 "最新" 需求的人。
2. FreeBSD 社区的成： 促使 FreeBSD-CURRENT 保持尽可能的健全而花去解决的的者； 以及那些意提出于 FreeBSD 化和体方向的建性建并且提供丁它的人。
3. 那些只是想注或了参考目的使用当前 (current) 源的人 (如， 了， 而不是行)。 些人也偶做做注或献代。

#### 25.5.1.3. FreeBSD-CURRENT 不是什么？

1. 追求最新功能， 听里面有一些很酷的新功能， 并希望成周围的人中第一个它的人。 尽管能因此首先了解到最新的功能， 但也意味着在出新的 bug 也首当其冲。
2. 修漏的快捷方式。 任何 FreeBSD-CURRENT 的既定版本在修已知漏的同又可能会生新的漏。
3. 无所不在的"官方支持"。 我尽最大努力在3个"合法的" FreeBSD-CURRENT 之一真人提供助， 但是我 没有提供技支持。 并不是因我是那不喜助人解困的无耻之徒 (如果我的是， 就不会制作 FreeBSD 了)。 我不能天回上百的消息， 而且我展 FreeBSD！ 在改善 FreeBSD 和回大量于代的之如果要做个的， 人会同前者。

#### 25.5.1.4. 使用 FreeBSD-CURRENT

1. 加入 FreeBSD-CURRENT 件列表 和 SVN src 头/-current 分支的修息 列表。 个不

是个好主意，而且很重要。如果你不去 [FreeBSD-CURRENT 邮件列表](#)，你就不会看到人所做的关于系统当前状态的说明，你就有可能在人们已经解决了一大堆问题面前跌倒。更重要的是你会看到一些重要的公告——关于系统的网络安全可能是至为重要的。

[SVN src 的 head/current 分支的修改进程](#) 列表允许你看到个性化的提交，因为有些提交与其它相关信息是同时的。

要加入这些列表，或其它可能的列表，看看 <https://lists.freebsd.org>，并且点击你想看的列。关于其它邮件列表的说明那里有提供。如果你有追踪整个原代码的更新，我建议你看看 [SVN 整个 src 的修改进程](#) (除了 "user" 与 "projects") 邮件列表。

## 2. 从 FreeBSD 镜像站点取源码。有几种方式：

与称作 standard-supfile 的 supfile 一起使用 [cvsup](#)，你可以从 /usr/shared/examples/cvsup 得到。它是最被推荐的方式，因为它允许一次取整个集合，以后就只取更改的部分。多人从 [cron](#) 运行 [cvsup](#)，以保持他们的源码自更新。你要定制上文的 supfile 脚本，并且配置 [cvsup](#) 以你的环境。

standard-supfile 例子是追踪指定的 FreeBSD 安全分支而指定的，而不是 FreeBSD-CURRENT。你需要三个文件并把如下行：

```
*default release=cvs tag=RELENG_X_Y
```



替换：

```
*default release=cvs tag=.
```

可以参手册中的 [CVS Tags](#) 章节得更多可用 tag 的说明。

使用工具 CTM。如果你的连接性能不太好(高价连接或只能通小子件存取)，CTM 是个。但它也有争并且常常得到坏文件。因此很少使用它，它也注定了不能长期用它来工作。于使用 9600 bps 或更快连接的人，我推荐使用 CVSup。

3. 如果你取源码是用于运行，而不只是看看，那你就取整个 FreeBSD-CURRENT，不要部分。你这样做的原因是源码的大部分都依赖于其他部分，要是你只取其中一部分的，保证会陷入麻烦。

在 FreeBSD-CURRENT 之前，你仔细看看 /usr/src 里的 Makefile 文件。尽管是部分的升级，但至少也要首先[安装新的内核和重建系统](#)。看看 [FreeBSD-CURRENT 邮件列表](#) 邮件列表和 /usr/src/UPDATING，会看到在其它循序的过程中保持最新，至于我向下一个行版移动是很有必要的。

4. 小心一点！如果你正运行 FreeBSD-CURRENT，我很想知道关于它的一些想法，尤其是关于漏洞或新的建设。非常欢迎有代的建设！

## 25.5.2. 使用最新的 FreeBSD STABLE

### 25.5.2.1. FreeBSD-STABLE 是什么？

FreeBSD-STABLE 是我自己的发展分支，我的主要行版就由此而来。这个分支会以不同速度进化，并且假定有些是第一次引入 FreeBSD-CURRENT 运行。然而，它仍然是个发展中的分支，意味着在一定的

候，FreeBSD-STABLE 源可能或不可能满足一些特殊的要求。它只不过是工程发展途径，并不是终端用户的源。

#### 25.5.2.2. 需要 FreeBSD-STABLE ？

如果你对追踪 FreeBSD 的进程或其做贡献，尤其是和下一个 "非" 的 FreeBSD 发行版有，考虑采用 FreeBSD-STABLE。

尽管安全更新也会加入 FreeBSD-STABLE 分支，但并不必使用 FreeBSD-STABLE 来达到目的。一个 FreeBSD 的安全公告都会解释如何修复受到影响的发行版中的，而因安全原因而去采用一个分支可能会引入一些不希望的修改。

尽管我们尽力保持 FreeBSD-STABLE 分支在任何时候都能正确运行，但没有人能担保它在任何时候都可以。此外，尽管代在加入 FreeBSD-STABLE 之前都是在 FreeBSD-CURRENT 上完成，但使用 FreeBSD-STABLE 的人要比使用 FreeBSD-CURRENT 的更多。有数据显示，角色里的各有些时候仍然会由于在 FreeBSD-CURRENT 不那明而在 FreeBSD-STABLE 暴露出来。

基于些原因，不推荐盲目地追踪 FreeBSD-STABLE，并且，在粗略地替代之前不要更新任何生服器到 FreeBSD-STABLE 也非常重要。

如果没有用于完成些工作的源，我们推荐使用最新的 FreeBSD 发行版，并使用发行版提供的二进制更新机制来在发行版之间完成转移。

#### 25.5.2.3. 使用 FreeBSD-STABLE

1. 加入 [FreeBSD-STABLE; 组件列表](#) 列表。随了解可能出在 FreeBSD-STABLE 里的 "build 依赖性" 或其它需要特别注意的。当正在考虑某些有争的修或更新，他就会在个组件列表里表声明，用机会回，看他于提出的化是否有什么。

加入相关的 SVN 列表来追踪所关心的分支。比如，如果在追踪 7-STABLE 分支，加入 [svn-src-stable-7](#) 列表。每次个分支上有改的时候就能看到提交，包括了修改可能引起的副作用之的相关信息。

要加入些列表或其他可用的，点 <https://lists.freebsd.org> 并点希望列表的。于其它明可以在那里看到。如果有兴趣追踪整个原代的更新，我们建整个 SVN 整个 src 的修息 (除了 "user" 与 "projects") 列表。

2. 如果正安装一个新系，并希望它行月从 FreeBSD-STABLE 快照，察看 [Snapshots](#) 网以了解更多信息。外，也可以从 [镜像站点](#) 安装最新的 FreeBSD-STABLE 发行版，并按照其中的明将系更新到最新的 FreeBSD-STABLE 源代。

如果已在行早的 FreeBSD 版本，并希望通源代方式升，可以通过 FreeBSD [镜像站点](#) 来完成。可以通过方式来行：..

与称作 stable-supfile 的 supfile 一起使用 [cvsup](#)，个可以从 /usr/shared/examples/cvsup 得到。是最被推的方式，因它允一次取整个集合，以后就只取更改的部分。多人从 [cron](#) 行 [cvsup](#)，以保持他的源自更新。要定制上的 supfile 本，并且配置 [cvsup](#) 以的环境。..

使用工具 CTM。如果的接性能不太好(高价接或只能通子件存取)，CTM 是个。但也也有争并且常常得到坏文件。因此很少使用它，也注定了不能期用它来工作。于使用 9600 bps 或更快接的人，我们推使用 CVSup。

3. 本上，如果需要快速存取源并且不通信的，可以使用 `cvsup` 或 `ftp`。否，就使用 CTM。。

在 FreeBSD-STABLE 之前，仔 /usr/src 里的 Makefile。至少安装一个新的内核并重建系，首先做升程的一部分。 FreeBSD-STABLE; 件列表 件列表和 /usr/src/UPDATING，可能在其它循序的程中保持更新，在我向下一行版移是很有必要的。

## 25.6. 同的源

有多方式通互网(或子件)与 FreeBSD 目源特定域或所有域保持更新，主要依于的趣。 我提供的主要服是匿名 CVS、CVSup，和 CTM。



然只更新源中的部分是可能的，唯一被支持的更新程是更新整个、并且重用区(如：在用空行的所有程序，像 /bin 和 /sbin 下的)和内核源。只更新源中的部分，或只有内核，或只有用区 (userland) 通常会出。些包括有、内核崩 (kernel panics)、数据出。

匿名 CVS 和 CVSup 使用下拉(pull)模式来更新源代。在 CVSup 中，用 (或者 cron 脚本) 会用 `cvsup` 程序，后者会同某一个 `cvsupd` 服行交互，以更新的文件。接到的更新是更新刻最新的，并且只会收到那些需要的更新。可以很容易地限制更新的，只更新那些需要的文件。服器端会根据手已有的文件即地生成更新内容。匿名 CVS 相于 CVSup 而言要一些，因它只是 CVS 的一展，可以从程的 CVS 代得到更新。CVSup 相而言，要比匿名 CVS 更有效率，然后后者却更容易使用。

一方法是 CTM。方法并不能将手的代与中央代中的版本行比，也不能下它。在主 CTM 服器上行的脚本会天行多次，次行都能自地所有文件自上次行以来所生的化，如果有文件生了，就会、上一个序列号，并行便于使用子件行送的操作(其中只包括可打印的 ASCII 字符)。一旦接收到，些"CTM deltas"就会被送 `ctm_rmail(1)` 工具---可以自行解、校和用些化到用的制的源里。个程比 CVSup 更有效，而且更少占用我的服器源，因它不采用下拉(pull)模式，采用上推(push)模式。

当然，做也会来一些不便。如果不意除了的包的部分内容，CVSup 会到并重建破坏的部分。CTM 是不会做的，如果除了的源中的某部分(并已不能恢)，那就必从破坏(从最新的 CVS "base delta")始，使用 CTM 或匿名 CVS 行重建，除坏的数据并再同。

## 25.7. 重新 "world"

只要根据一定版本的 FreeBSD (FreeBSD-STABLE、FreeBSD-CURRENT 等等)，已同了本地的源，那就可以使用些源来重建系。



做好

无需在行之前整个系是多的重要。尽管重新系是(如果按照文的指示做的)一件很容易完成的工作，但出也是在所免的，外，人在源里面引入的也可能造成系无法引。

信自己已做，并且在手有恢或可以引的光。可能永也不会用到它，但安全第一嘛！

恰当的组件列表

FreeBSD-STABLE 和 FreeBSD-CURRENT 分支自然是 展示中的。 FreeBSD 做 贡献的都是人，偶尔也会犯错。



有些错误没什么危害，只是引起你的系统生成新的断警告。 有些是永久性的，并导致你的系统不能启动或破坏你的文件系统（甚至更糟）。

如果出了类似的错误，一封“小心(heads up)”帖到相关的组件列表里， 清楚你的本意以及受影响你的系统。在问题解决后，再一封“解除(all clear)”声明。

如果使用 FreeBSD-STABLE 或 FreeBSD-CURRENT 而又不看 [FreeBSD-STABLE; 组件列表](#) 和 [FreeBSD-CURRENT 组件列表](#) 各自的组件列表，那你是自找麻烦。



不要使用 `make world`

许多早期的文档推荐使用 `make world` 来完成这项工作。 这样做会跳过一些必要的步骤， 因此只有在你知道自己在做什么的时候才可以这样做。 几乎所有的情况下 `make world` 都是不应该做的事情， 你应该使用这里描述的方法。

### 25.7.1. 更新系统的方法

在更新系统，一定要首先看看 `/usr/src/UPDATING` 文件， 以便了解在 `buildworld` 之前需要执行的操作， 然后按照下面列出的步骤进行操作：

有些更新假定你使用的是包含旧设备、 内核以及用你的工具及配置的旧版 FreeBSD。 我们使用 “world” 来表示系统中的核心运行文件、 函数库和程序文件。 设备是 “world” 的一部分， 但有其特殊性。

此外， 我们假定你已经得到了新版本操作系统的源代码。 如果你正更新的系统中的源代码也是旧版系统所附带的， 你需要参考 [同源的源](#) 来把代码同步到新的版本。

从源代码更新系统， 有可能会比初看上去的时候更麻烦一些， 一方面， FreeBSD 的社区有人可能会不得不修改你的更新， 特别是当出了一些无法避免的依赖关系的时候。 另一方面， 将介绍目前推荐的更新背后的原理。

成功的更新操作必须解决下面的一些问题：

- 旧的设备可能无法运行新的内核。（一方面， 旧的设备很可能有 bug。）因此， 新的内核必须以新的设备。 更具体地说， 新的设备在新内核启动之前已经完成了初始化。 注意， 新的设备并不一定需要在新的内核之前 安装 到系统中。
- 新的 world 有可能依赖一些新的内核特性。 因此， 新内核必须在新的 world 之前安装。

这个就是为什么我们将在后面的章节中介紹的， 需要按照 `buildworld`、 `buildkernel`、 `installkernel`、 `installworld` 的顺序来更新系统的原因。 这并不是需要遵守推荐的更新操作的全部原因， 除了这个最重要的理由之外， 还有一些并不那么显而易见的原因：

- 旧的 world 可能无法配合新的内核正常工作， 因此， 在安装完新内核之后， 尽快将 world 也随之更新。
- 有些配置文件的修改必须在安装新的 world 之前完成， 而一些配置文件的修改有可能导致旧 world 工作不正常。 因此， 通常而言会需要多次不同的配置文件更新。

- 多数情况下，更新只会替换或添加文件；换言之，有的旧文件并不会被删除。有，可能会致一些其他。因此，有安装操作会指明，必在某些操作之前手工删除一些文件。些在未来可能会被自动化，也可能不会自动化。

由于有些考虑，因此一般情况下我建议用户使用下列更新。注意，具体的更新操作中可能会需要一些附加的，但核心的过程是不会轻易变化的：

#### 1. `make buildworld`

操作会新的编译器，以及少量相关工具，并在随后使用新的编译器来构建 world。构建的结果会存放在 /usr/obj。

#### 2. `make buildkernel`

与旧式的、使用 `config(8)` 和 `make(1)` 的方法不同，这种做法会使用存放于 /usr/obj 中的新的编译器。这种做法使得免去了由于编译器与内核源代码不一致导致的。

#### 3. `make installkernel`

安装新的内核及其模块，使系统能以更新后的内核。

#### 4. 重启系统并进入单用户模式。

单用户模式使得更新正在进行的文件可能导致的减少到最少。此外，它也使配合新内核运行旧 world 可能出现的减少到最少。

#### 5. `mergemaster -p`

操作会完成安装新的 world 所需的配置文件更新操作。例如，它可能会在系统的密码数据中添加新的用户或用户。些操作通常在上次更新之后加了新的用户或特殊系统用户之后是需要的，因此 `installworld` 操作会需要些用户或才能顺利完成。

#### 6. `make installworld`

从 /usr/obj 中复制 world。操作之后，在上的系统，包括内核和 world 就都是新的了。

#### 7. `mergemaster`

更新余下的配置文件，因为 world 已更新完成了。

#### 8. 重启系统。

操作将加入新的内核，以及新的 world 和更新后的配置文件。

注意，如果正从同一 FreeBSD 版本分支升级，例如，从 7.0 到 7.1，上述过程可能没有那必要，因为不太可能遇到重的编译器、内核源代码、用户程序源代码或配置文件不匹配的情形。旧式的 `make world` 然后再新内核的升级方法，很可能有机会能正常工作而完成升级工作。

但是，在大版本升级的过程中，不按照前面所介绍的操作来进行升级，便很可能遇到一些。

此外，需要注意的是，有些时候升级的过程中（例如从 4.X 到 5.0）可能会需要一些额外的（例如在 `installworld` 之前更名或删除一些文件）。仔细查看 /usr/src/UPDATING 个文件，特别是它的尾部分所介



的推的升操作顺序。

由于人不可能完全避免一些不匹配方面的， 个程一直在演化程中。 不幸的是， 目前推的个升， 能在很一段内不需要做任何整。

一下， 目前推的从源代升 FreeBSD 的方法是：

```
# cd /usr/src
# make buildworld
# make buildkernel
# make installkernel
# shutdown -r now
```



有， 可能需要外地行一次 `mergemaster -p` 才能完成 `buildworld` 。 些要求， 会在 `UPDATING` 中行描述。 一般而言， 可以地跳一， 只要行的不是大跨度的 FreeBSD 版本升。

在 `installkernel` 成功完成之后， 需要引到用模式（例而言， 可以在加器提示后入 `boot -s`）。 接下来行：

```
# adjkerntz -i
# mount -a -t ufs
# mergemaster -p
# cd /usr/src
# make installworld
# mergemaster
# reboot
```



的明

前面所出的， 只是助始工作的要明。 要清楚地理解一， 特是如果打算自行定制内核配置， 就下面的内容。

## 25.7.2. /usr/src/UPDATING

在做其它事之前， 在 `/usr/src/UPDATING` （或在的源里的等效的文件）。 个文件要包含有于可能遇到的的重要信息， 或指定了可能使用到的命令的行顺序。 如果 `UPDATING` 与里到相矛盾， 那就先依据 `UPDATING`。



正如先前所述， `UPDATING` 并不能替代正的件列表。 都是互的， 并不彼此排斥。

## 25.7.3. /etc/make.conf

`/usr/shared/examples/etc/make.conf` 以及 `/etc/make.conf`。 第一个文件包含了一些默的定 - 它中的大多数都注掉了。 了在重新系能使用它， 把些加入到 `/etc/make.conf`。 注意在 `/etc/make.conf` 中的任何置同也会影次行 `make` 的果， 因此置一些合自己系的是一个好。

一般的用␣通常会从 `/usr/shared/examples/etc/make.conf` ␣制 `CFLAGS` 和 `NO_PROFILE` ␣␣的␣置到 `/etc/make.conf` 中并令它␣生效。

␣考␣其他的一些␣␣ (例如 `COPTFLAGS`、`NOPORTDOCS` 等等)，看看是否合用。

#### 25.7.4. 更新 `/etc` 里的文件

`/etc` 目␣包含有除了␣的系␣␣␣␣行的脚本外大部分的系␣配置信息。有些脚本随 FreeBSD 的版本而不同。

有些配置文件在天天␣行的系␣里也是要使用到的。尤其是 `/etc/group`。

偶␣，作␣安装␣程的一部分，`make installworld` 会要求事先␣建某些特定的用␣或␣。在␣行升␣，它␣可能并不存在。␣会␣升␣造成␣␣。有␣，`make buildworld` 会␣␣它␣是否已␣存在。

最近就有个␣␣的例子，当␣新␣了 `smmsp` 用␣。当用␣␣完成安装操作␣，在 `mtree(8)` ␣␣建立 `/var/spool/clientmqueue` ␣失␣了。

解决␣法是通过使用 `-p` ␣␣以␣建前 (pre-buildworld) 模式␣行 `mergemaster(8)`。␣表示只␣比那些␣于成功␣行 `buildworld` 或 `installworld` 起␣␣作用的文件。在第一次␣␣做␣，如果使用的是早期的不支持 `-p` 的 `mergemaster` 版本的␣，使用源␣中的新版本即可。

```
# cd /usr/src/usr.sbin/mergemaster
# ./mergemaster.sh -p
```



如果␣是个偏␣狂 (paranoid)，␣可以␣␣的系␣看看␣个文件属于␣已更名或␣除了的那个␣。

```
# find / -group GID -print
```

将␣示所有 `GID` ␣ (可以是␣名也可以是数字地␣ ID) 所有的文件。

#### 25.7.5. 改␣␣用␣模式

␣可能想在␣用␣模式下␣␣系␣。除了␣更快␣理事情␣然有好␣外，重装系␣将触及␣多重要的系␣文件，包括所有␣准系␣二␣制文件、␣文件、包含 (include) 文件等等。在正␣行的系␣ (尤其是在有活␣的用␣的␣候) 中更改␣些文件是自␣␣。

␣一␣模式是在多用␣模式下␣␣系␣，然后␣␣到␣用␣模式下安装。如果␣喜␣␣␣方式，只需在建立 (build) 完成后才␣行下␣的␣␣。␣推␣␣␣到␣用␣模式下直到␣必␣ `installkernel` 或 `installworld`。

从␣行的系␣里，以超␣用␣方式␣行：

```
# shutdown now
```

␣␣就会␣␣到␣用␣模式。

除此之外，也可以重␣系␣，并在␣菜␣␣␣ "single user" (␣用␣) ␣␣。␣␣系␣将以␣用␣模式␣␣。接着，在 shell 提示符␣行：



```
# fsck -p
# mount -u /
# mount -a -t ufs
# swapon -a
```

会文件系统，重新将 / 以只写模式挂接，参考 `/etc/fstab` 挂接其它所有的 UFS 文件系统，然后交回区。

如果的 CMOS 是置本地，而不是 GMT（如果 `date(1)` 命令出不能示正的区和地区也有其事），可能也需要行下的命令：



```
# adjkerntz -i
```

可以定正的本地区置-不不做，以后可能会到一些。

### 25.7.6. 除 `/usr/obj`

随着重新建系的行，结果会放到（默情况下）`/usr/obj` 下。些目会映射到 `/usr/src`。

通除个目，可以加速 `make buildworld` 的程，并避免相互依系等的。

`/usr/obj` 中的某些文件可能置了不可改（情参 `chflags(1)`），需要首先去掉些志。

```
# cd /usr/obj
# chflags -R noschg *
# rm -rf *
```

### 25.7.7. 重新基本系

#### 25.7.7.1. 保存出

建把行 `make(1)` 后得到的出存成一个文件。如果什地方出了，就会有信息。尽管不能分析里出了，但如果把的到某个件列表里就能助其他的人。

做最的法是使用 `script(1)` 命令，同是上参数指定存放出的文件名。在重建系之前立即做，然后在程完成入 `exit`。

```
# script /var/tmp/mw.out
Script started, output file is /var/tmp/mw.out
# make TARGET
... compile, compile, compile ...
# exit
Script done, ...
```

如果做，就 不要 把文件存到 `/tmp` 里。下次，个目就会被清除掉。存放的最好地方是 `/var/tmp`（如上个例）或 `root` 的主目。

### 25.7.7.2. 基本系统

必须在 `/usr/src` 目录里：

```
# cd /usr/src
```

(当然，除非源的目录是在其它地方，真是目录的更改成那个目录就行了)。

使用 `make(1)` 命令重建系统。这个命令会从 Makefile (描述或 FreeBSD 的程序如何被重建，以什么顺序建立等等) 里取指令。

输入的一般命令格式如下：

```
# make -x -D VARIABLE target
```

这个例子里，`-x` 是会调用 `make(1)` 的一个选项。看 `make(1)` 手册有可用的选项例子。

`-D VARIABLE` 是一个选项 Makefile。这些选项控制了 Makefile 的行。这些同 `/etc/make.conf` 设置的选项一样，只是提供了另一种设置它们的方法。

```
# make -D NO_PROFILE target
```

这是一种指定不被建立 (built) 的先定 (profiled libraries) 的方式，同 `/etc/make.conf` 里的

```
NO_PROFILE=    true    #    避免性能分析
```

一起使用。

目标 (target) 告诉 `make(1)` 做什么。一个 Makefile 定义了一定数量不同的"目标 (targets)"，然后目标就决定了什么会生成。

有些目标列在 Makefile 里的，但并不意味着要运行。相反，建立过程 (build process) 利用它们把重建系统的一些必要的部分分割成几个子目标。

大部分的选项不需要向 `make(1)` 传参数，因此命令看起来可能象：

```
# make target
```

此 `target` 表示的是若干目标。多数情况下，第一个 target 都是 `buildworld`。

正如名字所暗示的，`buildworld` 在 `/usr/obj` 下建立了一个全新的目录，然后使用一个 target，`installworld` 在当前的机器里安装它。

将选项分有四个点。首先，它允许安全地完成建立 (build)，而不正在运行的系统的部件产生影响。建立过程是"自主的 (self hosted)"。因此，可以安全地在以多用户模式运行的机器里运行 `buildworld`，而不用担心不良影响。但是依然推荐在单用户模式运行 `installworld`。

第二，允许使用 NFS 挂载 (NFS mounts) 升网里的多台计算机。如果有三台 A、B 和 C 想行升，在 A 行 `make buildworld` 和 `make installworld`。然后将 A 上的 `/usr/src` 和 `/usr/obj` 通 NFS 挂接到 B 和 C 上，接下来，只需在 B 和 C 上使用 `make installworld` 来安装建的结果就可以了。

尽管 `world` target 仍然存在，烈建不要用它。

行

```
# make buildworld
```

我提供了一个性的功能，可以在建程中 `make` 指定 `-j` 参数，令其在建程中同多个并的程。于多 CPU 的机器而言，做有助于其性能。不，由于程中的瓶主要是在 IO 而不是 CPU 上，因此它也会 CPU 的机器来好。

典型的 CPU 机器，可以使用：

```
# make -j4 buildworld
```

，`make(1)` 会最多同 4 个程。从到件列表中的看，做能来最佳的性能。

如果使用的机器有多 CPU，并且配置了 SMP 的内核，也可以看 6 到 10 的数，并察是否能来建性能上的改善。

### 25.7.7.3. 耗

基本系所需的会受到很多因素的影响，不，新的机器都能在一个小时之内完成 FreeBSD-STABLE 源代的建，而无任何技巧或捷径。完成 FreeBSD-CURRENT 源代的，通常需要更一些的。

## 25.7.8. 和安装新内核

要充分利用的新系，重新内核。是很有必要的，因特定的内存已生了改，像 `ps(1)` 和 `top(1)` 的程序会不能工作，除非内核同源的版本是一的。

最、最安全的方式是 `build` 并安装一个基于 `GENERIC` 的内核。然 `GENERIC` 可能没有合的系的所有必要的，但它包括了到的系到用模式所必需的内容。是个不的的新系是否工作正常的。在从 `GENERIC` 系、核系可以工作后，就可以建立 `(build)` 一个基于的正常内核配置文件的新的内核了。

在 FreeBSD 中，首先完成 `build world` 然后再新内核非常重要。

如果想建立一个定制内核，而且已有了配置文件，只需象使用 `KERNCONF=MYKERNEL`：



```
# cd /usr/src
# make buildkernel KERNCONF=MYKERNEL
# make installkernel KERNCONF=MYKERNEL
```

注意，如果已把 `内核安全(kern.securelevel)` 高到了 1 以上，而且置了 `noschg` 或相似的到了

的内核二进制里，可能会用到用模式里使用 `installkernel` 是很有必要的。如果没有设置它，也能毫无问题地在多用模式运行命令。参考 [init\(8\)](#) 以了解更多关于 [内核安全\(kern.securelevel\)](#) 的信息；看 [chflags\(1\)](#) 了解更多关于不同文件的信息。

### 25.7.9. 重回到用模式

用模式新内核。照[改用模式](#)的说明去做。

### 25.7.10. 安装好的新系

在使用 `installworld` 来安装新的系二进制。

行

```
# cd /usr/src
# make installworld
```

如果在 `make buildworld` 的命令行指定了量，就必须要在 `make installworld` 命令行里指定同的量。关于其它的并不是必需的，如，`-j` 就不能同 `installworld` 一起使用。

例，行了：

```
# make -DNO_PROFILE buildworld
```

就必须使用：

```
# make -DNO_PROFILE installworld
```

来安装果，否则就要安装先定 (profiled) 的在 `make buildworld` 阶段没有建立 (built) 的二进制文件。

### 25.7.11. 不是由 `make installworld` 更新的更新文件

重新整个系不会使用新的或改的配置文件的更新某些目 (尤其像 `/etc`、`/var` 和 `/usr`)

更新些文件最的方式就是使用 [mergemaster\(8\)](#)，手工去做也是可以的，只要注意。不管，一定得 `/etc` 以防出。

#### 25.7.11.1. `mergemaster`

[mergemaster\(8\)](#) 工具是个 Bourne 脚本，用于 `/etc` 和 `/usr/src/etc` 源里的配置文件的不同点。是保持系配置文件同源里的一起更新的推方式。

在提示符里地入 `mergemaster` 就可以开始，并看它的始程。`mergemaster` 会建立一个的根 (root) 境，在 / 下，放置各系配置文件。些文件然后同当前安装到系里的行比。此，不同的文件会以 [diff\(1\)](#) 格式行示，使用 `+` 符号加或修改的行，`-` 将完全除的行或将被替

成新行。看 [diff\(1\)](#) 手册可以得到更多关于 [diff\(1\)](#) 法和文件不同点显示的信息。

[mergemaster\(8\)](#) 会显示两个文件的不同，就可以删除新文件（相文件），是以未改状态安装文件，是以当前安装的文件合并文件，是再看一次 [diff\(1\)](#) 果。

"删除文件"将使 [mergemaster\(8\)](#) 知道我 希望保留我当前的文件不改，并删除新的。并不推个，除非没有更改当前文件的理由。任何候在 [mergemaster\(8\)](#) 提示符里入 `?`，就会得到助。如果跳文件，将在其它文件理完后再次行。

"安装未修改文件"将会使新文件替当前的。大部分未改的文件，是个最好的。

"合并文件"将打一个文本器，里是文件的内容。在就可以一合并它，一在屏幕里看，同从者中取部分生成最文件。当个文件一起比，`l` 会左的内容，`r` 会右的。最的出是由个部分成的一个文件，用它就可以安装了。个通常用于用修改了置的文件。

"再次看 [diff\(1\)](#) 果"将会在提供之前，示文件的不同，就象 [mergemaster\(8\)](#) 所做的一。

在 [mergemaster\(8\)](#) 完成了系文件的理后，会得到其它的。[mergemaster\(8\)](#) 可能会是否要重建密文件，并在最后提示是否要除余下的文件。

#### 25.7.11.2. 手更新

如果想要手工更新，但不要只是从 `/usr/src/etc` 把文件制到 `/etc` 就了事。有些文件是必先"安装"的。是因 `/usr/src/etc` 目并 不是 想像的那是 `/etc` 目的一个制。事上，有些是文件是 `/etc` 有的，而 `/usr/src/etc` 里没有。

如果使用 [mergemaster\(8\)](#) (作推)，可以向前跳到 下一。

手工做最的方式是安装些文件到一个新的目，完成后再来不同。



已有的 `/etc`

然，理上，没有什会自个目，事情是做操当一点。制已有 `/etc` 到一个安全的地方，如：

```
# cp -Rp /etc /etc.old
```

**-R** 完成制 (者注：即可以制目以下的所有内容)，**-p** 保留文件的、所属等等。

需要建立一个虚目 (a dummy set of directories) 来安装新的 `/etc` 和其它文件。`/var/tmp/root` 是个不的，除此之外，有一些子目是需要的。

```
# mkdir /var/tmp/root
# cd /usr/src/etc
# make DESTDIR=/var/tmp/root distrib-dirs distribution
```

就建好了需要的目，然后安装文件。在 `/var/tmp/root` 下建立的大分子目是空的，而且要除掉。最的方式是：

```
# cd /var/tmp/root
# find -d . -type d | xargs rmdir 2>/dev/null
```

它会删除所有的空目录。(标准的信息被重定向到了 /dev/null，以防止由于非空目录的警告。)

/var/tmp/root 现在包含了放在 / 下某个位置的所有文件。现在必须仔细检查一个文件，看看它与已有的文件有多大不同。

注意，有些已经安装在 /var/tmp/root 下的文件有个 "." 在里面。在写的时候，像唯一的文件是 /var/tmp/root/ 和 /var/tmp/root/root/ 里 shell 文件，尽管可能有其它的(依赖于什么时候取它)。相信使用 `ls -a` 可以看到它。

最好的方式是使用 [diff\(1\)](#) 去比较两个文件：

```
# diff /etc/shells /var/tmp/root/etc/shells
```

它会显示出 /etc/shells 文件和新的 /var/tmp/root/etc/shells 文件的不同。用这些来决定是合并已经做的变化还是复制旧的旧文件回来。

使用日期 (Time Stamp) 命名新的 Root(根)目录(/var/tmp/root), 这样可以轻松地比较各个版本的不同

频繁重建系统意味着必须频繁更新 /etc, 而可能有点麻烦。

在合并到 /etc 的文件里, 最新更改的目录可以做个复制, 由此加快目录(指更新)过程。下面就想出了一个可行的主意。



1. 像平常一样建立系统 (Make the world)。当想更新 /etc 和其它目录里, 目录目录一个含有当前日期的名字。假如是 1998 年 2 月 14 日做的, 可以运行下的:

```
# mkdir /var/tmp/root-19980214
# cd /usr/src/etc
# make DESTDIR=/var/tmp/root-19980214 \
  distrib-dirs distribution
```

2. 如上列出的, 从目录合并到化。

在完成之后, 不要删除 /var/tmp/root-19980214 目录。

3. 在目录下最新版的源并改后, 运行第一。将得到一个新的目录, 可能叫做 /var/tmp/root-19980221 (如果等了一周做的升级)。
4. 现在能看到目录的不同了---在隔周的目录里使用 `diff(1)` 建立 diff 生成的不同:

```
# cd /var/tmp
# diff -r root-19980214 root-19980221
```

一般情况下, 目录的不同比 /var/tmp/root-19980221/etc 和 /etc 之间的不同要小很多。因不同点更小, 也就更容易把一些变化移到 /etc 目录里。

5. 现在可以删除早先的目录 /var/tmp/root-\* 目录:

```
# rm -rf /var/tmp/root-19980214
```

6. 下次需要合并一些变化到 /etc 里, 就重复这个流程。

可以使用 `date(1)` 自动生成目录的名称:

```
# mkdir /var/tmp/root-`date "+%Y%m%d"`
```

## 25.7.12. 重启

现在完成了。在系统所有内容都放置正后, 可以重启系统了。只是系统的 `shutdown(8)` 可以这样做:

```
# shutdown -r now
```

### 25.7.13. 结束

恭喜！你在成功升级了 FreeBSD 系。

如果你有微小的，可以易地重建系的特定部分。例如，在部分升级或合并 `/etc` 时，不小心除了 `/etc/magic`, [file\(1\)](#) 命令就会停止工作。这种情况下，行下修：

```
# cd /usr/src/usr.bin/file
# make all install
```

### 25.7.14. 更新

#### 25.7.14.1. 个性化都要重建系？

个不好，因要看化的情况。如，如果行了 CVSup，并得到下更新的文件：

```
src/games/cribbage/instr.c
src/games/sail/pl_main.c
src/release/sysinstall/config.c
src/release/sysinstall/media.c
src/shared/mk/bsd.port.mk
```

就不必重建整个系。只需到相的子目里行 `make all install`，此而已。但是，如果有重大化，如 `src/lib/libc/stdlib`，那就要重建系或至少静接的那些部分(除了加的部分都是静接的)。

在天，就是的事了。要是个星期重建一下系的，可能会高。或者可能只想重做改的部分，信能出所有依系。

当然，所有些依于想升的率，和是否想跟踪 FreeBSD-STABLE 或 FreeBSD-CURRENT。

#### 25.7.14.2. 我的失，并伴随有多 11 信号 11 (或其它的数字信息) 号。是回事呀？

个通常表示硬件。(重)建系是个系硬件的有效方式，并且常常生内存。些正好表示它自己做器奇地死于收到的奇怪信息。

一个信的指示器是如果重新始 `make`，并且整个程中会死在不同的点上。

于情况，没有什可做的，除了更机器里的部件，看是一个坏了。

#### 25.7.14.3. 我完成后可以除 `/usr/obj` ？

短地，可以。

`/usr/obj` 包含了所有在段生成的目文件。通常，在 `make buildworld` 程中第一之一就是除个目重新始。情况下，在完成，保留 `/usr/obj` 没有多大意，可放一大堆磁空(目前是 2 GB 左右)。



不，如果很了解整个工程，也可以 `make buildworld` 跳——。会后——程行得更快，因大部分的源都不必再行了。做的面效果是它可能会触一些由于敏感的依——系——致的，些会致——以奇怪的方式出并失。在 FreeBSD 件列表里常引起沸，当有人抱怨他 build 失，并没意识到是因——自己是想抄近路。

#### 25.7.14.4. 中断的 `build` 可以被恢——？

依——于——在——到——之前整个工程行了多。

一般而言 (当然——并不是硬性——定)，`make buildworld` 的——程中将会首先——建新版的基本——建工具 (例如 `gcc(1)`，以及 `make(1)`) 和系——。随后会安装些工具和——。些新版本的工具和——在随后将被用于重新——和——接它——本身。整个系—— (——在包括了常——的用——程序，例如 `ls(1)` 或 `grep(1)`) 会同新版的系——文件一起被重新——建。

如果——正——于最后一个——段，并且了解它 (因——已——看——了所保存的——出) ——可以 (相当安全地) 做：

```
... ——修—— ...  
# cd /usr/src  
# make -DNO_CLEAN all
```

——就不会取消先前的 `make buildworld` 所做的工作了。

在“`make buildworld`”的——出中如果看到如下信息：

```
-----  
Building everything..  
-----
```

出——在 `make buildworld` 的——出中，——做——不会有什——。

如果没有看到——的信息，或者——不——定，——从——始——建将是万无一失的做法。

#### 25.7.14.5. 我——加快建立系——的速度？

- 以——用——模式——行
- 把 `/usr/src` 和 `/usr/obj` 目——放到不同磁——里的独立文件系——里。如果可能，些磁——在不同的磁——控制器里。
- 更好的，是把些文件系——放置到多个使用 `ccd(4)` (——接磁——器——concatenated disk driver)——的磁——里。
- ——掉 profiling (在 `/etc/make.conf` 里——置 “`NO_PROFILE=true`”)。——差不多用不了它。
- 在 `/etc/make.conf` 里也—— `CFLAGS` ——置上 `-O -pipe`。最佳——化 `-O2` 会更慢，而且 `-O` 和 `-O2` 之——的——化差——基本上可以忽略。`-pipe` ——器使用管道而不用——文件——行通信，——可以——少磁——存取 (以内存作——代价)。
- —— `-jn` —— `make(1)` 以便并——行多个——程。——就不会考——的是否是——个或多个——理器机器。
- 存放 `/usr/src` 的文件系——可以使用 `noatime` ——来挂接 (或重新挂接)。——会防止文件系————文件的存取——。——可能并不需要——些信息。

```
# mount -u -o noatime /usr/src
```



个例子里假定 `/usr/src` 是在它自己的文件系统里。如果不是 (例如假如它是 `/usr` 的部分), 那它就需要那个文件系统挂载点, 而不是 `/usr/src`。

- 存放 `/usr/obj` 的文件系统可以使用 `async` 被挂载 (或重新挂载)。它做将数据写入时, 对应用程序而言写会立即完成, 而数据延迟几秒才会写到磁盘。它能成批地写下数据, 从而大大地改善性能。



注意, 这个选项会使你的文件系统变得脆弱。使用这个选项会提高在电源断掉或机器非正常重启, 文件系统陷入不可恢复状态的概率。

如果在某个文件系统里 `/usr/obj` 是很小的, 它不是问题。如果它有其它有价值的文件, 那你在使用这个选项前, 备份一下。

```
# mount -u -o async /usr/obj
```



同上, 如果 `/usr/obj` 不在自己的文件系统里, 使用相应挂载点的名字把它从例子里替换掉。

### 25.7.15. 如果出了问题我该怎么办？

相信你的环境没有先前 `build` 留下的残余。点这里。

```
# chflags -R noschg /usr/obj/usr
# rm -rf /usr/obj/usr
# cd /usr/src
# make cleandir
# make cleandir
```

不, `make cleandir` 真的要执行多次。

然后重新开始整个工程, 使用 `make buildworld` 开始。

如果你有, 就把它和 `uname -a` 的输出送到 [FreeBSD 一般邮件列表](#) 邮件列表。准回答其它关于它的配置的问题！

## 25.8. 删除旧的文件、目录和函数

在 FreeBSD 的更新过程中, 随时可能会出一些文件或其内容的问题。这种情况有可能是由于其功能在其它地方变了, 函数的版本号增加, 或完全从基本系统中去, 等等。一般的更新工程并不会去删掉些旧的文件、函数或目录, 在更新系统之后, 它们得以清理。清理的好是些文件不会再占用存空间 (以及空间) 空, 此外, 如果旧的函数或文件中存在安全或可靠性问题, 它们也更新到新的函数, 以避免安全隐患或崩溃情形的产生。旧的文件、目录和函数会列在 `/usr/src/ObsoleteFiles.inc` 中。接下来将介绍在系统更新过程中如何删掉些旧的文件。

我假定你已经按照 [更新系统的正确方法](#) 介绍的完成了更新操作。在 `make installworld` 和 `mergemaster`

命令完成之后，使用下面的命令系统上是否存在旧的文件或目录：

```
# cd /usr/src
# make check-old
```

如果有旧的文件，可以用下面的命令来删除：

```
# make delete-old
```



参看 `/usr/src/Makefile` 可以了解其他 target 的功用。

在删除文件时，系统会询问每个文件都给出提示。可以跳过这些提示，并系统自动完成删除操作，方法是使用 `make BATCH_DELETE_OLD_FILES`，具体做法如下：

```
# make -DBATCH_DELETE_OLD_FILES delete-old
```

也可以用 `yes` 命令和管道来达到类似的目的：

```
# yes|make delete-old
```



删除旧的文件，有可能会破坏有的依赖于这些文件的旧应用程序。对于旧的函数库来说，删除的可能性更大。大多数情况下，可以重新安装使用旧库的所有程序、port 或函数库之后再运行 `make delete-old-libs`。

在 Ports Collection 中提供了一些可以检测依赖关系的工具，例如 [sysutils/libchk](#) 和 [sysutils/bsdadminscripts](#)。

旧的依赖库可能会与新库冲突，导致类似的警告消息：

```
/usr/bin/ld: warning: libz.so.4, needed by /usr/local/lib/libtiff.so, may conflict
with libz.so.5
/usr/bin/ld: warning: librpcsvc.so.4, needed by /usr/local/lib/libXext.so, may
conflict with librpcsvc.so.5
```

要解决这样的问题，需要安装旧库的 port：

```
# pkg_info -W /usr/local/lib/libtiff.so
/usr/local/lib/libtiff.so was installed by package tiff-3.9.4
# pkg_info -W /usr/local/lib/libXext.so
/usr/local/lib/libXext.so was installed by package libXext-1.1.1,1
```

接着卸载、重新安装并安装 port。可以使用 [ports-mgmt/portmaster](#) 或 [ports-mgmt/portupgrade](#) 工具来自行完成这些操作。在所有的 port 都重新安装，并且不再使用旧库以后，就可以用下面的命令来

除它了：

```
# make delete-old-libs
```

## 25.9. 跟踪多台机器

如果有多台机器想跟踪同源的源，那它都下源并重建所有西，看起来有点浪费源：磁盘空、网络以及 CPU 周期。解决的办法是一台机器处理大部分的工作，而其它的机器通过 NFS 挂载（mount）些工作。部分列了一做的方法。

### 25.9.1. 准

首先，定一批机器，行的二进制代码是同一套——我称作“建集群”（*build set*）。每台机器可以使用不同的定制内核，但它行的是相同的用户区二进制文件（*userland binaries*）。从批机器中一台机器做“建机器”（*build machine*）。将用于建（*build*）系和内核的机器。想像一下，它是一台快速的机器，有足够的空余的 CPU 来行 `make buildworld`。也想要一台机器做“机器”（*test machine*），个将用于件的更新生成品之前他行。个必是一台能提供的平也可使用的机器。它可以是“建机器”，但没个必要。

在个“建集群”里的所有机器需要从同一台机器、同一个点上挂载 `/usr/obj` 和 `/usr/src`。理想地，它在“建机器”上的个不同的器里，但是在那台机器上可以行 NFS 挂载。如果有多多个“建集群”，`/usr/src`在某个“建机器”上，而在其它机器上行 NFS 挂载。

最后，个“建集群”里所有机器上的 `/etc/make.conf` 和 `/etc/src.conf` 与“建机器”里的相同。意味着“建机器”必建部分基本系用于“建集群”里所有机器的安装。同，台“建机器”要有它自己的内核名字，使用 `/etc/make.conf` 里的 `KERNCONF` 行置，并且台“建机器”把它列在 `KERNCONF` 里，同把自己的内核列在最前。“建机器”的 `/usr/src/sys/arch/conf` 里一定要有台机器的内核配置文件，如果它想建它的内核的。

### 25.9.2. 基本系

既然所有的妥当了，就准建所有的西。如基本系中描述的一在“建机器”上建内核和系，但是什也不安装。在建束后，到“机器”上，安装建的内核。如果台机器通 NFS 挂载了 `/usr/src` 和 `/usr/obj`，在重到用模式里，需要网然后挂接他。最的方式是多用模式下，然后行 `shutdown now` 到用模式。一旦入，就可以安装新的内核和系，并行 `mergemaster`，就像平常一。完成后，重返回到一般多用模式操作台机器。

在信所有在“机”里都工作正常后，就使用相同的程在“建集群”里的其它机器里安装新的件。

### 25.9.3. Ports

似的想法是使用 ports。第一个的是从同一台计算机上挂载 `/usr/ports` 到“建集群”里的全部计算机。然后正置 `/etc/make.conf` 共享 `distfiles`。把 `DISTDIR` 置到一个共享的目里，那里可以被任何一个 `root` 用写入，并且是由的 NFS 挂载映射的。置一台机器的 `WRKDIRPREFIX` 到一个本地建（*build*）目。最后，如果要建和布包（*packages*），那置 `PACKAGES` 到一个似于 `DISTDIR` 的目。

# Chapter 26. DTrace

## 26.1. 概述

DTrace, 也称跟踪, 是由 Sun™ 的一个用来在生和性生系上出系瓶的工具。在任何情况下它都不是一个工具, 而是一个系分析出性能及其他工具。

DTrace 是个特好的分析工具, 有大量的助断系特性。 可以使用先写好的脚本利用它的功能。用也可以通使用 DTrace D 言建他自己定制的分析工具, 以足特定的需求。

在了一章之后, 将了解:

- DTrace 是什, 它提供了些些特性。
- DTrace 在 Solaris™ 与 FreeBSD 上的的差。
- 如何在 FreeBSD 上和使使用 DTrace。

在了一章之前, 了解:

- 了解 UNIX® 和 FreeBSD 的基本知 (UNIX 基)。
- 熟悉基本的内核配置/ (配置FreeBSD的内核)。
- 熟悉 FreeBSD 有的安全知 (安全)。
- 了解如何取和重新 FreeBSD 源代 (更新与升 FreeBSD)。



特性目前仍被是性的。 有些功能性缺失, 有一些可能无法行。最, 个特性会合用于生, 届篇文也会做些当的修改。

## 26.2. 上的差

然 FreeBSD 上的 DTrace 与 Solaris™ 上的非常相似, 在深入之前我需要明一下存在的差。 用首先会注意到的便是 FreeBSD 上的 DTrace 需要明地被用。 DTrace 相的内核和模必后才能正常工作。 后我会作介。

有一个 `DDB_CTF` 内核用来从内核与内核模加 CTF 数据。CTF 是 Solaris™ Compact C Type Format 封装了似于 DWARF 和 venerable stabs 化的信息。CTF 数据是由 `ctfconvert` 和 `ctfmerge` 工具加入二制文件的。`ctfconvert` 工具分析由器生成的 DWARFELF 的 section, `ctfmerge` 合并目的 CTFELF section 到可行文件或共享。更多于在用 FreeBSD 内核上用此的内容即将完成。

比起 Solaris™, FreeBSD 有几个不同提供器。最得注意的是 `dtmalloc` 提供器, 可以根据型追踪 FreeBSD 内核中的 `malloc()`。

只有 `root` 可以使用 FreeBSD 上的 DTrace。 是由系安全上的差造成的, Solaris™ 提供了一些 FreeBSD 上未的低的安全。同, `/dev/dtrace/dtrace` 也被格的限制供 `root` 用。

最后, DTrace 的 Sun™ CDDL 可下布的件。随 FreeBSD 行的 `Common Development and Distribution License` 可以在 `/usr/src/cddl/contrib/opensolaris/OPENSOLARIS.LICENSE` 或者通 <http://www.opensolaris.org/os/licensing> 看版本。

它可表示有 DTrace 的 FreeBSD 内核仍 BSD 可；然而，以二制布模，或者加二制模需遵守 CDDL。

## 26.3. 用 DTrace 支持

在内核配置文件中加入以下几行来 DTrace 的支持：

```
options      KDTRACE_HOOKS
options      DDB_CTF
```



使用 AMD64 架的需要在内核配置文件中加入如下行：

```
options      KDTRACE_FRAME
```

此提供了 FBT 特性的支持。DTrace 可以在没有此的情况下正常工作，但是函数界跟踪便会有所限制。

所有的源代都必须重新使用 CTF 安装。重新 FreeBSD 源代可以通过以下的命令完成：

```
# cd /usr/src
# make WITH_CTF=1 kernel
```

系需要重新。

在重新和新内核入内存之后，需要添加 Korn shell 的支持。因 DTrace 工具包有一些工具是由 ksh 写的。安装 shells/ksh93。同也可以通 shells/pdksh 或者 shells/mksh 使用些工具。

最后是得最新的 DTrace 工具包。当前版本可以通过下面的链接到 <http://www.opensolaris.org/os/community/dtrace/dtracetoolkit/>。个工具包含有一个安装机制，尽管如此，并不需要安装便可使用它。

## 26.4. 使用 DTrace

在使用 DTrace 的功能之前，DTrace 必须存在。使用如下的命令装此：

```
# kldload dtraceall
```

DTrace 支持在可以使用了。管理在可以使用如下的命令看所有的探器：

```
# dtrace -l | more
```

所有的出都 more 工具，因它会很快超出屏幕的示区域。此，DTrace 被是能正常工作的了。

现在是考察工具包的时候了。

工具包是一堆写好的脚本，与 DTrace 一起运行来收集系统信息。有脚本用来打印已打开的文件，内存，CPU 使用率和很多东西。使用如下的命令解包脚本：

```
# gunzip -c DTraceToolkit* | tar xvf -
```

使用 `cd` 命令切换到那个目录，并修改所有文件的可执行权限，把那些名字小写的文件权限改为 `755`。

有些脚本都需要修改它的内容。那些指向 `/usr/bin/ksh` 需要修改成 `/usr/local/bin/ksh`，另外使用 `/usr/bin/sh` 需要更改 `/bin/sh`，最后有使用 `/usr/bin/perl` 的需要更改 `/usr/local/bin/perl`。



此刻需提醒一下读者 FreeBSD 的 DTrace 支持仍是不完整的和实验性的。有些脚本中的大多数都无法运行，因为它基于 Solaris™ 或者使用了目前不支持的探测器。

在撰写这篇文章的时候，DTrace 工具包中只有两个脚本在 FreeBSD 上是完全支持的：`hotkernel` 和 `procsystime` 脚本。这两个脚本便是我下一部分将要探索的：

`hotkernel` 被改成两个函数占用了内核空间。正常运行的，它将生成类似以下的输出：

```
# ./hotkernel
Sampling... Hit Ctrl-C to end.
```

系统管理员必须使用 `Ctrl + C` 组合键停止这个进程。接着中止之后，脚本便会打印内核函数与定义的列表，使用数量排序输出：

|                               |       |       |
|-------------------------------|-------|-------|
| 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\_isitmxx               | 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% |

□个脚本也能与内核模□一起工作。要使用此特性， 用 **-m** □志□行脚本：

```
# ./hotkernel -m
Sampling... Hit Ctrl-C to end.
^C
MODULE                                COUNT    PCNT
0xc107882e                            1        0.0%
0xc10e6aa4                            1        0.0%
0xc1076983                            1        0.0%
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%
```

procsystime 脚本捕捉并打印□定 PID 的系□用□□。 在下面的例子中， 新生成了一个 /bin/csh □ 例。procsystime □行后□等待在新□行的 **cs**h 上□入一些命令。 □是□□的□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 |

正如显示的那，`read` 系统调用似乎使用了最多的秒位，`getpid()` 系统调用使用了最少的。

## 26.5. D 语言

DTrace 工具包括了很多由 DTrace 特殊语言写成的脚本。在 Sun™ 的文档中称该语言为 "D 语言"，它与 C++ 非常相似。此语言更深入的描述超出了这篇文章的范围。更多相关的可以在

<http://wikis.sun.com/display/DTrace/Documentation> 找到。

# Part IV: 网 通

FreeBSD 是目前以高性能网服为目的而部署最广泛的操作系之一。 些的章包括：

- 串口通
- PPP 和以太网上的 PPP
- 子件
- 行网服
- 防火
- 其他网

些章主要供在需要参考。 不必按特定的序来它， 此外， 始在网中使用 FreeBSD 之前也不需要先把它都完。

# Chapter 27. 串口通信

## 27.1. 概述

UNIX® 一直都是支持串口通信的。事实上，早期的 UNIX® 系统就是利用串口来输入和输出数据的。那时常见的“终端”包括一个每秒10个字符的串口打印机和调制解调器，现在有些已发生了很大的变化。本章将介绍一些利用 FreeBSD 进行串口通信的方法。

读完本章，你将了解到：

- 如何通信终端连接到FreeBSD系统。
- 如何使用modem号码到远程主机。
- 如何允许程序通过modem登录到系统。
- 如何从串口控制台引导系统。

本章之前，应当了解：

- 如何配置和安装一个新的内核 ([配置FreeBSD的内核](#))。
- 理解 UNIX® 的限制和编程 ([UNIX 基础](#))。
- 准备打算在 FreeBSD 中使用的串口设备 (modem 或多串口)的技术参考手册。

## 27.2. 介绍



从 FreeBSD 8.0 开始，用于串口的设备点从 `/dev/cuaN` 改为了 `/dev/cuauN`；从 `/dev/ttydN` 改为了 `/dev/ttyuN`。FreeBSD 7.X 用户需要根据具体情况参考文档中的例子进行必要的调整。

### 27.2.1. 术语

#### bps

每秒位- 数据的传输速度

#### DTE

数据终端设备 - 如你的计算机

#### DCE

数据通信设备 - 如你的modem

#### RS-232

用于硬件串口通信的EIA标准

当通信数据速度的时候，你不会使用“baud”。Baud指电气标准率，它已使用了很长时间，而“bps” (bits per second) 才是正在使用的（至少它不会打那些竞争的家口）。

27.2.2. 串口和端口

要将 modem 或端口与您的 FreeBSD 系统相连，您的计算机需要一个串口，以及用于连接串口所需的电缆。如果您不熟悉硬件及所需的电缆，您可以跳过去。

27.2.2.1. 串口

串口有许多不同的类型。最常使用的类型是 null-modem 电缆和标准 ("直连") RS-232 电缆。您的硬件说明中会介绍使用的电缆。

27.2.2.1.1. Null-modem 电缆

null-modem 电缆会直接传送某些信号，如 "Signal Ground" (信号地)，但其他信号交叉。例如，"Transmitted Data" (数据发送) 引脚是到另一端 "Received Data" (数据接收) 引脚的。

也可以自行制作 null-modem 电缆端使用 (例如，为了产品的要求)。下面的表格展示了 RS-232C 信号，以及 DB-25 连接器上的引脚。注意，标准也要求一根直通引脚 1 到引脚 1 的保护地 (Protective Ground)，但通常都被省掉。某些端口在只有引脚 2、3 和 7 的时候，就已能正常使用了，而其他一些，则需要下面例子中所展示的不同配置。

表 9. DB-25 to DB-25 Null-Modem Cable

| 信号  | 引脚 # |     | 引脚 # | 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 |

这里还有目前比其他流行的其他连接方式。

表 10. DB-9 到 DB-9 Null-Modem 电缆

| 信号  | 引脚 # |    | 引脚 # | 信号  |
|-----|------|----|------|-----|
| RD  | 2    | 接到 | 3    | TD  |
| TD  | 3    | 接到 | 2    | RD  |
| DTR | 4    | 接到 | 6    | DSR |
| DTR | 4    | 接到 | 1    | DCD |
| SG  | 5    | 接到 | 5    | SG  |
| DSR | 6    | 接到 | 4    | DTR |

| 信号  | 引脚 # |    | 引脚 # | 信号  |
|-----|------|----|------|-----|
| DCD | 1    | 接到 | 4    | DTR |
| RTS | 7    | 接到 | 8    | CTS |
| CTS | 8    | 接到 | 7    | RTS |

表 11. DB-9 到 DB-25 Null-Modem 连接

| 信号  | 引脚 # |                            | 引脚 # | 信号  |
|-----|------|----------------------------|------|-----|
| RD  | 2    | DB-9 到 DB-25 Null-Modem 连接 | 2    | TD  |
| TD  | 3    | 接到                         | 3    | RD  |
| DTR | 4    | 接到                         | 6    | DSR |
| DTR | 4    | 接到                         | 8    | DCD |
| SG  | 5    | 接到                         | 7    | SG  |
| DSR | 6    | 接到                         | 20   | DTR |
| DCD | 1    | 接到                         | 20   | DTR |
| RTS | 7    | 接到                         | 5    | CTS |
| CTS | 8    | 接到                         | 4    | RTS |



当某一段连接器上的一个引脚需要连接到另一端的一个引脚时，通常是将那一引脚使用一短跳线，而使用另一端的那个引脚。

上面的连接似乎更流行。在其他连接中 (在 *RS-232 Made Easy* 手册中进行了介绍) 是 SG 接 SG，TD 接 RD、RTS 和 CTS 接 DCD、DTR 接 DSR，反之亦然。

#### 27.2.2.1.2. 标准 RS-232C 连接

标准的串口卡会直接发送所有 RS-232C 信号。也就是，一块的 "Transmitted Data" 引脚，会直接接到另一块的 "Transmitted Data" 引脚。这包括将调制解调器接到 FreeBSD 系统上的那根线，同时也用于某些型号的终端。

#### 27.2.2.2. 端口

串口是 FreeBSD 主机与终端数据通信的接口。本章描述了端口的配置和它在 FreeBSD 上是如何寻址的。

##### 27.2.2.2.1. 端口的类型

有好几种串口。在采购或制作之前，要确保它能符合你的终端以及 FreeBSD 系统。

大多数终端都提供 DB-25 端口。个人计算机，也包括运行 FreeBSD 的 PC 机，通常会有 DB-25 或 DB-9 口。如果你的 PC 上有多串口卡，可能有 RJ-12 或 RJ-45 口。

参考硬件的文档以了解所用接口的规格。此外，你也可以通过观察外线来了解所用的端口。

#### 27.2.2.2. 端口名称Port Names

在FreeBSD中，你可以通 `/dev` 目中的一个来访问个串口。有不同的：

- 呼入端口的名字是 `/dev/ttyuN`，其中  $N$  是端口的号，从零开始数。一般来，使用呼入端口作端。呼入端口要求数据使用波 (DCD) 信号来工作。
- 呼出端口的名字是 `/dev/cuaN`。通常并不使用呼出端口作端，而只用于制解器。如果串口或端不支持波信号，可能必使用呼出端口。

如果你已连接一个端到第一个串口（在 MS-DOS® 上是COM1），可以使用 `/dev/ttyu0` 来作端。如果它是在第二个串口 (COM2)，那就是 `/dev/ttyu1`，等等。

#### 27.2.3. 内核配置

FreeBSD默认支持4个串口。在MS-DOS®下，些是 COM1, COM2, COM3, 和 COM4。FreeBSD 目前支持 "dumb" 多串口，如 BocaBoard 1008 和 2016, 以及多 Digiboard 和 Stallion Technologies 制造的智能多接口。不，默认的内核只会标准的COM端口。

要看看的内核是否支持的串口，只要在内核看看一下信息，或使用 `/sbin/dmesg` 命令重新内核信息。特的，以 `sio` 字符的信息。



如果想只察看包含 `sio` 一的消息，可以使用下面的命令：

```
# /sbin/dmesg | grep 'sio'
```

例如，在一个有4个串口的系上，些是串口特定的内核信息：

```
sio0 at 0x3f8-0x3ff irq 4 on isa
sio0: type 16550A
sio1 at 0x2f8-0x2ff irq 3 on isa
sio1: type 16550A
sio2 at 0x3e8-0x3ef irq 5 on isa
sio2: type 16550A
sio3 at 0x2e8-0x2ef irq 9 on isa
sio3: type 16550A
```

如果内核未能出所有的串口，可能需要通修改 `/boot/device.hints` 文件来行一些配置。此外，也可以注或完全除掉没有的。

参 [sio\(4\)](#) 机手册来了解于串口，以及多口配置的。如果正使用一个在不同版本的 FreeBSD 上的文件必小心，因参数和法生了化。



里端口 `IO_COM1` 代替了 `0x3f8`，端口 `IO_COM2` 代替了 `0x2f8`，端口 `IO_COM3` 代替了 `0x3e8`，端口 `IO_COM4` 代替了 `0x2e8`，些都是各自端口相的端口地址。中断4, 3, 5, 9都是常用的中断。也要注意有些正常的串口可能无法在一些ISA的PC上共享中断（多口板有板的子，允在板上所有 16550A 的共享一个或个中断求）。

## 27.2.4. 特殊文件

在内核中，大多数都是通“特殊文件”来访问的，这些文件一般位于 `/dev` 目录中。sio 是通 `/dev/ttyuN` (呼入) 和 `/dev/cuauN` (呼出) 来访问的。此外，FreeBSD 也提供了初始化 ( `/dev/ttyuN.init` 和 `/dev/cuauN.init` ) 以及锁 ( `/dev/ttyuN.lock` 和 `/dev/cuauN.lock` )。初始化用于在打开端口时初始化其通参数，例如使用 `RTS/CTS` 信号行流控制的调制解调器的 `crtcts`。锁用于在端口上提供一个标志，防止用户或程序改变特定的参数；参 [termios\(4\)](#)、[sio\(4\)](#)，以及 [stty\(1\)](#) 的手册，以了解关于端口配置、锁和初始化，以及配置端口参数的信息。

## 27.2.5. 串口配置

`ttyuN` (或 `cuauN`) 是用户将要打开的程序的通用名称。当程序打开某个端口，它将有一个端口 I/O 设置的默认配置。可以在命令行看看这些设置：

```
# stty -a -f /dev/ttyu1
```

当修改了一个端口的设置，这个设置会生效，除非它被锁。当它被重新打开，它将回到默认设置。要修改默认设置，可以打开和调整“初始状态”的设置。例如，要让 `ttyu5` 打开 `CLOCAL` 模式，8位通和默认的 `XON/XOFF` 流控制，输入：

```
# stty -f /dev/ttyu5.init clocal cs8 ixon ixoff
```

串口的系统初始化，是由 `/etc/rc.d/serial` 来控制的。这个文件会影响串口的默认设置。

为了防止用户程序修改某些设置，修改“lock state”(状态) 锁。例如，要把 `ttyu5` 的速率固定为 57600 bps，输入：

```
# stty -f /dev/ttyu5.lock 57600
```

在，一个打开 `ttyu5` 和无法改变端口速度的程序将被固定在 57600bit/s。很自然地，需要固定初始状态，然后用 root 固定状态的写入功能。

很自然，只有 `root` 用户可以初始化或固定状态。

## 27.3. 终端



从 FreeBSD 8.0 开始，用于串口的端点从 `/dev/cuadN` 改为了 `/dev/cuauN`；从 `/dev/ttydN` 改为了 `/dev/ttyuN`。FreeBSD 7.X 用户需要根据具体情况参考文档中的例子进行必要的调整。

当在计算机控制台或是在一个接入的网络上，终端提供了一个方便和低成本的方法。FreeBSD 系统的方法。描述了如何在 FreeBSD 上使用终端。

### 27.3.1. 端的用法和型

早期的 UNIX® 系没有控制台。人通将端接到计算机的串口来登和使用程序。它很像用 modem 和一些端件来号入一个程的系，只能行文本的工作。

今天的 PC 已可以使用高量的形了，但与今天的其他UNIX®操作系一，建立一个登会的能力仍然存在。通使用一个端接到一个没有使用的串口，就能登和行任何文本程序或在 X 系中行一个 `xterm` 口程序。

于商用，可以把任何端接到 FreeBSD 系，然后把它放在工的面上。于家庭用，可以使用一台比老的 IBM PC 或 Macintosh 行一个端接到一台行 FreeBSD 的高性能机器上。

于FreeBSD，有三端：

- 端
- 充当端的PC
- X 端

下面一小将描述一。

#### 27.3.1.1. 端

端需要的好几硬件，通串口接到计算机。它被叫做 "0" 是因它只能用来示，送和接收文本。不能在它上面行任何程序。

有好几百端，包括Digital Equipment Corporation 的VT-100和Wyse的WY-75。只有几可以在FreeBSD上工作。一些高端的端可以示形，但只有某些件包可以使用些高特性。

端被广泛用于那些不需要形用的工作中。

#### 27.3.1.2. 充当端的PC

假如 端 的功能限于示、送和接收文本的，那然任何一台置的个人计算机，都完全能任端的工作。因此需要的是合的，以及一些在台计算机上行 端真 件。

配置在家庭中用十分广泛。例如，如果的人正忙于在的 FreeBSD 系的控制台上工作，就可以将一台功能弱的计算机挂在个 FreeBSD 系上来同完成一些文本界面的工作。

在 FreeBSD 的基本系中至少有能用于行串口接的工具：[cu\(1\)](#) 和 [tip\(1\)](#)。

如果要从行 FreeBSD 的计算机上通串口接到一系，可以使用：

```
# cu -l 串口
```

此 "串口" 表示计算机上某个串口的名。/dev/cuaN。

此的 "N" 表示串口的号。





注意在 FreeBSD 中 COM 的编号是从零而非一开始的（这一点与一些系统，如基于 MS-DOS® 的系统不同）。因此，在基于 MS-DOS® 系统中的 COM1 在 FreeBSD 中通常叫做 `/dev/cuau0`。



其他一些人可能喜欢使用一些来自 Ports 套件的程序。Ports 中提供了几个与 `cu(1)` 和 `tip(1)` 类似的工具，例如 `comms/minicom`。

### 27.3.1.3. X 端

X 端是最简单的端系统。它通常需要使用以太网来连接。它不能运行任何 X 应用程序。

我们介绍 X 端只是出于兴趣。然而，本章不会涉及 X 端的安装，配置或使用。

## 27.3.2. 配置

本章描述了一个端系统上运行一个登录会话，需要在 FreeBSD 系统上进行的配置。假设已经配置好了内核来支持串口，就可以直接开始连接了。

在 [FreeBSD 引导程序](#) 中曾提到，`init` 程序依赖于系统所有的管理控制和初始化。通过 `init` 来运行的一些任务将先读取 `/etc/ttys` 文件，然后在可用的端系统上运行一个 `getty` 程序。`getty` 程序可用来为一个登录名和 `login` 程序。

然而，要配置 FreeBSD 系统配置端，需要以 `root` 身份运行下面的命令：

1. 如果它不在那里，需要串口在 `/dev` 目录下添加一行到 `/etc/ttys`。
2. 指定 `/usr/libexec/getty` 在端口上运行，然后从 `/etc/gettytab` 文件指定适当的 `getty` 类型。
3. 指定默认端系统类型。
4. 置端口为 "on"。
5. 设定端口是否为 "secure"。
6. 迫使 `init` 重新读取 `/etc/ttys` 文件。

作为可选项，可以通过在 `/etc/gettytab` 中建立一个条目，在第 2 步建立一个定制的 `getty` 类型来使用。本章不会介绍如何做。可以参考 [gettytab\(5\)](#) 和 [getty\(8\)](#) 的手册了解更多信息。

### 27.3.2.1. 添加一个条目到 `/etc/ttys`

`/etc/ttys` 文件列出了 FreeBSD 系统上允许登录的所有端口。例如，第一个虚拟控制台 `ttyv0` 在该文件中有一个条目。可以使用该条目登录到控制台。该文件也包含其他虚拟控制台的条目，串口，和 `tty` 端口。对于一个硬件的端口，只要列出串口的 `/dev` 部分而不需要 `/dev` 部分（例如，`/dev/ttyv0` 可以被列为 `ttyv0`）。

默认的 FreeBSD 安装包括一个支持最初四个串口 `ttyu0` 到 `ttyu3` 的 `/etc/ttys` 文件。如果从那些端口中某一个使用端口，不需要添加一个条目。

假设我们连接两个终端系统：一个 Wyse-50 和一个老的串行 Procomm 终端设备，一个 VT-100 终端的 286 IBM PC。在 `/etc/ttys` 文件中的相应的行是：

```
ttyu1 "/usr/libexec/getty std.38400" wy50 on insecure
ttyu5 "/usr/libexec/getty std.19200" vt100 on insecure
```

- 第一部分指定了终端指定文件的名称，它可以在 `/dev` 中找到。
- 第二部分是在串行行的命令，通常是 `getty(8)`。`getty` 初始化然后打印一行，设置速度，用命名的命令和串行登录程序。`getty` 程序在它的命令行接收一个参数（可选），`getty` 类型。一个 `getty` 类型会在终端行描述一个特征，像波特率和奇偶校验。`getty` 程序从 `/etc/gettytab` 文件取得一些特征。文件 `/etc/gettytab` 包含了多老的和新的终端行。在很多例子中，文本 `std` 的终端将用硬终端来工作。有些忽略了奇偶性。它是一个从 110 到 115200 bit/s 的 `std` 终端。当然，可以添加自己的终端到这个文件。`gettytab` 的机手册提供了更多的信息。当在 `/etc/ttys` 中设置 `getty` 类型的時候，确信在终端上的通设置匹配。在我的例子中，Wyse-50 不使用奇偶性，用 38400 bit/s 来连接。286 PC 不使用奇偶性，用 19200 bit/s 来连接。
- 第三部分是通常连接到那个 tty 行的终端类型。对于号端口，`unknown` 或 `dialup` 通常被用在地方。对于硬终端的终端，终端类型不会改变，所以可以从 `termcap` 数据文件中放一个真正的终端类型。在我的例子中，Wyse-50 使用真正的终端类型，而串行 Procomm 的 286 PC 将被置成在 VT-100 上的模式。
- 如果端口被用，可以指定第四部分。在第二部分，把它放在那儿将行初始化程序来登录程序 `getty`。如果在部分延迟，将没有 `getty`，在端口上因此就没有登录。
- 最后部分被用来指定端口是否安全。一个安全的端口意味着信任它允许用 `root` 从那个端口登录。不安全的端口不允许 `root` 登录。在一个不安全的端口上，用户必须用无特权的登录，然后使用 `su` 或一个相似的机制来得到超用户的权限。

### 27.3.2.2. 重新取得 `/etc/ttys` 来控制 `init`

`/etc/ttys` 文件做一个必要的修改后，必须发送一个 `SIGHUP` 信号给初始化程序来迫使它重新取得配置文件，例如：

```
# kill -HUP 1
```



`init` 是系统运行的第一个进程，因此它是 PID 1。

如果能正确设置，所有的终端都是适当的，终端将可以用了，然后一个 `getty` 进程将在每个终端行，将在终端上看到登录命令行。

### 27.3.3. 的接口可能出现的

即使小心翼翼地注意，仍然可能会在设置终端出现。有一个有问题和解决方法的列表：

### 27.3.3.1. 没有登录命令输出：

终端被嵌入和打印了。如果把一台个人计算机充当一个终端，终端模式软件运行在正端的串口上。

终端被固定地连接到终端和 FreeBSD 计算机上。终端用了正端的。

终端和 FreeBSD 的速度和奇偶配置已一致了。 如果有一个像示终端，终端比度已好了。如果它是一个可打印的终端，终端和墨水已就了。

定一个 **getty** 程序正在运行和服终端。例如， 可以用 **ps** 命令得到运行 **getty** 程序的列表， 输入：

```
# ps -axww|grep getty
```

将看到一个终端的。例如， 下面的表示表明一个 **getty** 正在第二个串口 **ttyu1** 运行， 正在 **/etc/gettytab** 中使用 **std.38400** 的：

```
22189  d1  Is+    0:00.03 /usr/libexec/getty std.38400 ttyu1
```

如果没有 **getty** 程序运行， 终端已在 **/etc/ttys** 中用了端口。 在修改完 **/etc/ttys** 文件后， 得运行 **kill -HUP 1**。

如果 **getty** 程序在运行， 但终端上仍然没有显示出登录提示， 或者然示了缺不允许入， 的终端或可能不支持硬件握手。 将 **/etc/ttys** 中的 **std.38400** 改 **3wire.38400** (注意在改完 **/etc/ttys** 之后要 **kill -HUP 1**)。 **3wire** 和 **std** 似， 但忽略硬件握手。 可能需要在使 **3wire** 少波特率或用件流控制以避免缓冲区溢出。

### 27.3.3.2. 输出一个 " " 而不是一个登录命令行

终端和 FreeBSD 使用相同的 bit/s 速率和奇偶校验。 一下 **getty** 程序信当前使用正端的 **getty** 型。如果没有， 在 **/etc/ttys** 然后运行 **kill -HUP 1**。

### 27.3.3.3. 当输入密码， 字符一个个输出

将终端 (或终端模式软件) 从 "半双工" 或 "本地回" 成 "全双工"。

## 27.4. 入服



从 FreeBSD 8.0 始， 用于串口的点从 **/dev/cuaN** 改了 **/dev/cuauN**； 从 **/dev/ttydN** 改了 **/dev/ttyuN**。 FreeBSD 7.X 用需要根据情况文例子行必要的整。

入服配置 FreeBSD 系与接到终端是非常相似的， 除非正在使用 modem 来号而不是终端。

### 27.4.1. 外置vs.内置modem

外置 modem 看起来很容易号。 因， 外置 modem 可以通存在非易失性的 RAM 中的参数来配置， 它通常提供指示器来示重要的 RS-232 信号的状态。 不停光的信号灯能用留下比深刻的印象， 而且指示器也可以用来看 modem 是否正常地工作。

内置modem通常缺乏非易失性的RAM，所以它的配置可能会限制在通过 DIP 开关来设置。如果内置modem有指示灯，它也很可能被看到。

27.4.1.1. Modem和IO

如果你使用一个外置的 modem，那将需要适当的IO。一个标准的串口应当足以普通的信号能直接接上：

表 12. 信号名称

| 缩写  | 全名   |
|-----|--|
| RD  | 收到数据 (Received Data)                             |
| TD  | 发出数据 (Transmitted Data)                          |
| DTR | 数据终端就绪 (Data Terminal Ready)                     |
| DSR | 数据集就绪 (Data Set Ready)                           |
| DCD | 数据载波检测 (Data Carrier Detect) (RS-232 的收到线路信号检测器) |
| SG  | 信号地 (Signal Ground)                              |
| RTS | 要求发送数据 (Request to Send)                         |
| CTS | 允许对方发送数据 (Clear to Send)                         |

FreeBSD 在速度超过 2400 bps 的情形需要通过 RTS 和 CTS 信号来完成流控制，通过 CD 信号来呼叫IO和挂机，并通过 DTR 信号来在会结束调制解调器行位置。某些IO在接口没有提供全部需要的信号，可能会带来问题，例如在挂断时灯会不消失，它就有可能是IO的问题。

与其它 UNIX® 操作系统类似，FreeBSD 使用硬件信号来呼叫IO，以及在挂断时挂断并位置调制解调器。FreeBSD 避免发送命令调制解调器，或其状态。如果你熟悉通过调制解调器来连接基于 PC 的 BBS 系统，它可能看起来有点无用。

27.4.2. 串口的考虑

FreeBSD支持基于 NS8250， NS16450， NS16550 和 NS16550A 的EIA RS-232C通信接口。8250和16450IO有IO字符缓冲。16550IO提供了一个 16 个字符的缓冲，可以提高更多的系统性能。因为IO字符缓冲比 16 个字符的缓冲需要更多的系统资源来工作，所以基于16550A的接口可能更好。如果系统没有活动的串口，或有大的IO，16 字符缓冲的IO于低速率的通信来更好。

27.4.3. 快速IO

在IO端，init 会在IO配置串口上IO个IO接入产生一个 getty 进程。例如，如果一个 modem 被附在 /dev/ttyu0 中，用命令ps ax可以显示下面IO些：

```
4850 ?? I      0:00.09 /usr/libexec/getty V19200 ttyu0
```

当用IO上modem，并使用它行IO接，CD IO就会被 modem IO出。内核注意到IO波信号已被IO到，需要完成 getty IO端口的打开。getty IO送一个登录：在指定的初始IO速度上的命令行。Getty IO合法的字符是否被接收，在典型的配置中，如果IO "IO"， getty IO就会IO法

000速度，直到它接收到合理的字符。

用户在输入他的登录名称后，`getty`行`/usr/bin/login`，它会要求用户输入密码来完成登录，然后用户使用他的shell。

#### 27.4.4. 配置文件

如果希望允许输入的 FreeBSD 系统，在 `/etc` 目录中有三个系统配置文件需要关注。其一是 `/etc/gettytab`，其中包含用于 `/usr/libexec/getty` 服务的配置信息。其二是 `/etc/ttys`，它的作用是告诉 `/sbin/init` 哪些 tty 上运行 `getty`。最后，关于端口的初始化命令，放到 `/etc/rc.d/serial` 脚本中。

关于在 UNIX® 上配置调制解调器有主要的流派。一一是将本地计算机到调制解调器的 RS-232 接口配置固定速率。做得好的是，程序能够立即得到系统的登录提示符，而其缺点是，系统并不知道用真的数据速率是多少，因而，类似 Emacs 的程序，也就无法调整它控制屏幕的方式，以便慢速地接改善输入。

另一流派将调制解调器的 RS-232 接口速率配置随程序的连接速率变化。例如，V.32bis (14.4 Kbps) 连接，调制解调器会自己的 RS-232 接口以 19.2 Kbps 的速率运行，而 2400 bps 连接，会使调制解调器的 RS-232 接口以 2400 bps 的速率运行。由于 `getty` 并不能具体的调制解调器的连接速率反馈信息，因此，`getty` 会以初始速度出一个 `login:` 提示，并用它的空字符。如果用看到乱码，他知道此按下 `Enter`，直到看到可以辨别的提示符为止。如果数据速率不匹配，`getty` 会将用户输入的任何信息均“乱”，并以下一速率来再次出 `login:` 提示符。一程序可能需要令人作呕地重下去，不一般而言，用只要敲一下就能看到正的提示符了。然，登录程序看起来不如前面所介绍的“固定速率”方法那明了，但使用低速连接的用户，却可以在运行全屏程序得到更好的交互。

一将尽可能公平地介绍于配置的信息，但更着力于介绍调制解调器速率随连接速率变化的配置方法。

##### 27.4.4.1. `/etc/gettytab`

`/etc/gettytab`是一个用来配置 `getty` 信息的 `termcap` 格式的文件。看看 `gettytab` 的机手册了解完整的文件格式和功能列表。

###### 27.4.4.1.1. 固定速度的配置

如果把modem的数据通率固定在一个特殊的速度上，不需要 `/etc/gettytab` 文件作任何化。

###### 27.4.4.1.2. 匹配速度的配置

将需要在 `/etc/gettytab` 中置一个来告诉 `getty` 希望在 modem 上使用的速度。如果的 modem 的速率是 2400 bit/s，可以使用有的 `D2400` 的。

```
#
# Fast dialup terminals, 2400/1200/300 rotary (can start either way)
#
D2400|d2400|Fast-Dial-2400:\
        :nx=D1200:tc=2400-baud:
3|D1200|Fast-Dial-1200:\
        :nx=D300:tc=1200-baud:
5|D300|Fast-Dial-300:\
        :nx=D2400:tc=300-baud:
```

如果你有一个更高速度的 modem，必在 `/etc/gettytab` 中添加一个。下面是一个可以以最高 19.2 Kbit/s 的用在 14.4 Kbit/s 的 modem 上的接口：

```
#
# 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:
```

做的结果是 8-数据位，没有奇偶校验的接口。

上面使用 19.2 Kbit/s 的接口速度的例子，也可以使用 9600 bit/s (for V.32), 2400 bit/s, 1200 bit/s, 300 bit/s, 直到 19.2 Kbit/s。通率的仍使用 `nx=` ("next table") 来。它仍使用一个 `tc=` ("table continuation") 的仍来加速于一个特殊率的仍置。

如果你有 28.8 Kbit/s 的 modem，或想使用它的 14.4 Kbit/s 模式，就需要使用一个更高的超 19.2 Kbit/s 的通速度的 modem。它是一个 57.6 Kbit/s 的 `gettytab` 的例子：

```
#
# 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:
```

如果你的 CPU 速度低，或系的负荷很重，而且没有 16550A 的串口，可能会在 57.6 Kbit/s 上得到 `silo`。

#### 27.4.4.2. `/etc/ttys`

`/etc/ttys` 文件的配置在在 `/etc/ttys` 中加端口中介。配置 modem 是相似的，但我必指定一个不同的端口型。定速度和匹配速度配置的通用格式是：

```
ttyu0  "/usr/libexec/getty xxx"  dialup on
```

上面的第一条是一个特定的文件 - `ttyu0` 表示 `/dev/ttyu0` 是一个 `getty` 将被使用的文件。第二条 `"/usr/libexec/getty xxx"` 是将行在上的程序 `init`。第三条, `dialup`, 是默认的终端型。第四个参数, `on`, 指出了路是可操作的 `init`。也可能会有第五个参数, `secure`, 但它将只被用作有物理安全的终端(如系终端)。

默认的终端型可能依赖于本地参考。号是默认的终端型, 以至用可以定制它的登录脚本来注意终端什么时候号, 和自它的终端型。然而, 作者它很容易在它的站点上指定 `vt102` 作默认的终端型, 因用在它的程序系上使用的是VT102模拟器。

`/etc/ttys`作修改之后, 可以发送 `init` 程序一个 HUP 信号来重文件。可以使用下面的命令来送信号:

```
# kill -HUP 1
```

如果是的第一次置系, 可能要在信号 `init` 之前等一下, 等到 modem 正确地配置并接好。

#### 27.4.4.2.1. 固定速度的配置

于一个固定速度的配置, 的 `ttys` 必须有一个 `getty` 提供固定速度的。于一个速度被定在 19.2kbit/s 的 modem, `ttys` 是:

```
ttyu0  "/usr/libexec/getty std.19200"  dialup on
```

如果的 modem 被定在一个不同的数据速度, `std.speed` 使用当的速度来代替 `std.19200`。信使用了一个在 `/etc/gettytab` 中列出的正的型。

#### 27.4.4.2.2. 匹配速度的置

在一个匹配速度的置中, 的 `ttys` 需要参考在 `/etc/gettytab` 当的起始 "auto-baud" 行。例如, 如果一个以 19.2 Kbit/s 始的可匹配速度的 modem 添加上面建的行, 的 `ttys` 可能是:

```
ttyu0  "/usr/libexec/getty V19200"  dialup on
```

#### 27.4.4.3. `/etc/rc.d/serial`

高速制解器, 如使用 V.32、V.32bis, 以及 V.34 的那些, 需要使用硬件 (RTS/CTS) 流控制。可以在 `/etc/rc.d/serial` 中加 `stty` 命令来在 FreeBSD 内核中, 制解器置硬件流控制志。

例如, 在 1 号串口 (COM2) 入和出上配置 `termios` 志 `crtsets`, 可以通过在 `/etc/rc.d/serial` 加下面的置来:

```
# Serial port initial configuration
stty -f /dev/ttyu1.init crtsets
stty -f /dev/cuau1.init crtsets
```



## 27.4.5. Modem 配置

如果你有一个 modem，它的参数能被存储在非易失性的 RAM 中，你将必须使用一个终端程序来配置参数（比如 MS-DOS® 下的 Telix 或者 FreeBSD 下的 **tip**）。使用同口的通信速度来连接 modem 作为初始速度 **getty** 将使用和配置 modem 的非易失性 RAM 来满足一些要求：

- 连接宣告 CD
- 操作宣告 DTR；DTR 消失挂断线路并复位调制解调器
- CTS 数据流控制
- 禁用 XON/XOFF 流控制
- RTS 接收数据流控制
- 静模式 (无返回)
- 无命令回

modem 的文档到需要用什命令和 DIP 接口配置。

例如，要在一个 U.S. Robotics® Sportster® 14400 的外置 modem 上配置上面的参数，可以用下面些命令：

```
ATZ
AT&C1&D2&H1&I0&R2&W
```

也可能想要在 modem 上有机会个配置，例如它是否使用 V.42bis 和 MNP5 。

外置 modem 也有一些用来配置的 DIP 开关，也可以使用些配置作一个例子：

- Switch 1: UP - DTR Normal
- Switch 2: N/A (Verbal Result Codes/Numeric Result Codes)
- Switch 3: UP - Suppress Result Codes
- Switch 4: DOWN - No echo, offline commands
- Switch 5: UP - Auto Answer
- Switch 6: UP - Carrier Detect Normal
- Switch 7: UP - Load NVRAM Defaults
- Switch 8: N/A (Smart Mode/Dumb Mode)

在号 modem 上的果代被禁用/抑制，以避免当 **getty** 在 modem 于命令模式并回入地出 **login:** 提示可能造成的。可能致 **getty** 与 modem 之生更的不必要交互。

### 27.4.5.1. 固定速度的配置

于固定速度的配置，需要配置 modem 来得一个不依赖于通信率的固定的 modem 到计算机的率。在一个 U.S. Robotics® Sportster® 14400 外置 modem 上，些命令将固定 modem 到计算机的率：



```
ATZ
AT&B1&W
```

### 27.4.5.2. 匹配速度的配置

由于一个速度的配置，需要配置 modem 的它的串口速率匹配接收的速率。 在一个 U.S. Robotics® Sportster® 14400 的外置 modem 上，一些命令将设定 modem 的修正速率符合命令要求的速度，但允许串口速度没有直接：

```
ATZ
AT&B2&W
```

### 27.4.5.3. 设置 modem 的配置

大多数高速的 modem 提供了用来查看当前操作参数的命令。 在USR Sportster 14400外置modem上，命令 **ATI5** 显示了存储在非易失性RAM中的设置。 要查看正确的 modem 操作参数，可以使用命令 **ATZ** 然后是 **ATI4**。

如果有一个不同牌子的 modem， modem 的使用手册看看如何双重设置 modem 的配置参数。

## 27.4.6. 解答

这儿是几个号码modem的。

### 27.4.6.1. 设置FreeBSD系统

把 modem 接到 FreeBSD 系统， 系统， 然后， 如果 modem 有一个指示灯， 当登录看看 modem 的 DTR 指示灯是否亮： 会在系统控制台输出命令行——如果它亮， 意味着 FreeBSD 已在适当的通信端口开了一个 **getty** 进程， 等待 modem 接收一个呼叫。

如果DTR指示灯不亮， 通信控制台登录到 FreeBSD系统， 然后运行一个 **ps ax** 命令来看 FreeBSD 是否正在正确的端口运行 **getty** 进程。 将在进程显示中看到像下面的一行：

```
114 ?? I      0:00.10 /usr/libexec/getty V19200 ttyu0
115 ?? I      0:00.10 /usr/libexec/getty V19200 ttyu1
```

如果看到是这样的：

```
114 d0 I      0:00.10 /usr/libexec/getty V19200 ttyu0
```

modem 不接收呼叫， 意味着 **getty** 已在通信端口打了。 可以指出有或 modem 配置， 因为 **getty** 无法打通信端口。

如果没有看到任何 **getty** 进程等待打想要的 ttyuN 端口， 在 /etc/ttys 中双倍的看看那儿是否有。 另外， 日志文件 /var/log/messages 看看是否有一些来自 **init** 或 **getty** 的日志。 如果有任何信息， 仔细配置文件 /etc/ttys 和 /etc/gettytab， 有相关的文件 /dev/ttyuN， 是否有， 丢失， 或指定文件。

### 27.4.6.2. 接入 Try Dialing In

拨号入系。 信使用8位， 没有奇偶， 在程系上的1阻止位。 如果不能立刻得到一个命令行， 隔一秒按一下 `Enter`。 如果仍没有看到一个登： 拨号送一个 `BREAK`。 如果正使用一个高速的 modem 来号， 在定号 modem 的接口速度后再。

如果不能得到一个登：prompt， 再一下 `/etc/gettytab`， 重：

- 在 `/etc/ttys` 中指定的初始可用的名称与 `/etc/gettytab` 的一个可用的相匹配。
- 个 `nx=` 与一个 `gettytab` 可用名称匹配。
- 个 `tc=` 与一个 `gettytab` 可用名称相匹配。

如果号但 FreeBSD 系上的 modem 没有回， 信 modem 能回。 如果 modem 看起来配置正了， 通 modem 的指示灯来 DTR 接正。

如果做了好几次， 它仍然无法工作， 打断一会， 等会再。 如果不能工作， 也一封子件 [FreeBSD 一般件列表](#) 求助。

## 27.5. 出



从 FreeBSD 8.0 始， 用于串口的点从 `/dev/cuadN` 改了 `/dev/cuauN`； 从 `/dev/ttydN` 改了 `/dev/ttyuN`。 FreeBSD 7.X 用需要根据情况文 中的例子 行必要的整。

下面将的主机通 modem 接到一台计算机上。 只要当地建立一个端作程主机就可以。

可以用来登一个BBS。

如果用 PPP 有， 那接可以用来从 Internet 上下一个文件。 如果必 FTP 一些西， 而 PPP 断了， 使用端会来 FTP 它。 然后使用 `zmodem` 来把它到 的机器上。

### 27.5.1. 我的Stock Hayes Modem不被支持， 我怎么办？

事上， 机手册于 个的描述已了。 一个通用的 Hayes 号已 内建其中。 只要在 的 `/etc/remote` 文件中使用 `at=hayes`。

Hayes 不 "明" 只能出一些比 新的 modem 的高特性 - 如 `BUSY`、`NO DIALTONE`， 或 `CONNECT 115200` 的信息将被乱。 当 使用的候， 必把些信息掉。(通 `ATX0&W`)。

外， 号的延是 60 秒。 的 modem 可能使用 外的 或提示 有其他的通。 `ATS7=45&W`。

### 27.5.2. 我如何入些 AT 命令？

在 `/etc/remote` 文件中加一个 "direct" 。 例而言， 如果 的制解器挂在第一个串口， 即 `/dev/cuau0` 上， 添加下面行：

```
cuau0:dv=/dev/cuau0:br#19200:pa=none
```

此命令使用modem所支持的最高波特率。接下来，输入 `tip cuau0` 就可以连接到 modem 上了。

此外，也可以 `root` 身份运行 `cu` 命令：

```
# cu -l line -s speed
```

`line` 是串口 (例如 `/dev/cuau0`) 而 `speed` 是速率 (如 `57600`)。当输入完 AT 之后，按 `~.` 即可退出。

### 27.5.3. 在 `pn @` 不能工作？

在电话号码中的 `@` 告诉计算机在 `/etc/phones` 文件中找一个号码。但 `@` 也是一个在像 `/etc/remote` 的可用文件中的特殊字符。用一个反斜杠符号退出：

```
pn=\@
```

### 27.5.4. 我如何在命令行输入号码？

在 `/etc/remote` 文件中通常放着一个叫做 "generic" 的条目。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:
```

然后，可以运行：

```
# tip -115200 5551234
```

如果喜欢 `cu` 而不是 `tip`，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
```

### 27.5.5. 做什么是否每次都需要重新输入波特率？

添加一个 `tip1200` 或 `cu1200`，并将波特率改成更合适的。 `tip` 的默认是 1200 bps，也就是为什么会有 `tip1200` 这条的原因。当然并不需要使用 1200 bps。

### 27.5.6. 我通一个端服务器到了很多主机。

除非次都要等到接到主机然后入 `CONNECT host`, 否使用 `tip` 的 `cm` 功能。例如, 在 `/etc/remote` 中的些:

```
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:
```

将入 `tip pain` 或 `tip muffin` 接到主机 `pain` 或 `muffin`, 和 `tip deep13` 接到端服务器。

### 27.5.7. `tip`能个站点用多个路?

常有一个, 一个大学有几个modem路, 几千个学生法使用它。

在 `/etc/remote` 中的大学添加一个, 然后 `pn` 功能使用 `@`:

```
big-university:\
      :pn=\@:tc=dialout
dialout:\
      :dv=/dev/cuau3:br#9600:at=courier:du:pa=none:
```

接着, 在 `/etc/phones` 中列出大学的号:

```
big-university 5551111
big-university 5551112
big-university 5551113
big-university 5551114
```

`tip` 将按序用一个, 然后就停止。如果想, 隔一段再行 `tip`。

### 27.5.8. 什我必入 `Ctrl + P` 次才能出 `Ctrl + P` 一次?

`Ctrl + P` 是默的"制"字符, 被用来告 `tip` 下一个字符是文字的数据。可以用 `~s` 任何其他的字符置制字符, 意思是"置一个量"。

在新的一行入 `~sforce=single-char`。 `single-char` 是任何字符。如果漏了 `single-char`, 那制字符就是空字符, 可以入 `Ctrl + 2` 或 `Ctrl + Space` 来完成。更好的 `single-char` 是 `Shift + Ctrl + 6`, 只用在一些端服务器上。

通在的 `$HOME/.tiprc` 文件中指定下面行, 就可以得到想要的任何制字符:

```
force=single-char
```

### 27.5.9. 突然我输入的一串西都成了大写??

一定是输入了 `Ctrl + A`，即 `tip` 的 "raise character"，会地指定成坏掉的 caps-lock。使用上面的 `~s` 来合理地置各 `raisechar`。事实上，如果不想使用些特性的，可以用同的方法置制字符。

儿有一个很好的示例 `.tiprc` 文件，Emacs用来，需要常按 `Ctrl + 2` 和 `Ctrl + A`：

```
force=^^
raisechar=^^
```

`^^` 是 `Shift + Ctrl + 6`。

### 27.5.10. 如何用 `tip` 做文件??

如果正在与一台 UNIX® 系，可以用 `~p`(put) 和 `~t` (take) 送和接收文件。些命令可以在程系上行 `cat` 和 `echo` 来接收和送文件。法是：

`~p local-file [ remote-file ]`

`~t remote-file [ local-file ]`

由于没有校，所以需要使其他，如 `zmodem`。

### 27.5.11. 我如何用 `tip` 行 `zmodem` ?

要接收些文件，可以在程端送程序。然后，入 `~C rz` 在本地始接收它。要送文件，可以在程端接收程序。然后，入 `~C sz files` 把它送到程系。

## 27.6. 置串口控制台



从 FreeBSD 8.0 始，用于串口的点从 `/dev/cuadN` 改了 `/dev/cuauN`；从 `/dev/ttydN` 改了 `/dev/ttyuN`。FreeBSD 7.X 用需要根据情况文例中的例子行必要的整。

### 27.6.1. 介

FreeBSD可以通一个串口只使用一个 (dumb) 端就可以一个系。一配置只有人能使用：希望在机器上安装 FreeBSD 的系管理，他没有或示器，有就是要内核或程序的入。

就象 [FreeBSD 引程](#) 描述的，FreeBSD 采用一个三的程。最先存在 FreeBSD 磁的 slice 的代中。引然后就被加，接着行第三引器 (`/boot/loader`)。

了置串口控制台，必配置代，引器代和内核。

### 27.6.2. 串口控制台的配置，明版

一假定使用默的配置，只希望迅速地得于配置串口控制台的概。

1. 使用串口线接 COM1 和控制端。
2. 要在串口控制台上显示所有的引导信息，需要以超级用户的身执行下面的命令：

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

3. 编辑 `/etc/ttys` 并把 `ttyu0` 的 `off` 改为 `on`，`dialup` 改为 `vt100`。否则通过串口控制台上将不会提示输入命令，从而导致潜在的安全漏洞。
4. 重新引导并观察是否生效。

如果需要不同的配置，更详细的配置可以在 [串口控制台的配置](#) 找到。

### 27.6.3. 串口控制台的配置

### 1. 准一根串口。

需要使用一个 null-modem 的或准的串口和一个 null-modem 配器。参考 [和端口](#) 中有串口的。

### 2. 拔掉。

大多数的 PC 在机的时候会到，如果没有到，会出。一些机器会提示缺少，就不会引。

如果的计算机出，但仍能，可以不必要理它。

如果的计算机没有拒，那需要配置 BIOS 来避免它。参考的主板的使用明了解更多。



在 BIOS 中将 "Not installed" (未安装)。在仍然无法使用。做只是告 BIOS 在不要探。的 BIOS 不抱怨不存在。即使一置 "Not installed"，只要把上，它就仍可使用。如果以上的不存在于 BIOS 中，可 "Halt on Error"。把一置 "All but Keyboard" 或者是 "No Errors"，都能起到相同的作用。



如果系有 PS/2 鼠，如果幸的，也可以象一把它拔下来，是因 PS/2 鼠与的一些硬件是共享的，的鼠上去，系会仍在那儿。

### 3. 一个 (dumb) 端到 COM1：(sio0)。

如果没有端，可以使用一个比老的有一个 modem 程序的 PC/XT 机器，或在其他 UNIX® 机器上的串口。如果没有 COM1：(sio0)，去一个。，就不能只能 COM1：来系。如果已在上一台上使用 COM1，必除那个，然后安装一个新的系和内核。

### 4. 信的内核配置文件已 COM1：(sio0) 置了当的：

有的：

#### 0x10

用控制台支持。如果没有置它，其他的控制台都会被忽略。在，大多数的置都有控制台的支持。个的第一个就是首的。个独是不能保串口用于控制台的，置下面的或加上下面描述的 -h，和个放在一起。

#### 0x20

无是否使用了下面将要的 -h，都制个元作控制台（除非使用了更高先的控制台）。志 0x20 必与 0x10 一起使用。

#### 0x40

留个元（配合 0x10）并它不能用于普通的使用。不希望在作控制台的串口元上置个志。一志是内核程准的。参 [作者手册](#) 以了解于程更一的情况。

例如：

```
device sio0 flags 0x10
```

看看 [sio\(4\)](#) 的 [手册](#) 了解更多信息。

如果 `uart` 没有被 `uart`， 必须运行 `UserConfig` 或重新 `uart` 内核。

## 5. 在 `boot` 磁区的 `a` 分区的根目录建 `boot.config` 文件。

这个文件将指引 `boot` 如何 `uart`。 为了激活串口控制台， 必须有一个或多个下面的 `uart`——如果 `uart` 要多个 `uart`， 在同一行必须都包含它：

### -h

切换内部和串口控制台。 使用 `uart` 来交 `uart` 控制台。 例如， 如果 `uart` 从内部控制台 `uart`， `uart` 可以使用 `-h` 来直接使用 `uart` 和内核来使用串口作 `uart` 的控制台。 另外， 如果 `uart` 从串口 `uart`， `uart` 可以使用 `-h` 来告 `uart` 和内核使用 `uart` 作 `uart` 控制台。

### -D

切换 `uart` 和双重控制台配置。 在 `uart` 配置中， 控制台将是本机的控制台 (`uart`) 或串口。 在双重控制台配置中， `uart` 和串口将同 `uart` 成 `uart` 控制台， 无 `-h` 的 `uart` 的情形。 然而， 双控制台配置只在 `uart` 行的 `uart` 中起作用。 一旦 `uart` 得控制， 由 `-h` `uart` 指定的控制台将成为 `uart` 唯一的控制台。

### -P

在 `uart`， 探 `uart`。 如果 `uart` 不到， `-D` 和 `-h` `uart` 会自 `uart`。



由于当前版本 `uart` 的空 `uart` 限制， `-P` `uart` 只能探 `uart` 展的 `uart`。 少于 1010 的 `uart` 将无法被探 `uart` 到。 如果 `uart` 到 `uart` 个情况， 必须避免使用 `-P` `uart`。 目前 `uart` 没有 `uart` 个 `uart` 的 `uart`。

使用 `-P` `uart` 来自 `uart` 控制台， 或使用 `-h` `uart` 来激活控制台。

`uart` 也可以使用 `boot` 机文中所描述的其他 `uart`。

除了 `-P` `uart`， 所有 `uart` 将被 `uart` 和 `uart` (`/boot/loader`)。 `uart` 和 `uart` 将通 `uart` `-h` `uart` 的 `uart` 来决定是 `uart` 成 `uart` 控制台， 是串口成 `uart` 控制台。 `uart` 表示如果 `uart` 指定 `-D` `uart`， 但在 `/boot.config` 中没有 `-h` `uart`， `uart` 在 `uart` 代 `uart` 使用串口作 `uart` 控制台。 `uart` 和 `uart` 将使用内部 `uart` 作 `uart` 控制台。

## 6. `uart` 机器

当 `uart` 的 `FreeBSD`， `uart` 将把 `/boot.config` 的内容 `uart` 控制台。 例如：

```
/boot.config: -P
Keyboard: no
```

如果 `uart` 把 `-P` 放在 `/boot.config` 中并指出 `uart` 存在或不存在， 那将只出 `uart` 第二行。 些信息会被定位到串口或内部控制台， 或者同 `uart`， `uart` 完全取决于 `/boot.config` 中的 `uart`。



|         |             |
|---------|-------------|
| 00      | 送出消息的00     |
| 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00 | 内部控制台       |
| -P, 无00 | 串口控制台       |

出0上面信息后，在引00加000引0器和更多信息被映到屏幕之前将有一个小小的停0。在通常情况下，0不需要打断000程，但0了信0置是否正0，0也可以00做。

在控制台上按 `Enter` 以外的任意0就能打断000程。引00将0入命令行模式。0将看到：

```
>> FreeBSD/i386 BOOT
Default: 0:ad(0,a)/boot/loader
boot:
```

00上面出0的信息，可能是串口，或内部控制台，或0个同0，完全取决于0在 `/boot.config` 中的00。如果信息出0在正0的控制台，按 `Enter` 00000程。

如果0要使用串口控制台，但0没有看到命令行，那可能0置有00。00，0入 `-h` 然后按 `Enter` 或 `Return` 来告引00（然后是00引0器和内核）00串口作0控制台。一旦系0起来了，就可以回去00一下是什0出了00。

00引0器加0完后，0将0入000程的第三0，0仍然可以在00引0器通00定0喜0的0境来切0内部控制台和串口控制台。参考 [从00引0器修改控制台](#)。

### 27.6.4. 摘要

0是几个在0章要00的几个0置和00的控制台的摘要。

#### 27.6.4.1. 例1：00 sio0 0置00 0x10

```
device sio0 flags 0x10
```

| 在 <code>/boot.config</code> 中的00 | 引000行0所用的控制台 | 引0加0器0行0所用的控制台 | 内核所用的控制台 |
|----------------------------------|--------------|----------------|----------|
| 无                                | 内部           | 内部             | 内部       |
| -h                               | 串口           | 串口             | 串口       |
| -D                               | 串口和内部        | 内部             | 内部       |
| -Dh                              | 串口和内部        | 串口             | 串口       |
| -P, 有00                          | 内部           | 内部             | 内部       |

| 在 <code>/boot.config</code> 中的 | 引导行所用的控制台 | 引导加载器行所用的控制台 | 内核所用的控制台 |
|--------------------------------|-----------|--------------|----------|
| <code>-P</code> ，没有            | 串口和内部     | 串口           | 串口       |

27.6.4.2. 例2： `sio0` 置 `0x30`

```
device sio0 flags 0x30
```

| 在 <code>/boot.config</code> 中的 | 引导行所用的控制台 | 引导加载器行所用的控制台 | 内核所用的控制台 |
|--------------------------------|-----------|--------------|----------|
| 无                              | 内部        | 内部           | 串口       |
| <code>-h</code>                | 串口        | 串口           | 串口       |
| <code>-D</code>                | 串口和内部     | 内部           | 串口       |
| <code>-Dh</code>               | 串口和内部     | 串口           | 串口       |
| <code>-P</code> ，有             | 内部        | 内部           | 串口       |
| <code>-P</code> ，没有            | 串口和内部     | 串口           | 串口       |

27.6.5. 串口控制台的提示

27.6.5.1. 置更高的串口速度

在默认配置中，串口的置是：速率 9600 波特、8 数据位、无奇偶校位、1 停止位。如果你希望修改默认的控制台速率，可以采用下列几种方法之一：

- 将 `BOOT_COMCONSOLE_SPEED` 配置你希望的速率，并重新引导。参 使用 `sio0` 以外的串口 作控制台 以了解如何和安装新的引导。

如果串口控制台已配置使用 `-h` 以外的其它方式引导， 或者内核使用的速率与引导不同， 必需在内核配置文件中加入下述置，并重新新内核：

```
options CONSPEED=19200
```

- 使用内核引导 `-S`。 `-S` 个命令行可以加到 `/boot.config` 中。参 手册 `boot(8)` 以得如何在 `/boot.config` 中加，以及其它的可用。
- 在你的 `/boot/loader.conf` 文件中用 `comconsole_speed`。

使用个，你需要在 `/boot/loader.conf` 中配置 `console`、 `boot_serial`， 以及 `boot_multicons`。 下面是一个利用 `comconsole_speed` 改串口控制台速率的例子：

```
boot_multicons="YES"
boot_serial="YES"
comconsole_speed="115200"
console="comconsole,vidconsole"
```

### 27.6.5.2. 使用 **sio0** 以外的串口 作控制台

使用串口而不是 **sio0** 作控制台需要做一些重。如果无论如何都要使用一个串口，重新引导，引导器和内核。

1. 取得内核源代码 (参考 [更新与升级 FreeBSD](#))。
2. 编辑 `/etc/make.conf` 文件，然后置 `BOOT_COMCONSOLE_PORT` 作要使用 (`0x3f8`、`0x2f8`、`0x3E8` 或 `0x2E8`) 端口的地址。只有 **sio0** 到 **sio3** (**COM1** 到 **COM4**) 都可以使用；但多串口将不会工作。不需要任何中断置。
3. 建一个定制的内核配置文件，在要使用的串口添加合的。例如，如果要将 **sio1** (**COM2**) 作控制台：

```
device sio1 flags 0x10
```

或

```
device sio1 flags 0x30
```

其他端口的控制台也不要。

4. 重新引导和安装引导：

```
# cd /sys/boot
# make clean
# make
# make install
```

5. 重建和安装内核。
6. 用 [bsdlabeled\(8\)](#) 将引导写到上，然后从新内核。

### 27.6.5.3. 通串口入 **DDB** 器

```
options BREAK_TO_DEBUGGER
options DDB
```

### 27.6.5.4. 在串口控制台上得到一个登录命令行

可能希望通串口入登录提示，在可以看到信息，通串口控制台入内核信息。可以。

用一个器打 `/etc/ttys` 文件，然后到下面的行：

```
ttyu0 "/usr/libexec/getty std.9600" unknown off secure
ttyu1 "/usr/libexec/getty std.9600" unknown off secure
ttyu2 "/usr/libexec/getty std.9600" unknown off secure
ttyu3 "/usr/libexec/getty std.9600" unknown off secure
```

ttyu0 到 ttyu3 相当于 COM1 到 COM4。可以打开或关闭某个端口。如果已经改变了串口的速度，必须改掉标准的 9600 与当前的例如 19200 相匹配。

也可以改变端口的类型从不知名的到串口端口的真类型。写完个文件，必须 `kill -HUP 1` 来使个修改生效。

## 27.6.6. 从引导器修改控制台

前面一节描述了如何通过整引用来定义串口控制台。将到在引导器中通过一些命令和环境量来指定控制台。由于引导器会被进程的第三所引用，引用以后，在引导器中的设置将忽略在引中的设置。

### 27.6.6.1. 配置串口控制台

可以很容易地指定引导器和内核来使用串口控制台，只需要在 `/boot/loader.conf` 中写入下面行：

```
console="comconsole"
```

无论前一行中的引如何配置，个设置都会生效。

最好把上面一行放在 `/boot/loader.conf` 文件的第一行，以便尽早地在看到串口控制台的设置信息。

同样地，可以指定内部控制台：

```
console="vidconsole"
```

如果不设置引环境量控制台，引导器和内核将使用在引用 `-h` 指定的控制台。

控制台可以在 `/boot/loader.conf.local` 或者是在 `/boot/loader.conf` 中指定。

看看 [loader.conf\(5\)](#) 的手册了解更多信息。



目前，引尚不提供与引加器的 `-P` 等价的，此外，它也不能根据是否有存在自己决定使用内部控制台还是串口控制台。

### 27.6.6.2. 使用串口而不是 sio0 作控制台

要使用一个串口而不是 sio0 作串口控制台需要重新配置引导器。下面的跟使用 sio0 以外的串口作控制台描述的相似。

### 27.6.7. 警告

这篇文章本意是想告诉人如何确定没有显示或的服务器。不幸的是，大多数系统没有可以，而没有显示就不。使用 AMI BIOS 的机器可以通过在 CMOS 中将 "graphics adapter" 为 "Not installed" 来在不要求显示器。

然而，多机器并不支持个，如果的系统没有显示硬件就拒。于些机器，即使没有显示器，也必须在机器上上显示器。建议采用 AMI BIOS 的机器。

# Chapter 28. PPP 和 SLIP

## 28.1. 概述

FreeBSD 有很多方法可以将计算机与计算机连接起来。通常使用调制解调器 modem 来建立网络或 Internet 连接，或允许其他人通过他们的机器来上网，有些都要求使用 PPP 或 SLIP。本章将介绍配置一些基于 modem 的通信服务的方法。

读完一章，你将了解：

- 如何配置用户 PPP。
- 如何配置内核 PPP。（仅限 FreeBSD 7.X）。
- 如何配置 PPPoE (PPP over Ethernet)。
- 如何配置 PPPoA (PPP over ATM)。
- 如何配置和安装 SLIP 客户端和服务端。（仅限 FreeBSD 7.X）。

在开始本章之前，应：

- 熟悉基本的网络。
- 理解调制解调器和 PPP、SLIP 的基础知识。

你可能想知道用户 PPP 与内核 PPP 之间的不同之处。回答很简单：用户 PPP 管理用户的输入和输出数据，而不是内核。在内核与用户空间之间复制数据的花费要大一些，但它能提供具有更多特性的 PPP。用户 PPP 使用 tun 与外界通信而内核 PPP 使用 ppp 。



在本章中，如果没有特殊说明，ppp 指的是用户 PPP，除非需要和其它 PPP 组件，例如 pppd（仅限 FreeBSD 7.X）加以区分。此外，若没有额外的注明，本章所介绍的所有命令都需要以 root 身份来执行。

## 28.2. 使用用户 PPP



从 FreeBSD 8.0 开始，uart(4) 取代了 sio(4) 。

用以表示串口的设备点由分 /dev/cuadN 改为了 /dev/cuaun，并从 /dev/ttydN 改为了 /dev/ttyun。FreeBSD 7.X 用户在升级时需要因之配置文件进行必要的更改。

### 28.2.1. 用户 PPP

#### 28.2.1.1. 前提条件

本章假定具备如下条件：

- 有一个 ISP 提供的用于连接使用 PPP 的调制解调器。
- 需要有连接在系统上，并做了正确配置的 modem，或其他能连接 ISP 的设备。
- ISP 的电话号码。

- 的登录名称和密码 (可能是一般的 UNIX 风格的登录名和密码, 也可能是 PAP 或 CHAP 登录名和密码)。
- 一个或多个域名服务器 IP 地址。通常, 会从ISP得到个的IP地址。如果至少得到了一个, 就可以在文件 `ppp.conf` 中加入 `enable dns` 命令使 ppp 置域名服。个功能取决于 ISP 支持 DNS 商的具体。

下面的信息由的 ISP 提供, 但不是必需的:

- ISP的网IP地址。网是准接, 并默路由的主机。如果没有个信息, 可以虚一个, 在接ISP 的 PPP 服务器会自告正的。

个虚的 IP 地址在 ppp 中做 `HISADDR`。

- 准使用的子网掩。如果ISP没有提供, 一般使用 `255.255.255.255` 是没有的。
- 如果 ISP 提供了静的IP地址和主机名, 可以入它。反之, 方主机指定它合的 IP 地址。

如果不知道些信息, 与的 ISP 系。



在中, 所有作例子展示的配置文件中都有行号。些行号只是了使解和得方便, 在真的文件中并不存在。此外, 在必要使用 Tab 和空格来行。

#### 28.2.1.2. PPP自化配置

`ppp`和`pppd`(PPP的内核, 限 FreeBSD 7.X) 都使用 `/etc/ppp` 目中的配置文件。用 PPP 的例子可以在 `/usr/shared/examples/ppp/` 中到。

配置`ppp`要求根据的需要几个文件。几个文件取决于的 IP 是静分配 (次都使用同一个地址) 是分配的 (次接到 ISP 都会得不同的 IP 地址)。

##### 28.2.1.2.1. PPP和静IP地址

需要配置文件`/etc/ppp/ppp.conf`, 如下所示。



以冒号:尾的行从第一列 (行首)始, 其它所有的行都要使用空格或制表符 (Tab) 来。

```

1  default:
2      set log Phase Chat LCP IPCP CCP tun command
3      ident user-ppp VERSION (built COMPILATIONDATE)
4      set device /dev/cuau0
5      set speed 115200
6      set dial "ABORT BUSY ABORT NO\\sCARRIER TIMEOUT 5 \
7              \"\" AT OK-AT-OK ATE1Q0 OK \\dATDT\\t TIMEOUT 40 CONNECT"
8      set timeout 180
9      enable dns
10
11  provider:
12      set phone "(123) 456 7890"
13      set authname foo
14      set authkey bar
15      set login "TIMEOUT 10 \"\" \"\" gin:--gin: \\U word: \\P col: ppp"
16      set timeout 300
17      set ifaddr x.x.x.x y.y.y.y 255.255.255.255 0.0.0.0
18      add default HISADDR

```

#### 行1：

指定默认的行为。当PPP运行一个会话中的命令将来自该行。

#### 行2：

启用登录参数。工作正常后，避免产生过多的日志文件，进行优化：

```
set log phase tun
```

#### 行3：

告诉PPP向何方自我介绍。如果在建立或使用连接遇到任何麻烦，PPP就会向何方主机自我介绍。何方主机管理在管理该会话，这些信息会有用。

#### 行4：

指明modem要连接的端口号。COM1 对应的设备是 /dev/cuau0 而 COM2 对应的设备是 /dev/cuau1。

#### 行5：

设置连接的速度。如果 115200 有问题，试试 38400。

#### 行6 & 7：

号码字符串。用PPP使用一与 [chat\(8\)](#) 程序相似的方法。参考设备手册了解该命令的更多信息。

注意，为了便于该命令运行了该行。任何 ppp.conf 里的命令都可以这样做，前提是行的最后一个字符必须是 \。

#### 行8：

设置连接的超时间隔。默认是 180 秒，所以这一行是多余的。



行 9：

告诉PPP向何方主机本地域名解析位置。如果进行了本地的域名服务器，要注释或删除一行。

行 10：

为了可能性的需要设置一个空行。空行会被PPP忽略。

行 11：

"provider"指定一个。可以改成ISP的名字，以后就可以使用 `load ISP` 来连接。

行 12：

设置提供商的号码。多个号码可以使用冒号 (:) 或管道符号 (|) 分隔。每个字符的区域在 [ppp\(8\)](#) 的联机手册中有介绍。的来，如果要循环使用些号码，可以使用冒号。如果想使用第一个号码，当第一个号码失效了再用第二个号码，就使用管道符号。如所示的那，要整个号码加上引号(")。

如果号码里有空格，必须用引号(")将其括起来。否则会造成却以察的。

行 13 & 14：

指定用户名和密码。当使用 UNIX® 风格的命令提示符登录，些可以用有 \U \P 参数的 `set login` 命令进行修改。当使用PAP或CHAP行连接，些在中使用。

行 15：

如果使用的是PAP或者CHAP，在里就不会有登录。要注释或删除一行。参考 [PAP 和 CHAP](#) 以了解更多。

登录命令的格式是chat型的。在个例子中是这样的：

```
J. Random Provider
login: foo
password: bar
protocol: ppp
```

需要改脚本以合自己的需要。当第一次写个脚本，当保已用 "chat" 并于登录状态，才能通信是否正在按行。

行 16：

设置默认的超。里，连接若在 300 秒内无将被断。如果不想置成超，将个置成0，或在命令行使用 `-ddial` 。

行 17：

设置接口地址。需要用ISP提供的IP地址替换字符串 `x.x.x.x`，用ISP的网IP地址(即要连接的主机)替换字符串 `y.y.y.y`。如果ISP没有提供网地址，可以使用 `10.0.0.2/0`。如果需要用一个"猜到"的地址，保在 `/etc/ppp/ppp.linkup` 中个 [PPP和IP地址](#) 指令建了一。如果没有一行，`ppp` 将无法以 `-auto` 模式行。

第18行：

添加一个到ISP网的默认路由。 `HISADDR` 个字会被第17行所指定的网地址替换。行必出在第17行之后，以免在 `HISADDR` 初始化之前使用它的。

如果不想使用 `-auto` 的 PPP，则行到 `ppp.linkup` 文件中。

若有一个静态IP地址，且使用 `-auto` 模式运行ppp(因在连接之前已正确配置了路由表)，那就不需要再向ppp.linkup添加。可能希望在连接以后建立一个来用程序。在以后的sendmail的例子中会解。

示例配置文件可以在目录 `/usr/shared/examples/ppp/` 中找到。

#### 28.2.1.2.2. PPP和静态IP地址

如果ISP没指定静态的IP地址，ppp要被配置成能与对方商定本地和远程地址。要完成这项工作，先要"猜"一个IP地址，然后允许ppp在连接后使用IP配置(PPCP)行正确配置。ppp.conf的配置是与PPP和静态IP地址一致的，除了以下的改：

```
17      set ifaddr 10.0.0.1/0 10.0.0.2/0 255.255.255.255 0.0.0.0
```

再次，不要包括行号，它只是一个引用。排一个空格是必需的。

行17：

/ 字符后面是 PPP 所要求的地址掩。可以根据需要使用不同 IP 地址，但以上的例子永远是可行的。

最后的参数(0.0.0.0)告诉 PPP 从 0.0.0.0 而不是 10.0.0.1 开始商定地址，由于有些ISP，这是必需的。不要将 0.0.0.0 作为 set ifaddr 的第一个参数，因为这使得 PPP 在 `-auto` 模式不能设置初始路由。

如果不运行 `-auto` 模式，就需要在 `/etc/ppp/ppp.linkup` 中建立一个。连接建立之后，ppp.linkup被调用。时候，ppp将指派接口地址，接着再添加路由表：

```
1      provider:
2      add default HISADDR
```

行 1：

为了建立连接，ppp 会按照如下在 ppp.linkup：首先，相同的 (如同在ppp.conf)。如果失败了，则作为网 IP 地址的，此是四个八位字的格式。如果依旧没有得到，就 MYADDR

行 2：

行告诉ppp添加指向 HISADDR的默认路由。HISADDR由通过IPCP商得到的IP号替。

参考 `/usr/shared/examples/ppp/ppp.conf.sample` 和 `/usr/shared/examples/ppp/ppp.linkup.sample` 中的 `pmdemand` 以取个性化的例子。

#### 28.2.1.2.3. 接收输入

当要配置ppp接受来自LAN上的输入，需要决定是否将包到LAN。如果是的，就必须从 LAN 子网中地方分配一个IP，需要在文件 `/etc/ppp/ppp.conf` 中使用命令 `enable proxy`。指定文件 `/etc/rc.conf` 中包含以下内容：

```
gateway_enable="YES"
```

#### 28.2.1.2.4. 使用 `getty` ？

配置 FreeBSD 的 `号服` 描述了如何用 `getty(8)` 来 `号服`。

除了 `getty` 之外还有 `mgetty` (可通过 `comms/mgetty+sendfax` port 来安装)，它是 `getty` 的智能版本，是按照 `号` 的思想的。

使用 `mgetty` 的好处是它能 `地` 与 modem `行 会`， `就意味着` 如果在 `/etc/ttys` 中的端口被 `，` 的 modem 就不会回 `入`。

`新版本的 mgetty` (从 0.99beta 起) 也支持自 `PPP` 数据流， `即便客 端不使用脚本也能 服 器了`。

参考 `Mgetty` 和 `AutoPPP` 的 `机手册` 了解更多信息。

#### 28.2.1.2.5. `PPP` 限

`ppp` 命令通常必 `以 root` 用 `的身 行`。 如果希望以普通用 `的身` `ppp` 服 (就像下面描述的那)， 就必 `把此用 加入 network`， 使其 `得 行 ppp` 的 `限`。

`需要使用 allow` 命令使用 `能 配置文 件的一个或多个部分`：

```
allow users fred mary
```

如果 `个命令` 被用在 `default` 部分中， `可以 指定的用 任何 西`。

#### 28.2.1.2.6. `IP` 用 `的 PPP Shell`

`建一个名 /etc/ppp/ppp-shell` 文件， 加入以下内容：

```
#!/bin/sh
IDENT=`echo $0 | sed -e 's/^.*-\(.*\)$/\1/'`
CALLEDAS="$IDENT"
TTY=`tty`

if [ x$IDENT = xdialup ]; then
    IDENT=`basename $TTY`
fi

echo "PPP for $CALLEDAS on $TTY"
echo "Starting PPP for $IDENT"

exec /usr/sbin/ppp -direct $IDENT
```

`个脚本` 要有可 `行属性`。 然后通 `如下命令` `建一个指向此脚本且名 ppp-dialup` 的符号 `接`：

```
# ln -s ppp-shell /etc/ppp/ppp-dialup
```

`将 个脚本` 作 `所有 入用 的 shell`。 以下是在文件 `/etc/passwd` 中 `于 PPP` 用 `pchil ds` 的例子 (切，

不要直接修改那个密码文件，用 [vipw\(8\)](#) 来修改它)。

```
pchlds:*:1011:300:Peter Childs PPP:/home/ppp:/etc/ppp/ppp-dialup
```

创建一个名为 `/home/ppp` 的目录作为入用的主目录，其中包含以下一些空文件：

```
-r--r--r-- 1 root wheel 0 May 27 02:23 .hushlogin
-r--r--r-- 1 root wheel 0 May 27 02:22 .rhosts
```

这样就可以防止 `/etc/motd` 被显示出来。

#### 28.2.1.2.7. 静态IP用的Shell

像上面那样创建 `ppp-shell` 文件，每个静态分配IP用创建一个到 `ppp-shell` 的符号链接。

例如，如果你希望三个号码，`fred`，`sam`，和 `mary` 路由 /24 CIDR 的网络，你需要输入以下内容：

```
# ln -s /etc/ppp/ppp-shell /etc/ppp/ppp-fred
# ln -s /etc/ppp/ppp-shell /etc/ppp/ppp-sam
# ln -s /etc/ppp/ppp-shell /etc/ppp/ppp-mary
```

每个用的Shell必须被链接成一个符号链接(例如用 `mary` 的Shell是 `/etc/ppp/ppp-mary`)。

#### 28.2.1.2.8. 静态IP用配置 `ppp.conf`

`/etc/ppp/ppp.conf` 文件包含下面一些行：

```
default:
    set debug phase lcp chat
    set timeout 0

ttyu0:
    set ifaddr 203.14.100.1 203.14.100.20 255.255.255.255
    enable proxy

ttyu1:
    set ifaddr 203.14.100.1 203.14.100.21 255.255.255.255
    enable proxy
```



必须。

`default:` 在每次都会加。它在 `/etc/ttys` 中用的行都必须其建立一个类似于 `ttyu0:` 的。一行从IP地址池中取得唯一的IP地址。

#### 28.2.1.2.9. 静音 IP 用配置 ppp.conf

根据上面 /usr/shared/examples/ppp/ppp.conf 文件的内容，必须给每个静音号用添加一个。我可以以 fred、sam 以及 mary 为例。

```
fred:
  set ifaddr 203.14.100.1 203.14.101.1 255.255.255.255

sam:
  set ifaddr 203.14.100.1 203.14.102.1 255.255.255.255

mary:
  set ifaddr 203.14.100.1 203.14.103.1 255.255.255.255
```

如果需要，/etc/ppp/ppp.linkup 也包含给静音 IP 用的路由信息。下面一行客户直接添加到了 203.14.101.0/24 网的路由。

```
fred:
  add 203.14.101.0 netmask 255.255.255.0 HISADDR

sam:
  add 203.14.102.0 netmask 255.255.255.0 HISADDR

mary:
  add 203.14.103.0 netmask 255.255.255.0 HISADDR
```

#### 28.2.1.2.10. mgetty 和 AutoPPP

默认情况下，comms/mgetty+sendfax port 在用了 AUTO\_PPP 后，它使 mgetty 能进入 PPP 连接的 LCP 状态，并自动生成 PPP shell。不过，由于在默认配置中的 login/password 序列并不输出，因此，就必须使用 PAP 或 CHAP 来重用自身。

假定你已经在此系统中成功地并安装了 comms/mgetty+sendfax。

它的 /usr/local/etc/mgetty+sendfax/login.config 文件中包含以下内容：

```
/AutoPPP/ - - /etc/ppp/ppp-pap-dialup
```

执行 mgetty 运行 ppp-pap-dialup 脚本来听 PPP 连接。

建/etc/ppp/ppp-pap-dialup 文件写入以下内容 (此文件是可执行的)：

```
#!/bin/sh
exec /usr/sbin/ppp -direct pap$IDENT
```

由于每个在/etc/ttys 的引用行，都要在/etc/ppp/ppp.conf 中建立相应的。和上面的定义是相同的。

```
pap:
  enable pap
  set ifaddr 203.14.100.1 203.14.100.20-203.14.100.40
  enable proxy
```

每个以该方式登录的用户，都必须在 `/etc/ppp/ppp.secret` 文件中给出用户名/口令，或者使用以下命令，来通过 PAP 方式以 `/etc/passwd` 文件提供的信息来完成身份验证。

```
enable passwdauth
```

如果想给某些用户分配静态 IP，可以在 `/etc/ppp/ppp.secret` 中将 IP 号作为第三个参数指定。请参考 `/usr/shared/examples/ppp/ppp.secret.sample` 中的例子。

#### 28.2.1.2.11. MS Extensions

可以配置 PPP 以提供 DNS 和 NetBIOS 域名服务器地址。

要在 PPP 1.x 版本中使用这些扩展，需要在 `/etc/ppp/ppp.conf` 的 `init` 中加入下列配置：

```
enable msext
set ns 203.14.100.1 203.14.100.2
set nbns 203.14.100.5
```

PPP 版本 2 及以上：

```
accept dns
set dns 203.14.100.1 203.14.100.2
set nbns 203.14.100.5
```

将告诉客户端首台域名服务器和备用域名服务器。

在版本 2 及以上版本中，如果省略了 `set dns`，PPP 会使用 `/etc/resolv.conf` 中的配置。

#### 28.2.1.2.12. PAP 和 CHAP 对比

一些 ISP 将系统配置成使用 PAP 或 CHAP 机制来完成连接。如果遇到这种情况，在连接 ISP 就不会看到 **login:** 提示符，而是立即开始 PPP 连接。

PAP 安全性要比 CHAP 差一些，但在串口安全性并不是问题，因为密码（即使用明文发送）只是通过串行线发送，攻击者并没有太多机会去“窃听”它。

参考 [PPP 与静态 IP 地址](#) 或 [PPP 与动态 IP 地址](#) 小节，并完成下列改动：

```
13      set authname MyUserName
14      set authkey MyPassword
15      set login
```

第 13 行：

一行指明 PAP/CHAP 用户名。需要 `MyUserName` 输入正值的。

第 14 行：

一行指明 PAP/CHAP password 密码。需要 `MyPassword` 输入正值的。此外，可能希望加入一些额外的，例如：

```
16      accept PAP
```

或

```
16      accept CHAP
```

以明确的意思，不，默认情况下 PAP 和 CHAP 都会被接受。

行 15：

如果使用的是 PAP 或 CHAP，一般来 ISP 就不会要求登录服务器了。因此，就必须禁用 "set login" 配置。

#### 28.2.1.2.13. 即修改的 ppp 配置

与后台运行的 ppp 程序运行是可能的，前提是配置了一个合适的断端口。做到这一点，需要把下面的行加入到配置中：

```
set server /var/run/ppp-tun%d DiagnosticPassword 0177
```

行告诉 PPP 在指定的 UNIX 域 socket 中监听，当用连接需要输出指定的密码。%d 用 tun 号替换。

一旦用了 socket，就可以在脚本中用程序 [pppctl\(8\)](#) 来管理正在运行的 PPP。

#### 28.2.1.3. 使用 PPP 网地址翻

PPP 可以使用内建的 NAT，而无需内核支持。可以在 `/etc/ppp/ppp.conf` 中加入如下配置来用它：

```
nat enable yes
```

PPP NAT 也可以使用命令行 `-nat`。在 `/etc/rc.conf` 文件中也有 `ppp_nat`，并默认用。

如果使用了个特性，会等在 `/etc/ppp/ppp.conf` 中以下用于 incoming connections forwarding 是有用的：

```
nat port tcp 10.0.0.2:ftp ftp
nat port tcp 10.0.0.2:http http
```

或者完全不信任外来的请求

```
nat deny_incoming yes
```

#### 28.2.1.4. 最后的系统配置

在已配置了 `ppp`，但在真正工作之前有一些事情要做。即修改 `/etc/rc.conf`。

从上依次往下看，已正确地配置了 `hostname=`，例如：

```
hostname="foo.example.com"
```

如果你的ISP提供一个静态的IP和名字，将这个名字 `hostname` 是最合适的。

`network_interfaces` 变量。如果要配置系统通过号接入ISP，一定要将 `tun0` 加入个列表，否则就删除它。

```
network_interfaces="lo0 tun0"
ifconfig_tun0=
```

`ifconfig_tun0` 变量是空的，且要建立一个名 `/etc/start_if.tun0` 的文件。这个文件包含一行：



```
ppp -auto mysystem
```

此脚本在网络配置被执行时，启动PPP守护进程进入自模式。如果这台机器充当一个LAN的网络，可能希望使用 `-alias`。参考相关手册了解更多。

必在 `/etc/rc.conf` 中，把路由程序置 `NO`：

```
router_enable="NO"
```

不 `routed` 服务程序非常重要，因为 `routed` 会掉由 `ppp` 所建立的默认路由。

此外，我建议检查一下 `sendmail_flags` 一行中没有指定 `-q` 参数，否则 `sendmail` 将会不断地扫描网，而做将会致机器不断地运行号。可以考：

```
sendmail_flags="-bd"
```

替代的做法是当次 PPP 连接建立时必须通过以下命令制 `sendmail` 重新组件列：



```
# /usr/sbin/sendmail -q
```

也可以在ppp.linkup使用**!bg**命令自完成些工作：

```
1 provider:
2 delete ALL
3 add 0 0 HISADDR
4 !bg sendmail -bd -q30m
```

如果不喜做，可以立一个 "dfilter" 以阻止 SMTP 。参考相文件了解更多。

在唯一要做的事是重新算机。重之后，可以入：

```
# ppp
```

然后是**dial provider**以 PPP会。 或者如果想**ppp**自建立会， 因有一条广域网接（且没有建start\_if.tun0 脚本），入：

```
# ppp -auto provider
```

#### 28.2.1.5.

当第一次置PPP，下面几是必的：

客端：

1. 保 tun了核。
2. 保 /dev 目中名 tunN 的文件是可用的。
3. 在 /etc/ppp/ppp.conf中建一个。 pmdemand示例合于大多数ISP。
4. 如果使用IP地址，在/etc/ppp/ppp.linkup建一个。
5. 更新/etc/rc.conf 文件。
6. 如果要求按需号，建一个start\_if.tun0脚本。

服器端：

1. 确保tun已调入内核。
2. 确保 /dev 目录中名为 tunN 的文件是可用的。
3. 在/etc/passwd中建立一个 (使用vipw(8)程序)。
4. 在用户的home目录建立一个行 `ppp -direct direct-server` 或相似命令的profile。
5. 在/etc/ppp/ppp.conf中建立一个。direct-server示例可能满足要求。
6. 在 /etc/ppp/ppp.linkup中建立一个。
7. 更新 /etc/rc.conf 文件。

## 28.3. 使用内核PPP



该内容只在 FreeBSD 7.X 上可用。

### 28.3.1. 建立内核PPP

在开始配置 PPP 之前，确保 `pppd` 已存放在 `/usr/sbin` 中，并且 `/etc/ppp` 目录是存在的。

`pppd` 能在以下模式下工作：

1. 作为一个 "客" - 要通过PPP串行或modem把您的机器连接到互联网上。
2. 作为 "服务器" - 计算机已位于网络上，且被用于通过PPP与其它计算机连接。

两种情况都需要建立一个配置文件，(`/etc/ppp/options` 或者是 `~/ppprc` 如果您的计算机有多个用户使用PPP)。

您可能需要一些modem/serial硬件([comms/kermit](#)就很合适)，使您能拨号并与远程主机建立连接。

### 28.3.2. 使用pppd作为客户端

下面是一个 `/etc/ppp/options` 文件可能被用来与CISCO端服务器的PPP连接。

```
crtstcts      # enable hardware flow control
modem         # modem control line
noipdefault   # remote PPP server must supply your IP address
              # if the remote host does not send your IP during IPCP
              # negotiation, remove this option
passive       # wait for LCP packets
domain ppp.foo.com  # put your domain name here

:remote_ip    # put the IP of remote PPP host here
              # it will be used to route packets via PPP link
              # if you didn't specified the noipdefault option
              # change this line to local_ip:remote_ip

defaultroute  # put this if you want that PPP server will be your
              # default router
```

□接：

1. 使用 Kermit (或其他 modem 程序来□号), 然后□入□的用□名和□令 (或在□程主机上□用 PPP 所需的其他信息)。
2. 退出 Kermit (并不挂断□接)。
3. □入下面□行：

```
# /usr/sbin/pppd /dev/tty01 19200
```

一定要使用正□的速度和□□名。

□在□的□算机已□用PPP□接。 如果□接失□, □可在文件 `/etc/ppp/options` 中添加 `debug` □□, 并□看控制台信息以跟踪□□。

下面□个/etc/ppp/pppup脚本能自□完成□三个□□：

```
#!/bin/sh
pgrep -l pppd
pid=`pgrep pppd`
if [ "X${pid}" != "X" ] ; then
    echo 'killing pppd, PID=' ${pid}
    kill ${pid}
fi
pgrep -l kermit
pid=`pgrep kermit`
if [ "X${pid}" != "X" ] ; then
    echo 'killing kermit, PID=' ${pid}
    kill -9 ${pid}
fi

ifconfig ppp0 down
ifconfig ppp0 delete

kermit -y /etc/ppp/kermit.dial
pppd /dev/tty01 19200
```

`/etc/ppp/kermit.dial` 是一个 Kermit 脚本, 它会完成□号, 并在□程主机上完成所有需要的身□□□□程 (□□文□的最后有一个脚本□例)。

使用下面□个脚本/etc/ppp/pppdown断□PPP□□：

```
#!/bin/sh
pid=`pgrep pppd`
if [ X${pid} != "X" ] ; then
    echo 'killing pppd, PID=' ${pid}
    kill -TERM ${pid}
fi

pgrep -l kermi
pid=`pgrep kermi`
if [ "X${pid}" != "X" ] ; then
    echo 'killing kermi, PID=' ${pid}
    kill -9 ${pid}
fi

/sbin/ifconfig ppp0 down
/sbin/ifconfig ppp0 delete
kermi -y /etc/ppp/kermi.hup
/etc/ppp/ppptest
```

通□行/usr/etc/ppp/ppptest, 看看pppd 是否仍在□行：

```
#!/bin/sh
pid=`pgrep pppd`
if [ X${pid} != "X" ] ; then
    echo 'pppd running: PID=' ${pid-NONE}
else
    echo 'No pppd running.'
fi
set -x
netstat -n -I ppp0
ifconfig ppp0
```

□行脚本 /etc/ppp/kermi.hup以挂起modem, □个文件包含：

```

set line /dev/tty01 ; put your modem device here
set speed 19200
set file type binary
set file names literal
set win 8
set rec pack 1024
set send pack 1024
set block 3
set term bytesize 8
set command bytesize 8
set flow none

pau 1
out +++
inp 5 OK
out ATH0\13
echo \13
exit

```

也可以用`chat`代替`kermit`：

以下两个文件用以建立`pppd`连接。

`/etc/ppp/options`：

```

/dev/cuad1 115200

rtscts      # enable hardware flow control
modem       # modem control line
connect "/usr/bin/chat -f /etc/ppp/login.chat.script"
noipdefault # remote PPP serve must supply your IP address
             # if the remote host doesn't send your IP during
             # IPCP negotiation, remove this option
passive     # wait for LCP packets
domain your.domain # put your domain name here

:           # put the IP of remote PPP host here
             # it will be used to route packets via PPP link
             # if you didn't specified the noipdefault option
             # change this line to local_ip:remote_ip

defaultroute # put this if you want that PPP server will be
             # your default router

```

`/etc/ppp/login.chat.script`：



以下的内容放在一行内。

```
ABORT BUSY ABORT 'NO CARRIER' "" AT OK ATDTphone.number
CONNECT "" TIMEOUT 10 ogin:-\\r-ogin: login-id
TIMEOUT 5 sword: password
```

一旦它被安装且修改正确，它所要做的就是运行pppd，就像：

```
# pppd
```

### 28.3.3. 使用pppd作服务器

/etc/ppp/options要包括下面些内容：

```
crtcts                # Hardware flow control
netmask 255.255.255.0  # netmask (not required)
192.114.208.20:192.114.208.165 # IP's of local and remote hosts
                                # local ip must be different from one
                                # you assigned to the Ethernet (or other)
                                # interface on your machine.
                                # remote IP is IP address that will be
                                # assigned to the remote machine
domain ppp.foo.com     # your domain
passive                # wait for LCP
modem                  # modem line
```

下面个脚本/etc/ppp/pppserv 使pppd以服务器方式：

```
#!/bin/sh
pgrep -l pppd
pid=`pgrep pppd`
if [ "X${pid}" != "X" ] ; then
    echo 'killing pppd, PID=' ${pid}
    kill ${pid}
fi
pgrep -l kermit
pid=`pgrep kermit`
if [ "X${pid}" != "X" ] ; then
    echo 'killing kermit, PID=' ${pid}
    kill -9 ${pid}
fi

# reset ppp interface
ifconfig ppp0 down
ifconfig ppp0 delete

# enable autoanswer mode
kermit -y /etc/ppp/kermit.ans

# run ppp
pppd /dev/tty01 19200
```

使用脚本/etc/ppp/pppservdown停止服务器：

```
#!/bin/sh
pgrep -l pppd
pid=`pgrep pppd`
if [ "X${pid}" != "X" ] ; then
    echo 'killing pppd, PID=' ${pid}
    kill ${pid}
fi
pgrep -l kermit
pid=`pgrep kermit`
if [ "X${pid}" != "X" ] ; then
    echo 'killing kermit, PID=' ${pid}
    kill -9 ${pid}
fi
ifconfig ppp0 down
ifconfig ppp0 delete

kermit -y /etc/ppp/kermit.noans
```

下面的 Kermit 脚本 (/etc/ppp/kermit.ans) 能启用/禁用 modem 的自应答模式。其内容似下面：

```

set line /dev/tty01
set speed 19200
set file type binary
set file names literal
set win 8
set rec pack 1024
set send pack 1024
set block 3
set term bytesize 8
set command bytesize 8
set flow none

pau 1
out +++
inp 5 OK
out ATH0\13
inp 5 OK
echo \13
out ATS0=1\13 ; change this to out ATS0=0\13 if you want to disable
                ; autoanswer mode

inp 5 OK
echo \13
exit

```

一个名`/etc/ppp/kermidial`的脚本用于向程主机 行号和。 要根据需要定制它。 要加入的登  
名和密， 要根据 modem 和程主机的反修改入句。

```

;
; put the com line attached to the modem here:
;
set line /dev/tty01
;
; put the modem speed here:
;
set speed 19200
set file type binary ; full 8 bit file xfer
set file names literal
set win 8
set rec pack 1024
set send pack 1024
set block 3
set term bytesize 8
set command bytesize 8
set flow none
set modem hayes
set dial hangup off
set carrier auto ; Then SET CARRIER if necessary,
set dial display on ; Then SET DIAL if necessary,
set input echo on

```



```

set input timeout proceed
set input case ignore
def \%x 0                                ; login prompt counter
goto slhup

:slcmd                                    ; put the modem in command mode
echo Put the modem in command mode.
clear                                    ; Clear unread characters from input buffer
pause 1
output +++                              ; hayes escape sequence
input 1 OK\13\10                          ; wait for OK
if success goto slhup
output \13
pause 1
output at\13
input 1 OK\13\10
if fail goto slcmd                        ; if modem doesn't answer OK, try again

:slhup                                    ; hang up the phone
clear                                    ; Clear unread characters from input buffer
pause 1
echo Hanging up the phone.
output ath0\13                            ; hayes command for on hook
input 2 OK\13\10
if fail goto slcmd                        ; if no OK answer, put modem in command mode

:sldial                                    ; dial the number
pause 1
echo Dialing.
output atdt9,550311\13\10                 ; put phone number here
assign \%x 0                             ; zero the time counter

:look
clear                                    ; Clear unread characters from input buffer
increment \%x                             ; Count the seconds
input 1 {CONNECT }
if success goto sllogin
reinput 1 {NO CARRIER\13\10}
if success goto sldial
reinput 1 {NO DIALTONE\13\10}
if success goto slnodial
reinput 1 {\255}
if success goto slhup
reinput 1 {\127}
if success goto slhup
if < \%x 60 goto look
else goto slhup

:sllogin                                  ; login
assign \%x 0                             ; zero the time counter
pause 1

```

```

echo Looking for login prompt.

:sloop
increment \%x                ; Count the seconds
clear                        ; Clear unread characters from input buffer
output \13
;
; put your expected login prompt here:
;
input 1 {Username: }
if success goto sluid
reinput 1 {\255}
if success goto slhup
reinput 1 {\127}
if success goto slhup
if < \%x 10 goto slloop      ; try 10 times to get a login prompt
else goto slhup              ; hang up and start again if 10 failures

:sluid
;
; put your userid here:
;
output ppp-login\13
input 1 {Password: }
;
; put your password here:
;
output ppp-password\13
input 1 {Entering SLIP mode.}
echo
quit

:slnodial
echo \7No dialtone. Check the telephone line!\7
exit 1

; local variables:
; mode: csh
; comment-start: "; "
; comment-start-skip: "; "
; end:

```

## 28.4. PPP 连接故障排除



从 FreeBSD 8.0 开始, `uart(4)` 取代了 `sio(4)`。用以表示串口的点由分 `/dev/cuadN` 改为了 `/dev/cuauN`, 并从 `/dev/ttydN` 改为了 `/dev/ttyuN`。FreeBSD 7.X 用户在升级时需要因之配置文件行必要的更改。

本文将描述通过 modem 连接使用 PPP 可能出现的。

例如, 您可能需要确切地知道输入的系会出一个

的命令行提示符。有些 ISP 会提供 `ssword` 提示符，而其它的可能会出 `password`；如果没有根据情况的不同相应地写 `ppp` 脚本，登录就会失败。中断 `ppp` 最常用的方法是手动行连接。以下的信息会一步步地完成手动连接。

### 28.4.1. 配置点

如果使用的是定制内核，在其配置中包含下列配置：

```
device    uart
```

默认的 `GENERIC` 内核中包含了 `uart`，因此如果你使用的是它的，就不需要担心了。只要看 `dmesg` 输出中是否有 `modem`：

```
# dmesg | grep uart
```

看到与 `uart` 有关的输出。一些就是我需要的 `COM` 端口。如果你的 `modem` 按照标准串行端口工作，就会在 `uart1` 或 `COM2` 上找到它。如果 `modem` 直接接在 `uart1` 接口（在 `DOS` 中称 `COM2`），那你的 `modem` 将会是 `/dev/cuau1`。

### 28.4.2. 手动连接

通过手动控制 `ppp` 来连接 Internet 是中断连接及通知 ISP 处理 PPP 客户端方式的一个快速，简单的方法。我从 PPP 命令行开始，在所有的例子中我使用 `example` 表示运行 PPP 服务的主机名。输入 `ppp` 命令打 `ppp`：

```
# ppp
```

在我已打了 `ppp`。

```
ppp ON example> set device /dev/cuau1
```

设置 `modem`，在本例子中是 `cuau1`。

```
ppp ON example> set speed 115200
```

设置连接速度，在本例中我使用 15,200 kbps。

```
ppp ON example> enable dns
```

使 `ppp` 配置域名服务，在文件 `/etc/resolv.conf` 中添加域名服务器行。如果 `ppp` 不能确定我的主机名，可以在后面设置。

```
ppp ON example> term
```

切换到“终端”我就能手动地控制台 modem 的模式。

```
deflink: Entering terminal mode on /dev/cuau1
type '~h' for help
```

```
at
OK
atdt123456789
```

使用命令 **at** 初始化 modem，然后使用 **atdt** 和 ISP 的号码行号。

```
CONNECT
```

连接配置，如果我遇到了与硬件无关的连接问题，可以在这里解决。

```
ISP Login:myusername
```

这里提示输入用户名，输入 ISP 提供的用户名然后按回车。

```
ISP Pass:mypassword
```

提示我输入密码，输入 ISP 提供的密码。如同登录 FreeBSD，密码不会显示。

```
Shell or PPP:ppp
```

由于 ISP 的不同，这个提示符可能不会出现。这里我需要考：是使用行于提供商端的 Shell，还是 **ppp**？本例中，我使用 **ppp**，因为我希望得到 Internet 连接。

```
Ppp ON example>
```

注意在这个例子中，第一个 **p** 已大写。表示我已成功地连接上了 ISP。

```
PPp ON example>
```

我已成功通了 ISP 的，正在等待分配 IP 地址。

```
PPP ON example>
```

我得到了一个 IP 地址，成功地完成了连接。

```
PPP ON example>add default HISADDR
```

这样就完成了添加默认路由所需的配置。这是与外界通信所必需的。因为之前我们只是与服务端建立了连接。如果由于已存在的路由而导致操作失败，可以在 `add` 前加 `!` 号。除此之外，也可以在真正连接之前设置一些（指 `add default HISADDR`），`ppp` 会根据约定商定取得新的路由。

如果一切顺利，在我们这里能得到一个活动的 Internet 连接，可以使用 `CTRL + Z` 使其进入后台。如果 `PPP` 重新设置 `ppp`，表示连接被断开。大写的 `P` 表明建立了到 ISP 的连接，而小写的 `p` 表示连接由于某种原因被断开，有助于帮助我们了解连接的状态。`ppp` 只有两个状态。

#### 28.4.2.1. 断排

如果有一根直线且似乎不能建立连接，要使用 `set ctsrts off` 以关闭字流的 CTS/RTS。这种情况一般发生在连接兼容 PPP 的终端服务器。当它向通信连接写入数据，PPP 就会挂起，一直等待一个 CTS，或者一个不可能出现的 Clear to Send 信号。如果使用了这个，使用 `set accmap`，某些存在缺陷的硬件在完成端端发送特定字符，特别是 XON/XOFF 可能会遇到困难。参看 [ppp\(8\)](#) 手册以了解关于可用选项的更多选项，以及如何使用它们。

如果的 modem 比旧，就需要使用 `set parity even` 了。奇偶校验的默认设置是 `none`，但在旧式的（当流量大量增加）调制解调器和某些 ISP 被用来。需要使用这个才能使用 Compuserve ISP。

PPP 可能并不返回命令模式，通常是 ISP 等待一端启动另一端生成了。此时，使用 `~p` 命令将控制 `ppp` 开始发送配置信息。

如果没有看到登录提示，很可能需要使用 PAP 或 CHAP 来代替前面例子中的 UNIX® 风格。要使用 PAP 或 CHAP 只需在进入终端模式之前把下面的加入 PPP：

```
ppp ON example> set authname myusername
```

此 `myusername` 更改的 ISP 分配的用户名。

```
ppp ON example> set authkey mypassword
```

此 `mypassword` 的 ISP 分配的口令。

如果连接正常，但无法访问域名，看看 [ping\(8\)](#) 某个 IP 地址来看看是否返回了信息。如果百分之百（100%）包，那很可能没有分配默认路由。仔细看看 `add default HISADDR` 是否在连接被设置了。如果能连接到程序的 IP 地址有可能域名解析服务器的地址没有被加入到 `/etc/resolv.conf`。这个文件是下面的样子：

```
domain example.com
nameserver x.x.x.x
nameserver y.y.y.y
```

此 `x.x.x.x` 和 `y.y.y.y` 更改的 ISP 的 DNS 服务器的 IP 地址。一信息在注册可能会提供，不过通常只需 ISP 打个电话就能知道了。

可以配置 `syslog(3)` 的 PPP 连接提供日志。只需添加：

```
!ppp
*. * /var/log/ppp.log
```

到 `/etc/syslog.conf` 中。大多数情况下，该功能默认已打开了。

## 28.5. 使用基于以太网的PPP(PPPoE)

本文将介绍如何建立基于以太网的PPP (PPPoE)。

### 28.5.1. 配置内核

对于PPPOE，并没有必需的内核配置。如果必需的 netgraph 支持没有引入内核，它可以由 `ppp` 添加。

### 28.5.2. 配置ppp.conf

以下是一个 `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:
    set device PPPoE:x11 # replace x11 with your Ethernet device
    set authname YOURLOGINNAME
    set authkey YOURPASSWORD
    set dial
    set login
    add default HISADDR
```

### 28.5.3. 运行ppp

以 `root` 身份运行：

```
# ppp -ddial name_of_service_provider
```

### 28.5.4. 配置运行ppp

在 `/etc/rc.conf` 中加入以下内容：

```
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"
```

### 28.5.5. 使用 PPPoE 服务

在某些时候，有必要使用一个服务来建立连接。服务用于区分同一网中的不同服务器。

可以在ISP提供的文档中找到必要的服务信息。若不能找到，可向ISP寻求技术支持。

作为最后的方法，可以看看 [Roaring Penguin PPPoE](#)，它可以在 [Ports Collection](#) 中找到。然而需要注意的是，它可能会清除 modem 的固件，并使其无法正常工作，因此一定要仔细考虑之后再做一个操作。正确地安装由服务提供商随 modem 提供的程序。随后，在 **System** 菜单。它的配置文件会在里面列出。一般来说它的名字是 *ISP*。

配置文件名 (service tag, 服务) 将被用于 PPPoE 在 `ppp.conf` 中的配置，作为服务商 `set device` 命令的一部分 (参看 [ppp\(8\)](#) 手册以了解更多)。它类似于下面的例子：

```
set device PPPoE:x11:ISP
```

记住将 `_x11_` 成功的以太网。

记住将 *ISP* 成功的profile名。

得更多的信息，参考：

- [Cheaper Broadband with FreeBSD on DSL](#) by Renaud Waldura.
- [Nutzung von T-DSL und T-Online mit FreeBSD](#) by Udo Erdelhoff (in German).

### 28.5.6. 我有一个3Com® HomeConnect™ ADSL Modem的PPPOE双重连接

这个 modem 不遵循 [RFC 2516](#) (*A Method for transmitting PPP over Ethernet (PPPoE)*，其作者 L. Mamakos、K. Lidl、J. Evarts、D. Carrel、D. Simone 以及 R. Wheeler)。而是使用不同的数据包格式作为以太网的框架。我向 [3Com](#) 抱怨，如果它能遵守 PPPoE 的。

为了FreeBSD能与这个通信，必须设置sysctl。通过更改/etc/sysctl.conf，它可以在启动时完成：

```
net.graph.nonstandard_pppoe=1
```

或者，也可以直接运行下面的命令：

```
# sysctl net.graph.nonstandard_pppoe=1
```

很不幸，由于它是系统全局设置，无法同时与正常的PPP客户端(或服务)和3Com®HomeConnect™ ADSL Modem通信。

## 28.6. 使用 ATM 上的 PPP (PPPoA)

以下将介绍如何配置基于ATM的PPP(PPPoA)。PPPoA是欧洲DSL提供商的普遍。

### 28.6.1. 使用 Alcatel SpeedTouch™ USB 的 PPPoA

目前 FreeBSD 的 PPPoA 支持，在 FreeBSD 中是由 `port` 提供的，因其固件使用了 [Alcatel SpeedTouch](#)，因而不能与 FreeBSD 的基本系统一起免安装地再部署。

使用 [Ports 套件](#) 可以非常方便地安装 `net/pppoe` port，之后按照它提供的指示操作就可以了。

和许多 USB 设备类似，Alcatel 的 SpeedTouch™ USB 需要从主机上下固件才能正常工作。在 FreeBSD 中可以将此操作自动化，在有空闲到某个 USB 接口的时候自动下固件。可以在 `/etc/usbd.conf` 文件中加入下面的信息来让它自动完成固件的推送。注意，必须以 `root` 用户的身临其境。

```
device "Alcatel SpeedTouch USB"
    devname "ugen[0-9]+"
    vendor 0x06b9
    product 0x4061
    attach "/usr/local/sbin/modem_run -f /usr/local/libdata/mgmt.o"
```

要启用 USB 守护进程 `usbd`，在 `/etc/rc.conf` 加入以下行：

```
usbd_enable="YES"
```

也可以将 `ppp` 置成 `enable`。向 `/etc/rc.conf` 加入以下几行。同样地也需要以 `root` 用户登录。

```
ppp_enable="YES"
ppp_mode="ddial"
ppp_profile="adsl"
```

为了使其正常工作，需要使用 `net/pppoe` port 提供的 `ppp.conf` 示例。

### 28.6.2. 使用 `mpd`

可以使用 `mpd` 来连接多种类型的服务器，特别是 PPTP 服务器。可以在 Ports Collection 中找到 `mpd`，它的位置是 [net/mpd](#)。许多 ADSL modem 需要在 modem 和计算机之间建立一条 PPTP 隧道，而 Alcatel SpeedTouch™ Home 正是其中之一。

首先需要从 `port` 完成安装，然后才能配置 `mpd` 来满足的需要，并完成服务提供商的配置。`port` 会把一系列包括了注释的配置文件示例放到 `PREFIX/etc/mpd/`。注意，这里的 `PREFIX` 表示 `ports` 安装的目标，默认情况下，它是 `/usr/local/`。配置 `mpd` 的完整说明，会以 HTML 格式随 `port` 一起安装。这些文件将放在 `PREFIX/shared/doc/mpd/`。下面是通过 `mpd` 连接 ADSL 服务器的一个例子。配置被分放到了两个文件中，第一个是 `mpd.conf`：



```

default:
    load adsl

adsl:
    new -i ng0 adsl adsl
    set bundle authname username ①
    set bundle password password ②
    set bundle disable multilink

    set link no pap acfcomp protocomp
    set link disable chap
    set link accept chap
    set link keep-alive 30 10

    set ipcp no vjcomp
    set ipcp ranges 0.0.0.0/0 0.0.0.0/0

    set iface route default
    set iface disable on-demand
    set iface enable proxy-arp
    set iface idle 0

    open

```

① username用来向ISP行。

② password用来向ISP行。

mpd.links包含接的信息：

```

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 行mpd的主机的IP地址。

② ADSL modem的IP地址。Alcatel SpeedTouch™ Home 默的是 10.0.0.138。

初始化接：

```
# mpd -b adsl
```

可以通以下命令看接状：

```
% 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连接ADSL服务是推荐的方式。

### 28.6.3. 使用pptpclient

也可以使用net/pptpclient连接其它的 PPPoA。

要使用 net/pptpclient 连接 DSL 服务，需要安装 port 或 package 并编辑 /etc/ppp/ppp.conf。需要有 root 权限才能完成这些操作。以下是 ppp.conf 中的一个示例。参考 ppp 的用户手册 pptp(8)，以了解更多有关 ppp.conf 的信息。

```
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
```

① 在 DSL 服务提供商那里的用户名

② 密码的口令。



由于必须将账号密码以明文的方式放入 ppp.conf，确保没有任何人能看到此文件的内容。以下一系列命令将会确保此文件只被 root 用户可访问。参考 chmod(1) 和 chown(8) 的用户手册以了解有关如何操作的一些信息。

```
# chown root:wheel /etc/ppp/ppp.conf
# chmod 600 /etc/ppp/ppp.conf
```

以下将连接到 DSL 路由器的会打一个 tunnel。以太网DSL modem有一个设置的局域网IP地址。以 Alcatel SpeedTouch™ Home 为例，这个地址是 10.0.0.138。路由器的文档会告诉它使用的地址。运行以下命令以打 tunnel 并始会：

```
# pptp address adsl
```



请在命令的最后加上("&")号，否则 pptp 无法返回到命令行提示符。

要建立一个 tun 虚拟用于程序 pptp 和 ppp 之间的交互。一旦回到了命令行，或者 pptp 程序开了一个连接，可以看看 tunnel：

```
% ifconfig tun0
tun0: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST> mtu 1500
    inet 216.136.204.21 --> 204.152.186.171 netmask 0xffffffff00
    Opened by PID 918
```

如果无法连接，一般可以通过telnet或者web浏览器配置路由器(modem)的配置。如果依旧无法连接，可以查看pptp的输出及ppp的日志文件 /var/log/ppp.log 以得到线索。

## 28.7. 使用SLIP



该内容只在 FreeBSD 7.X 上可用。

### 28.7.1. 配置 SLIP 客户端

下面是在静主机网上配置 FreeBSD 机器使用 SLIP 的方法。由于主机名分配 (IP 的地址会随接口号而不同)，可能需要做一些配置。

首先，需要调制解调器所连接的串口。多人会设置一个符号链接，例如 /dev/modem，用以指向实际的设备名，如 /dev/cuadN。这样就可以对设备名进行抽象，以调制解调器到其他串口方便调整之用。不然，修改 /etc 和遍布于系统中的 .kermrc 文件将是一件很麻烦的事情！



/dev/cuad0 是 COM1，而 /dev/cuad1 是 COM2，等等。

保留的内核文件包含以下内容：

```
device    sl
```

包含在GENERIC内核，所以不会是个问题，除非已移除了它。

#### 28.7.1.1. 只需做一次的事情

1. 把本地网路上的机器、网口以及域名服务器，都加入到 /etc/hosts 文件中。我指的是下面几个子：

```
127.0.0.1          localhost loghost
136.152.64.181     water.CS.Example.EDU water.CS water
136.152.64.1       inr-3.CS.Example.EDU inr-3 slip-gateway
128.32.136.9       ns1.Example.EDU ns1
128.32.136.12      ns2.Example.EDU ns2
```

2. 保存在 /etc/nsswitch.conf 中的 **hosts:** 小口里面，**files** 先于 **dns** 出口。如果不是口的，可能会产生一些不希望的现象。

3. /etc/rc.conf。

- a. 以下行置主机名(hostname)：

```
hostname="myname.my.domain"
```

用主机的Internet全名代替。 ..

改一行以指明默认的路由：

```
defaultrouter="NO"
```

改：

```
defaultrouter="slip-gateway"
```

4. 建文件/etc/resolv.conf，写入以下内容：

```
domain CS.Example.EDU
nameserver 128.32.136.9
nameserver 128.32.136.12
```

正如看到的，些行置了域名服务器。当然，口的域名和IP地址取决于口的境。

5. 置**root**和 **toor**的密(其它任何没有密的号)。
6. 重算机，然后使用了正的主机名。

### 28.7.1.2. 建一个SLIP接

1. 在命令提示符之后输入 `slip` 后跟行号，后跟要连接的机器名和口令。具体需要输入什么，与你的环境密切相关。如果使用 Kermit，可以使用以下脚本：

```
# kermit setup
set modem hayes
set line /dev/modem
set speed 115200
set parity none
set flow rts/cts
set terminal bytesize 8
set file type binary
# The next macro will dial up and login
define slip dial 643-9600, input 10 =>, if failure stop, -
output slip\x0d, input 10 Username:, if failure stop, -
output silvia\x0d, input 10 Password:, if failure stop, -
output ***\x0d, echo \x0aCONNECTED\x0a
```

当然，你可能需要修改用名和口令来满足你的需要。完成这些操作之后，只需在 Kermit 提示符之后输入 `slip` 就可以连接了。



将密码以文本的形式存放在文件系统无论如何都是个坏主意。考虑你做的。

2. 在提示符里退出 Kermit (也可以用 `Ctrl + Z` 将其挂起)，以 `root` 用户登录：

```
# slattach -h -c -s 115200 /dev/modem
```

如果你能 ping 通路由器一端的主机，就是连接好了！如果不行，你可以使用 `-a` 代替 `-c` 作为 `slattach` 的参数。

### 28.7.1.3. 断开连接

按下面的步骤做：

```
# kill -INT `cat /var/run/slattach.modem.pid`
```

来关掉 `slattach`。上述操作只有以 `root` 身份才能完成。接下来回到 `kermit` (如果之前是将它挂起了，使用 `fg`) 并退出 (`q`)。

在 `slattach(8)` 手册中提到，必须使用 `ifconfig sl0 down` 才能将接口关闭，但和你做似乎没什么区别。( `ifconfig sl0` 仍然报告同的东西。)

有时，你的 modem 可能会拒绝挂断。这种情况下，只需重新运行 `kermit` 并再次退出它就可以了。一般来二次就可以了。

#### 28.7.1.4. 问题解答

如果不行，尽管到 [freebsd-net](#) 问题列表来提问。常见的问题包括：

- 运行 `slattach` 不使用 `-c` 和 `-a` (它们不是必须的，但有些用它们告别人做解决了)。
- 使用 `sl0` 替 `sl0` (在一些字体下很易看出不同)。
- 用 `ifconfig sl0` 来看它的接口状况。例如，它可以这样做：

```
# ifconfig sl0
sl0: flags=10<POINTOPOINT>
    inet 136.152.64.181 --> 136.152.64.1 netmask fffffff0
```

- 如果在使用 `ping(8)` 得到了 `no route to host` 的提示，表明它的路由表可能有误。可以用 `netstat -r` 命令来显示当前的路由：

```
# netstat -r
Routing tables

Destination      Gateway            Flags        Refs      Use  IfaceMTU    Rtt
Netmasks:

(root node)
(root node)

Route Tree for Protocol Family inet:
(root node) =>
default          inr-3.Example.EDU  UG           8    224515  sl0 -        -
localhost.Exampl localhost.Example. UH           5     42127  lo0 -        0.438
inr-3.Example.ED water.CS.Example.E UH           1         0  sl0 -        -
water.CS.Example localhost.Example. UGH          34  47641234  lo0 -        0.438
(root node)
```

前述的例子来自于一个非常繁忙的系统。系统上的某些数字会因网络活动的不同而改变。

### 28.7.2. 配置SLIP服务器

本文提供了在 FreeBSD 上配置 SLIP 服务，也就是如何配置它的系统，使其能在远程 SLIP 客户端登录自本地直接建立的连接。

#### 28.7.2.1. 前提条件

它技术性很强，所以要求它有一定的背景知识。本书假定熟悉 TCP/IP 网络，特别是网络和点地址、子网掩码、子网划分、路由、路由协议（如RIP）等知识。在电话服务器上配置 SLIP 需要一些概念性的知识。如果不熟悉它们，先读 Craig Hunt 的 *TCP/IP 网络管理* 由 O'Reilly & Associates, Inc. 出版 (ISBN 0-937175-82-X)，或 Douglas Comer 有 *TCP/IP* 的书。

此外假定已配置好了它的调制解调器以及相关的系统文件，以允许通过调制解调器进行登录。如果没有配置好系统，参看 [入门](#) 以了解它于如何运行电话服务器的配置。可能也会想看一看 [sio\(4\)](#) 的机手册，

以了解于串口设备的唯一信息，以及 [ttys\(5\)](#)、[gettytab\(5\)](#)、[getty\(8\)](#) & [init\(8\)](#) 上于配置系统来接受来自调制解调器的登录请求的具体情况，有 [stty\(1\)](#) 以了解于置串口参数 (例如 [clocal](#) 表示串口直连) 等。

### 28.7.2.2. 快速

使用FreeBSD作SLIP服务器，在典型配置，它是工作的：一个SLIP客户号并以用的login ID登录到FreeBSD SLIP服务器。它使用 `/usr/sbin/sliplogin` 作 shell。 `sliplogin` 程序会在文件 `/etc/sliphome/slip.hosts` 中每个用的，如果到了匹配，就将串行接到一个可用的 SLIP 接口，然后行 shell 脚本 `/etc/sliphome/slip.login` 以配置 SLIP 接口。

#### 28.7.2.2.1. 一个SLIP服务器登录的例子

例如，如果一个SLIP用的ID是 `Shelmerg`，在 `/etc/master.passwd` 中 `Shelmerg` 的如下所示：

```
Shelmerg:password:1964:89::0:0:Guy Helmer -
SLIP:/usr/users/Shelmerg:/usr/sbin/sliplogin
```

`Shelmerg` 登录， `sliplogin` 在文件 `/etc/sliphome/slip.hosts` 中搜索与用ID匹配的行;如下所示：

|          |                   |            |          |
|----------|-------------------|------------|----------|
| Shelmerg | dc-slip sl-helmer | 0xfffffc00 | autocomp |
|----------|-------------------|------------|----------|

`sliplogin` 到条区分行，并将串行与一个可用的SLIP接口起来，然后行 `/etc/sliphome/slip.login` 脚本：

```
/etc/sliphome/slip.login 0 19200 Shelmerg dc-slip sl-helmer 0xfffffc00 autocomp
```

如果一切顺利 `/etc/sliphome/slip.login` 将在 `sliplogin` 定的 SLIP 接口上出 `ifconfig` (前述的例子中是 SLIP 接口 0，是 `slip.login` 的第一个参数)，以置本地 IP 地址 (`dc-slip`)、程 IP 地址 (`sl-helmer`)、一 SLIP 接口的子网掩 (`0xfffffc00`)，以及任何其他志 (`autocomp`)。如果生， `sliplogin` 通常会通 `syslogd` 的 daemon facility 下有用的信息，前者会把些信息保存到 `/var/log/messages` (参 [syslogd\(8\)](#) 和 [syslog.conf\(5\)](#) 以及 `/etc/syslog.conf` 的机手册，以了解 `syslogd` 在什，以及些内容将被在里)。

### 28.7.2.3. 内核配置

FreeBSD 的默认内核 (GENERIC) 提供了 SLIP ([sl\(4\)](#)) 支持；使用定制的内核，必把下面的置加入到配置文件：

```
device sl
```

默情况下，的 FreeBSD 计算机不会包。如果希望将 FreeBSD SLIP 服务器作路由器使用，就需要修改 `/etc/rc.conf` 文件，将 `gateway_enable` 量 YES。下次系引就能保持一配置了。

要立即用些配置，可以 `root` 的身行：

```
# /etc/rc.d/routing start
```

参 配置FreeBSD的内核 以了解如何配置 FreeBSD 内核， 并 得在重新配置内核方面的指 。

#### 28.7.2.4. Sliplogin配置

正如先前所提到的， /etc/sliphome 目 中有三个文件， 它 共同 成 /usr/sbin/sliplogin 的配置（参考 [sliplogin](#) 的 机手册 [sliplogin\(8\)](#)）： 用于定 SLIP 用 和相 的 IP 地址的 slip.hosts、 通常 用于配置 SLIP 接口的 slip.login， 以及（可 的）slip.logout， 用以撤 由 slip.login 所 行的 作。

##### 28.7.2.4.1. 配置 slip.hosts

/etc/sliphome/slip.hosts里的 行包含至少四个元素， 元素之 由空格隔 ：

- SLIP用 的登 ID
- SLIP 接的本地地址(指SLIP服 器)
- SLIP 接的 程地址
- 网 掩网

本地和 程地址可以是主机名（通 文件/etc/hosts或者域名服 解析 IP地址， 取 取决于文件/etc/nsswitch.conf 中的 置）， 网 掩网可以是一个 能通 文件/etc/networks解析的名字。 在一个 例系 中， /etc/sliphome/slip.hosts是 的：

```
#
# login local-addr      remote-addr      mask      opt1      opt2
#                      (normal,compress,noicmp)
#
Shelmerg dc-slip      sl-helmerg      0xfffffc00      autocomp
```

在 行末尾是一或多个 ：

- **normal** -不 任何
- **compress** - 任何
- **autocomp** -如果 程端允 ， 任何
- **noicmp** -禁用ICMP数据包（ 就会 所有的"ping"数据包， 不占用 的 ）

SLIP 接的本地及 程地址的 取 决是 是准 在SLIP服 器上使用 TCP/IP 子网 是使用"ARP代理"（它并不是"真正的"ARP代理， 而是我 在本 用于介 的 ）。 如果 不能 定 何 方式或者如何分配地址， 参 考"前提条件"([前提条件](#))里列出的TCP/IP 籍 或者向 的IP网 管理 教。

如果打算 的 SLIP 客 使用一个独立的子网， 就需要先从分配得到的网 号中取出一个子网号， 然后再在 个子网里 个 SLIP 客 分配 IP 地址。 接下来， 需要通 SLIP 服 器在最近的 IP 路由器上配置一个指向 SLIP 子网的静 路由。

如果要使用 "代理 ARP" 的方式， 需要从 SLIP 服 器的以太子网中 个 SLIP 客 分配IP地址， 必 修改/etc/sliphome/slip.login 和 /etc/sliphome/slip.logout脚本以使用 [arp\(8\)](#)来管理在 SLIP 服 器 ARP



表中的 "代理 ARP" 。

#### 28.7.2.4.2. slip.login Configuration

典型的/etc/sliphome/slip.login 如下所示：

```
#!/bin/sh -
#
#      @(#)slip.login  5.1 (Berkeley) 7/1/90

#
# generic login file for a slip line.  sliplogin invokes this with
# the parameters:
#      1      2      3      4      5      6      7-n
#  slipunit ttyspeed loginname local-addr remote-addr mask opt-args
#
/sbin/ifconfig sl$1 inet $4 $5 netmask $6
```

这个slip.login脚本有相本地及程地址和掩的SLIP接口行 **ifconfig**。

如果决定使用"ARP代理" 方式(而非的SLIP客使用独立的子网)， 的/etc/sliphome/slip.login 是：

```
#!/bin/sh -
#
#      @(#)slip.login  5.1 (Berkeley) 7/1/90

#
# generic login file for a slip line.  sliplogin invokes this with
# the parameters:
#      1      2      3      4      5      6      7-n
#  slipunit ttyspeed loginname local-addr remote-addr mask opt-args
#
/sbin/ifconfig sl$1 inet $4 $5 netmask $6
# Answer ARP requests for the SLIP client with our Ethernet addr
/usr/sbin/arp -s $5 00:11:22:33:44:55 pub
```

slip.login新加的行**arp -s \$5 00:11:22:33:44:55 pub** 在 SLIP 服务器的 ARP 表中加入了一个表。 个ARP使得当个以太网上的其它 IP 点 SLIP 客端 IP 地址行 ARP 求， SLIP 服务器会以自己的以太网MAC地址作回。

当使用以上的例子， 一定要将 以太网MAC地址 (00:11:22:33:44:55) 替成系网的MAC地址， 否"ARP代理" 将完全无法工作！ 可以看 **netstat -i** 出果以取得以太网 MAC 地址； 出的第二行是：

```
ed0    1500    <Link>0.2.c1.28.5f.4a          191923      0    129457      0    116
```

行表明个系的以太网MAC地址是**00:02:c1:28:5f:4a** **-netstat** **-i**出的以太网MAC地址必

改成用冒号隔, 并且要个十六数前加上。 是arp(8)要求的格式; 参考arp(8) 的机手册以 取完整的使用方法。



在写 /etc/sliphome/slip.login 和 /etc/sliphome/slip.logout , 一定要置 "可执行" (execute) 位 (言之, chmod 755 /etc/sliphome/slip.login /etc/sliphome/slip.logout), 否 sliplogin将无法行它。

#### 28.7.2.4.3. slip.logout配置

/etc/sliphome/slip.logout并不是必需的 (除非使用了"ARP代理"), 如果准建它, 里有一个基本的 slip.logout 脚本的例子:

```
#!/bin/sh -
#
#      slip.logout

#
# logout file for a slip line.  sliplogin invokes this with
# the parameters:
#      1      2      3      4      5      6      7-n
#  slipunit ttyspeed loginname local-addr remote-addr mask opt-args
#
/sbin/ifconfig sl$1 down
```

如果使用了 "代理 ARP", 可能希望 /etc/sliphome/slip.logout 在用注自 SLIP 客端除 ARP:

```
#!/bin/sh -
#
#      @(#)slip.logout

#
# logout file for a slip line.  sliplogin invokes this with
# the parameters:
#      1      2      3      4      5      6      7-n
#  slipunit ttyspeed loginname local-addr remote-addr mask opt-args
#
/sbin/ifconfig sl$1 down
# Quit answering ARP requests for the SLIP client
/usr/sbin/arp -d $5
```

arp -d \$5 将除由 "代理 ARP"slip.login 在 SLIP 客程序登所生成的 ARP。

再次: 建立 /etc/sliphome/slip.logout 之后, 一定要置可执行位 (也就是, chmod 755 /etc/sliphome/slip.logout)。

#### 28.7.2.5. 路由考

如果没有使用 "代理 ARP" 的方法来在的 SLIP 客机和网的其余部分 (也可能是 Internet) 之

路由数据包， 可能需要添加最近的默认路由器的静路由， 以便通过 SLIP 服务器来在 SLIP 客户机子网上  
行路由。

#### 28.7.2.5.1. 静路由

向最近的默认路由添加一个静路由可以是很麻烦（或者是不可能， 如果没有限制做）。 如果在的  
中使用多路由器网， 有些路由器（比如 Cisco 和 Proteon 生的）不但要配置指向 SLIP 子网的路由，  
而且需要配置将些静路由到其它的路由器。 所以一些家意见和解答于使基于静  
路由表的路由正常工作很有必要。

# Chapter 29. 子件

## 29.1. 概述

"子件", 或通常所说的 email, 是现今使用最广泛的通信方式之一。本章将如何在 FreeBSD 上行件服务, 以及如何使用 FreeBSD 来收子件作基本的介绍; 然而, 它并不是一本完整的参考手册, 而且, 许多需要考量的重要事项都没有提及。我们推荐者参考[参考文献](#)中的参考书籍, 以获得于本部分的全面知识。

读完本章, 你将了解:

- 哪些件与收子件有。
- FreeBSD 下的基本 sendmail 配置文件在哪里。
- 本地和远程箱之间的区别。
- 如何阻止邮件制造者非法地使用件的服务器作中继。
- 如何安装和配置用于替代 sendmail 的其他件代理。
- 如何管理常件的服务器。
- 如何使用 SMTP 和 UUCP。
- 如何置系使其只能送件。
- 如何在号接使用件。
- 如何配置 SMTP 以增加安全性。
- 如何安装并使用件代理, 如 mutt 来收件。
- 如何从远程的 POP 或 IMAP 服务器上收件。
- 如何在入的件上自地用器和。

在本章之前, 你需要:

- 正确地配置的网络连接 ([高网络](#)).
- 正确地件的服务器配置 DNS 信息 ([网服](#)).
- 知道如何安装第三方件 ([安装应用程序](#), [Packages](#) 和 [Ports](#)).

## 29.2. 使用子件

件交互可以分为 5 部分。它是: [用端程序](#)、[服端守护程](#)、[DNS](#)、[程或本地的箱](#)、当然, [有件主机自己](#)。

### 29.2.1. 用端程序

包括一些基于命令行的程序, 例如 mutt、alpine、elm 和 [mail](#), 以及似 balsa、xfmail 的 GUI 程序。此外, 有我们更熟悉的"WWW 器"的程序。些程序地通用[服守程](#)把件事交本地的["件主机"](#), 或者通 TCP 把件出去。

## 29.2.2. 邮件主机上使用的服务程序

FreeBSD 默认情况下采用 sendmail，但它也支持数量多的其它邮件服务程序，其中包括：

- exim;
- postfix;
- qmail.

邮件服务器后台守护程序通常有四个功能 - 接收外面来的邮件和把邮件送出去。但它不能使用类似 POP 或 IMAP 的邮件来访问邮件，也不能接到本地的 mbox 或 Maildir 信箱。它可能需要其它的 [服务程序](#) 来完成某些任务。



旧版本的 sendmail 有一些严重的安全问题，它们可能致攻击者从本地和/或远程操作的系统。用户应该自己使用的是最新版本以避免这些问题。另外，也可以从 [FreeBSD Ports Collection](#) 来安装其它的 MTA。

## 29.2.3. Email 和 DNS

域名系统 (DNS) 及其服务程序 `named` 在 email 的投递过程中扮演着很重要的角色。为了能从一个站点向其它的站点发送邮件，服务程序需要知道 DNS 接收邮件的远程站点的位置。类似地，在远程站点向本地主机发送邮件也会产生问题。

DNS 将主机名映射到 IP 地址，同时，也需要保存发送邮件所需要的信息，这些信息称作 MX 记录。MX (Mail eXchanger, 邮件交换) 记录指定了一个，或某些主机能接收特定域下的邮件。如果没有主机名或域名置 MX 记录，邮件将被直接交到主机名到 IP 所在的主机。

可以通过 `host(1)` 命令来查看任何域或主机名的 MX 记录，如下面的例子所示：

```
% host -t mx FreeBSD.org
FreeBSD.org mail is handled (pri=10) by mx1.FreeBSD.org
```

## 29.2.4. 接收邮件

本地的域接收邮件是通过邮件服务器来完成的。它收集发送到本地域的那些邮件，并保存到 mbox (存邮件默认的方法) 或 Maildir 格式，这取决于采用的配置。一旦邮件被保存下来，就可以在本地通过类似 `mail(1)` 或 `mutt` 的程序，或在远程通过 POP 或 IMAP 的邮件来收取了。本地地，如果只在本地的邮件，那就没有必要安装 POP 或 IMAP 服务。

### 29.2.4.1. 通过 POP 和 IMAP 远程程序的邮件

如果希望在远程邮箱，就需要 POP 或 IMAP 服务器。这些允许用户从远程方便地访问他的信箱。尽管 POP 和 IMAP 都允许用户从远程访问信箱，但 IMAP 有很多特点，包括：

- IMAP 既可以从远程服务器上取邮件，也可以把邮件放上去。
- IMAP 支持并发性更新。
- IMAP 易于使用低速网络的用尤其有用，因为它能允许用户把邮件的部分下载下去，而无需立即下载整个邮件。它可以在服务器端进行类似的操作，以减少客户端和服务器之间的通信量。

可以按照下面的来安装和配置 POP 或 IMAP 服务器：

1. 选择一个最符合需要的 IMAP 或 POP 服务器。下列 POP 和 IMAP 服务器是最著名的，而且都有很多成功案例：
  - qpopper;
  - teapop;
  - imap-uw;
  - courier-imap;
  - dovecot;
2. 通过 ports collection 安装 POP 或 IMAP 服务。
3. 根据需要修改 /etc/inetd.conf 来加入 POP 或 IMAP 服务。



此外需要注意的是 POP 和 IMAP 服务的信息，包括用户名和密码等等，通常都是明文的。这意味着如果希望加密进程中的信息，可能需要考虑使用 [ssh\(1\)](#) 隧道或者使用 SSL。关于如何实施隧道在 [SSH 隧道](#) 中进行了描述，SSL 部分在 [OpenSSL](#)。

#### 29.2.4.2. 操作本地的信箱

信箱可以在邮件服务器本地直接用 MUA 来进行操作。通常是通 mutt 或 [mail\(1\)](#) 的的应用程序的。

#### 29.2.5. 邮件服务器

邮件服务器是通服务器的一个名字（注意：来自主机），也正是它能在的主机和网络上发送和接收邮件的原因。

### 29.3. sendmail 配置

[sendmail\(8\)](#) 是 FreeBSD 中的默认邮件代理（MTA）。sendmail 的任务是从邮件用户代理（MUA）接收邮件然后根据配置文件的定义把它发送配置好的的寄送程序。sendmail 也能接受网连接，并且发送邮件到本地信箱或者发送它到其它程序。

sendmail 使用下列配置文件：

| 文件名                        | 功能                |
|----------------------------|-------------------|
| /etc/mail/access           | sendmail 数据库文件    |
| /etc/mail/aliases          | 信箱名               |
| /etc/mail/local-host-names | sendmail 接收邮件主机列表 |
| /etc/mail/mailer.conf      | 邮寄配置程序            |
| /etc/mail/mailertable      | 邮件分列表             |
| /etc/mail/sendmail.cf      | sendmail的主配置文件    |
| /etc/mail/virtusertable    | 虚拟用户和域列表          |

### 29.3.1. /etc/mail/access

数据决定了什么主机或者 IP 地址可以本地邮件服务器和它是哪种类型的。主机可能会列出 **OK**、**REJECT**、**RELAY** 或者通的 `sendmail` 的输出程序一个特定的邮件。主机默列出 **OK**，允许发送邮件到主机，只要邮件的最后目的地是本地主机。列出 **REJECT** 将拒所有的邮件接收。如果有 **RELAY** 的主机将被允许通个邮件服务器发送邮件到任何地方。

例 33. 配置 `sendmail` 的可数据

```
cyberspammer.com      550 We do not accept mail from spammers
FREE.STEALTH.MAILER@  550 We do not accept mail from spammers
another.source.of.spam REJECT
okay.cyberspammer.com OK
128.32                RELAY
```

在上面的例子中我有 5 条。与左列表匹配的邮件人受到右列表作的影响。前的例子出了 `sendmail` 的输出程序到的代码。当一个邮件与左列表相匹配，个信息会被打印到程主机上。下一条拒来自 Internet 上的一个特主机的邮件 **another.source.of.spam**。接下来的允许来自 **okay.cyberspammer.com** 的邮件接收，条比上面那行 **cyberspammer.com** 更准。更多的准匹配使不准的匹配无效。最后一行允许子邮件从主机和 **128.32** 的 IP 地址。些主机将被允许通台邮件服务器前往其它邮件服务器发送邮件。

当个文件被升的时候，必在 `/etc/mail/` 行 **make** 升数据。

### 29.3.2. /etc/mail/aliases

名数据包含一个展到用，程序或者其它名的虚箱列表。下面是一些在 `/etc/mail/aliases` 中使用的例子：

例 34. 邮件名

```
root: localuser
ftp-bugs: joe,eric,paul
bit.bucket: /dev/null
procmail: "|/usr/local/bin/procmail"
```

个文件的格式很；冒号左的箱名，会被展成右的形式。第一个例子地将 **root** 箱展 **localuser**，之后将个在名数据中行。如果没有到匹配的，邮件会被本地用 **localuser**。第二个例子展示了一个邮件列表。送到 **ftp-bugs** 的邮件会被展成 **joe**，**eric** 和 **paul** 三个箱。注意也可以通 **user@example.com** 的形式来指定程的箱。接下来的例子展示了如何把邮件写入到文件中，个例子中是 `/dev/null`。最后一个例子展示了如何将邮件一个程序，具体而言是通 UNIX® 管道到 `/usr/local/bin/procmail` 的准入。

更新此文件，需要在 `/etc/mail/` 中使用 **make** 来更新数据。

### 29.3.3. /etc/mail/local-host-names

`/etc/mail/local-host-names` 是一个 `sendmail(8)` 被接受的一个本地主机名的主机名列表。可以放入任何 `sendmail` 将从那里收件件的域名或主机。例如，如果一个邮件服务器从域 `example.com` 和主机 `mail.example.com` 接收邮件，它的 `local-host-names` 文件，可以看起来象如下：

```
example.com
mail.example.com
```

当这个文件被升级，`sendmail(8)` 必须重新配置，以便更新配置。

### 29.3.4. /etc/mail/sendmail.cf

`sendmail` 的主配置文件 `sendmail.cf` 控制着 `sendmail` 的所有行，包括从重写邮件地址到打印拒绝邮件服务器信息等所有事。当然，作为一个不同的角色，这个配置文件是相当复杂的，它的复杂部分已经超出了本书的范围。幸运的是，这个文件对于标准的邮件服务器来说很少需要被修改。

`sendmail` 的主配置文件可以用 `m4(1)` 宏定义 `sendmail` 的特性和行。它的文档看 `/usr/src/contrib/sendmail/cf/README`。

当这个文件被修改，`sendmail` 必须重新配置以便新修改生效。

### 29.3.5. /etc/mail/virtusertable

`virtusertable` 映射虚拟域名和邮箱到真实的邮箱。有些邮箱可以是本地的、远程的、`/etc/mail/aliases` 中定义的别名或一个文件。

例 35. 虚拟域邮件映射的例子

```
root@example.com      root
postmaster@example.com postmaster@noc.example.net
@example.com           joe
```

在上面这个例子中，我映射了一个域 `example.com`。这个文件是按照从上到下，首个匹配的方式来处理的。第一，将 `root@example.com` 映射到本地邮箱 `root`。第二，将 `postmaster@example.com` 映射到位于 `noc.example.net` 的 `postmaster`。最后，如果没有来自 `example.com` 的匹配，将使用最后一条映射，它表示将所有的其它邮件到 `example.com` 域的某个人，将映射到本地信箱 `joe`。

## 29.4. 改变邮件代理程序

先前已经提到，FreeBSD 中的 `sendmail` 已经安装了 MTA（邮件代理程序）。因此它做着接收邮件的工作。

然而，基于不同的理由，一些系统管理员想要改换其它的 MTA。有些理由从简单的想要换一个 MTA，到需要一个特殊的特性或者 package 依赖某个库程序等等。幸运的是，不管是什理由，FreeBSD 都能容易的改换它。



### 29.4.1. 安装一个新的 MTA

对于可用的 MTA 有很多的。一个好的出发点是 [FreeBSD Ports Collection](#)，在那里你能得到很多。当然你可以从任何位置不受任何限制的使用 MTA，只要它能运行在 FreeBSD 下。

开始安装新的 MTA。一旦它被安装，它可以有机会判定它是否能满足的需要，并且在它接管 sendmail 之前有机会配置新的邮件。当完成这些之后，确信安装的新邮件不会更改系统的二进制文件例如 /usr/bin/sendmail。除此以外，新的邮件邮件用之前要已配置好它。

具体配置参考所给的 MTA 邮件的配置文或其它相关资料。

### 29.4.2. 禁用 sendmail



如果你打算禁用 sendmail 的邮件服务，保持系统中有一个替代它的、可用的邮件送系统就非常重要。如果你不做的，类似 [periodic\(8\)](#) 的系统功能就无法如期期的那，通过邮件来送其行结果。系统中的多部分可能都假定有可用的 sendmail-兼容系统。如果些应用程序使用 sendmail 的行文件来送邮件，而又禁用了它，邮件将入 sendmail 的非活 (inactive) 列，而永不会被送。

要彻底禁用包括邮件送出服务在内的所有 sendmail 功能，必须将

```
sendmail_enable="NO"
sendmail_submit_enable="NO"
sendmail_outbound_enable="NO"
sendmail_msp_queue_enable="NO"
```

写入 /etc/rc.conf。

如果只是想要停止 sendmail 的接收邮件服务，可以在 /etc/rc.conf 文件中置

```
sendmail_enable="NO"
```

更多的有关 sendmail 可用的选项，参看 [rc.sendmail\(8\)](#) 手册。

### 29.4.3. 机器引导时运行的新 MTA

可以向 /etc/rc.conf 中加入配置使新的 MTA 在系统启动时运行，下面是一个 postfix 的例子：

```
# echo 'postfix_enable="YES"' >> /etc/rc.conf
```

新的 MTA 就能在系统是自动运行了。

### 29.4.4. 替系统默认的邮件程序 sendmail

因为 sendmail 程序是一个在 UNIX® 系统下普遍存在的一个标准的邮件，一些邮件就假定它已被安装并且配置好。基于这个原因，多其它的 MTA 提供者都提供了兼容 sendmail 的命令行界面来运行。

使它像“混入”sendmail 一样的很容易掌握。

因此，如果你使用其它的寄程序，你必须定这个件是去哪个行准的 sendmail 二进制，就象 /usr/bin/sendmail，它是行自己自己的替寄程序。幸运的是，FreeBSD 提供了一个系用 [mailwrapper\(8\)](#)，它能做件工作。

当 sendmail 安装后被行，可以在 /etc/mail/mailer.conf 中看到如下行：

```
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
```

个的意思就是当些公共命令（例如 sendmail 它本身）行，系上用了个 sendmail 指定的 mailwrapper 的副本，它 mailer.conf 并且行 /usr/libexec/sendmail/sendmail 做替代。当默的 sendmail 功能被用，系将很容易的改上行的二进制文件。

因此如果你想 /usr/local/supermailer/bin/sendmail-compat 替 sendmail 被行，改 /etc/mail/mailer.conf 文件：

```
sendmail    /usr/local/supermailer/bin/sendmail-compat
send-mail   /usr/local/supermailer/bin/sendmail-compat
mailq       /usr/local/supermailer/bin/mailq-compat
newaliases  /usr/local/supermailer/bin/newaliases-compat
hoststat    /usr/local/supermailer/bin/hoststat-compat
purgestat   /usr/local/supermailer/bin/purgestat-compat
```

## 29.4.5. 最后

一旦做完想要配置的件事，掉 sendmail 程并且属于的新件的程，或者的重。重也将提供了系的已行了正的配置的机会。在引的候自的行新的 MTA。

## 29.5. 疑解答

### 29.5.1. 什必在我的站点的主机上使用 FQDN？

可能会主机上是在外一个域里面，例如，如果你是在 `foo.bar.edu` 里，而要一台叫 `mumble` 的主机，它在 `bar.edu` 域里，就必须用完整的域名 `mumble.bar.edu`，而不是用 `mumble`。

上，在 BSD BIND resolvers 中是可行的。然而目前随 FreeBSD 附的 BIND 已不同一域外提供写服。所以，个不完整的主机名 `mumble` 必须以 `mumble.foo.bar.edu` 形式才能被到，或者将在根域中搜索它。

跟以前的理是不同的，以前版本将会 `mumble.bar.edu` 和 `mumble.edu`。如果想要了解方式是否是好，或者它有什安全方面的漏洞，参 RFC 1535 文。

如果你想要一个好的工作环境，你可以使用如下行：

```
search foo.bar.edu bar.edu
```

替你先前的版本：

```
domain foo.bar.edu
```

把行放在的 `/etc/resolv.conf` 文件中。然而，你一定要定搜索顺序不会造成 RFC 1535 里提到的"boundary between local and public administration"。

### 29.5.2. sendmail 提示信息 mail loops back to myself

下面是 sendmail FAQ 中的回答：

我得到了如下的信息：

```
553 MX list for domain.net points back to relay.domain.net
554 <user@domain.net>... Local configuration error
```

我如何解决这个问题？

你已经通过 MX 指定把送特定的域（例如，domain.net）的邮件被寄到指定的主机（在这个例子中，relay.domain.net），而台机器并不它自己是 domain.net。把 domain.net 添加到 `/etc/mail/local-host-names` 文件中 [在 8.10 版之前是 `/etc/sendmail.cw`]（如果你使用 `FEATURE(use_cw_file)` 的）或者在 `/etc/mail/sendmail.cf` 中添加 `Cw domain.net`。  
MX record

sendmail 的 FAQ 可以在 <http://www.sendmail.org/faq/> 找到，如果你想要的邮件做任何的调整，你推首先看一看它。

### 29.5.3. 我如何在一个号主机上行一个邮件服务？PPP

想要把局域网上的 FreeBSD 主机连接到互联网上，而台 FreeBSD 主机将会成个局域网的邮件网，个号连接不必一直保持在连接状态。

最少有方法可以满足的要求。一方法就是使用 UUCP。

另一方法是到一个的服务器来为的域提供副 MX 主机服务。例如，如果你的域名是 `example.com`，的互联网服务提供商把 `example.net` 作为域的副 MX 服务：

|              |    |    |              |
|--------------|----|----|--------------|
| example.com. | MX | 10 | example.com. |
|              | MX | 20 | example.net. |

只有一台主机被指定当做您的最可靠收信主机（在 `example.com` 主机的 `/etc/mail/sendmail.cf` 文件中添加 `Cw example.com`）。

当 `sendmail` 区分信件的时候，它会通过 modem 连接到（`example.com`）。因您并不在，所以它是会得到一个超信的。 `sendmail` 将会把信件送到副 MX 主机，也就是，您的互联网服务提供商（`example.net`）。副 MX 主机将周期性的接收并送信件到您的主机（`example.com`）。

您也可以使用下面的这个登录脚本：

```
#!/bin/sh
# Put me in /usr/local/bin/pppmyisp
( sleep 60 ; /usr/sbin/sendmail -q ) &
/usr/sbin/ppp -direct pppmyisp
```

如果您想要一个用您建立一个分登录的脚本，您可以使用 `sendmail -qRexample.com` 替换上面的脚本。这将使所有的信件按照您的 `example.com` 列立即被处理。

更深入的方法可以参考下面一段：

这段信息是从 [FreeBSD Internet 服务提供商的信件列表](#) 拿来的。

- > 我用提供副 MX 主机服务。用每天都会上好几次
- > 并且自己把信件取回主 MX 主机
- > （当有他的信件我并没有通知他）。
- > 我的 mailqueue 程序 30 分钟清一次信件列。那段他
- > 就必须上 30 分钟以保他的信件送他的主 MX 主机。
- >
- > 有任何指令可以用 `sendmail` 寄出所有信件？
- > 普通用在我机器上当然没有 root 权限。

在 `sendmail.cf` 的 `privacy flags` 部分，有您的设定 `Opgoaway,restrictqrun`

移除 `restrictqrun` 可以非 root 用队列处理的程序。可能也要重新安排您的 MX 设定。我是用您的 MX 主机，而且我设定了个：

```
# If we are the best MX for a host, try directly instead of generating # local config error.
OwTrue
```

您的程序机器会直接把信送，而不会接收的用您的机器。然后就可以把信件送到您的用。这个设定只主机有效，所以必须要您的用在 DNS 中把他信件主机置 `customer.com` 或者 `hostname.customer.com`。只要 `customer.com` 在 DNS 里添加一个 A 记录就可以了。

#### 29.5.4. 为什么当我发送邮件到其它主机时会有 Relaying Denied 出错信息？

默认 FreeBSD 安装中，sendmail 会配置为只发送邮件来自它所在主机上的邮件。例如，如果有可用的 POP 服务器，用户可以将邮件从学校、公司或其他什么地方接收，但他仍然无法从远程直接发送邮件。通常，在几次尝试之后，MAILER-DAEMON 将发出一封包含 5.7 Relaying Denied 信息的邮件。

有很多方法可以避免这种现象。最直接了当的方法是把你的 ISP 的地址放到 /etc/mail/relay-domains 文件中。完成这项工作的方法是：

```
# echo "your.isp.example.com" > /etc/mail/relay-domains
```

建立或修改这个文件以后你必须重新配置 sendmail。如果你是一个管理员并且不希望在本地上发送邮件，或者想要在其它的机器甚至其它的 ISP 上使用一个客户端系统，这个方法是很方便的。如果你有一到几个邮件，它也非常有用。如果有大量的地址需要添加，你可以很轻松地使用喜欢的文本编辑器打开这个文件添加域名，再行一个：

```
your.isp.example.com
other.isp.example.net
users-isp.example.org
www.example.org
```

你的邮件可以通过系统发送，这个列表中存在的主机（前提是用你的系统在本地已有一个）将可以成功地发送。这是一个允许正常的程序从你的系统发送邮件，并且阻止其它非法用通过系统发送邮件的好方法。

## 29.6. 高级主题

下面将介绍邮件配置和整个域安装邮件。

### 29.6.1. 基本配置

在邮箱外，只要配置 /etc/resolv.conf 或者运行自己的名字服务器，你就可以发送邮件到外部的主机。如果你想要邮件发送到某个特定的 MTA(例如，sendmail) 在你的 FreeBSD 主机上，有以下几个方法：

- 运行自己的域名服务器和自己的域。例如，FreeBSD.org
- 得直接分给主机发送邮件。可以直接使用当前的 DNS 名称。例如，example.FreeBSD.org。

不管上面那方法，为了直接在你的主机上发送邮件，必须有一个静态的 IP 地址(不是象 PPP 号码一样的地址)。如果你在防火后面，它必须 SMTP 连通。如果你想要在主机上直接的收取邮件，你必须定几件事：

- 定在 DNS 中的 MX 记录(最小号的)指向你的 IP 地址。
- 定在 DNS 中的 MX 记录没有禁止你的主机。

上面的几条都允许在你的主机直接接收邮件。

几个：

```
# hostname
example.FreeBSD.org
# host example.FreeBSD.org
example.FreeBSD.org has address 204.216.27.XX
```

如果你看到这些，你直接往 [yourlogin@example.FreeBSD.org](mailto:yourlogin@example.FreeBSD.org) 你已经可以正常工作了（假如 `sendmail` 已经在 [example.FreeBSD.org](http://example.FreeBSD.org) 上正运行了）。

如果你看到这些：

```
# host example.FreeBSD.org
example.FreeBSD.org has address 204.216.27.XX
example.FreeBSD.org mail is handled (pri=10) by hub.FreeBSD.org
```

所有送到主机（[example.FreeBSD.org](http://example.FreeBSD.org)）的邮件在相同的用户名下将会被 `hub` 阻止的收集，而不是直接送到你的主机。

上面的信息是通过你的 DNS 服务器来处理的。支持邮件路由信息的 DNS 记录是邮件交换记录。如果 MX 记录不存在，邮件将通过它自己的 IP 地址被直接的送到主机。

[freefall.FreeBSD.org](http://freefall.FreeBSD.org) 的 MX 记录如下所示：

```
freefall      MX  30  mail.crl.net
freefall      MX  40  agora.rdrop.com
freefall      MX  10  freefall.FreeBSD.org
freefall      MX  20  who.cdrom.com
```

正如你看到的，`freefall` 有很多 MX 记录。最小编号的 MX 记录是直接接收邮件的主机。如果因一些原因它不可用，其它（有可能会“backup MXes”）接收信息将会接替并做记录的排列。

有了有效的使用交换式 MX 站点，你当从你的机器上分一些 Internet 连接。你的 ISP 或者其它友好的站点可以没有任何记录的提供这个服务。

## 29.6.2. Mail for Your Domain

你设置一个“邮件主机”（又称邮件服务器）你必须要把多邮件送到与它相关的几个工作站中。基本上，你想要“要求”在你的域的所有邮件（在你的例子里是 [\\*.FreeBSD.org](http://*.FreeBSD.org)）向到你的邮件服务器，从而使你的用户可以在主邮件服务器里接收他的邮件。

要使工作最简单，你有同用户名和密码同时存在于多台机器上。使用 `adduser(8)` 来这样做。

你将使用的邮件主机必须向你的工作站指定一个邮件交换记录。你可以在 DNS 中配置：

```
example.FreeBSD.org A    204.216.27.XX      ; Workstation
                    MX  10  hub.FreeBSD.org  ; Mailhost
```

无 A 指向，将工作站重新定位到邮件主机。邮件将被送到 MX 主机。

不能自己做除非运行着一个 DNS 服务器。如果不是，或者不能运行自己的 DNS 服务器，告诉的 ISP 或者提供 DNS 服务的人。

如果正在使用虚拟邮件主机，下面的信息将会有用。在这个例子中，我假定有一个客户并且他有自己的域，这个例子中是 `customer1.org`，要把 `customer1.org` 所有的邮件送到自己的邮件主机 `mail.myhost.com`。的 DNS 记录是：

```
customer1.org      MX 10 mail.myhost.com
```

不需要有个 A 记录，如果只域 `customer1.org` 处理邮件。



必须清楚 `customer1.org` 将不能工作，除非存在一个 A 记录。

最后一件必须要做的事是告诉 `sendmail` 接受邮件的是什域和(或)主机名。里面有好几种方法。下面方法可以任一：

- 添加自己的主机到 `/etc/mail/local-host-names` 文件中，如果使用的是 `FEATURE(use_cw_file)`。如果使用 `sendmail 8.10` 或者更高版本，文件是 `/etc/sendmail.cw`。
- 添加一行 `Cyour.host.com` 到自己的 `/etc/sendmail.cf` 或 `/etc/mail/sendmail.cf` 文件，如果使用 `sendmail 8.10` 或者更高版本。

## 29.7. SMTP 与 UUCP

`sendmail` 的配置，在 `FreeBSD` 中已配置好自己的站点直接的接 Internet。如果站点希望他的邮件通过 `UUCP` 交互，必须安装其它的 `sendmail` 配置文件。

手工的配置 `/etc/mail/sendmail.cf` 是一个高主。 `sendmail 8` 版本通过 `m4(1)` 处理生成一个配置文件，上面这个配置生在一个比高的抽象。 `m4(1)` 配置文件可以在 `/usr/shared/sendmail/cf` 下找到。 `cf` 目录中的 `README` 文件是关于 `m4(1)` 配置的基本的介绍。

最好的支持 `UUCP` 送的方法是使用 `mailertable` 的特点。建立一个材料 `sendmail` 可以使用它自己的路由决策。

首先，必须建立自己的 `.mc` 文件。 `/usr/shared/sendmail/cf/cf` 目录包含一些例子。假定已命名自己的文件叫做 `foo.mc`，要做的只是把它变成一个有效的 `sendmail.cf`：

```
# cd /etc/mail
# make foo.cf
# cp foo.cf /etc/mail/sendmail.cf
```

一个典型的 `.mc` 文件看起来可能象：

```

VERSIONID('Your version number') OSTYPE(bsd4.4)

FEATURE(accept_unresolvable_domains)
FEATURE(nocanonify)
FEATURE(mailertable, 'hash -o /etc/mail/mailertable')

define('UUCP_RELAY', your.uucp.relay)
define('UUCP_MAX_SIZE', 200000)
define('confDONT_PROBE_INTERFACES')

MAILER(local)
MAILER(smtp)
MAILER(uucp)

Cw    your.alias.host.name
Cw    youruucpnodename.UUCP

```

`accept_unresolvable_domains`、`nocanonify` 和 `confDONT_PROBE_INTERFACES` 特性将避免在发送邮件时使用DNS的机会。`UUCP_RELAY` 是支持 UUCP 发送邮件所必需的。在配置文件中放入一个 Internet 上可以处理 UUCP 虚拟域地址的主机名。通常，在配置文件中填入 ISP 邮件的回复。

一旦做完这些，你需要一个 `/etc/mail/mailertable` 文件。如果你只有一个用来处理所有邮件的外通道的，以下的文件就足够了：

```

#
# makemap hash /etc/mail/mailertable.db < /etc/mail/mailertable
.
                                uucp-dom:your.uucp.relay

```

一个更复杂的例子象：

```

#
# makemap hash /etc/mail/mailertable.db < /etc/mail/mailertable
#
horus.interface-business.de    uucp-dom:horus
.interface-business.de         uucp-dom:if-bus
interface-business.de          uucp-dom:if-bus
.heep.sax.de                   smtp8:%1
horus.UUCP                     uucp-dom:horus
if-bus.UUCP                    uucp-dom:if-bus
.                               uucp-dom:

```

三行处理域地址邮件，不会被送出默认的路由，而由某些 UUCP 占据取代的特殊情况，是走了"捷径"。下一行处理本地网的邮件，它可以使用 SMTP 来发送。最后，UUCP 占据提起。UUCP 虚拟域的，允许一个 `uucp-neighbor !recipient` 推翻默认。最后一行以一个单独的句点最末，以 UUCP 送到提供所有的邮件网的 UUCP 占据。所有在 `uucp-dom:` 后面的点名称必须是有效的 UUCP 占据，可以用 `uname` 去。

提醒这个文件在使用前必须被转换成 DBM 数据文件。最好在 `mailertable` 最上面用注解写出命令来完成



的工作。当再次更改的 mailertable 后是需要行命令。

最后提示：如果你不指定某个特定的路径可用，你得把 `-bt` 加到 `sendmail`。会将 `sendmail` 在地址模式。只要按下 `3,0`，接着输入希望的路径位置。最后一行告诉你使用邮件代理程序，代理程序会通知目的主机以及（可能的）地址。要此模式按 `Ctrl + D`。

```
% sendmail -bt
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
> 3,0 foo@example.com
canonify          input: foo @ example . com
...
parse            returns: $# uucp-dom $@ your.uucp.relay $: foo < @ example . com . >
> ^D
```

## 29.8. 只送邮件的配置

多时候，可能只希望通过服务器来发送邮件。典型的情况包括：

- 使用界面，但希望通过似 `send-pr(1)` 的程序发送邮件。就需要使用 ISP 的邮件服务器。
- 不在本地管理邮件的服务器，但它需要把邮件交给服务器来处理。

几乎任何一个 MTA 都能胜任的工作。然而不幸的是，要把一个全功能的 MTA 正地配置成只把邮件交给其他服务器是一件很困难的事情。使用 `sendmail` 以及 `postfix` 的程序，多少有些用牛刀的感觉。

此外，如果你使用典型的 Internet 服务，的服务可能会包含禁止行“邮件服务器”的条款。

满足些需要最好的办法是安装 `mail/ssmtp` port。以 `root` 身行下面的命令：

```
# cd /usr/ports/mail/ssmtp
# make install replace clean
```

一旦装好，`mail/ssmtp` 就可以用四行 `/usr/local/etc/ssmtp/ssmtp.conf` 来配置：

```
root=yourrealemail@example.com
mailhub=mail.example.com
rewriteDomain=example.com
hostname=_HOSTNAME_
```

使用了真的子邮件地址。用你的 ISP 提供的外邮件服务器名称，替掉 `mail.example.com`（某些 ISP 可能将其称“外邮件服务器”或“SMTP 服务器”）。

接下来需要禁用了 `sendmail`，包括邮件出服务在内。参 [禁用 sendmail](#) 以了解一的。

`mail/ssmtp` 也提供了一些其他。参在 `/usr/local/etc/ssmtp` 中的示例配置，或者 `ssmtp` 的机手册来得到一些例子和更多的其他信息。

以这种方式配置 `ssmtp`，能确保计算机上的任何需要发送邮件的邮件都正常，而不必冒反 ISP 的使用政策，或使你的邮件被劫持用于发送邮件的。

## 29.9. 通过接口使用邮件发送

如果你有静态的 IP 地址，就不用修改任何默认的配置。将主机名置为分配的 Internet 名称，其他的事情 `sendmail` 都会替你做好。

如果你的 IP 地址是动态分配的，并使用 PPP 接入 Internet，你可能会从 ISP 的邮件服务器上得到一个信箱。这里我假设你的 ISP 的域名是 `example.net`，你的用户名是 `user`，你把自己的机器称作 `bsd.home`，而你的 ISP 告诉你可以使用 `relay.example.net` 来发送邮件。

为了从信箱收取邮件，需要安装一个收信代理。`fetchmail` 是一个能支持多种不同协议的不间断的程序。这个程序可以通过 `package` 或 Ports Collection ([mail/fetchmail](#)) 来安装。通常，你的 ISP 会提供 POP。如果你使用 PPP，你可以在 Internet 上建立自己的邮件接收点，你可以通过在 `/etc/ppp/ppp.linkup` 中添加如下的内容：

```
MYADDR:
!bg su user -c fetchmail
```

如果你正使用 `sendmail` (如下所示) 来非本地地发送邮件，你可能需要 `sendmail` 在你的 Internet 上建立立即发送邮件队列。要完成这项工作，你把下面的命令放到 `/etc/ppp/ppp.linkup` 中的 `fetchmail` 之后

```
!bg su user -c "sendmail -q"
```

假设你在 `bsd.home` 上有一个 `user` 用户。在 `bsd.home` 上的 `user` 主目录中建立一个 `.fetchmailrc` 文件：

```
poll example.net protocol pop3 fetchall pass MySecret
```

因为它包含了密码 `MySecret`，这个文件只能由 `user` 可读。

要使用正确的 `from:` 来发送邮件，你必须告诉 `sendmail` 使用 `user@example.net` 而不是 `user@bsd.home`。另外，你可能也需要要求 `sendmail` 通过 `relay.example.net` 来发送邮件，以便更快地发送它。

以下的 `.mc` 文件可以满足需求：

```

VERSIONID('bsd.home.mc version 1.0')
OSTYPE(bsd4.4)dn1
FEATURE(nouucp)dn1
MAILER(local)dn1
MAILER(smtp)dn1
Cwlocalhost
Cwbsd.home
MASQUERADE_AS('example.net')dn1
FEATURE(allmasquerade)dn1
FEATURE(masquerade_envelope)dn1
FEATURE(nocanonify)dn1
FEATURE(nodns)dn1
define('SMART_HOST', 'relay.example.net')
Dmbsd.home
define('confDOMAIN_NAME', 'bsd.home')dn1
define('confDELIVERY_MODE', 'deferred')dn1

```

如何将这个 .mc 文件到 sendmail.cf 文件的末尾，请参考前面的章节。另外，在更新 sendmail.cf 文件后，不要忘记重启 sendmail。

## 29.10. SMTP 简介

在邮件服务器上启用 SMTP 有很多好处。SMTP 不仅可以为 sendmail 提供多一重安全保障，而且也使得使用不同机器的漫游用户能够使用同一个邮件服务器，而不需要每次都修改它的邮件客户端配置。

1. 从 ports 安装 [security/cyrus-sasl2](#)。这个 port 位于 [security/cyrus-sasl2](#)。 [security/cyrus-sasl2](#) port 支持很多可以在 `sendmail.cf` 指定的可选项。由于我要使用 SMTP 身份验证，因此要禁用 `LOGIN` 选项。
2. 安装完 [security/cyrus-sasl2](#) 之后，编辑 `/usr/local/lib/sasl2/Sendmail.conf` (如果不存在则建立一个) 并在其中添加下列配置：

```
pwcheck_method: saslauthd
```

3. 接下来，安装 [security/cyrus-sasl2-saslauthd](#)，编辑 `/etc/rc.conf` 并加入下列配置：

```
saslauthd_enable="YES"
```

最后启用 saslauthd 服务：

```
# /usr/local/etc/rc.d/saslauthd start
```

这个服务将充当 sendmail 使用 FreeBSD 的 passwd 数据库来完成身份验证的代理人角色。这避免了需要为 SMTP 身份验证的用户建立用户名和密码的麻烦，也保证了登录与邮件的密码一致。

4. 在 `/etc/make.conf` 文件，添加如下行：

```
SENDMAIL_CFLAGS=-I/usr/local/include/sasl -DSASL
SENDMAIL_LDFLAGS=-L/usr/local/lib
SENDMAIL_LDADD=-lsasl2
```

这些配置将告诉 sendmail 使用适当的配置来在程序中引入 [cyrus-sasl2](#)。在重新安装 sendmail 之前，系统已安装了 [cyrus-sasl2](#)。

5. 重新安装 sendmail 执行如下命令：

```
# 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` 和共享库没有大的优化并且它们都必须可用，sendmail 安装没有任何问题。

6. sendmail 被重新安装和安装后，编辑 `/etc/mail/freebsd.mc` 文件 (或者无系统使用的任何 `.mc` 文件。多管理使用跟 [hostname\(1\)](#) 一致的唯一的 `.mc` 文件输出)。添加下列行在该文件：

```
dn1 set SASL options
TRUST_AUTH_MECH('GSSAPI DIGEST-MD5 CRAM-MD5 LOGIN')dn1
define('confAUTH_MECHANISMS', 'GSSAPI DIGEST-MD5 CRAM-MD5 LOGIN')dn1
```

有些配置有不同的方法，于 `sendmail` 用。如果想要使用除 `pwcheck` 之外的方法，参考相文。

- 最后，在 `/etc/mail` 行 `make(1)`。它将建立的新 `.mc` 文件并建立一个 `.cf` 文件命名 `freebsd.cf` (或者想使用的其它名字的 `.mc` 文件)。接着使用命令 `make install restart`，将制文件到 `sendmail.cf`，并且正的重新 `sendmail`。更多有个程的信息，可以参考 `/etc/mail/Makefile` 文件。

如果所个都做，可以通件的客户端入的登信息并且送一个信息。更多的分析，置 `sendmail` 的 `LogLevel` 到 13 并且看 `/var/log/maillog` 中的信息。

如欲了解更多的信息，参看 `sendmail` 网站上的 于 `SMTP` 的介。

## 29.11. 件用代理

件用代理 (MUA) 是一个用于收件的用程序。更一，随着子的“演化”并愈，MUA 在和子件相合方面得日大；用提供了更多的功能和活性。FreeBSD 包含了于多件用代理的支持，所有些都可以通 [FreeBSD Ports Collection](#) 来松安装。用可以似 `evolution` 以及 `balsa` 的形界面程序，也可以似 `mutt`、`alpine` 或 `mail` 的控制台程序，或者某些大型机使用的 web 界面。

### 29.11.1. mail

`mail(1)` 是 FreeBSD 中默的件用代理 (MUA)。它是一个基于控制台的 MUA，提供了所有用于收文本形式的子件所需的基本功能，然它理附件的能力有限，而且只支持本地的信箱。

然 `mail` 没有内建的 POP 或 IMAP 服器支持，然而些信箱可以通似 `fetchmail` 的用程序，来下到本地的 `mbox` 文件中。一用程序在本章的后部分 (使用 `fetchmail`) 行了介。

要收件，只需地地使用 `mail` 命令，如下所示：

```
% mail
```

用保存在 `/var/mail` 中的信箱的内容会被 `mail` 程序自地取。如果信箱是空的，程序会退出并出一个消息表示没有件。一旦完了信箱，将用程序的界面，并列出件。所有的件会被自号，似下面的子：

```
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"
```

在，通过使用 `mail` 的 `t` 命令，并输出邮件的编号，就可以看到邮件了。在这个例子中，我将显示第一封邮件：

```
& 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.
```

正如在上面的例子中所看到的，`t` 将显示完整的邮件。要再次查看邮件的列表，可以使用 `h`。

如果需要回复邮件，也可以使用 `mail` 来完成，方法是使用 `R` 或 `r` 命令。 `R` 会要求 `mail` 只回复发送邮件的人，而 `r` 不回复发送邮件的人，而且也会将回复抄送原来邮件的其他接收者。如果需要，也可以在命令后面指定邮件的编号。做完这些之后，就可以输入回复了，在邮件的最后有一个只有一个 `.` 的行，例如：

```
& R 1
To: root@localhost
Subject: Re: test

Thank you, I did get your email.
.
EOT
```

要发出新邮件，可以使用 `m`，后面接收件人的邮件地址。多个收件人之间，使用 `,` 分隔。接下来需要输入邮件的主题，然后是正文。同样的，在邮件最后需要一个只有 `.` 的空行表示结束。

```
& mail root@localhost
Subject: I mastered mail

Now I can send and receive email using mail ... :)
.
EOT
```

在 `mail` 工具中，可以用 `?` 来显示帮助，而参考 `mail(1)` 手册可以了解更多关于 `mail` 的帮助信息。



正如前面所提到的那样，`mail(1)` 命令在 Linux 没有考虑到要管理附件，因而在这方面他的功能很弱。新的 MUA，如 `mutt`，能更好地管理附件。但如果仍然希望使用 `mail` 命令，那么 `converters/mpack` port 是一个值得考虑的附加工具。

## 29.11.2. mutt

mutt 是一个短小精悍的邮件代理，它提供了许多卓越的功能，包括：

- 能按索引邮件；
- 支持使用 PGP 邮件进行数字签名和加密；
- 支持 MIME；
- 支持 Maildir；
- 高度可定制。

所有这些特性，都使得 mutt 得以跻身于目前最先行的邮件代理的行列。参考 <http://www.mutt.org> 以了解更多关于 mutt 的资料。

稳定版本的 mutt 可以通过 [mail/mutt](#) port 来安装，而开发版本，可以通过使用 [mail/mutt-devel](#) port 安装。通过 port 安装之后，可以通过下面的命令来安装 mutt：

```
% mutt
```

mutt 会自动取 /var/mail 中的用户信箱，并显示其内容。如果用户信箱中没有邮件，mutt 将等待来自用户的命令。下面的例子展示了 mutt 列出邮件的情形：

```
q:Quit  d:Del  u:Undel  s:Save  m:Mail  r:Reply  g:Group  ?:Help
 1 N   Mar 09 Super-User      ( 1) test
 2 N   Mar 09 Super-User      ( 1) user account
 3 N   Mar 09 Super-User      ( 1) sample

--*--Mutt: /var/mail/marcs [Msgs:3 New:3 1.6K]---(date/date)----- (all)---
```

要查看邮件，只需用光标选中它，然后按 `Enter`。以下是 mutt 显示邮件的例子：

```

i:Exit  -:PrevPg  <Space>:NextPg  u:View Attachm.  d:Del  r:Reply  j:Next  ?:Help
X-Original-To: marcs@localhost
Delivered-To: marcs@localhost
To: marcs@localhost
Subject: test
Date: Tue, 9 Mar 2004 10:28:36 +0200 (SAST)
From: Super-User <root@localhost>

This is a test message, please reply if you receive it.


-N  - 1/1: Super-User          test          -- (all)

```

和 `mail(1)` 类似，mutt 允许你只回复附件人，或者回复所有人。如果只想回复信人，使用 `r` 快捷。要回复所有人 (group reply)，可以用 `g` 快捷。



mutt 会使用 `vi(1)` 命令作编辑器，用于创建和回复邮件。你一行可以通过建立用自己的 `.muttrc` 文件来控制，方法是修改 `editor` 变量或配置 `EDITOR` 环境变量。参看 <http://www.mutt.org/> 以了解配置 mutt 的一些信息。

要撰写新邮件，需要首先按 `m`。在输入了有效的邮件主题之后，mutt 将进入 `vi(1)`，你可以在其中撰写邮件。写好邮件的内容之后，存盘并退出 `vi`，mutt 将保存，并显示一些关于将发出的邮件的摘要信息。要发送邮件，只需按 `y`。下面给出了摘要信息的一个例子：



```
y:Send q:Abort t:To c:CC s:Subj a:Attach file d:Descrip ?:Help
  From: Marc Silver <marcs@localhost>
  To: Super-User <root@localhost>
  Cc:
  Bcc:
  Subject: Re: test
Reply-To:
  Fcc:
Security: Clear

-- Attachments
- I      1 /tmp/mutt-bsd-c0hobscQ      [text/plain, 7bit, us-ascii, 1.1K]

-- Mutt: Compose [Approx. msg size: 1.1K  Atts: 1]-----
```

mutt 也提供了相当丰富的帮助，在大多数菜单中，都可以使用 `?` 将其呼出。屏幕一行中也会显示出常用的快捷键。

### 29.11.3. alpine

alpine 主要是给初学者用的，但也提供了一些高级功能。



过去，alpine 邮件客户端被发现有多个漏洞，这些漏洞会允许攻击者在用户的本地系统上，通过发送精心炮制的邮件来执行任意的代码。所有的已知漏洞都已被修正了，但 alpine 的代码是以很不安全的方式写的，并且 FreeBSD 安全官相信仍然有一些尚未被修复的安全漏洞。当考虑并安装 alpine 可能带来的风险。

最新版本的 alpine 可以通过使用 [mail/alpine](#) port 来安装。装好之后，alpine 可以通过下面的命令：

```
% alpine
```

第一次运行 alpine 时，它会显示出一个欢迎界面，并输出必要的介绍，以及 alpine 小提示要求用匿名邮件发送一封邮件，以便帮助他了解有多少用户在使用他的客户端程序的请求。要发送一封匿名的邮件，按 `Enter`，也可以按 `E` 退出，而不发送匿名邮件。下面是欢迎界面的一个例子：

```
PINE 4.58      GREETING TEXT      No Messages

      <<<This message will appear only once>>>

      Welcome to Pine ... a Program for Internet News and Email

We hope you will explore Pine's many capabilities. From the Main Menu,
select Setup/Config to see many of the options available to you. Also
note that all screens have context-sensitive help text available.

SPECIAL REQUEST: This software is made available world-wide as a public
service of the University of Washington in Seattle. In order to justify
continuing development, it is helpful to have an idea of how many people
are using Pine. Are you willing to be counted as a Pine user? Pressing
Return will send an anonymous (meaning, your real email address will not
be revealed) message to the Pine development team at the University of
Washington for purposes of tallying.

      Pine is a trademark of the University of Washington.

[ALL of greeting text]
? Help      E Exit this greeting      - PrevPage  Z Print
Ret [Be Counted!]      Spc NextPage
```

接下来展示的将是主菜单，可以很容易地通过光标在上面行。这个主菜单提供了用于撰写新邮件、邮件目录，甚至管理地址簿等等的快捷方式。主菜单下面是完成各功能的快捷说明。

由 alpine 打开的默认目录是 inbox。要查看邮件索引，请按 **I**，或下面所示的 MESSAGE INDEX ：

```
PINE 4.58      MAIN MENU      Folder: INBOX  3 Messages

      ?      HELP      -  Get help using Pine

      C      COMPOSE MESSAGE      -  Compose and send a message

      I      MESSAGE INDEX      -  View messages in current folder

      L      FOLDER LIST      -  Select a folder to view

      A      ADDRESS BOOK      -  Update address book

      S      SETUP      -  Configure Pine Options

      Q      QUIT      -  Leave the Pine program

      Copyright 1989-2003.  PINE is a trademark of the University of Washington.

? Help      P PrevCmd      R RelNotes
O OTHER CMDS > [Index]      N NextCmd      K KBLock
```

邮件索引展示了当前目录下的邮件，可以使用光标翻页。按 **Enter** 高亮选定的邮件。

```
PINE 4.58      MESSAGE INDEX                               Folder: INBOX  Message 1 of 3 ANS
A   1 Mar  9 Super-User          (471) test
A   2 Mar  9 Super-User          (479) user account
A   3 Mar  9 Super-User          (473) sample

? Help      < FldrList  P PrevMsg      - PrevPage  D Delete      R Reply
O OTHER CMDS > [ViewMsg] N NextMsg    Spc NextPage  U Undelete  F Forward
```

在上面的截屏中，使用 alpine 显示了一封示例邮件。在屏幕底部也显示了快捷键供参考。其中的一个例子是 `r`，它告诉 MUA 回复正在显示的邮件。

```
PINE 4.58      MESSAGE TEXT                               Folder: INBOX  Message 1 of 3 ALL ANS
Date: Tue,  9 Mar 2004 10:28:36 +0200 (SAST)
From: Super-User <root@localhost>
To: marcs@localhost
Subject: test

This is a test message, please reply if you receive it.

[ALL of message]
? Help      < MsgIndex  P PrevMsg      - PrevPage  D Delete      R Reply
O OTHER CMDS > ViewAttch N NextMsg    Spc NextPage  U Undelete  F Forward
```

在 alpine 中回复邮件，是通过 pico 编辑器完成的，后者默认情况下会随 alpine 一起安装。而 pico 工具使得邮件编辑得更加简单，并且要比 `vi(1)` 或 `mail(1)` 更能容忍操作。回复写好之后，可以用 `Ctrl + X` 来退出它。此前，alpine 程序会要求你。

```
PINE 4.58      COMPOSE MESSAGE REPLY      Folder: INBOX  3 Messages

To      : Super-User <root@localhost>
Cc      :
Attchmnt:
Subject : Re: test
----- Message Text -----

I did recieve your message...

^G Get Help  ^X Send      ^R Read File ^Y Prev Pg  ^K Cut Text ^O Postpone
^C Cancel    ^J Justify    ^W Where is ^U Next Pg  ^U UnCut Text ^T To Spell
```

alpine 程序可以通过使用主菜单中的 `SETUP` 选项来进行定制。 参考 <http://www.washington.edu/alpine/> 来了解更多信息。

## 29.12. 使用 fetchmail

fetchmail 是一个全功能的 IMAP 和 POP 客户端程序，它允许用户自动地从远程的 IMAP 和 POP 服务器上下载邮件，并保存到本地的信箱中；这样，用户下载邮件就更方便了。fetchmail 可以通过 [mail/fetchmail](#) port 安装，它提供了许多有用的功能，其中包括：

- 支持 POP3、APOP、KPOP、IMAP、ETRN 以及 ODMR 选项。
- 通过 SMTP 发送邮件，这使得本地、远程，以及邮件名称都能正常工作。
- 能够以服务器程序的方式运行，并周期性地下载邮件。
- 能够从多个信箱收取邮件，并根据配置，将某些邮件下载到不同的本地用户。

尽管介绍全部 fetchmail 的功能超出了本书的范围，但这里仍然介绍了其基本的功能。fetchmail 工具需要一个名为 `.fetchmailrc` 的配置文件才能正常工作。这个文件中包含了服务器信息，以及登录使用的凭据。由于这个文件包含敏感内容，建议将其设置成只有属主所有，使用下面的命令：

```
% chmod 600 .fetchmailrc
```

下面的 `.fetchmailrc` 提供了一个将某一用户的信箱通过 POP 下载到本地的例子。它告诉 fetchmail 连接到 `example.com`，并使用用户名 `joesoap` 和密码 `XXX`。这个例子假定 `joesoap` 同时也是本地的系统用户。

```
poll example.com protocol pop3 username "joesoap" password "XXX"
```

下一个例子将连接多个 POP 和 IMAP 服务器，并根据需要连接到不同的本地用户：

```
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;
```

另外，fetchmail 也可以通过指定 `-d` 参数，并输出 fetchmail 在 `.fetchmailrc` 文件中列出的服务器的间隔，来以服务器程序的方式运行。下面的例子会让 fetchmail 每 600 秒运行一次：

```
% fetchmail -d 600
```

更多关于 fetchmail 的资料，可以在 <http://fetchmail.berlios.de/> 找到。

## 29.13. 使用 procmail

procmail 是一个强大得惊人的邮件输入件的实用程序。它允许用特定“规则”，并用一些规则来匹配输入的邮件，而执行某些特定的功能，或将某些邮件重定向到其他信箱和/或邮件地址。procmail 可以通过 [mail/procmail](#) port 来安装。装好之后，可以直接把它集成到大多数 MTA 中；参考你使用的 MTA 的文档了解具体的作法。另外，procmail 可允许通过把下面的配置加入到用户主目录中的 `.forward` 文件中，来使用 procmail 功能：

```
"|exec /usr/local/bin/procmail || exit 75"
```

接下来我将介绍一些基本的 procmail 规则，以及它们都是做什么的。各个规则，都写到 `.procmailrc` 文件中，而该文件必须放在用户的主目录下。

主要的规则，也可以在 [procmailex\(5\)](#) 手册中找到。

将所有来自 [user@example.com](#) 的邮件，重定向到外部地址 [goodmail@example2.com](#)：

```
:0
* ^From.*user@example.com
! goodmail@example2.com
```

将所有不超过 1000 字节的邮件到外部地址 [goodmail@example2.com](#)：

```
:0
* < 1000
! goodmail@example2.com
```

把所有送到 [alternate@example.com](#) 的邮件放到信箱 `alternate` 中：

```
:0
* ^TOalternate@example.com
alternate
```

将所有 "Spam" 的邮件到 /dev/null :

```
:0
^Subject:.*Spam
/dev/null
```

将收到的所有 **FreeBSD.org** 邮件列表的邮件， 到各自的信箱：

```
:0
* ^Sender:.owner-freebsd-\[^@]+\@FreeBSD.ORG
{
    LISTNAME=${MATCH}
    :0
    * LISTNAME??^\[^@]+
    FreeBSD-${MATCH}
}
```

# Chapter 30. 网 服 器

## 30.1. 概要

本章将覆 某些在 UNIX® 系 上常用的网 服 。 将会 及 如何安装、配置、 和 多 不同 型的网 服 。 本章 中将提 供大量配置文件的 例，期望能 有所裨益。

在 完本章之后， 将会知道：

- 如何管理 inetd。
- 如何 置 行一个网 文件系 。
- 如何配置一个网 信息服 器以共享用 号。
- 如何通 DHCP自 配置网 。
- 如何配置一个域名服 器。
- 如何 置Apache HTTP 服 器。
- 如何 置文件 (FTP) 服 器。
- 如何使用Samba Windows® 客 端 置文件和打印服 。
- 如何同 和日期，以及如何 置使用NTP 的 服 器。
- 如何配置 准的日志守 程， `syslogd`，接受 程主机的日志。

在 此章 之前， 当：

- 理解有 /etc/rc中脚本的基本知 。
- 熟悉基本网 。
- 得如何安装 外的第三方 件（[安装 用程序](#)、[Packages](#) 和 [Ports](#)）。

## 30.2. inetd"超 服 器"

### 30.2.1. 。

[inetd\(8\)](#) 有 也被称作 "Internet 超 服 器"，因 它可以 多 服 管理 接。当 inetd 收到 接，它能 定 接所需的程序， 相 的 程，并把 socket 交 它 (服 socket 会作 程序的 准 入、 出和 出描述符)。使用 inetd 来 行那些 不重的服 有助于降低系 ，因 它不需要 个服 都 独立的服 程序。

一般 来，inetd 主要用于 其它服 程序，但它也有能力直接 理某些 的服 ，例如 chargen、auth，以及 daytime。

一 将介 于如何通 命令行，以及配置文件 /etc/inetd.conf 来 inetd 行配置的一些基 知 。

### 30.2.2. 置

inetd 是通 [rc\(8\)](#) 系 的。`inetd_enable` 默 为 `NO`，但可以在安装系 ，由 根据需要通 `sysinstall` 来打 。将：

```
inetd_enable="YES"
```

或

```
inetd_enable="NO"
```

写入 `/etc/rc.conf` 可以启用或禁用系统 `inetd` 的服务。命令：

```
# /etc/rc.d/inetd rcvar
```

可以显示目前的配置。

此外，还可以通过 `inetd_flags` 参数来向 `inetd` 设置外的其它参数。

### 30.2.3. 命令行选项

与多数服务程序类似，`inetd` 也提供了很多用以控制其行为的参数。完整的参数列表如下：

```
inetd [-d] [-l] [-w] [-W] [-c maximum] [-C rate] [-a address | hostname] [-p filename] [-R rate] [-s maximum] [configuration file]
```

这些参数都可以通过 `/etc/rc.conf` 的 `inetd_flags` 来设置 `inetd`。默认情况下，`inetd_flags` 为 `-wW -C 60`，前者表示希望 `inetd` 的服务使用 TCP wrapping，并阻止来自同一 IP 每分钟超过 60 次的请求。

虽然我们下面介绍限制连接率的选项，但初学的用户可能会很高地认为这些参数通常并不需要行修改。在收到超大量的连接请求，这些选项有可能会起作用。完整的参数列表，可以在 [inetd\(8\)](#) 手册中找到。

#### -c maximum

指定每个服务的最大并发的数量，默认为不限。也可以在此服务的具体配置里面通过 `max-child` 改掉。

#### -C rate

指定每个服务一分钟内能被一个 IP 地址使用的最大次数，默认为不限。也可以在此服务的具体配置里面通过 `max-connections-per-ip-per-minute` 改掉。

#### -R rate

指定每个服务一分钟内能被使用的最大次数，默认为 256。0 允许不限次数使用。

#### -s maximum

指定同一 IP 同时请求同一服务允许的最大数；默认为不限制。可以通过 `max-child-per-ip` 参数来以服务为单位进行限制。

### 30.2.4. inetd.conf

对于 `inetd` 的配置，是通过 `/etc/inetd.conf` 文件来完成的。

在修改了 `/etc/inetd.conf` 之后，可以使用下面的命令来控制 `inetd` 重新读取配置文件：



例 36. 重新加 inetd 配置文件

```
# /etc/rc.d/inetd reload
```

配置文件中的每一行都是一个独立的服程序。在个文件中， 前面有 "#" 的内容被是注。 /etc/inetd.conf 文件的格式如下：

```
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
```

下面是 IPv4 的 [ftpd\(8\)](#) 服的例子：

```
ftp      stream tcp      nowait  root    /usr/libexec/ftpd      ftpd -l
```

service-name

指明各个服的服名。其服名必与/etc/services中列出的一致。 将决定inetd会听个port。一旦有新的服需要添加，必先在/etc/services里面添加。

socket-type

可以是stream、dgram、raw或者 seqpacket。 stream 用于基于接的 TCP 服；而 dgram 用于使用 UDP 的服。

protocol

下列之一：

|           | 明                    |
|-----------|----------------------|
| tcp, tcp4 | TCP IPv4             |
| udp, udp4 | UDP IPv4             |
| tcp6      | TCP IPv6             |
| udp6      | UDP IPv6             |
| tcp46     | Both TCP IPv4 and v6 |
| udp46     | Both UDP IPv4 and v6 |

{wait|nowait}[/max-child[/max-connections-per-ip-per-minute[/max-child-per-ip]]]

wait|nowait 指明从inetd 里用的服是否可以自己理socket。 dgram socket型必使用wait，而stream socket daemons， 由于通常使用多程方式， 当使用 nowait。 wait 通常把多个 socket 个服程， 而 nowait 会个新的 socket 生成一个子程。

`max-child` 能配置 `inetd` 能本服派生出的最大子程序数量。如果某特定服需要限定最高10个例，把`/10`放到`nowait`后就可以了。指定`/0`表示不限制子程序的数量。

除了 `max-child` 之外，有个可以限制来自同一位置到特定服的最大连接数。`max-connections-per-ip-per-minute` 可以限制特定 IP 地址分的连接数，例如，限制任何 IP 地址分最多连接十次。`max-child-per-ip` 可以限制某一 IP 地址在任何候所的子程序数量。些用于防止服器有意或无意的源耗竭和拒服 (DoS) 攻十分有用。

个字段中，必指定 `wait` 或 `nowait` 者之一。而 `max-child`、`max-connections-per-ip-per-minute` 和 `max-child-per-ip` 是可。

流式多程序服，并且不配置任何 `max-child`、`max-connections-per-ip-per-minute` 或 `max-child-per-ip` 限制，其配置：`nowait`。

同一个服，但希望将服的数量限制十个，是：`nowait/10`。

同配置，限制个 IP 地址分最多连接二十次，而同的子程序最多十个，写作：`nowait/10/20`。

下面是 `fingerd(8)` 服的默认配置：

```
finger stream tcp      nowait/3/10 nobody /usr/libexec/fingerd fingerd -s
```

最后个例子中，将子程序数限制100个，而任意 IP 最多同建立5个连接：`nowait/100/0/5`。

## user

指定服将以什用身行。一般而言，服行身是 `root`。基于安全目的，可以看到有些服以 `daemon` 身，或者是最小特的 `nobody` 身行。

## server-program

当连接到，行服程序的全路径。如果服是由 `inetd` 内置提供的，以 `internal` 代替。

## server-program-arguments

当 `server-program` 用到，通 `argv[0]` 通服而工作。如果命令行：`mydaemon -d`，`mydaemon -d` `server-program-arguments` 的。同的，如果服是由 `inetd` 内置提供的，里是 `internal`。

## 30.2.5. Security

随安装所的模式不同，多 `inetd` 的服可能已默认用。如果不需要某个特定的服，考虑禁用它。在 `/etc/inetd.conf` 中，将服的那行前面加上 `"#"`，然后 [重新加 inetd 配置](#) 就可以了。某些服，例如 `fingerd`，可能是完全不需要的，因它提供的信息可能攻者有用。

某些服在是缺少安全意的，或者有或根本没有连接求的超机制。使得攻者能通慢地些服起连接，并耗尽可用的源。于情况，置 `max-connections-per-ip-per-minute`、`max-child` 或 `max-child-per-ip` 限制，来制服的行是个好法。

默认情况下，TCP wrapping 是打的。参考 [hosts\\_access\(5\)](#) 手册，以得更多于在各 `inetd` 用的服上置 TCP 限制的信息。

### 30.2.6. 服务

daytime、time、echo、discard、chargen，以及auth都是由inetd提供的内建服务。

auth服务提供了网络身份验证，它可以配置为提供不同的服务，而其它服务通常只能提供打印或网络。

参考 [inetd\(8\)](#) 手册获得更多信息。

## 30.3. 网络文件系统（NFS）

网络文件系统是FreeBSD支持的文件系统之一， 也被称作 NFS。 NFS允许一个系统在网络上与其它人共享目录和文件。通过使用NFS，用户和程序可以象操作本地文件一样操作远端系统上的文件。

以下是NFS最显而易见的好处：

- 本地工作站使用更少的磁盘空间，因为通常的数据可以存放在一台机器上而且可以通过网络访问到。
- 用户不必在多个网络上机器里都有一个home目录。Home目录可以被放在NFS服务器上并且在网络上访问可用。
- 例如，CDROM，和 Zip® 之类的存储设备可以在网络上面被任何的机器使用。 这可以减少整个网络上的可移动介质的数量。

### 30.3.1. NFS是如何工作的

NFS 至少包括两个主要的部分：一台服务器，以及至少一台客户端，客户端远程地保存在服务器上的数据。要使它一切运转起来，需要配置并运行几个程序。

服务器必须运行以下服务：

| 服务      | 描述                              |
|---------|---------------------------------|
| nfsd    | NFS，它来自NFS客户端的请求。               |
| mountd  | NFS挂载服务，处理nfsd(8)发来的请求。         |
| rpcbind | 此服务允许 NFS 客户端程序正在被 NFS 服务使用的端口。 |

客户端同样运行一些程序，比如 nfsiod。 nfsiod处理来自NFS的请求。 它是可选的，而且可以提高性能，对于普通和正常的操作来并不是必需的。 参考[nfsiod\(8\)](#)手册获得更多信息。

### 30.3.2. 配置NFS

NFS的配置程序相当简单。每个程序只需要 /etc/rc.conf 文件作一些修改。

在NFS服务器端，在/etc/rc.conf 文件里以下项目都配上了：

```
rpcbind_enable="YES"
nfs_server_enable="YES"
mountd_flags="-r"
```

只要NFS服务被置为enable，mountd 就能自行运行。

在客户端一，下面个出在 /etc/rc.conf里：

```
nfs_client_enable="YES"
```

/etc/exports文件指定了个文件系统 NFS出（有被称"共享"）。 /etc/exports里面行指定一个出的文件系统和一些机器可以文件。在指定机器限的同，也可以被指定。有很多可以被用在个文件里，不不会在里。可以通过[exports\(5\)](#) 手册来些。

以下是一些/etc/exports的例子：

下面是一个出文件系的例子，不配置与所的网络境及其配置密切相。例如，如果要把 /cdrom 出与服务器域名相同的三台计算机（因此例子中只有机器名，而没有出些计算机的域名），或在 /etc/hosts 文件中行了配置。 **-ro** 志表示把出的文件系统置只。由于使用了个志，程系在出的文件系统上就不能写入任何了。

```
/cdrom -ro host1 host2 host3
```

下面的例子可以出/home三个以IP地址方式表示的主机。于在没有配置DNS服务器的私有网里，很有用。此外， /etc/hosts 文件也可以用以配置主机名；参看 [hosts\(5\)](#) 。 **-alldirs** 允子目被作挂点。也就是，客户端可以根据需要挂需要的目。

```
/home -alldirs 10.0.0.2 10.0.0.3 10.0.0.4
```

下面几行出 /a ，以便个来自不同域的客户端可以文件。 **-maproot=root** 授端系上的 **root** 用在被出的文件系统上以**root**身行写。如果没有特指定 **-maproot=root** ，即使用在端系上是 **root** 身，也不能修改被出文件系统上的文件。

```
/a -maproot=root host.example.com box.example.org
```

了能到被出的文件系统，客户端必被授。客端在的 /etc/exports 被列出。

在 /etc/exports 里，一行里面，出信息和文件系统一。一个程主机次只能一个文件系统。而且只能有一个默入口。比如，假 /usr 是独立的文件系统。个 /etc/exports 就是无效的：

```
# Invalid when /usr is one file system
/usr/src client
/usr/ports client
```

一个文件系统， /usr，有行指定出到同一主机， **client**. 解决一的正的格式是：

```
/usr/src /usr/ports client
```

在同一文件系中，出到指定客机的所有目，都必写到同一行上。没有指定客机的行会被是

□一主机。□限制了□可以□□□出的文件系□，但□□大多数人来□□不是□□。

下面是一个有效□出列表的例子，/usr 和 /exports 是本地文件系□：

```
# 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
```

在修改了 /etc/exports 文件之后，就必□□ mountd 服□重新□□它，以便使修改生效。一□方法是通□□正在□行的服□程序□送 HUP 信号来完成：

```
# kill -HUP `cat /var/run/mountd.pid`
```

或指定□当的参数来□行 `mountd rc(8)` 脚本：

```
# /etc/rc.d/mountd onereload
```

□于使用 rc 脚本的□□，□参□ [在 FreeBSD 中使用 rc](#)。

□外，系□重□□可以□ FreeBSD 把一切都弄好。尽管如此，重□不是必□的。以 `root` 身□□行下面的命令可以□定一切。

在 NFS 服□器端：

```
# rpcbind
# nfsd -u -t -n 4
# mountd -r
```

在 NFS 客□端：

```
# nfsiod -n 4
```

□在□件事情都□□就□，以□挂□一个□端文件系□。在□些例子里□，服□器名字将是：`server`，而客□端的名字将是：`client`。如果□只打算□□挂□一个□端文件系□或者只是打算作□□配置正□与否，只要在客□端以 `root` 身□□行下面的命令：

```
# mount server:/home /mnt
```

□条命令会把服□端的 /home 目□挂□到客□端的 /mnt 上。如果配置正□，□□可以□入客□端的 /mnt 目

并且看到所有服务器端的文件。

如果打算系统每次在重启的时候都自动挂载端点的文件系统，把那个文件系统加到 `/etc/fstab` 文件里去。下面是例子：

```
server:/home    /mnt    nfs rw 0 0
```

[fstab\(5\)](#) 手册里有所有可用的选项。

### 30.3.3. 锁

某些应用程序（例如 `mutt`）需要文件上锁支持才能正常运行。在使用 NFS 时，可以用 `rpc.lockd` 来支持文件上锁功能。要用它，需要在服务器和客户端的 `/etc/rc.conf` 中加入（假定两端均已配好了 NFS）：

```
rpc_lockd_enable="YES"
rpc_statd_enable="YES"
```

然后使用下述命令启动程序：

```
# /etc/rc.d/lockd start
# /etc/rc.d/statd start
```

如果并不需要真的在 NFS 客户端和 NFS 服务器间保持上锁的选项，可以在 NFS 客户端在本地上锁，方法是使用 [mount\\_nfs\(8\)](#) 指定 `-l` 参数。参看 [mount\\_nfs\(8\)](#) 手册以了解更多。

### 30.3.4. 网络应用

NFS 有很多应用。下面是比较常见的一些：

- 多个机器共享一台 CDROM 或者其他资源。便于在多台机器中安装软件来更加便宜跟方便。
- 在大型网络中，配置一台中心 NFS 服务器用来放置所有用户的 `home` 目录可能会很便利。有些目录能被导出到网络以便用户不管在哪个工作站上登录，都能得到相同的 `home` 目录。
- 几台机器可以有通用的 `/usr/ports/distfiles` 目录。通常的，当需要在几台机器上安装 `port`，可以无需在每台机器上下网而快速下载源。

### 30.3.5. 通过 amd 自动地挂载

[amd\(8\)](#)（自动挂载服务）能够自动地在启动挂载进程的文件系统。如果文件系统在一分钟内没有活动，它会被 `amd` 自动卸下。通过使用 `amd`，能够提供持久挂载以外的选项，而后者往往需要列入 `/etc/fstab`。

`amd` 通常将自己以 NFS 服务器的形式，附加到 `/host` 和 `/net` 目录上来工作。当某些目录中的文件，`amd` 将相应相的进程挂载点，并自动地挂载。`/net` 用于挂载进程 IP 地址上导出的文件系统，而 `/host` 用于挂载进程主机名上的文件系统。

在 `/host/foobar/usr` 中的文件，相当于告诉 `amd` 挂载在主机 `foobar` 上导出的 `/usr`。

可以通过使用 `showmount` 命令来看程序主机上出的文件系统。例如，要看 `foobar` 上出的文件系统，可以用：

```
% showmount -e foobar
Exports list on foobar:
/usr                      10.10.10.0
/a                        10.10.10.0
% cd /host/foobar/usr
```

如同在前面例子中所看到的，`showmount` 显示了出的 `/usr`。当输入 `/host/foobar/usr` 这个目录，`amd` 将解析主机名 `foobar` 并自动地挂载需要的文件系统出。

`amd` 可以通过脚本来，方法是在 `/etc/rc.conf` 中加入：

```
amd_enable="YES"
```

除此之外，可以通过 `amd` 通过 `amd_flags` 来设置外的参数。默认情况下，`amd_flags`：

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

`/etc/amd.map` 文件定义了挂载出文件系统所使用的默认。 `/etc/amd.conf` 文件，定义了更多关于 `amd` 的高级功能。

参考 [amd\(8\)](#) 和 [amd.conf\(8\)](#) 手册，以了解一的情况。

### 30.3.6. 与其他系统集成时的常见问题

某些特定的 ISA PC 系上的以太网适配器上有一些限制，这些限制可能会导致严重的网，特别是与 NFS 配合使用。这些并非 FreeBSD 所特有的，但 FreeBSD 系会受到这些的影响。

的，几乎是在当 (FreeBSD) PC 系与高性能的工作站，例如 Silicon Graphics, Inc., 和 Sun Microsystems, Inc. 的工作站网。NFS 挂载能正常工作，而且一些操作也可能成功，但服务器会很快得客户机不太理会，然而其他客户机的请求仍然能正常理。情况通常生在客户端，无论它是一个 FreeBSD 系或是端。在多数系上，一旦生了的，通常没法正常地客户机。唯一的法通常是端位，因此一 NFS 状况没有办法被解决。

尽管 "正的" 解决法，是 FreeBSD 系配一高性能的、用的以太网适配器，然而也有法并得到相意的果。如果 FreeBSD 系是服务器，在客户机挂载，指定 `-w=1024`。如果 FreeBSD 系是客户机，加入 `-r=1024` 参数。这些可以通过在的 `fstab` 的第四个字段加入，以便客户机能自地挂载，或者通过 [mount\(8\)](#) 的 `-o` 参数在手工挂载指定。

需要注意的是一个，有可能会是和上面一的。个多关于 NFS 服务器和客户机在不同的网上。如果是情况，一定要定 的路由器把必需的 UDP 信息路由到了目的地，否则将什也做不了。

下面的例子中，`fastws` 是主机（接口）的名字，它是一台高性能的端口，而 `freebox` 是一台主机（接口）的名字，它是一个使用低性能的以太网适配器的 FreeBSD 系统。同时，`/sharedfs` 将被导出成 NFS 文件系统（参看 [exports\(5\)](#)），而 `/project` 将是客户机上挂接一出口文件系统的挂接点。所有的场景中，注意附加选项，例如 `hard` 或 `soft` 以及 `bg` 可能是端口用所需要的。

对于 FreeBSD 系统（`freebox`）作服务器的示例 `/etc/fstab` 文件，位于 `freebox` 之上：

```
fastws:/sharedfs /project nfs rw,-r=1024 0 0
```

在 `freebox` 上手工挂接：

```
# mount -t nfs -o -r=1024 fastws:/sharedfs /project
```

以 FreeBSD 系统作服务器的例子，是 `fastws` 上的 `/etc/fstab`：

```
freebox:/sharedfs /project nfs rw,-w=1024 0 0
```

在 `fastws` 上手工挂接的命令是：

```
# mount -t nfs -o -w=1024 freebox:/sharedfs /project
```

几乎所有的 16-位 以太网控制器，都能在没有上述写尺寸限制的情况下正常工作。

对于那些关心到底是什么的人，下面是失如何产生的解释，同时它也说明了什么是一个无法恢复的。典型情况下，NFS 会使用一个 "0" 端口位行操作，其尺寸是 8 K（虽然它可能会将操作分成更小尺寸的分片）。由于最大的以太网包尺寸大约是 1500 字节，因此 NFS "0" 会分成多个以太网包，虽然在更高代的看来它仍然是一个完整的单元，并在接收方重新组装，作为一个整体来。高性能的工作站，可以将成 NFS 单元的包迅速发出，其节奏会快到允许的最大限度。在容量小的上，后来的包会冲掉同一单元内的最早的包，因而整个单元无法被重建或。其结果是，工作站将超并重，但仍然是完整的 8 K 单元，一旦程序将无休止地重下去。

如果将单元尺寸限制在以太网包尺寸之下，我就能保证一个以太网包都能被独立地接收和，从而避免了上面的死锁情形。

溢出在高性能工作站将数据投向 PC 系统仍会产生，但在更好的网络上，能保证溢出不会在一个 NFS "单元" 上都产生。当出溢出，被影响的单元被重，因此有很大的机会它将被正确接收、重，并。

## 30.4. 网络信息服务 (NIS/YP)

### 30.4.1. 它是什么？

NIS，表示网络信息服务 (Network Information Services)，最初由 Sun Microsystems 开发，用于 UNIX®（最初是 SunOS™）系统的集中管理。目前，它基本上已成为业界标准；所有主流的 UNIX® 系统 (Solaris™, HP-UX, AIX®, Linux, NetBSD, OpenBSD, FreeBSD, 等等) 都支持 NIS。



NIS 也就是人所熟知的黄页(Yellow Pages), 但由于商业的原因, Sun 将其改名现在的名字。旧的黄页 (以及 yp), 仍然经常可以看到, 并被广泛使用。

Yp 是一个基于 RPC 的客户/服务器系统, 它允许在一个 NIS 域中的一台机器共享一系列配置文件。通过它, 系统管理员就可以配置只包含最基本配置数据的 NIS 客户机系统, 并在需要时添加、删除或修改配置数据。

尽管 Yp 的内部结构截然不同, 它和 Windows NT® 域系统非常相似, 以至于可以将两者的基本功能相互对比。

30.4.2. 你可能知道的命令和进程

有一系列命令和重要的用户进程将在下面 FreeBSD 上 NIS 用用到, 无论是在构建 NIS 服务器, 或作为 NIS 客户机:

| 命令            | 说明  |
|---------------|---|
| NIS 域名        | NIS 主服务器和所有其客户机 (包括从服务器) 会使用同一 NIS 域名。和 Windows NT® 域系统类似, NIS 域名与 DNS 无关。  |
| rpcbind       | 必须运行这个程序, 才能使用 RPC (进程使用, NIS 用到的一网络)。如果没有运行 rpcbind, 将无法运行 NIS 服务器, 或作为 NIS 客户机。   |
| ypbind        | "绑定(bind)" NIS 客户机到它的 NIS 服务器上。通过它, 它将从系统中获取 NIS 域名, 并使用 RPC 连接到服务器上。ypbind 是 NIS 环境中, 客户机-服务器通信的核心; 如果客户机上的 ypbind 死掉的, 它将无法访问 NIS 服务器。  |
| ypserv        | 只应在 NIS 服务器上运行它; 它是 NIS 的服务器进程。如果 ypserv(8) 死掉的, 服务器将不再具有访问 NIS 请求的能力 (此时, 如果有从服务器的, 它会接管操作)。有一些 NIS 的进程 (但不是 FreeBSD 的进程) 的客户机上, 如果之前用一个服务器, 而那台服务器死掉的, 并不重新连接到另一个服务器。通常, 产生这种情况, 唯一的办法就是重新安装服务器进程 (或者, 甚至重新安装服务器) 或客户机上的 ypbind 进程。 |
| rpc.yppasswdd | 一个只在 NIS 主服务器上运行的进程; 它是一个服务器程序, 其作用是允许 NIS 客户机更改它的 NIS 口令。如果没有运行这个服务, 用户将必须登录到 NIS 服务器上, 并在那里修改口令。  |

30.4.3. 它是如何工作的?

在 NIS 环境中, 有三种类型的主机: 主服务器, 从服务器, 以及客户机。服务器的作用是充当主机配置信息的中央数据库。主服务器上保存着这些信息的权威副本, 而从服务器是保存这些信息的冗余副本。客户机依赖于服务器向它提供这些信息。

许多文件的信息可以通过一种方式来共享。通常情况下, master.passwd、group, 以及 hosts 是通过 NIS 分发的。无论什么时候, 如果客户机上的某个进程请求某些数据在本地文件中的材料的时候, 它都会向所指定的 NIS 服务器发出请求, 而不使用本地的版本。

### 30.4.3.1. 机器类型

- 一台 NIS 主服务器。一台服务器，和 Windows NT® 域控制器类似，会拥有所有 NIS 客户机所使用的文件。passwd, group, 以及许多其他 NIS 客户机所使用的文件，都被存放到主服务器上。



可以将一台 NIS 主服务器用在多个 NIS 域中。然而，本书不打算详细配置如何进行，因详细配置，通常只出现在小规模的 NIS 环境中。

- NIS 从服务器。同一概念，与 Windows NT® 的域控制器类似。NIS 从服务器，用于保存 NIS 主服务器的数据文件副本。NIS 从服务器提供了一定的冗余，在许多重要的环境中是必需的。此外，它也帮助减轻了主服务器的负荷：NIS 客户机是挂接到最先访问它的 NIS 服务器上，而这也包括来自从服务器的数据。
- NIS 客户机。NIS 客户机，和多数 Windows NT® 工作站类似，通过 NIS 服务器 (或等于 Windows NT® 工作站，或是 Windows NT® 域控制器) 来完成登录的身份验证过程。

## 30.4.4. 使用 NIS/YP

本节将通例介绍如何配置 NIS 环境。

### 30.4.4.1. 背景

假定正在管理大学中的一个小型办公室。在这个办公室中，有 15 台 FreeBSD 机器，目前尚没有集中的管理点；一台机器上有自己的 /etc/passwd 和 /etc/master.passwd。这些文件通过人工干预的方法来保持与其他机器上版本的同步；目前，如果在办公室中添加一个用户，将不得不在所有 15 台机器上手工进行 `adduser` 命令。毋庸置疑，这一状态必须改变，因此决定将整个办公室使用 NIS，并使用一台机器作为服务器。

因此，办公室的配置将是这样的：

| 机器名       | IP 地址         | 机器的角色    |
|-----------|---------------|----------|
| ellington | 10.0.0.2      | NIS 主服务器 |
| coltrane  | 10.0.0.3      | NIS 从服务器 |
| basie     | 10.0.0.4      | 教师工作站    |
| bird      | 10.0.0.5      | 客户机      |
| cli[1-11] | 10.0.0.[6-17] | 其他客户机    |

如果是首次配置 NIS，仔细思考如何进行就十分重要。无论网络的规模大小如何，都必须进行几个决策。

#### 30.4.4.1.1. 选择 NIS 域名

可能不是要去使用的 "域名(domainname)"。它的通常的叫法，则是 "NIS 域名"。当客户机广播此信息的要求时，它会将 NIS 域的名字作为请求的一部分发出。这样，同一网络上的多个服务器，就能知道如何回应请求。可以把 NIS 域名想象成以某种方式相关的一台主机的名字。

一些机器会直接使用它的 Internet 域名来作为 NIS 域名。并不推荐这样做，因为在网络中，可能会导致不必要的困惑。NIS 域名应该是在网络上唯一的，并且有助于了解它所描述的到底是哪一台机器。例如对于 Acme 公司的美工部，可以考虑使用 "acme-art" 作为 NIS 域名。在例子中，使用的域名是 `test-domain`。

然而，某些操作系统（最著名的是 SunOS™）会使用其 NIS 域名作为 Internet 域名。如果网络上存在包含限制的机器，就必须使用 Internet 域名来作为 NIS 域名。

#### 30.4.4.1.2. 服务器的物理要求

NIS 服务器，需要牢牢钉死一些东西。NIS 的一个不太好的特性就是其客户机对于服务器的依赖程度。如果客户机无法与其 NIS 域的服务器联系，这台机器通常会陷于不可用的状态。缺少用户和密码信息，会使大多数系统陷入短暂的故障。基于此的考虑，可能需要一台不常重新启动，或用于别的机器来承担其任务。如果网络不太忙，也可以使用运行着其他服务的机器来安放 NIS 服务器，只是需要注意，一旦 NIS 服务器不可用，所有的 NIS 客户机都会受到影响。

#### 30.4.4.2. NIS 服务器

所有的 NIS 信息的正本，都被保存在一台单独的称作 NIS 主服务器的机器上。用于保存这些信息的数据，称作 NIS 映射(map)。在 FreeBSD 中，这些映射被保存在 `/var/yp/[domainname]` 里，其中 `[domainname]` 是提供服务的 NIS 域的名字。一台 NIS 服务器，可以同时支持多个域，因此可以建立很多不同的目录，所支持一个域一个。一个域都会有一组独立的映射。

NIS 主和从服务器，通过 `ypserv` 服务程序来处理所有的 NIS 请求。`ypserv` 有责任接收来自 NIS 客户机的请求，翻请求的域，并将名字映射成数据文件的路径，然后将来自数据的数据回答客户机。

##### 30.4.4.2.1. 配置 NIS 主服务器

配置主 NIS 服务器相对而言十分的简单，而其具体步骤取决于你的需要。FreeBSD 提供了一组到位的 NIS 支持。需要做的全部事情，只是在 `/etc/rc.conf` 中加入一些配置，其他工作会由 FreeBSD 完成。

```
nisdomainname="test-domain"
```

1. 一行将在网络（例如重新启动）时，把 NIS 域名配置成 `test-domain`。

```
nis_server_enable="YES"
```

2. 将要求 FreeBSD 在网络子系统之后立即启动 NIS 服务。

```
nis_yppasswdd_enable="YES"
```

3. 将使用 `rpc.yppasswdd` 服务程序，如前面提到的，它允许用户在客户机上修改自己的 NIS 口令。



随 NIS 配置的不同，可能需要添加其他一些项目。请参考 [关于 NIS 服务器同时充当 NIS 客户机](#) 一节，以了解更多的情况。

配置好前面这些配置之后，需要以超用户身份运行 `/etc/netstart` 命令。它会根据 `/etc/rc.conf` 的配置来配置系统中的其他部分。最后，在初始化 NIS 映射之前，需要手工启动 `ypserv` 服务程序：

```
# /etc/rc.d/ypserv start
```

#### 30.4.4.2.2. 初始化 NIS 映射

NIS 映射 是一些数据文件，它位于 `/var/yp` 目录中。这些文件基本上都是根据 NIS 主服务器的 `/etc` 目录自己生成的，唯一的例外是：`/etc/master.passwd` 文件。一般来说，会有非常充分的理由不将 `root` 以及其他管理账号的口令放到所有 NIS 域上的服务器上。因此，在开始初始化 NIS 映射之前，我：

```
# cp /etc/master.passwd /var/yp/master.passwd
# cd /var/yp
# vi master.passwd
```

里，除掉和系统有关的账号的（`bin`、`tty`、`kmem`、`games`，等等），以及其他不希望被散到 NIS 客户端的账号（例如 `root` 和任何其他 UID 0（超用）的账号）。



这个 `/var/yp/master.passwd` 文件是同用，以及其他用不可的（模式 600）！如果需要的，用 `chmod` 命令来改它。

完成些工作之后，就可以初始化 NIS 映射了！FreeBSD 提供了一个名 `ypinit` 的脚本来帮助完成工作（更多信息，见其手册）。注意，这个脚本在大多数 UNIX® 操作系统上都可以到，但并不是所有操作系统都提供。在 Digital UNIX/Compaq Tru64 UNIX 上它的名字是 `ypsetup`。由于我正在生成的是 NIS 主服务器的映射，因此使用 `ypinit` 的 `-m` 参数。如果已完成了上述，要生成 NIS 映射，只需行：

```
ellington# ypinit -m test-domain
Server Type: MASTER Domain: test-domain
Creating an YP server will require that you answer a few questions.
Questions will all be asked at the beginning of the procedure.
Do you want this procedure to quit on non-fatal errors? [y/n: n] n
Ok, please remember to go back and redo manually whatever fails.
If you don't, something might not work.
At this point, we have to construct a list of this domains YP servers.
rod.darktech.org is already known as master server.
Please continue to add any slave servers, one per line. When you are
done with the list, type a <control D>.
master server : ellington
next host to add: coltrane
next host to add: ^D
The current list of NIS servers looks like this:
ellington
coltrane
Is this correct? [y/n: y] y

[..output from map generation..]

NIS Map update completed.
ellington has been setup as an YP master server without any errors.
```

`ypinit` 会根据 `/var/yp/Makefile.dist` 来创建 `/var/yp/Makefile` 文件。创建完之后，这个文件会假定正在操作只有 FreeBSD 机器的服务器 NIS 环境。由于 `test-domain` 有一个从服务器，必须修改 `/var/yp/Makefile`：

```
ellington# vi /var/yp/Makefile
```

能看到一行，其内容是

```
NOPUSH = "True"
```

(如果没有注释掉的)。

#### 30.4.4.2.3. 配置 NIS 从服务器

配置 NIS 从服务器，甚至比配置主服务器要简单。登录到从服务器上，并按照前面的方法，修改 `/etc/rc.conf` 文件。唯一的区别是，在运行 `ypinit` 需要使用 `-s` 参数。这里的 `-s` 选项，同样要求提供 NIS 主服务器的名字，因此我的命令行是：

```
coltrane# ypinit -s ellington test-domain

Server Type: SLAVE Domain: test-domain Master: ellington

Creating an YP server will require that you answer a few questions.
Questions will all be asked at the beginning of the procedure.

Do you want this procedure to quit on non-fatal errors? [y/n: n] n

Ok, please remember to go back and redo manually whatever fails.
If you don't, something might not work.
There will be no further questions. The remainder of the procedure
should take a few minutes, to copy the databases from ellington.
Transferring netgroup...
ypxfr: Exiting: Map successfully transferred
Transferring netgroup.byuser...
ypxfr: Exiting: Map successfully transferred
Transferring netgroup.byhost...
ypxfr: Exiting: Map successfully transferred
Transferring master.passwd.byuid...
ypxfr: Exiting: Map successfully transferred
Transferring passwd.byuid...
ypxfr: Exiting: Map successfully transferred
Transferring passwd.byname...
ypxfr: Exiting: Map successfully transferred
Transferring group.bygid...
ypxfr: Exiting: Map successfully transferred
Transferring group.byname...
ypxfr: Exiting: Map successfully transferred
Transferring services.byname...
```

```

ypxfr: Exiting: Map successfully transferred
Transferring rpc.bynumber...
ypxfr: Exiting: Map successfully transferred
Transferring rpc.byname...
ypxfr: Exiting: Map successfully transferred
Transferring protocols.byname...
ypxfr: Exiting: Map successfully transferred
Transferring master.passwd.byname...
ypxfr: Exiting: Map successfully transferred
Transferring networks.byname...
ypxfr: Exiting: Map successfully transferred
Transferring networks.byaddr...
ypxfr: Exiting: Map successfully transferred
Transferring netid.byname...
ypxfr: Exiting: Map successfully transferred
Transferring hosts.byaddr...
ypxfr: Exiting: Map successfully transferred
Transferring protocols.bynumber...
ypxfr: Exiting: Map successfully transferred
Transferring ypservers...
ypxfr: Exiting: Map successfully transferred
Transferring hosts.byname...
ypxfr: Exiting: Map successfully transferred

coltrane has been setup as an YP slave server without any errors.
Don't forget to update map ypservers on ellington.

```

在会会有一个叫做 `/var/yp/test-domain` 的目。在个目中，保存 NIS 主服务器上的映射的副本。接下来需要定些文件都及地同更新了。在从服务器上，下面的 `/etc/crontab` 将助保一点：

```

20      *      *      *      *      root    /usr/libexec/ypxfr passwd.byname
21      *      *      *      *      root    /usr/libexec/ypxfr passwd.byuid

```

行将制从服务器将映射与主服务器同。由于主服务器会保所有其 NIS 映射的都知会从服务器，因此些并不是必需的。不，由于保持其他客户端的口令信息正性十分重要，而依赖于从服务器，烈推明明指定系常制更新口令映射。于繁忙的网而言，一点尤其重要，因有可能出映射更新不完全的情况。

在，在从服务器上行 `/etc/netstart`，就可以 NIS 服了。

### 30.4.4.3. NIS 客机

NIS 客机会通 `ypbind` 服程序来与特定的 NIS 服务器建立一称作定的系。 `ypbind` 会系的默域 (是通 `domainname` 命令来置的)，并始在本地网上广播 RPC 求。些求会指定 `ypbind` 定的域名。如果已配置了服务器，并且些服务器接到了广播，它将回 `ypbind`，后者服务器的地址。如果有多个可用的服务器 (例如一个主服务器，加上多个从服务器)， `ypbind` 将使用第一个的地址。从一刻始，客机会把所有的 NIS 求直接那个服务器。 `ypbind` 偶会 "ping" 服务器以其仍然在正常行。如果在合理的内没有得到， `ypbind` 会把域未定，并再次起广播，以期到一台服务器。

#### 30.4.4.3.1. 配置 NIS 客户端

配置一台 FreeBSD 机器作为 NIS 客户端是非常简单的。

1. 编辑 `/etc/rc.conf` 文件，并在其中加上下面几行，以配置 NIS 域名，并在网络接口上启用 `ypbind`：

```
nisdomainname="test-domain"
nis_client_enable="YES"
```

2. 要从 NIS 服务器输入所有的口令， 需要从它的 `/etc/master.passwd` 文件中删除所有用 `!` 开头的行， 并使用 `vipw` 在这个文件的最后一行加入：

```
+:::
```



这一行将 NFS 服务器的口令映射中的 `!` 号能登录。也有很多修改这一行来配置 NIS 客户端的方法。 参阅后的 [netgroups 小节](#) 以了解一些的情况。 要了解更多信息，可以参阅 O'Reilly 的 [Managing NFS and NIS](#) 书本。



需要至少保留一个本地 `!` 号（也就是不通 NIS 输入）在它的 `/etc/master.passwd` 文件中， 而 `!` 号是 `wheel` 组的成员。 如果 NIS 组生不， `!` 号可以用来登录， 成为 `root`， 并修正。

3. 要从 NIS 服务器上输入信息， 需要在 `/etc/group` 文件末尾加入：

```
+:*::
```

想要立即启动 NIS 客户端， 需要以超级用户身份运行下列命令：

```
# /etc/netstart
# /etc/rc.d/ypbind start
```

完成这些之后， 就可以通过运行 `ypcat passwd` 来看到 NIS 服务器的口令映射了。

### 30.4.5. NIS 的安全性

基本上， 任何程序都可以发起一个 RPC 到 [ypserv\(8\)](#) 并得到 NIS 映射的内容， 如果程序了解域的域名。 要避免未授权的访问， [ypserv\(8\)](#) 支持一个称为 "securenets" 的特性， 用以将访问限制在一个特定的机器上。 在程序中， [ypserv\(8\)](#) 会从 `/var/yp/securenets` 中加载 `securenet` 信息。



这个路径随 `-p` 参数改变。 这个文件包含了一些行， 每一行中包含了一个网络和子网掩码， 中间用空格分隔。 以 `!` 开头的行会被认为是注释。 示例的 `securenets` 文件如下所示：



```
# 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)** 接到了来自匹配上述任一 IP 地址的请求，它将会正常处理请求。反之，请求将被忽略，并产生一条警告信息。如果 `/var/yp/securenets` 文件不存在，**ypserv** 会允许来自任意主机的请求。

**ypserv** 程序也支持 Wietse Venema 的 TCP Wrapper 软件包。这样，管理员就能使用 TCP Wrapper 的配置文件来代替 `/var/yp/securenets` 完成访问控制。



尽管访问控制机制都能提供某程度的安全，但是，和特端口一样，它无法避免 "IP 伪造" 攻击。防火墙阻止所有与 NIS 有关的访问。

使用 `/var/yp/securenets` 的服务器，可能会无法让某些使用旧版的 TCP/IP 的 NIS 客户机服务。有些客户机可能会在广播时，将主机位都置为 0，或在计算广播地址时忽略子网掩码。尽管有些问题可以通过修改客户机的配置来解决，其他一些问题也可能导致不得不淘汰那些客户机系统，或者不使用 `/var/yp/securenets`。

在使用旧版的 TCP/IP 的系统中，使用 `/var/yp/securenets` 是一个非常糟糕的做法，因为它将导致网络上的 NIS 丧失大部分功能。

使用 TCP Wrapper 软件包，会导致 NIS 服务器的延迟增加。而增加的延迟，可能会导致客户端程序超时，特别是在繁忙的网络或者很慢的 NIS 服务器上。如果某个客户机因此而产生一些异常，它将某些客户机从 NIS 服务器，并限制其绑定自己。

### 30.4.6. 不允许某些用户登录

在我的办公室中，**basie** 是一台机器，是一台教师用的工作站。我不希望将这台机器拿出 NIS 域，而主 NIS 服务器上的 `passwd` 文件，同时包含了教师和学生的账号。应该怎么做？

有一种方法来禁止特定的用户登录机器，即使他本身在 NIS 数据之中。要完成这一工作，只需要在客户机的 `/etc/master.passwd` 文件中加入一些 `-username` 的条目，其中，`username` 是希望禁止登录的用户名。一般推荐使用 **vipw** 来完成这个工作，因为 **vipw** 会在 `/etc/master.passwd` 文件上所作的修改进行合法性检查，并在结束时重新建立口令数据。例如，如果希望禁止用户 **bill** 登录 **basie**，我：



```

basie# vipw
[在末尾加入 -bill, 并退出]
vipw: rebuilding the database...
vipw: done

basie# cat /etc/master.passwd

root:[password]:0:0::0:0:The super-user:/root:/bin/csh
toor:[password]:0:0::0:0:The other super-user:/root:/bin/sh
daemon:*:1:1::0:0:Owner of many system processes:/root:/sbin/nologin
operator:*:2:5::0:0:System &:/sbin/nologin
bin:*:3:7::0:0:Binaries Commands and Source,,,:/sbin/nologin
tty:*:4:65533::0:0:Tty Sandbox:/sbin/nologin
kmem:*:5:65533::0:0:KMem Sandbox:/sbin/nologin
games:*:7:13::0:0:Games pseudo-user:/usr/games:/sbin/nologin
news:*:8:8::0:0:News Subsystem:/sbin/nologin
man:*:9:9::0:0:Mister Man Pages:/usr/shared/man:/sbin/nologin
bind:*:53:53::0:0:Bind Sandbox:/sbin/nologin
uucp:*:66:66::0:0:UUCP pseudo-user:/var/spool/uucppublic:/usr/libexec/uucp/uucico
xten:*:67:67::0:0:X-10 daemon:/usr/local/xten:/sbin/nologin
pop:*:68:6::0:0:Post Office Owner:/nonexistent:/sbin/nologin
nobody:*:65534:65534::0:0:Unprivileged user:/nonexistent:/sbin/nologin
+:+++++
-bill

basie#

```

### 30.4.7. 使用 Netgroups

前一节介绍的方法，在需要非常少的用户和/或机器行特殊的配置时算合适。在更大的网络上，一定会忘记禁止某些用户登录到敏感的机器上，或者，甚至必须单独地修改一台机器的配置，因而掉了 NIS 最重要的优越性：集中式管理。

NIS 开发人员提供的解决方案，被称作 *netgroups*。它的作用和配置，基本上可以等同于 UNIX® 文件系统上使用的。主要的区别是它没有数字化的 ID，以及可以在 *netgroup* 中同时包含用户和其他 *netgroup*。

*Netgroups* 被用来管理大的、复杂的包含数百用户和机器的网络。一方面，在不得不理清情形时，它是一个很有用的东西。而另一方面，它的复杂性又使得通非常简单的例子很解 *netgroup* 到底是什么。下面的其余部分的例子将展示这个。

假如在办公室中成功地部署 NIS 引起了上司的兴趣。接下来的任务是将 NIS 域扩展，以覆盖校中的一些其他的机器。下面这个表格中包括了新用户和新机器，及其要说明。

| 用户名            | 说明        |
|----------------|-----------|
| alpha, beta    | IT 部的普通雇员 |
| charlie, delta | IT 部的学徒   |

| 用户名                                    | 说明                              |
|--|---------------------------------|
| echo, foxtrott, golf, ...              | 普通雇员                            |
| able, baker, ...                       | 目前的实习生                          |
| 机器名                                    | 说明                              |
| war, death, famine, pollution          | 最重要的服务器。只有 IT 部门的雇员才允许登录这些机器。   |
| pride, greed, envy, wrath, lust, sloth | 不太重要的服务器，所有 IT 部门的成员，都可以登录这些机器。 |
| one, two, three, four, ...             | 普通工作站。只有真正的雇员才允许登录这些机器。         |
| trashcan                               | 一台不包含任何数据的旧机器。即使是实习生，也允许登录它。    |

如果可能通过一个一个地阻止用户来实施这些限制，就需要在每一个系统的 `passwd` 文件中，添加一个不允许登录该系统的用户添加新的 `-user` 行。如果忘了任何一个，就可能会造成问题。在行初始配置时，正确地配置也不是什么，但随着日复一日地添加新用户，有一天你会忘记新用户添加某一行。毕竟，Murphy 是一个聪明的人。

使用 `netgroups` 来管理这一状况可以来得多好多。不需要单独地管理一个用户；可以给予用户一个或多个 `netgroups` 身份，并允许或禁止某一个 `netgroup` 的所有成员登录。如果添加了新的机器，只需要定义 `netgroup` 的登录限制。如果增加了新用户，也只需要将用户加入一个或多个 `netgroup`。这些变化是相互独立的：不再需要“每一个用户和机器一行……”。如果系统的 `NIS` 配置出了问题的话，就只需要修改一个中央的配置文件，就能允许或禁止某些台机器的登录了。

第一步是初始化 `NIS` 映射 `netgroup`。FreeBSD 的 [ypinit\(8\)](#) 默认情况下并不创建一个映射，但它的 `NIS` 功能可以在创建映射之后立即为其提供支持。要创建空映射，可以地输入

```
ellington# vi /var/yp/netgroup
```

并开始添加内容。在我举的例子中，至少需要四个 `netgroup`：IT 雇员，IT 学徒，普通雇员和实习生。

```
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)
```

`IT_EMP`、`IT_APP` 等等，是 `netgroup` 的名字。每一个括号中的内容，都有一些用逗号。其中的三个字段是：

1. 在哪些机器上能使用这些。如果不指定主机名，那么在所有机器上都有效。如果指定了主机，则很容易造成混乱。
2. 属于哪个 `netgroup` 的逗号。
3. 逗号的 `NIS` 域。可以从其他 `NIS` 域中把逗号号输入到该 `netgroup` 中，如果管理多个 `NIS` 域的话。

一个字段都可以包括通配符。参 [netgroup\(5\)](#) 了解更多。

Netgroup 的名字一般来不超 8 个字符，特别是当的 NIS 域中有机器打算行其它操作系的时候。名字是区分大小写的；使用大写字母作 netgroup 的名字，能更容易地区分用、机器和 netgroup 的名字。

某些 NIS 客户程序 (FreeBSD 以外的那些) 可能无法理含有大量的 netgroup。例如，某些早期版本的 SunOS™ 会在 netgroup 中包含多于 15 个 出。要个，可以建多个子netgroup，一个中包含少于 15 个用，以及一个包含所有子netgroup 的真正的 netgroup：



```
BIGGRP1 (,joe1,domain) (,joe2,domain) (,joe3,domain) [...]
BIGGRP2 (,joe16,domain) (,joe17,domain) [...]
BIGGRP3 (,joe31,domain) (,joe32,domain)
BIGGROUP BIGGRP1 BIGGRP2 BIGGRP3
```

如果需要超 225 个用，可以重上面的程。

激活并分新的 NIS 映射非常：

```
ellington# cd /var/yp
ellington# make
```

个操作会生成三个 NIS 映射，即 netgroup、netgroup.byhost 和 netgroup.byuser。用 [ypcat\(1\)](#) 可以些 NIS 映射是否可用了：

```
ellington% ypcat -k netgroup
ellington% ypcat -k netgroup.byhost
ellington% ypcat -k netgroup.byuser
```

第一个命令的出，与 /var/yp/netgroup 的内容相近。第二个命令，如果没有指定本机有的 netgroup，没有出。第三个命令，用于示某个用的 netgroup 列表。

客机的置也很。要配置服务器 war，只需入 [vipw\(8\)](#) 并把

```
+:::
```

改

```
+@IT_EMP:::
```

在，只有 netgroup IT\_EMP 中定的用会被入到 war 的口令数据中，因此只有些用能登。

不，个限制也会作用于 shell 的 ~，以及所有在用名和数字用 ID 之施的函数的功能。言之，cd

`~user` 将不会正常工作，而 `ls -l` 也将显示数字的 ID 而不是用名，并且 `find . -user joe -print` 将失败，并输出 `No such user` 的信息。要修正这个问题，需要加入所有的用，而不允许他登录服务器。

可以通过在 `/etc/master.passwd` 加入一行来完成。行的内容是：

`+:::/sbin/nologin`，意思是“加入所有的，但加入的 shell 替换为 `/sbin/nologin`”。通过在 `/etc/master.passwd` 中加默认，可以替换掉 `passwd` 中的任意字段。



必须 `+:::/sbin/nologin` 一行出现在 `+@IT_EMP:::` 之后。否则，所有从 NIS 加入的用号将以 `/sbin/nologin` 作为登录 shell。

完成上面的修改之后，在 IT 部有了新员工，只需修改一个 NIS 映射就足够了。也可以用类似的方法，在不太重要的服务器上，把先前本地版本的 `/etc/master.passwd` 中的 `+:::` 改：

```
+@IT_EMP:::
+@IT_APP:::
+:::/sbin/nologin
```

相关的用于普通工作站的配置是：

```
+@IT_EMP:::
+@USERS:::
+:::/sbin/nologin
```

一切平安无事，直到数周后，有一天策略生了变化：IT 部也开始招收新员工了。IT 员工允许使用普通的终端，以及不太重要的服务器；而 IT 学徒，可以登录主服务器。加了新的 netgroup `IT_INTERN`，以及新的 IT 员工到这个 netgroup 并开始修改一台机器上的配置……老得好：“一，全身”。

NIS 通过 netgroup 来建立 netgroup 的能力，正可以避免的情形。一种可能的方法是建立基于角色的 netgroup。例如，可以建称 `BIGSRV` 的 netgroup，用于定义最重要的服务器上的登录限制，以及一个成 `SMALLSRV` 的 netgroup，用以定义次要的服务器，以及第三个，用于普通工作站的 netgroup `USERBOX`。三个 netgroup 中的一个，都包含了允许登录到一些机器上的所有 netgroup。的 NIS 映射中的新如下所示：

```
BIGSRV    IT_EMP  IT_APP
SMALLSRV  IT_EMP  IT_APP  ITINTERN
USERBOX   IT_EMP  ITINTERN  USERS
```

定义登录限制的方法，在可能将机器分类并加以限制的时候可以工作的相当好。不幸的是，是例外，而非情况。多数时候，需要按机器去定义登录限制。

与机器相关的 netgroup 定义，是理上述策略改的一种可能的方法。此，每台机器的 `/etc/master.passwd` 中，都包含一个“+”号的行。第一个用于添加允许登录的 netgroup 号，而第二个用于加其它号，并把 shell 置为 `/sbin/nologin`。使用“全大写”的机器名作为 netgroup 名是个好主意。言之，些行似乎于：

```
+@BOXNAME::::::::::
+::::::::::/sbin/nologin
```

一旦在所有机器上都完成了`/etc/master.passwd`的修改，就再也不需要修改本地的`/etc/master.passwd`了。所有未来的修改都可以在 NIS 映射中`行`。`里`是一个例子，其中展示了在`一`用情景中所需要的 netgroup 映射，以及其它一些常用的技巧：

```
# Define groups of users first
IT_EMP    (,alpha,test-domain)    (,beta,test-domain)
IT_APP    (,charlie,test-domain)  (,delta,test-domain)
DEPT1     (,echo,test-domain)     (,foxtrott,test-domain)
DEPT2     (,golf,test-domain)     (,hotel,test-domain)
DEPT3     (,india,test-domain)    (,juliet,test-domain)
ITINTERN  (,kilo,test-domain)     (,lima,test-domain)
D_INTERNS (,able,test-domain)     (,baker,test-domain)
#
# Now, define some groups based on roles
USERS     DEPT1  DEPT2  DEPT3
BIGSRV    IT_EMP IT_APP
SMALLSRV   IT_EMP IT_APP ITINTERN
USERBOX    IT_EMP ITINTERN  USERS
#
# And a groups for a special tasks
# Allow echo and golf to access our anti-virus-machine
SECURITY  IT_EMP (,echo,test-domain) (,golf,test-domain)
#
# machine-based netgroups
# Our main servers
WAR       BIGSRV
FAMINE    BIGSRV
# User india needs access to this server
POLLUTION BIGSRV (,india,test-domain)
#
# This one is really important and needs more access restrictions
DEATH     IT_EMP
#
# The anti-virus-machine mentioned above
ONE       SECURITY
#
# Restrict a machine to a single user
TWO       (,hotel,test-domain)
# [...more groups to follow]
```

如果`正`使用某`数据`来管理`号`，`可以使用`的数据`的`告工具来`建`映射的第一部分。`，` 新用`就`自`地`可以`些`机器了。

最后的提醒：使用基于机器的 netgroup 并不`是`用的。如果正在`学生`室部署数十台甚至上百台同`的`机器，`使用`基于角色的 netgroup，而不是基于机器的 netgroup，以便把 NIS

映射的尺寸保持在一个合理的范围内。

### 30.4.8. 需要牢记的事

这里是一些其它在使用 NIS 环境需要注意的地方。

- 首次需要在网室中添加新用户，必须只在 NIS 服务器上加入用户，而且一定要记得重建 NIS 映射。如果忘了这么做，新用户将无法登录除 NIS 主服务器之外的任何其它机器。例如，如果要在网室添加新用户 `jsmith`，我需要：

```
# pw useradd jsmith
# cd /var/yp
# make test-domain
```

也可以运行 `adduser jsmith` 而不是 `pw useradd jsmith`。

- 将管理用的 ID 号排除在 NIS 映射之外。一般来，不希望这些管理 ID 号和口令被分散到那些包含不使用它的用户的机器上。
- 保持 NIS 主和从服务器的安全，并尽可能少其停机时间。如果有人攻入或恶意地破坏这些机器，整个网室的任也就无法登录了。

网是集中式管理系统中最薄弱的环节。如果没有保护好 NIS 服务器，就有大批愤怒的用户需要对付了！

### 30.4.9. NIS v1 兼容性

FreeBSD 的 `ypserv` 提供了某些 NIS v1 客户端提供服务的支持能力。FreeBSD 的 NIS 软件，只使用 NIS v2 软件，但其它软件可能会包含 v1 软件，以提供旧系统的向下兼容能力。随某些系统提供的 `ypbind` 服务将首先确定 NIS v1 服务器，即使它并不真的需要它（有些甚至可能会一直广播搜索请求，即使已从某台 v2 服务器得到了回答也是如此）。注意，尽管支持一般的客户端应用，这个版本的 `ypserv` 并不能处理 v1 的映射请求；因而，它就不能与早期的支持 v1 软件的 NIS 服务器配合使用，无论是作主服务器还是从服务器。幸运的是，目前还没有仍然在用的旧的服务器了。

### 30.4.10. 同作 NIS 客户端的 NIS 服务器

在多服务器域的环境中，如果服务器同作 NIS 客户端，在运行 `ypserv` 时要特别小心。一般来，制服服务器规定自己要比允许它广播请求要好，因某些情况下它可能会相互规定。某些怪异的故障，很可能是由于某一台服务器停机，而其它服务器都依其服务所致的。最后，所有的客户端都会超并规定到其它服务器，但个延迟可能会相当可观，而且恢复之后仍然存在再次产生此问题的隐患。

可以控制一台机器规定到特定的服务器，是通过 `ypbind` 的 `-S` 参数来完成的。如果不希望每次 NIS 服务器都手工完成这项工作，可以在 `/etc/rc.conf` 中加入：

```
nis_client_enable="YES" # run client stuff as well
nis_client_flags="-S NIS domain,server"
```

参 [ypbind\(8\)](#) 以了解更多情况。

### 30.4.11. 口令格式

在 NIS 中，口令格式的兼容性是最常的。假如的 NIS 服务器使用 DES 加密口令，它只能支持使用 DES 的客户端。例如，如果的网路上有 Solaris™ NIS 客户端，几乎肯定需要使用 DES 加密口令。

要服务器的客户端使用的口令格式，需要看 `/etc/login.conf`。如果主机被配置使用 DES 加密的口令，`default class` 将包含似：

```
default:\n    :passwd_format=des:\n    :copyright=/etc/COPYRIGHT:\n    [Further entries elided]
```

其他一些可能的 `passwd_format` 包括 `blf` 和 `md5` (分属于 Blowfish 和 MD5 加密口令)。

如果修改了 `/etc/login.conf`，就必须重建登录性能数据，这是通以 `root` 身运行下面的程序来完成的：

```
# cap_mkdb /etc/login.conf
```



已在 `/etc/master.passwd` 中的口令的格式不会被更新，直到用在登录性能数据重建之后首次修改口令止。

接下来，了保所有的口令都按照的格式加密了，需要 `/etc/auth.conf` 中 `crypt_default` 出的先的口令格式。要完成此工作，将的格式放到列表的第一。例如，当使用 DES 加密的口令，：

```
crypt_default = des blf md5
```

在一台基于 FreeBSD 的 NIS 服务器和客户端上完成上述工作之后，就可以肯定网路上它都在使用同的口令格式了。如果在 NIS 客户端上做身生，也是第一个可能出的地方。注意：如果希望在混合的网路上部署 NIS 服务器，可能就需要在所有系上都使用 DES，因是所有系都能支持的最低限度的公共准。

## 30.5. 网自配置 (DHCP)

### 30.5.1. 什么是 DHCP？

DHCP，主机配置，是一系得以接到网路上，并取所需要的配置参数手段。FreeBSD 使用来自 OpenBSD 3.7 的 OpenBSD `dhclient`。里提供的所有于 `dhclient` 的信息，都是以 ISC 或 OpenBSD DHCP 客户端程序准的。DHCP 服务器是 ISC 件包的一部分。

### 30.5.2. 一都介些内容

描述了 ISC 和 DHCP 系中的客户端，以及和 ISC DHCP 系中的服务器端的件。客户端程序，`dhclient`，是随 FreeBSD 作它的一部分提供的；而服务器部分，可以通 `net/isc-dhcp31-server` port

得到。 [dhclient\(8\)](#)、 [dhcp-options\(5\)](#)、 以及 [dhclient.conf\(5\)](#) 的手册， 加上下面所介绍的参考文献， 都是非常有用的来源。

### 30.5.3. 它如何工作

当 DHCP 客户端程序， [dhclient](#) 在客户端机上运行， 它会开始广播请求配置信息的信息。 默认情况下， 这些请求是在 UDP 端口 68 上。 服务器通过 UDP 67 端口， 向客户端提供一个 IP 地址， 以及其他有用的配置参数， 例如子网掩码、 路由器， 以及 DNS 服务器。 所有这些信息都会以 DHCP "lease" 的形式发出， 并且只在一段特定的时间内有效（是由 DHCP 服务器的所有者配置的）。 因此， 那些已断开网络的客户端使用的旧的 IP 地址就能被自动地回收了。

DHCP 客户端程序可以从服务器端获取大量的信息。 关于能获取的信息的完整列表， 请参考 [dhcp-options\(5\)](#)。

### 30.5.4. FreeBSD 集成

FreeBSD 完全地集成了 OpenBSD 的 DHCP 客户端， [dhclient](#)。 DHCP 客户端支持在安装程序和基本系统中均有提供， 这使得不再需要去了解那些已运行了 DHCP 服务器的网络的具体配置参数。

sysinstall 能支持 DHCP。 在 sysinstall 中配置网络接口， 它的第二个问题便是： "Do you want to try DHCP configuration of the interface? (是否希望在此接口上安装 DHCP 配置?)". 如果做肯定的回答， 将运行 [dhclient](#)， 一旦成功， 将自动地填写网络配置信息。

要在系统中使用 DHCP， 必须做两件事：

- 系统的内核中， 必须包含 bpf 模块。 如果需要这么做， 需要将 `device bpf` 添加到内核的配置文件， 并重新编译内核。 要了解关于内核的更多信息， 请参考 [配置FreeBSD的内核](#)。

bpf 模块已是 FreeBSD 发行版中默认的 GENERIC 内核的一部分了， 因此如果没有内核定制， 不用构建一个新的内核配置文件， DHCP 就能工作了。



对于那些安全意识很强的人来， 他们知道 bpf 也是包监听工具能正确工作的条件之一（当然， 它需要以 `root` 身份运行才行）。 bpf 是使用 DHCP 所必需的， 但如果安全非常敏感， 很可能会有理由不把 bpf 加入到系统的内核配置中， 直到真的需要使用 DHCP 为止。

- 编辑 `/etc/rc.conf` 并加入下面的设置：

```
ifconfig_fxp0="DHCP"
```



必须将 `fxp0` 替换成希望自动配置的网络接口的名字， 可以在 [设置网络](#) 找到更详细的介绍。

如果希望使用同一位置的 [dhclient](#)， 或者需要 [dhclient](#) 的其他参数， 可以添加下面的配置（根据需要自行修改）：

```
dhclient_program="/sbin/dhclient"
dhclient_flags=""
```



DHCP 服务器，`dhcpcd`，是作为 [net/isc-dhcp31-server](#) port 的一部分提供的。这个 port 包括了 ISC DHCP 服务器及其文档。

### 30.5.5. 文件

- `/etc/dhclient.conf`

`dhclient` 需要一个配置文件，`/etc/dhclient.conf`。一般来说，这个文件中只包括注释，而默认值基本上都是合理的。这个配置文件在 [dhclient.conf\(5\)](#) 手册中进行了详细的描述。

- `/sbin/dhclient`

`dhclient` 是一个静悄悄的，它被安装到 `/sbin` 中。[dhclient\(8\)](#) 手册指出了关于 `dhclient` 的一些细节。

- `/sbin/dhclient-script`

`dhclient-script` 是一个 FreeBSD 使用的 DHCP 客户端配置脚本。在 [dhclient-script\(8\)](#) 中它进行了描述，但一般来说，用者不需要对其做任何修改，就能一切正常了。

- `/var/db/dhclient.leases`

DHCP 客户端程序会维护一个数据库来保存有效的 lease，它被以日志的形式保存到该文件中。[dhclient.leases\(5\)](#) 出了更详细的介绍。

### 30.5.6. 文档

DHCP 文档的完整描述是 [RFC 2131](#)。关于它的其他信息来源的站点 <http://www.dhcp.org/> 也提供了详尽的资料。

## 30.5.7. 安装和配置 DHCP 服务器

### 30.5.7.1. 本章包含哪些内容

本章提供了关于如何在 FreeBSD 系统上使用 ISC (Internet 系统组) 的 DHCP 套件来架设 DHCP 服务器的信息。

DHCP 套件中的服务器部分并没有作为 FreeBSD 的一部分来提供，因此需要安装 [net/isc-dhcp31-server](#) port 才能提供该服务。请参考 [安装应用程序. Packages 和 Ports](#) 以了解关于如何使用 Ports Collection 的情况。

### 30.5.7.2. 安装 DHCP 服务器

为了在 FreeBSD 系统上进行配置以便作为 DHCP 服务器来使用，需要把 [bpf\(4\)](#) 加载到内核。要完成这项工作，需要将 `device bpf` 加入到内核配置文件中，并重新加载内核。要得到关于如何加载内核的一些信息，请参考 [配置FreeBSD的内核](#)。

`bpf` 是 FreeBSD 所附带的 GENERIC 内核中已包含的组件，因此并不需要为了 DHCP 正常工作而特地定制内核。



如果有安全意， 注意 bpf 同也是监听程序能正工作的 (尽管程序仍然需要以特用身行)。 bpf是 使用 DHCP 所必需的， 但如果安全非常敏感， 可能会不希望将 bpf 放内核， 直到真的 DHCP 是必需的止。

接下来要做的是示的 dhcpd.conf， 它由 [net/isc-dhcp31-server](#) port 安装。默情况下， 它的名字是 /usr/local/etc/dhcpd.conf.sample， 在始修改之前， 需要把它制 /usr/local/etc/dhcpd.conf。

### 30.5.7.3. 配置 DHCP 服器

dhcpd.conf 包含了一系列于子网和主机的定， 下面的例子可以助理解它：

```
option domain-name "example.com";①
option domain-name-servers 192.168.4.100;②
option subnet-mask 255.255.255.0;③

default-lease-time 3600;④
max-lease-time 86400;⑤
ddns-update-style none;⑥

subnet 192.168.4.0 netmask 255.255.255.0 {
    range 192.168.4.129 192.168.4.254;⑦
    option routers 192.168.4.1;⑧
}

host mailhost {
    hardware ethernet 02:03:04:05:06:07;⑨
    fixed-address mailhost.example.com;⑩
}
```

- ① 个指定了提供客机作默搜索域的域名。参考 [resolv.conf\(5\)](#) 以了解于一概念的。
- ② 个用于指定一客机使用的 DNS 服器， 它之以逗号分隔。
- ③ 提供客机的子网掩。
- ④ 客机可以求租的有效期， 而如果没有， 服器将指定一个租有效期， 也就是个 (位是秒)。
- ⑤ 是服器允租出地址的最大。 如果客机求了更的租期， 它将得到一个地址， 但其租期限于 **max-lease-time** 秒。
- ⑥ 个用于指定 DHCP 服器在一个地址被接受或放是否更新 DNS。 在 ISC 中， 一是必指定的。
- ⑦ 指定地址池中可以用来分配客机的 IP 地址。 在个之， 以及其界的 IP 地址将分配客机。
- ⑧ 定客机的默网。
- ⑨ 主机的硬件 MAC 地址 (DHCP 服器就能在接到求知道求的主机身)。
- ⑩ 指定是得到同一 IP 地址的主机。 注意在此使用主机名是的， 因 DHCP 服器会在返回租借地址信息之前自行解析主机名。

在配制好 dhcpd.conf 之后， 在 /etc/rc.conf 中用 DHCP 服器， 也就是加：

```
dhcpd_enable="YES"
dhcpd_ifaces="dc0"
```

此处的 `dc0` 接口名可改为 DHCP 服务器需要监听 DHCP 客户端请求的接口 (如果有多个, 用空格分隔)。

接下来, 可以用下面的命令来启动服务:

```
# /usr/local/etc/rc.d/isc-dhcpd start
```

如果未来需要修改服务器的配置, 必须牢牢记送 `SIGHUP` 信号给 `dhcpd` 并不会导致配置文件的重新加载, 而在其他服务程序中则是比较普遍的规定。需要送 `SIGTERM` 信号来停止进程, 然后使用上面的命令来重新启动它。

#### 30.5.7.4. 文件

- `/usr/local/sbin/dhcpd`

`dhcpd` 是静默启动的, 并安装到 `/usr/local/sbin` 中。随 `port` 安装的 [dhcpd\(8\)](#) 手册提供了关于 `dhcpd` 更详尽的信息。

- `/usr/local/etc/dhcpd.conf`

`dhcpd` 需要配置文件, 即 `/usr/local/etc/dhcpd.conf` 才能向客户端提供服务。该文件需要包括提供客户端的所有信息, 以及关于服务器运行的其他信息。此配置文件的描述可以在随 `port` 安装的 [dhcpd.conf\(5\)](#) 手册上找到。

- `/var/db/dhcpd.leases`

DHCP 服务器会维护一个它的租用地址数据库, 并保存在该文件中, 该文件是以日志的形式保存的。随 `port` 安装的 [dhcpd.leases\(5\)](#) 手册提供了更详细的描述。

- `/usr/local/sbin/dhcrelay`

`dhcrelay` 在更复杂的场景中, 可以用来支持使用 DHCP 服务器请求一个独立网络上的 DHCP 服务器。如果需要该功能, 需要安装 `net/isc-dhcp31-relay` port。 [dhcrelay\(8\)](#) 手册提供了更详尽的介绍。

## 30.6. 域名系统 (DNS)

### 30.6.1. 简介

FreeBSD 在默认情况下使用一个版本的 BIND (Berkeley Internet Name Domain), 是目前最流行的 DNS 实现。DNS 是一个系统, 可以通过它将域名同 IP 地址相互映射。例如, 访问 `www.FreeBSD.org` 将得到 FreeBSD Project 的 web 服务器的 IP 地址, 而访问 `ftp.FreeBSD.org` 将得到 FTP 机器的 IP 地址。类似地, 也可以做相反的事情。即 IP 地址可以得到其主机名。当然, 完成 DNS 查询并不需要在系统中运行域名服务器。

目前, 默认情况下 FreeBSD 使用的是 BIND9 DNS 服务器。我们内建于系统中的版本提供了较高的安全特性、

新的文件目录，以及自己的 `chroot(8)` 配置。

在 Internet 上的 DNS 是一套权威的根域名系，顶级域名 (TLD)，以及一系列小模块的，提供少量域名解析服务并域名信息进行存储的域名服务器组成的。

目前，BIND 由 Internet Systems Consortium <https://www.isc.org/> 维护。

### 30.6.2. 术语

要理解本文，需要首先了解一些相关的 DNS 术语。

| 术语             | 定义  |
|----------------|---|
| 正向 DNS         | 将域名映射到 IP 地址                                    |
| 原点 (Origin)    | 表示特定域文件所在的域                                     |
| named, BIND    | 在 FreeBSD 中 BIND 域名服务器软件包的常用叫法。                 |
| 解析器 (Resolver) | 计算机用以向域名服务器查询域名信息的一个系统程序                        |
| 反向 DNS         | 将 IP 地址映射回主机名                                   |
| 根域             | Internet 域层次的起点。所有的域都在根域之下，类似文件系统中，文件都在根目录之下那样。 |
| 域 (Zone)       | 独立的域，子域，或者由同一机器管理的 DNS 的一部分。                    |

域的例子：

- `.` 在本文中通常指代根域。
- `org.` 是根域之下的一个顶级域名 (TLD)。
- `example.org.` 是在 `org.` TLD 之下的一个域。
- `1.168.192.in-addr.arpa` 是一个表示所有 `192.168.1.*` IP 地址空间中 IP 地址的域。

如前所述，域名中越左的部分会越靠左出。例如，`example.org.` 就比 `org.` 更小，类似地 `org.` 又比根域更小。域名各个部分的格局与文件系统十分相似：`/dev` 目录在根目录之下，等等。

### 30.6.3. 运行域名服务器的理由

域名服务器通常会有两种形式：权威域名服务器，以及缓存域名服务器。

下列情况需要有权威域名服务器：

- 想要向全世界提供 DNS 信息，并请求出权威答案。
- 注册了类似 `example.org` 的域，而需要将 IP 指定到其下的主机名上。
- 某个 IP 地址需要反向 DNS 查询 (IP 到主机名)。
- 主服务器，或常备的从 (slave) 服务器，会在主服务器出错或无法回来时回答请求。

下列情况需要有缓存域名服务器：

- 本地的 DNS 服务器能缓存， 并比直接向外界的域名服务器请求更快地得到回答。

当有人向 [www.FreeBSD.org](http://www.FreeBSD.org) 问， 解析器通常会向上游 ISP 的域名服务器发出请求， 并得到回答。 如果有本地的缓存 DNS 服务器， 则只有在第一次被缓存 DNS 服务器问到外部世界。 其他的则不会向局域网外， 因为它已缓存有本地的缓存了。

### 30.6.4. DNS 如何工作

在 FreeBSD 中， BIND 服务器程序被称为 `named`。

| 文件                                  | 描述              |
|-------------------------------------|-----------------|
| <a href="#">named(8)</a>            | BIND 服务器程序      |
| <a href="#">rndc(8)</a>             | 域名服务器控制程序       |
| <code>/etc/namedb</code>            | BIND 存放域名信息的位置。 |
| <code>/etc/namedb/named.conf</code> | 域名服务器配置文件       |

随在服务器上配置的域的性质不同， 域的配置文件一般会存放到 `/etc/namedb` 目录中的 `master`、 `slave`， 或 `dynamic` 子目录中。 这些文件中提供了域名服务器在操作所需要的 DNS 信息。

### 30.6.5. 安装 BIND

由于 BIND 是默认安装的， 因此配置它相对而言很简单。

默认的 `named` 配置， 是在 [chroot\(8\)](#) 环境中提供基本的域名解析服务， 并且只限于监听本地 IPv4 回环地址 (127.0.0.1)。 如果希望进一步配置， 可以使用下面的命令：

```
# /etc/rc.d/named onestart
```

如果希望 `named` 服务器在每次重启的时候都能启动， 需要在 `/etc/rc.conf` 中加入：

```
named_enable="YES"
```

当然， 除了本文所介绍的配置选项之外， 在 `/etc/namedb/named.conf` 中还有很多其它的选项。 不过， 如果你需要了解 FreeBSD 中用于配置 `named` 的那些选项， 你可以看看 `/etc/defaults/rc.conf` 中的 `named_*` 参数， 并参考 [rc.conf\(5\)](#) 手册。 除此之外， 在 [FreeBSD 中使用 rc](#) 也是一个不错的起点。

### 30.6.6. 配置文件

目前， `named` 的配置文件存放于 `/etc/namedb` 目录， 在使用前可根据需要自行修改， 除非你只打算让它完成基本的域名解析服务。 这个目录同时也是进行大多数配置的地方。

#### 30.6.6.1. /etc/namedb/named.conf

```
// $FreeBSD$
//
```



```
// set named_auto_forward=yes in /etc/rc.conf. You can also enable
// named_auto_forward_only (the effect of which is described above).
// include "/etc/namedb/auto_forward.conf";
```

正如注Ⓐ所言，如果希望从上Ⓐ存中受益，可以在此Ⓐ用 **forwarders**。正常情况下，域名服Ⓐ器会逐Ⓐ地Ⓐ Internet 来Ⓐ到特定的域名服Ⓐ器，直到得到答案Ⓐ止。Ⓐ个Ⓐ将Ⓐ它首先Ⓐ上Ⓐ域名服Ⓐ器（或Ⓐ外提供的域名服Ⓐ器），从而从它Ⓐ的Ⓐ存中得到Ⓐ果。如果上Ⓐ域名服Ⓐ器是一个繁忙的高速域名服Ⓐ器，Ⓐ用它将有助于改善服Ⓐ品Ⓐ。



**127.0.0.1**不会正常工作。一定要把地址改Ⓐ上Ⓐ服Ⓐ器的 IP 地址。

```
/*
Modern versions of BIND use a random UDP port for each outgoing
query by default in order to dramatically reduce the possibility
of cache poisoning. All users are strongly encouraged to utilize
this feature, and to configure their firewalls to accommodate it.

AS A LAST RESORT in order to get around a restrictive firewall
policy you can try enabling the option below. Use of this option
will significantly reduce your ability to withstand cache poisoning
attacks, and should be avoided if at all possible.

Replace NNNNN in the example with a number between 49160 and 65530.
*/
// query-source address * port NNNNN;
};

// If you enable a local name server, don't forget to enter 127.0.0.1
// first in your /etc/resolv.conf so this server will be queried.
// Also, make sure to enable it in /etc/rc.conf.

// The traditional root hints mechanism. Use this, OR the slave zones below.
zone "." { type hint; file "named.root"; };

/* Slaving the following zones from the root name servers has some
significant advantages:
1. Faster local resolution for your users
2. No spurious traffic will be sent from your network to the roots
3. Greater resilience to any potential root server failure/DDoS

On the other hand, this method requires more monitoring than the
hints file to be sure that an unexpected failure mode has not
incapacitated your server. Name servers that are serving a lot
of clients will benefit more from this approach than individual
hosts. Use with caution.

To use this mechanism, uncomment the entries below, and comment
the hint zone above.
*/
```

```

/*
zone "." {
    type slave;
    file "slave/root.slave";
    masters {
        192.5.5.241;    // F.ROOT-SERVERS.NET.
    };
    notify no;
};
zone "arpa" {
    type slave;
    file "slave/arpa.slave";
    masters {
        192.5.5.241;    // F.ROOT-SERVERS.NET.
    };
    notify no;
};
zone "in-addr.arpa" {
    type slave;
    file "slave/in-addr.arpa.slave";
    masters {
        192.5.5.241;    // F.ROOT-SERVERS.NET.
    };
    notify no;
};
*/

/* Serving the following zones locally will prevent any queries
   for these zones leaving your network and going to the root
   name servers. This has two significant advantages:
   1. Faster local resolution for your users
   2. No spurious traffic will be sent from your network to the roots
*/
// RFC 1912
zone "localhost"    { type master; file "master/localhost-forward.db"; };
zone "127.in-addr.arpa" { type master; file "master/localhost-reverse.db"; };
zone "255.in-addr.arpa" { type master; file "master/empty.db"; };

// RFC 1912-style zone for IPv6 localhost address
zone "0.ip6.arpa"   { type master; file "master/localhost-reverse.db"; };

// "This" Network (RFCs 1912 and 3330)
zone "0.in-addr.arpa"    { type master; file "master/empty.db"; };

// Private Use Networks (RFC 1918)
zone "10.in-addr.arpa"   { type master; file "master/empty.db"; };
zone "16.172.in-addr.arpa" { type master; file "master/empty.db"; };
zone "17.172.in-addr.arpa" { type master; file "master/empty.db"; };
zone "18.172.in-addr.arpa" { type master; file "master/empty.db"; };
zone "19.172.in-addr.arpa" { type master; file "master/empty.db"; };
zone "20.172.in-addr.arpa" { type master; file "master/empty.db"; };

```



```

zone "21.172.in-addr.arpa" { type master; file "master/empty.db"; };
zone "22.172.in-addr.arpa" { type master; file "master/empty.db"; };
zone "23.172.in-addr.arpa" { type master; file "master/empty.db"; };
zone "24.172.in-addr.arpa" { type master; file "master/empty.db"; };
zone "25.172.in-addr.arpa" { type master; file "master/empty.db"; };
zone "26.172.in-addr.arpa" { type master; file "master/empty.db"; };
zone "27.172.in-addr.arpa" { type master; file "master/empty.db"; };
zone "28.172.in-addr.arpa" { type master; file "master/empty.db"; };
zone "29.172.in-addr.arpa" { type master; file "master/empty.db"; };
zone "30.172.in-addr.arpa" { type master; file "master/empty.db"; };
zone "31.172.in-addr.arpa" { type master; file "master/empty.db"; };
zone "168.192.in-addr.arpa" { type master; file "master/empty.db"; };

// Link-local/APIPA (RFCs 3330 and 3927)
zone "254.169.in-addr.arpa" { type master; file "master/empty.db"; };

// TEST-NET for Documentation (RFC 3330)
zone "2.0.192.in-addr.arpa" { type master; file "master/empty.db"; };

// Router Benchmark Testing (RFC 3330)
zone "18.198.in-addr.arpa" { type master; file "master/empty.db"; };
zone "19.198.in-addr.arpa" { type master; file "master/empty.db"; };

// IANA Reserved - Old Class E Space
zone "240.in-addr.arpa" { type master; file "master/empty.db"; };
zone "241.in-addr.arpa" { type master; file "master/empty.db"; };
zone "242.in-addr.arpa" { type master; file "master/empty.db"; };
zone "243.in-addr.arpa" { type master; file "master/empty.db"; };
zone "244.in-addr.arpa" { type master; file "master/empty.db"; };
zone "245.in-addr.arpa" { type master; file "master/empty.db"; };
zone "246.in-addr.arpa" { type master; file "master/empty.db"; };
zone "247.in-addr.arpa" { type master; file "master/empty.db"; };
zone "248.in-addr.arpa" { type master; file "master/empty.db"; };
zone "249.in-addr.arpa" { type master; file "master/empty.db"; };
zone "250.in-addr.arpa" { type master; file "master/empty.db"; };
zone "251.in-addr.arpa" { type master; file "master/empty.db"; };
zone "252.in-addr.arpa" { type master; file "master/empty.db"; };
zone "253.in-addr.arpa" { type master; file "master/empty.db"; };
zone "254.in-addr.arpa" { type master; file "master/empty.db"; };

// IPv6 Unassigned Addresses (RFC 4291)
zone "1.ip6.arpa" { type master; file "master/empty.db"; };
zone "3.ip6.arpa" { type master; file "master/empty.db"; };
zone "4.ip6.arpa" { type master; file "master/empty.db"; };
zone "5.ip6.arpa" { type master; file "master/empty.db"; };
zone "6.ip6.arpa" { type master; file "master/empty.db"; };
zone "7.ip6.arpa" { type master; file "master/empty.db"; };
zone "8.ip6.arpa" { type master; file "master/empty.db"; };
zone "9.ip6.arpa" { type master; file "master/empty.db"; };
zone "a.ip6.arpa" { type master; file "master/empty.db"; };
zone "b.ip6.arpa" { type master; file "master/empty.db"; };

```

```

zone "c.ip6.arpa"      { type master; file "master/empty.db"; };
zone "d.ip6.arpa"      { type master; file "master/empty.db"; };
zone "e.ip6.arpa"      { type master; file "master/empty.db"; };
zone "0.f.ip6.arpa"    { type master; file "master/empty.db"; };
zone "1.f.ip6.arpa"    { type master; file "master/empty.db"; };
zone "2.f.ip6.arpa"    { type master; file "master/empty.db"; };
zone "3.f.ip6.arpa"    { type master; file "master/empty.db"; };
zone "4.f.ip6.arpa"    { type master; file "master/empty.db"; };
zone "5.f.ip6.arpa"    { type master; file "master/empty.db"; };
zone "6.f.ip6.arpa"    { type master; file "master/empty.db"; };
zone "7.f.ip6.arpa"    { type master; file "master/empty.db"; };
zone "8.f.ip6.arpa"    { type master; file "master/empty.db"; };
zone "9.f.ip6.arpa"    { type master; file "master/empty.db"; };
zone "a.f.ip6.arpa"    { type master; file "master/empty.db"; };
zone "b.f.ip6.arpa"    { type master; file "master/empty.db"; };
zone "0.e.f.ip6.arpa"  { type master; file "master/empty.db"; };
zone "1.e.f.ip6.arpa"  { type master; file "master/empty.db"; };
zone "2.e.f.ip6.arpa"  { type master; file "master/empty.db"; };
zone "3.e.f.ip6.arpa"  { type master; file "master/empty.db"; };
zone "4.e.f.ip6.arpa"  { type master; file "master/empty.db"; };
zone "5.e.f.ip6.arpa"  { type master; file "master/empty.db"; };
zone "6.e.f.ip6.arpa"  { type master; file "master/empty.db"; };
zone "7.e.f.ip6.arpa"  { type master; file "master/empty.db"; };

// IPv6 ULA (RFC 4193)
zone "c.f.ip6.arpa"    { type master; file "master/empty.db"; };
zone "d.f.ip6.arpa"    { type master; file "master/empty.db"; };

// IPv6 Link Local (RFC 4291)
zone "8.e.f.ip6.arpa"  { type master; file "master/empty.db"; };
zone "9.e.f.ip6.arpa"  { type master; file "master/empty.db"; };
zone "a.e.f.ip6.arpa"  { type master; file "master/empty.db"; };
zone "b.e.f.ip6.arpa"  { type master; file "master/empty.db"; };

// IPv6 Deprecated Site-Local Addresses (RFC 3879)
zone "c.e.f.ip6.arpa"  { type master; file "master/empty.db"; };
zone "d.e.f.ip6.arpa"  { type master; file "master/empty.db"; };
zone "e.e.f.ip6.arpa"  { type master; file "master/empty.db"; };
zone "f.e.f.ip6.arpa"  { type master; file "master/empty.db"; };

// IP6.INT is Deprecated (RFC 4159)
zone "ip6.int"         { type master; file "master/empty.db"; };

// NB: Do not use the IP addresses below, they are faked, and only
// serve demonstration/documentation purposes!
//
// Example slave zone config entries. It can be convenient to become
// a slave at least for the zone your own domain is in. Ask
// your network administrator for the IP address of the responsible
// master name server.
//

```

```
// Do not forget to include the reverse lookup zone!
// This is named after the first bytes of the IP address, in reverse
// order, with ".IN-ADDR.ARPA" appended, or ".IP6.ARPA" for IPv6.
//
// Before starting to set up a master zone, make sure you fully
// understand how DNS and BIND work. There are sometimes
// non-obvious pitfalls. Setting up a slave zone is usually simpler.
//
// NB: Don't blindly enable the examples below. :-) Use actual names
// and addresses instead.

/* An example dynamic zone
key "exampleorgkey" {
    algorithm hmac-md5;
    secret "sf87HJqjkqh8ac87a021la==";
};
zone "example.org" {
    type master;
    allow-update {
        key "exampleorgkey";
    };
    file "dynamic/example.org";
};
*/

/* Example of a slave reverse zone
zone "1.168.192.in-addr.arpa" {
    type slave;
    file "slave/1.168.192.in-addr.arpa";
    masters {
        192.168.1.1;
    };
};
*/
```

在 `named.conf` 中， 出了从域、 域和反解析域的例子。

如果新了域， 就必需在 `named.conf` 中加入 的 目。

例如， 用于 `example.org` 的域文件的描述 似下面 ：

```
zone "example.org" {
    type master;
    file "master/example.org";
};
```

如 `type` 句所 示的那， 是一个主域， 其信息保存在 `/etc/namedb/master/example.org` 中， 如 `file` 句所示。

```
zone "example.org" {
    type slave;
    file "slave/example.org";
};
```

在从域的情形中， 所指定的域的信息会从主域名服务器那里来， 并保存到该的文件中。 当主域名服务器启动或不可用， 从域名服务器就有一可用的域名信息， 从而能另外提供服务。

30.6.6.2. 域文件

下面的例子展示了用于 example.org 的主域文件 (存放于 /etc/namedb/master/example.org) :

```
$TTL 3600      ; 1 hour default TTL
example.org.   IN      SOA      ns1.example.org. admin.example.org. (
                                2006051501      ; Serial
                                10800            ; Refresh
                                3600             ; Retry
                                604800          ; Expire
                                300             ; Negative Reponse TTL
                                )

; DNS Servers
                                IN      NS      ns1.example.org.
                                IN      NS      ns2.example.org.

; MX Records
                                IN      MX 10    mx.example.org.
                                IN      MX 20    mail.example.org.

                                IN      A      192.168.1.1

; Machine Names
localhost     IN      A      127.0.0.1
ns1           IN      A      192.168.1.2
ns2           IN      A      192.168.1.3
mx            IN      A      192.168.1.4
mail          IN      A      192.168.1.5

; Aliases
www           IN      CNAME   example.org.
```

注意以 "." 结尾的主机名是全称主机名， 而结尾没有 "." 的是相对于原点的主机名。 例如， ns1 将被解析为 ns1.example.org.

域信息文件的格式如下：

| 域名 | IN 类型 |  |
|----|-------|--|
|----|-------|--|

最常用的 DNS 类型：

## SOA

域权威起始

## NS

域域名服务器

## A

主机地址

## CNAME

名称的正名称

## MX

邮件服务器

## PTR

域名指针 (用于反向 DNS)

```
example.org. IN SOA ns1.example.org. admin.example.org. (  
                2006051501      ; Serial  
                10800            ; Refresh after 3 hours  
                3600             ; Retry after 1 hour  
                604800           ; Expire after 1 week  
                300 )            ; Negative Reponse TTL
```

### example.org.

域名，同时也是该域信息文件的原点。

### ns1.example.org.

域的主/权威域名服务器。

### admin.example.org.

此域的负责人的电子邮件地址，其中 "@" 需要去掉 ([admin@example.org](mailto:admin@example.org) 即 [admin.example.org](mailto:admin@example.org))

### 2006051501

文件的序号。每次修改域文件都必须加一个数字。如今，许多管理员会使用 `yyyymmddrr` 的格式来表示序号。`2006051501` 通常表示上次修改于 05/15/2006，而后面的 `01` 表示在那天的第一次修改。序号非常重要，它用于通知从域服务器更新数据。

```
IN NS      ns1.example.org.
```

是一个 NS 类型。每个提供权威应答的服务器都必须有一个 NS 记录。

|           |    |   |             |
|-----------|----|---|-------------|
| localhost | IN | A | 127.0.0.1   |
| ns1       | IN | A | 192.168.1.2 |
| ns2       | IN | A | 192.168.1.3 |
| mx        | IN | A | 192.168.1.4 |
| mail      | IN | A | 192.168.1.5 |

A 指明了机器名。正如在前面所看到的, `ns1.example.org` 将解析 `192.168.1.2`。

|  |    |   |             |
|--|----|---|-------------|
|  | IN | A | 192.168.1.1 |
|--|----|---|-------------|

一行把当前原点 `example.org` 指定使用 IP 地址 `192.168.1.1`。

|     |          |   |
|-----|----------|---|
| www | IN CNAME | @ |
|-----|----------|---|

正名 (CNAME) 通常用于某台机器指定名。在这个例子中, 将 `www` 指定成了 "主" 机器的一个名, 后者的名字与域名 `example.org` (`192.168.1.1`) 相同。CNAME 不能同与之有相同名字的任何其它并存。

|  |       |    |                   |
|--|-------|----|-------------------|
|  | IN MX | 10 | mail.example.org. |
|--|-------|----|-------------------|

MX 表示一个邮件服务器接收到的域的邮件。 `mail.example.org` 是邮件服务器的主机名, 而 10 是它的先。

可以有多台邮件服务器, 其先分是 10、20 等等。向 `example.org` 投邮件的服务器, 会首先先最高的 MX (先数最小的)、接着次高的, 并重一程直到邮件止。

in-addr.arpa 域名信息文件 (反向 DNS), 采用的格式是同的, 只是 PTR 代替了 A 或 CNAME 的位置。

```
$TTL 3600
1.168.192.in-addr.arpa. IN SOA ns1.example.org. admin.example.org. (
                                2006051501      ; Serial
                                10800            ; Refresh
                                3600             ; Retry
                                604800           ; Expire
                                300 )            ; Negative Reponse TTL

                                IN      NS      ns1.example.org.
                                IN      NS      ns2.example.org.

1      IN      PTR      example.org.
2      IN      PTR      ns1.example.org.
3      IN      PTR      ns2.example.org.
4      IN      PTR      mx.example.org.
5      IN      PTR      mail.example.org.
```

一个文件出了上述假想域中 IP 地址到域名的映射关系。

需要明确的是，在 PTR 记录右面的名字必须是全称域名（也就是必须以 "." 结束）。

### 30.6.7. 缓存域名服务器

缓存域名服务器是一主要承担解析角色的域名服务器。它独立地自行运行，并将结果记住以后使用。

### 30.6.8. 安全

尽管 BIND 是最常用的 DNS 软件，但它还是有一些安全问题。经常会有人发现一些可能的甚至可以利用的安全漏洞。

尽管 FreeBSD 会自行将 named 放到 [chroot\(8\)](#) 环境中运行，但仍有一些其它可用的安全机制来帮助避免潜在的 DNS 服务器的攻击。

请阅读 [CERT](#) 的安全公告，并参考 [the FreeBSD 安全公告通知邮件列表](#) 是一个有助于帮助了解最新 Internet 及 FreeBSD 安全公告的好资源。



如果更新了，确保源代码是最新的，并重新编译 named 有可能会有所帮助。

### 30.6.9. 进一步阅读

BIND/named 手册：[rndc\(8\)](#) [named\(8\)](#) [named.conf\(8\)](#)

- [官方的 ISC BIND 页面](#)
- [Official ISC BIND Forum](#)
- [O'Reilly DNS 和 BIND 第 5 版](#)
- [RFC1034 - 域名 - 概念和工具](#)
- [RFC1035 - 域名 - 协议及其标准](#)

## 30.7. Apache HTTP 服务器

### 30.7.1. 简介

FreeBSD 被用于运行多全球最繁忙的 web 站点。大多数 Internet 上的 web 服务器，都使用 Apache HTTP 服务器。Apache 软件包可以在 FreeBSD 安装上得到。如果没有在首次安装时安装 Apache，可以通过 [www/apache13](#) 或 [www/apache22](#) port 来安装。

一旦成功地安装了 Apache，就必须进行配置。



本节介绍了 1.3.X 版本的 Apache HTTP 服务器的配置，因为它随 FreeBSD 一同使用的最多的版本。Apache 2.X 引入了很多新技术，但在此并不讨论。要了解关于 Apache 2.X 的更多资料，请参考 <http://httpd.apache.org/>。

### 30.7.2. 配置

主要的 Apache HTTP Server 配置文件，在 FreeBSD 上会安装 `/usr/local/etc/apache/httpd.conf`。它是一个典型的 UNIX® 文本配置文件，它使用 `#` 作注释符。由于全部配置文件的简介超出了本书的范围，这里将只介绍最常被修改的那些。

#### **ServerRoot "/usr/local"**

指定了 Apache 安装的目标。运行文件被放到服务器根目标 (server root) 的 `bin` 和 `sbin` 子目标中，而配置文件位于 `etc/apache`。

#### **ServerAdmin you@your.address**

这个地址是在服务器生成邮件时发送邮件的地址，它会出现在服务器生成的页面上，例如页面。

#### **ServerName www.example.com**

**ServerName** 允许配置返回给客户端的主机名，如果的服务器被用以的名字 (例如，使用 `www` 而不是主机本身的真名字)。

#### **DocumentRoot "/usr/local/www/data"**

**DocumentRoot**：这个目标是文档所在的目标。默认情况下，所有的请求都会从这个位置去取，但也可以通过符号链接和别名指定其它的位置。

在修改配置之前 Apache 的配置文件永远是一个好。一旦初始配置好了，就可以开始运行 Apache 了。

### 30.7.3. 运行 Apache

与许多其它网络服务不同，Apache 并不依赖于 `inetd` 超服务来运行。一般情况下会把它配置成一个独立的服务，以期在客户的 web 服务器输入 HTTP 请求，能获得更好的性能。它提供了一个 shell 脚本来使、停止和重新服务得尽可能地。首次运行 Apache，只需运行：

```
# /usr/local/sbin/apachectl start
```

可以在任何时候使用下面的命令来停止服务：

```
# /usr/local/sbin/apachectl stop
```

当由于某些原因修改了配置文件之后，需要重启服务：

```
# /usr/local/sbin/apachectl restart
```

要在重启 Apache 服务时不中断当前的连接，运行：

```
# /usr/local/sbin/apachectl graceful
```

更多的信息，可以在 [apachectl\(8\)](#) 手册中找到。



要在系统中启用 Apache，需要在 `/etc/rc.conf` 中加入：

```
apache_enable="YES"
```

或者对于 Apache 2.2：

```
apache22_enable="YES"
```

如果你希望在系统中引用 Apache `httpd` 程序并指定其它一些选项，你可以把下面的行加到 `rc.conf`：

```
apache_flags=""
```

在 web 服务器就启动行了，你可以使用 web 浏览器打 `http://localhost/`。默认显示的 web 页面是 `/usr/local/www/data/index.html`。

### 30.7.4. 虚拟主机

Apache 支持两种不同类型的虚拟主机。第一种方法是基于名字的虚拟主机。基于名字的虚拟主机使用客户端来的 HTTP/1.1 来辨别主机名。这使得不同的域得以共享同一个 IP 地址。

要配置 Apache 来使用基于名字的虚拟主机，需要把以下面的添加到 `httpd.conf` 中：

```
NameVirtualHost *
```

如果你的 web 服务器的名字是 `www.domain.tld`，而你希望建立一个 `www.someotherdomain.tld` 的虚拟域，需要在 `httpd.conf` 中加入：

```
<VirtualHost *>
ServerName www.domain.tld
DocumentRoot /www/domain.tld
</VirtualHost>

<VirtualHost *>
ServerName www.someotherdomain.tld
DocumentRoot /www/someotherdomain.tld
</VirtualHost>
```

需要把上面的地址和文档路径改为你使用的那些。

要了解关于虚拟主机的更多信息，请参考官方的 Apache 文档，有些文档可以在 <http://httpd.apache.org/docs/vhosts/> 找到。

### 30.7.5. Apache 模

有多种不同的 Apache 模，它可以在基本的服务器基上提供多附加的功能。FreeBSD 的 Ports Collection 安装 Apache 和常用的附加模提供了非常方便的方法。

#### 30.7.5.1. mod\_ssl

mod\_ssl 个模使用 OpenSSL，来提供通安全套接字 (SSL v2/v3) 和 安全 (TLS v1) 的加密能力。个模提供了从某一受信的署机申名所需的所有工具，可以藉此在 FreeBSD 上行安全的 web 服务器。

如果未曾安装 Apache，也可以直接安装一个包含了 mod\_ssl 的版本的 Apache 1.3.X，其方法是通 [www/apache13-modssl](#) port 来行。SSL 支持已作 Apache 2.X 的一部分提供，可以通 [www/apache22](#) port 来安装后者。

#### 30.7.5.2. 言定

Apache 于一些主要的脚本言都有相的模。些模使得完全使用某脚本言来写 Apache 模成可能。他通常也被嵌入到服务器作一个常内存的解器，以避免一个外部解器于下一个将描述的网网站所需和源上的。

### 30.7.6. 网网站

在过去的十年里，越来越多的企加了收益和曝光率而向了互网。也同了于互网内容的需求。有些公司，比如 Microsoft® 推出了基于他所有品的解决方案，源社区也做出了的回。比尚的包括 Django, Ruby on Rails, mod\_perl, and mod\_php.

#### 30.7.6.1. Django

Django 是一个以 BSD 可布的 framework，能者快速写出高性能高品的 web 用程序。它提供一个象系映射件，数据类型可以被当 Python 中的象，和一个富的数据 API，使者避免了写 SQL 句。它同提供了可扩展的模板系，用程序的与 HTML 的表分。

Django 依与 mod\_python, Apache, 和一个可的 SQL 数据引。在置了一些恰当的志后，FreeBSD 的 Port 系将会助安装些必需的依。

例 38. 安装 Django, Apache2, mod\_python3, 和 PostgreSQL

```
# cd /usr/ports/www/py-django; make all install clean -DWITH_MOD_PYTHON3
-DWITH_POSTGRESQL
```

在安装了 Django 和那些依的件之后，需要建一个 Django 目的目，然后配置 Apache，当有于网站上用程序的某些指定的 URL 用内嵌的 Python 解器。

需要在 Apache 的配置文件 httpd.conf 加入以下几行，把某些 URL 的请求交给 web 应用程序：

```
<Location "/">
    SetHandler python-program
    PythonPath ["'/dir/to/your/django/packages/'" + sys.path"]
    PythonHandler django.core.handlers.modpython
    SetEnv DJANGO_SETTINGS_MODULE mysite.settings
    PythonAutoReload On
    PythonDebug On
</Location>
```

### 30.7.6.2. Ruby on Rails

Ruby on Rails 是另外一个开源的 web framework，提供了一个全面的框架，能助 web 开发者工作更有效和快速写出大的应用。它能非常容易地从 ports 系安装。

```
# cd /usr/ports/www/rubygem-rails; make all install clean
```

### 30.7.6.3. mod\_perl

Apache/Perl 集成，将 Perl 程序语言的大功能，与 Apache HTTP 服务器紧密地合到了一起。通过 mod\_perl 模块，可以完全使用 Perl 来撰写 Apache 模块。此外，服务器中嵌入的持久性解释器，消除了由于外部的解释器 Perl 脚本的所造成的性能损失。

mod\_perl 通过多种方式提供。要使用 mod\_perl，要注意 mod\_perl 1.0 只能配合 Apache 1.3 而 mod\_perl 2.0 只能配合 Apache 2.X 使用。mod\_perl 1.0 可以通过 [www/mod\\_perl](#) 安装，而以静默方式安装版本，可以通过 [www/apache13-modperl](#) 来安装。mod\_perl 2.0 可以通过 [www/mod\\_perl2](#) 安装。

### 30.7.6.4. mod\_php

PHP，也称 "PHP: Hypertext Preprocessor"，是一种特别合于 Web 应用的通用脚本语言。它能很容易地嵌入到 HTML 之中，其语法接近于 C、Java™，以及 Perl，以期 web 开发人员的一迅速撰写生成的页面。

要得用于 Apache web 服务器的 PHP5 支持，可以从安装 [lang/php5](#) port 开始。

在首次安装 [lang/php5](#) port 的时候，系统会自显示可用的一系列 **OPTIONS** (配置项)。如果没有看到菜单，例如由于去曾安装 [lang/php5](#) port 等等，可以用下面的命令再次显示配置菜单，在 port 的目录中行：

```
# make config
```

在配置对话框中，选中 **APACHE** 一项，就可以显示出用于与 Apache web 服务器配合使用的可添加的 mod\_php5 模块了。



由于各式各样的原因（例如，出于已部署的 web 应用的兼容性考虑），许多网站仍在使用 PHP4。如果需要 `mod_php4` 而不是 `mod_php5`，请使用 `lang/php4` port。 `lang/php4` port 也支持许多 `lang/php5` port 提供的配置和选项。

前面我已成功地安装并配置了用于支持 PHP 应用所需的模块。并且已将下述配置加入到了 `/usr/local/etc/apache/httpd.conf` 中：

```
LoadModule php5_module          libexec/apache/libphp5.so
```

```
AddModule mod_php5.c
<IfModule mod_php5.c>
    DirectoryIndex index.php index.html
</IfModule>
<IfModule mod_php5.c>
    AddType application/x-httpd-php .php
    AddType application/x-httpd-php-source .phps
</IfModule>
```

些工作完成之后，需要使用 `apachectl` 命令来完成一次 graceful restart 以便加载 PHP 模块：

```
# apachectl graceful
```

在未来升级 PHP 时，`make config` 操作就不再是必需的了；所有的 `OPTIONS` 会由 FreeBSD 的 Ports 框架自行保存。

在 FreeBSD 中的 PHP 支持是高度模块化的，因此基本安装的功能十分有限。添加其他功能的支持非常简单，只需通过 `lang/php5-extensions` port 即可完成。该 port 提供了一个菜单式的界面来协助完成 PHP 扩展的安装。此外，也可以通过该 port 来独立安装扩展。

例如，要将用于 MySQL 数据库服务器的支持加入 PHP5，只需安装 `databases/php5-mysql`。

安装完扩展之后，必须重新加载 Apache 服务器，来令其加载新的配置项：

```
# apachectl graceful
```

## 30.8. 文件传输协议 (FTP)

### 30.8.1. 简介

文件传输协议 (FTP) 应用提供了一个简单的，与 FTP 服务器交互文件的方法。FreeBSD 系统中包含了 FTP 服务器件，`ftpd`。这使得在 FreeBSD 上建立和管理 FTP 服务器变得非常简单。

### 30.8.2. 配置

最重要的配置是决定允许哪些用户访问 FTP 服务器。一般的 FreeBSD 系统包含了一系列系统用户用于运行不同的服务程序，但未知的用户不被允许登录并使用某些用户。/etc/ftpusers 文件中，列出了不允许访问 FTP 的用户。默认情况下，它包含了前述的系统用户，但也可以在它里面加入其它不允许访问 FTP 的用户。

你可能会希望限制访问 FTP 登录的某些用户，而不是完全阻止他使用 FTP。可以通过 /etc/ftpchroot 文件来完成。它列出了希望 FTP 用户行限制的用户的表。而在 [ftpchroot\(5\)](#) 手册中，已对此行了详尽的介绍，故不再赘述。

如果你想要在服务器上启用匿名的 FTP 用户，你必须建立一个名为 **ftp** 的 FreeBSD 用户。然后，用户就可以使用 **ftp** 或 **anonymous** 和任意的口令（通常，它是那个用户的邮件地址作口令）来登录和访问的 FTP 服务器。FTP 服务器将在匿名用户登录时用 **chroot(2)**，以便将其限制在 **ftp** 用户的主目录中。

有一个文本文件可以用来指定显示在 FTP 客户端程序中的欢迎文字。/etc/ftpwelcome 文件中的内容将在用户接上之后，在登录提示之前显示。在成功的登录之后，将显示 /etc/ftpmotd 文件中的内容。注意后者是相对于登录环境的，因此对于匿名用户而言，将显示 ~ftp/etc/ftpmotd。

一旦正确地配置了 FTP 服务器，就必须在 /etc/inetd.conf 中启用它。里面需要做的全部工作就是将注释符 "#" 从已有的 ftpd 行之前去掉：

```
ftp stream tcp nowait root /usr/libexec/ftpd ftpd -l
```

如 [重新添加 inetd 配置文件](#) 所介绍的那，修改这个文件之后，必须让 inetd 重新加载它，才能使新的配置生效。参看 [配置](#) 以获取更多有关如何在系统上启用 inetd 的信息。

ftpd 也可以作为一个独立的服务。它的配置就需要在 /etc/rc.conf 中设置如下的变量：

```
ftpd_enable="YES"
```

在设置了上述变量之后，独立的服务将在下次系统重启的时候，或者通常以 **root** 身份手动运行如下的命令：

```
# /etc/rc.d/ftpd start
```

现在可以通入下面的命令来登录的 FTP 服务器了：

```
% ftp localhost
```

### 30.8.3. 日志

ftpd 服务程序使用 [syslog\(3\)](#) 来记录消息。默认情况下，系统日志将把和 FTP 相关的消息记录到 /var/log/xferlog 文件中。FTP 日志的位置，可以通过修改 /etc/syslog.conf 中如下所示的行来修改：

```
ftp.info /var/log/xferlog
```

一定要小心待在匿名 FTP 服务器中可能遇到的潜在问题。一般而言，允许匿名用户上传文件三思。可能自己的 FTP 站点成了交易未授权的商件的，或生更糟的情况。如果不需要匿名的 FTP 上，可以在文件上配置权限，使得不能在其它匿名用户下载某些文件之前它。

## 30.9. Microsoft® Windows® 客户机提供文件和打印服务 (Samba)

### 30.9.1. 简介

Samba 是一个流行的开源软件包，它提供了 Microsoft® Windows® 客户机的文件和打印服务。客户机可以连接并使用 FreeBSD 系统上的文件空间，就如同使用本地的磁盘一样，或者像使用本地打印机一样使用 FreeBSD 上的打印机。

Samba 软件包可以在 FreeBSD 安装上得到。如果没有在初次安装 FreeBSD 时安装 Samba，可以通过 [net/samba34](#) port 或 package 来安装。

### 30.9.2. 配置

默认的 Samba 配置文件会以 `/usr/local/shared/examples/samba34/smb.conf.default` 的名字安装。每个文件必须复制 `/usr/local/etc/smb.conf` 并进行定制，才能开始使用 Samba。

`smb.conf` 文件中包含了 Samba 的运行配置信息，例如关于打印机的定义，以及希望共享 Windows® 客户机的 "共享文件系统"。Samba 软件包包含了一个称为 `swat` 的 web 管理工具，后者提供了配置 `smb.conf` 文件的方法。

#### 30.9.2.1. 使用 Samba Web 管理工具 (SWAT)

Samba Web 管理工具 (SWAT) 是一个通过 `inetd` 运行的服务程序。因此，需要把 `/etc/inetd.conf` 中下面几行的注释去掉，才能使用 `swat` 来配置 Samba：

```
swat    stream  tcp     nowait 400      root    /usr/local/sbin/swat    swat
```

如 [重新添加 inetd 配置文件](#) 中所介绍的那样，在修改了这个配置文件之后，必须 `inetd` 重新添加配置，才能使其生效。

一旦在 `inetd.conf` 中用了 `swat`，就可以用浏览器 `connect to http://localhost:901` 了。将首先使用系统的 `root` 号登录。

只要成功地登录了 Samba 配置界面，就可以对系统的文件，或从 **Globals**(全局) 开始配置了。**Globals** 小模块于 `[global]` 小模块中的变量，前者位于 `/usr/local/etc/smb.conf` 中。

#### 30.9.2.2. 全局配置

无论是使用 `swat`，还是直接编辑 `/usr/local/etc/smb.conf`，通常首先要配置的 Samba 参数都是：

##### **workgroup**

NT 域名或工作名，其他计算机将通过这些名字来连接到服务器。

### netbios name

这个选项用于设置 Samba 服务器的 NetBIOS 名字。默认情况下，它是所在主机的 DNS 名字的第一部分。

### server string

这个选项用于设置通 `net view` 命令，以及某些其他网络工具可以看到的关于服务器的说明性文字。

### 30.9.2.3. 安全配置

在 `/usr/local/etc/smb.conf` 中的这个最重要的配置，是定义的安全模型，以及客户端上使用的口令存放后端。下面的语句控制这些：

### security

最常见的形式是 `security = share` 和 `security = user`。如果你的客户端使用用户名，并且这些用户名与你的 FreeBSD 机器一致，一般都用 `user` 安全。这是默认的安全策略，它要求客户端首先登录，然后才能访问共享的资源。

如果采用共享（share）安全，客户端不需要用有效的用户名和口令登录服务器，就能直接访问共享的资源。这是旧版本的 Samba 中的默认。

### passdb backend

Samba 提供了若干不同的后端模型。可以通过 LDAP、NIS+、SQL 数据库，或修改的口令文件，来完成客户端的身份验证。默认的模式是 `smbpasswd`，这也是本章将介绍的全部内容。

假如使用的是默认的 `smbpasswd` 后端，你必须首先建立一个 `/usr/local/etc/samba/smbpasswd` 文件，来允许 Samba 客户端进行身份验证。如果你打算用 UNIX® 用户名能从 Windows® 客户端上登录，可以使用下面的命令：

```
# smbpasswd -a username
```



目前推荐使用的后端是 `tdbsam`，使用下面的命令来添加用户名：

```
# pdbedit -a -u username
```

参考 [官方的 Samba HOWTO](#) 以了解关于配置的一些信息。按照前面给出的基本描述，你已经可以安装 Samba 了。

### 30.9.3. 安装 Samba

`net/samba34` port 会增加一个新的用于控制 Samba 的脚本。要用这个脚本，以便使用它来完成安装、停止或重启 Samba 的任一操作，需要在 `/etc/rc.conf` 文件中加入：

```
samba_enable="YES"
```

此外，也可以进行更细粒度的控制：

```
nmdb_enable="YES"
```

```
smbd_enable="YES"
```



也同时配置了在系统引导时启动 Samba。

配置好之后，就可以在任何时候通过下面的命令来启动 Samba 了：

```
# /usr/local/etc/rc.d/samba start
Starting SAMBA: removing stale tdb's :
Starting nmdb.
Starting smbd.
```

参看 [在 FreeBSD 中使用 rc](#) 以了解关于使用 rc 脚本的一些信息。

Samba 事实上包含了三个相互独立的服务器程序。你能看到 nmdb 和 smbd 这两个服务器程序都是通过 samba 脚本启动的。如果在 smb.conf 中启用了 winbind 名字解析服务，你可以看到 winbindd 服务被启动起来。

可以在任何时候通过下面的命令来停止运行 Samba：

```
# /usr/local/etc/rc.d/samba stop
```

Samba 是一个软件包，它提供了用于与 Microsoft® Windows® 网络集成集成各式各样的功能。要了解关于这里所介绍的基本安装以外的其它功能，请看 <http://www.samba.org>。

## 30.10. 通过 NTP 进行时钟同步

### 30.10.1. 简介

随着时钟的推移，计算机的时钟会倾向于漂移。网络时间协议 (NTP) 是一种保持时钟保持准确的方法。

许多 Internet 服务依赖、或大大地受益于本地计算机时钟的准确性。例如，web 服务器可能会接收到一个请求，要求如果文件在某一时刻之后修改才发送它。在局域网环境中，共享文件的计算机之间的时钟是否同步至重要，因此才能使它们保持一致。类似 cron(8) 的程序，也依赖于正确的系统时钟，才能准确地执行操作。

FreeBSD 附带了 ntpd(8) NTP 服务器，它可以用于同步其它的 NTP 服务器，并配置本地计算机的时钟，或者向其它机器提供服务。

### 30.10.2. 混合的 NTP 服务器

为了同步系统时钟，需要首先找到至少一个 NTP 服务器以供使用。网络管理员，或 ISP 都可能会提供用于此目的的 NTP 服务器——看他文档以了解是否是。此外，也有一个在 [公共的 NTP 服务器列表](#)，你可以从中选一个最近的 NTP 服务器。网络管理员的服务器策略，如果需要的，申明一下所需的时钟。

多个相互不连接的 NTP 服务器是一个好主意，当在某个服务器不可用，或者它不可用时可以有备无患。



是因，[ntpd\(8\)](#) 会智能地它收到的-它会更向于使用可的服器。

### 30.10.3. 配置的机器

#### 30.10.3.1. 基本配置

如果只想在系同， 可以使用 [ntpdate\(8\)](#)。 于常重新， 并且不需要常同的面系来比合， 但大多数机器都行 [ntpd\(8\)](#)。

在引使用 [ntpdate\(8\)](#) 来配合行 [ntpd\(8\)](#) 也是一个好主意。 [ntpd\(8\)](#) 地修正， 而 [ntpdate\(8\)](#) 直接置， 无机器的当前和正有多大偏差。

要用引的 [ntpdate\(8\)](#)， 需要把 `ntpdate_enable="YES"` 加到 `/etc/rc.conf` 中。 此外， 需要通 `ntpdate_flags` 来置同的服器和， 它将 [ntpdate\(8\)](#)。

#### 30.10.3.2. 一般配置

NTP 是通 `/etc/ntp.conf` 文件来行配置的， 其格式在 [ntp.conf\(5\)](#) 中行了描述。 下面是一个例子：

```
server ntplocal.example.com prefer
server timeserver.example.org
server ntp2a.example.net

driftfile /var/db/ntp.drift
```

里， `server` 指定了使用一个服器， 一个服器都独立一行。 如果某一台服器上指定了 `prefer` (偏好) 参数， 如上面的 `ntplocal.example.com`， 会先个服器。 如果偏好的服器和其他服器的存在著的差， 它的， 否将使用来自它的， 而不会理会其他服器。 一般来， `prefer` 参数注在非常精的 NTP 源， 例如那些包含特殊的控硬件的服器上。

而 `driftfile` ， 指定了用来保存系率偏差的文件。 [ntpd\(8\)](#) 程序使用它来自地自然漂移， 从而使即使在切断了外来源的情况下， 仍能保持相当的准度。

外， `driftfile` 也保存上一次所使用的 NTP 服器的信息。 个文件包含了 NTP 的内部信息， 它不被任何其他程修改。

#### 30.10.3.3. 控制服的服器的

默情况下， NTP 服器可以被整个 Internet 上的主机。 如果在 `/etc/ntp.conf` 中指定 `restrict` 参数， 可以控制允些机器服的服器。

如果希望拒所有的机器服的 NTP 服器， 只需在 `/etc/ntp.conf` 中加入：

```
restrict default ignore
```



做会禁止的服器在本地配置中列出的服器。 如果需要令 NTP 服器与外界的 NTP 服器同， 允指定服器。 参机手册 [ntp.conf\(5\)](#) 以了解一的。

如果只希望子网内的机器通⋯的服⋯器同⋯⋯，而不允⋯它⋯配置⋯服⋯器，或作⋯同⋯⋯的⋯点来⋯用，⋯加入

```
restrict 192.168.1.0 mask 255.255.255.0 nomodify notrap
```

⋯里，需要把 192.168.1.0 改⋯网⋯上的 IP 地址，并把 255.255.255.0 改⋯的子网掩⋯。

/etc/ntp.conf 可能包含多个 **restrict** ⋯⋯。要了解⋯一⋯的⋯⋯，⋯参⋯ [ntp.conf\(5\)](#) 的 **Access Control Support**(⋯控制支持) 小⋯。

#### 30.10.4. ⋯行 NTP 服⋯器

要⋯ NTP 服⋯器在系⋯⋯⋯随之⋯⋯，需要把 **ntpd\_enable="YES"** 加入到 /etc/rc.conf 中。如果希望向 [ntpd\(8\)](#) ⋯⋯更多参数，需要⋯⋯ /etc/rc.conf 中的 **ntpd\_flags**。

要在不重新⋯⋯机器的前提下⋯⋯服⋯器，需要手工⋯行 **ntpd**，并⋯上 /etc/rc.conf 中的 **ntpd\_flags** 所指定的参数。例如：

```
# ntpd -p /var/run/ntpd.pid
```

#### 30.10.5. 在⋯⋯性的 Internet ⋯接上使用 ntpd

[ntpd\(8\)](#) 程序的正常工作并不需要永久性的 Internet ⋯接。然而，如果⋯的⋯⋯性⋯接是配置⋯按需⋯号的，那⋯防止 NTP 通⋯繁触⋯号，或保持⋯接就有必要了。如果⋯使用用⋯ PPP，可以使用 **filter** ⋯句，在 /etc/ppp/ppp.conf 中⋯行必要的⋯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⋯一⋯的信息，⋯参考 [ppp\(8\)](#) 的 **PACKET FILTERING**(包⋯) 小⋯，以及 /usr/shared/examples/ppp/ 中的例子。



某些 Internet ⋯⋯提供商会阻止低⋯号的端口，⋯会⋯致 NTP 无法正常工作，因⋯⋯无法到⋯⋯的机器。

#### 30.10.6. ⋯一⋯的信息

⋯于 NTP 服⋯器的文⋯，可以在 /usr/shared/doc/ntp/ ⋯到 HTML 格式的版本。

## 30.11. 使用 `syslogd` 管理主机的日志

管理日志对于系统安全和管理是一个重要方面。当有多台分布在中型或大型网络的机器，再或者是位于各种不同的网络中，其他上面的日志文件会变得非常难以操作，在这种情况下，配置远程日志功能能使整个管理过程得更加轻松。

集中所有日志到一台指定的机器能减轻一些日志文件管理的负担。日志文件的收集，合并与循环可以在一台配置，使用 FreeBSD 原生的工具，比如 `syslogd(8)` 和 `newsyslog(8)`。在以下的配置示例中，主机 A，命名 `logserv.example.com`，将用来收集本地网络的日志信息。主机 B，命名 `logclient.example.com` 将把日志信息发送给服务器。在其中，每个主机都需要配置正向和反向的 DNS 或者在 `/etc/hosts` 中。否则，数据将被服务器拒收。

### 30.11.1. 日志服务器的配置

日志服务器是配置成用来接收远程主机日志信息的机器。在大多数的情况下是为了方便配置，或者是为了更好的管理。不论是何原因，在深入之前需要提一些必需条件。

一个正确配置的日志服务器必须符合以下几个最基本的条件：

- 服务器和客户端的防火墙允许 514 端口上的 UDP 文通。
- `syslogd` 被配置成接受从远程客户端来的消息。
- `syslogd` 服务器和所有的客户端都必须有配有正向和反向 DNS，或者在 `/etc/hosts` 中有相应配置。

配置日志服务器，客户端必须在 `/etc/syslog.conf` 中列出，并指定日志的 facility：

```
+logclient.example.com
*.* /var/log/logclient.log
```



更多关于各被支持并可用的 facility 能在 [syslog.conf\(5\)](#) 手册中找到。

一旦加入以后，所有此 facility 消息都会被写到先前指定的文件 `/var/log/logclient.log`。

提供服务的机器需要在其 `/etc/rc.conf` 中配置：

```
syslogd_enable="YES"
syslogd_flags="-a logclient.example.com -v -v"
```

第一个表示在系统上使用 `syslogd` 服务，第二个表示允许服务器接收来自指定日志源客户端的数据。第二行配置中最后的部分，使用 `-v -v`，表示增加日志消息的程度。在整个 facility 配置的时候，这个配置非常有用，因为管理能看到些消息将作这个 facility 的内容来。

可以同时指定多个 `-a` 来允许多个客户端。此外，可以指定 IP 地址或网段，参 [syslog\(3\)](#) 手册以了解可用配置的完整列表。

最后，日志文件被创建。不用任何方法创建，比如 `touch(1)` 能很好的完成此任务：

```
# touch /var/log/logclient.log
```

此时，重新并一下 `syslogd` 守护进程：

```
# /etc/rc.d/syslogd restart
# pgrep syslog
```

如果返回了一个 `PIC` 的，服务端被成功重启了，并开始配置客户端。如果服务端没有重启的，在 `/var/log/messages` 日志中相应出。

### 30.11.2. 日志客户端配置

日志客户端是一台发送日志信息到日志服务器的机器，并在本地保存拷。

与日志服务器似，客户端也需要满足一些最基本的条件：

- `syslogd(8)` 必须被配置成发送指定类型的消息到能接收他的日志服务器。
- 防火墙必须允许 514 端口上的 UDP 包通过；
- 必须配置正向与反向 DNS，或者在 `/etc/hosts` 中有正确的。

相比服务器来配置客户端更松一些。客户端的机器在 `/etc/rc.conf` 中做如下的置：

```
syslogd_enable="YES"
syslogd_flags="-s -v -v"
```

和前面似，些会在系统进程中用 `syslogd` 服务，并加日志消息的程度。而 `-s` 表示禁止服务接收来自其他主机的日志。

Facility 是描述某个消息由系统的部分生成的。例如，`ftp` 和 `ipfw` 都是 facility。当服务生成日志消息，它通常在日志消息中包含了工具。Facility 通常有一个先或等，就是用来一个日志消息的重要程度。最普通的 `warning` 和 `info`。参 `syslog(3)` 手册以得一个完整可用的 facility 与先列表。

日志服务器必须在客户端的 `/etc/syslog.conf` 中指明。在此例中，`@` 符号被用来表示发送日志数据到远程的服务，看上去差不多如下：

```
*.* @logserv.example.com
```

添加后，必须重启 `syslogd` 使得上述修改生效：

```
# /etc/rc.d/syslogd restart
```

日志消息是否能通过网络送，在准备出消息的客户端上用 `logger(1)` 来向 `syslogd` 出信息：

```
# logger "Test message from logclient"
```

该段消息将在同一台客户机的 `/var/log/messages` 以及日志服务器的 `/var/log/logclient.log` 中。

### 30.11.3. 日志服务器

在某些情况下，如果日志服务器没有收到消息的就需要一番了。有几个可能的原因，最常的一个是网络连接的和 DNS 的问题。了解了一些问题，所有的机器都能使用 `/etc/rc.conf` 中所定义的主机名到对方。如果它能正常工作的，那就需要 `/etc/rc.conf` 中的 `syslogd_flags` 做些修改了。

在以下的示例中，`/var/log/logclient.log` 是空的，`/var/log/message` 中也没有表明任何失败的原因。为了增加它的输出，修改 `syslogd_flags` 至类似于如下的示例，并重启服务：

```
syslogd_flags="-d -a logclien.example.com -v -v"
```

```
# /etc/rc.d/syslogd restart
```

在重启服务之后，屏幕上将立刻出现类似的数据：

```
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.
```

很明显，消息是由于主机名不匹配而被拒收的。在一点一点的改了配置文件之后，改了 `/etc/rc.conf` 中如下行有输入：

```
syslogd_flags="-d -a logclien.example.com -v -v"
```

该行包含有 `logclient`，而不是 `logclien`。在做了正确的修改并重启之后便能看到预期的效果了：

```
# /etc/rc.d/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
```

此刻，消息能够被正确接收并保存入文件了。

#### 30.11.4. 安全性方面的思考

就像其他的网服务一样，在配置之前需要考虑安全性。有些日志文件也包含了敏感信息，比如本地主机上所用的服务，用户名和密码数据。从客户端出的数据通过网络到服务器，早期既没有加密也没有密码保护。如果有加密需要的，可以使用 [security/stunnel](#)，它将在一个加密的隧道中传输数据。

本地安全也是个问题。日志文件在使用中或循环后都没有被加密。本地用户可能读取些文件以获得系统更深入的了解。出于这种情况，有些文件设置正确的权限是非常有必要的。 [newsyslog\(8\)](#) 工具支持新建和循环的日志设置权限。把日志文件的权限设置 `600` 能阻止本地用户不必要的探测。

# Chapter 31. 防火

## 31.1. 入

防火的存在，使得出入系的数据流成可能。防火可以使用一或多 " (rules)"，来出入的网接的数据包，并决定允或阻止它通。些通常可以数据包的某个或某些特征，些特征包括，但不必限于型、来源或目的主机地址，以及来源或目的端口。

防火可以大幅度地改善主机或网的安全。它可以用来完成下面的任：

- 保护和隔用程序、服程序，以及内部网上的机器，不受那些来自公共的 Internet 网上所不希望的数据流量的干。
- 限制或禁止从内部网公共的 Internet 上的服。
- 支持网地址 (NAT)，它使得的内部网能使用私有的 IP 地址，并分享一条通往公共的 Internet 的接 (使用一个 IP 地址，或者一公网地址)。

完章，将了解：

- 如何正确地定包。
- FreeBSD 中内建的几防火之的差。
- 如何使用和配置 OpenBSD 的 PF 防火。
- 如何使用和配置 IPFILTER。
- 如何使用和配置 IPFW。

章之前，需要：

- 理解基本的 FreeBSD 和 Internet 概念。

## 31.2. 防火的概念

建立防火集的基本方法有："明示允 (inclusive)"型 或 "明示禁止 (exclusive)"型。明示禁止的防火，默允所有数据通防火，而集中定，是不允通防火的流量，言之，与些不匹配的数据，全部是允通防火的。明示允的防火正好相反，它只允符合集中定集的流量通，而其他所有的流量都被阻止。

明示允型防火能提供于出流量更好的控制，使其更合那些直接 Internet 公网提供服的系的需要。它也能控制来自 Internet 公网到的私有网的型。所有和匹配的流量都会被阻止并案。一般来明示允防火要比明示禁止防火更安全，因它著地少了允不希望的流量通可能造成的。



除非特明，一章的配置和示的集都是建明示允防火的。

使用了 "状功能的防火 (stateful firewall)"，可以一地收安全机制。防火能通防火的接，而只允与有接匹配的接，或建新的接。状功能的防火的缺点是，在很短内有大量的接求，它可能会受到拒服 (DoS) 攻。大多数防火都提供了同用防火的能力，以便站点提供更好的保。

## 31.3. 防火⏏件包

FreeBSD 的基本系⏏内建了三⏏不同的防火⏏件包。 它⏏是 *IPFILTER* (也被称作 IPF)、 *IPFIREWALL* (也被称作 IPFW), 以及 *OpenBSD* 的 *PacketFilter* (也被称⏏ PF)。 FreeBSD 也提供了⏏个内建的、用于流量整形 (基本上是控制⏏占用) 的⏏件包 : [altq\(4\)](#) 和 [dummynet\(4\)](#)。 Dummynet 在⏏去一直和 IPFW ⏏密集成, 而 ALTQ ⏏需要配合 PF 使用。 IPFILTER 的流量整形功能可以使用 IPFILTER 的 NAT 和⏏功能以及 IPFW 的 [dummynet\(4\)](#) 配合, 或者 使用 PF 跟 ALTQ 的⏏合。 IPFW, 以及 PF 都是用⏏来控制是否允⏏数据包出入⏏的系⏏, ⏏然它⏏采取了不同的⏏方法和⏏法。

FreeBSD ⏏包含多个内建的防火⏏件包的原因在于, ⏏不同的人会有不同的需求和偏好。 任何一个防火⏏件包都很⏏是最好的。

作者⏏向于使用 IPFILTER, 因⏏它提供的状⏏式⏏, 在 NAT 的⏏境中要⏏多, 而且它内建了 ftp 代理, ⏏化了使用外部 FTP 服⏏所需的配置。

由于所有的防火⏏都基于⏏所⏏定的包控制字段来⏏功能, 撰写防火⏏集⏏, 就必⏏了解 TCP/IP 是如何工作的, 以及包的控制字段在正常会⏏交互中的作用。 ⏏可以在⏏个网站⏏到一⏏很好的解⏏文⏏ : <http://www.ipprimer.com/overview.cfm>.

## 31.4. OpenBSD Packet Filter (PF) 和 ALTQ

2003 年 7 月, OpenBSD 的防火⏏, 也就是常⏏的 PF 被成功地移植到了 FreeBSD 上, 并可以通⏏ FreeBSD Ports Collection 来安装了; 第一个将 PF 集成到基本系⏏中的版本是 2004 年 11 月⏏行的 FreeBSD 5.3。 PF 是一个完整的提供了大量功能的防火⏏件, 并提供了可⏏的 ALTQ (交⏏列, Alternate Queuing) 功能。 ALTQ 提供了服⏏品⏏ (QoS) ⏏整形功能。

OpenBSD ⏏目非常⏏出的⏏着一⏏ [PF FAQ](#)。 就其本身而言, ⏏一⏏注重于 FreeBSD 的 PF 和提供一些⏏于使用方面的一般常⏏。更⏏的使用信息⏏参⏏ [PF FAQ](#)。

更多的⏏信息, 可以在 FreeBSD 版本的 PF 网站上⏏到 : <http://pf4freebsd.love2party.net/>。

### 31.4.1. 使用 PF 可加⏏的内核模⏏

要加⏏ PF 内核模⏏, 可以在 `/etc/rc.conf` 中加入下面的⏏置 :

```
pf_enable="YES"
```

然后使用⏏脚本来加⏏模⏏ :

```
# /etc/rc.d/pf start
```

需要⏏明的是, 如果系⏏中没有⏏集配置文件, ⏏上述操作不会加⏏ PF 模⏏。 配置文件的默⏏位置是 `/etc/pf.conf`。 如果 PF ⏏集在其他位置, 可以用下面的 `/etc/rc.conf` 配置来告⏏ PF :

```
pf_rules="/path/to/pf.conf"
```



pf.conf 的例子可以在 /usr/shared/examples/pf/ 找到。

PF 模式也可以手工从命令行加载：

```
# kldload pf.ko
```

PF 的日志功能是由 `pflog.ko` 提供的，通常在 /etc/rc.conf 中加入下面的配置：

```
pflog_enable="YES"
```

然后使用脚本来加载模式：

```
# /etc/rc.d/pflog start
```

如果需要其他 PF 特性，需要将 PF 支持的内核。

### 31.4.2. PF 内核

虽然不必自己把 PF 的支持集成到 FreeBSD 内核，但是有仍然需要这样做来使用到 PF 的某些没有被接收到的可加载模式的高特性，比如 `pfsync(4)` 用来发送某些更改到 PF 状态表。它能配合 `carp(4)` 使用 PF 建立支持故障转移的防火墙。更多有关 CARP 的信息可以参看本手册的 [Common Address Redundancy Protocol \(CARP, 共用地址冗余\)](#)。

The PF kernel options can be found in /usr/src/sys/conf/NOTES and are reproduced below:

有关 PF 的内核配置可以在 /usr/src/sys/conf/NOTES 中找到，以下也略有描述：

```
device pf
device pflog
device pfsync
```

`device pf` 用于启用 "Packet Filter" 防火墙的支持 (`pf(4)`)。

`device pflog` 启用可的 `pflog(4)` 网络，用以通过 `bpf(4)` 描述符来捕获流量。 `pflogd(8)` 服务可以用来存储信息，并把它以日志形式写到磁盘上。

`device pfsync` 启用可的 `pfsync(4)` 支持，它是用于 "状态更改" 的网络。

### 31.4.3. 可用的 rc.conf

The following `rc.conf(5)` statements configure PF and `pflog(4)` at boot:

以下 `rc.conf(5)` 中的语句用于配置 PF 和 `pflog(4)`

```
pf_enable="YES"           # 启用 PF (如果需要的, 自动加载内核模块)
pf_rules="/etc/pf.conf"   # pf 使用的规则文件
pf_flags=""               # pfctl 的其他选项
pflog_enable="YES"        # 启用 pflogd(8)
pflog_logfile="/var/log/pflog" # pflogd 用于日志的文件名
pflog_flags=""            # pflogd 的其他选项
```

如果防火墙后面有一个 LAN，而且需要通它来 LAN 上的包，或行 NAT，需要同用下述：

```
gateway_enable="YES"      # 启用 LAN 网
```

### 31.4.4. 建立规则

PF 会从 [pf.conf\(5\)](#) (默认 `/etc/pf.conf`) 文件中取配置，并根据那里的修改、或数据包通。默认安装的 FreeBSD 已提供了一些的例子放在 `/usr/shared/examples/pf/` 目录下。参 [PF FAQ](#) 取完整的 PF 信息。



在 [PF FAQ](#) 中，时刻注意不同版本的 FreeBSD 可能会使用不同版本的 PF。目前，FreeBSD 8.X 和之前的系使用的是与 OpenBSD 4.1 相同版本的 PF。FreeBSD 9.X 和之后的系使用的是与 OpenBSD 4.5 相同版本的 PF。

[FreeBSD packet filter 组件列表](#) 是一个提有配置使用 PF 防火墙的好地方。在提之前组件列表的！

### 31.4.5. 使用 PF

使用 [pfctl\(8\)](#) 可以控制 PF。以下是一些用的命令 (参 [pfctl\(8\)](#) 得全部可用的)：

| 命令  | 作用  |
|---|---|
| <code>pfctl -e</code>                         | 启用 PF   |
| <code>pfctl -d</code>                         | 禁用 PF   |
| <code>pfctl -F all -f /etc/pf.conf</code>     | 清除所有 (nat, filter, state, table, 等等。)并取 <code>/etc/pf.conf</code> |
| <code>pfctl -s [ rules   nat   state ]</code> | 列出 filter 规则, nat 规则, 或状态表  |
| <code>pfctl -vnf /etc/pf.conf</code>          | 取 <code>/etc/pf.conf</code> 中的规则, 但不加载相的规则                        |

### 31.4.6. 启用 ALTQ

ALTQ 只有在作内核加入到 FreeBSD 内核才能使用。ALTQ 目前不是所有的可用网都能支持的。参 [altq\(4\)](#) 手册了解正使用的 FreeBSD 版本中的支持情况。

下面些将用 ALTQ 以及一些附加的功能：

|         |            |                   |
|---------|------------|-------------------|
| options | ALTQ       |                   |
| options | ALTQ_CBQ   | # 基于分流的排列 (CBQ)   |
| options | ALTQ_RED   | # 随机先期 (RED)      |
| options | ALTQ_RIO   | # 流入和出的包行 RED     |
| options | ALTQ_HFSC  | # 等包的包度器 (HFSC)   |
| options | ALTQ_PRIQ  | # 按先的排列 (PRIQ)    |
| options | ALTQ_NOPCC | # 在 SMP 内核必使用, 禁止 |

options ALTQ 将用 ALTQ 框架的支持。

options ALTQ\_CBQ 用于用 基于分流的排列 (CBQ) 支持。CBQ 允许将接口分成不同的, 或者, 列, 以便在中它指定不同的先。

options ALTQ\_RED 将用 随机 (RED)。RED 是一用于防止网塞的技。RED 度量列的度, 并将其与列的最大和最小度行比。如果列, 新的包将被。如名所示, RED 从不同的接口中随机地数据包。

options ALTQ\_RIO 将用 出入的随机。

options ALTQ\_HFSC 用 次式公平平滑包度器。要了解于 HFSC 一的信息, 参 <http://www-2.cs.cmu.edu/~hzhang/HFSC/main.html>。

options ALTQ\_PRIQ 用 先列 (PRIQ)。PRIQ 首先允高先列中的包通。

options ALTQ\_NOPCC 用 ALTQ 的 SMP 支持。如果是 SMP 系, 必使用它。

## 31.5. IPFILTER (IPF) 防火

IPFILTER 的作者是 Darren Reed。IPFILTER 是独立于操作系的: 它是一个放源代码的用, 并且已被移植到了 FreeBSD、NetBSD、OpenBSD、SunOS、HP/UX, 以及 Solaris 操作系上。IPFILTER 的支持和都相当活, 并且有律地布更新版本。

IPFILTER 提供了内核模式的防火和 NAT 机制, 些机制可以通过用模式行的接口程序行和控制。防火可以使用 [ipf\(8\)](#) 工具来地置和除。NAT 可以通过 [ipnat\(1\)](#) 工具来。 [ipfstat\(8\)](#) 工具可以用来示 IPFILTER 内核部分的数。最后, 使用 [ipmon\(8\)](#) 程序可以把 IPFILTER 的作到系日志文件中。

IPF 最初是使用一 "以最后匹配的准" 的策略来的, 方式只能支持无状的。随着代的, IPF 被逐, 并加入了 "quick" , 以及支持状的 "keep state" , 使得理得更富有代气息。IPF 的官方文只介了写的文件和文件理。新的功能只是作一些附加的出, 如果能完全理解些功能, 于建立更安全的防火就很有好。

一中主要是 "quick" , 以及支持状的 "keep state" 的介。是明示允防火集最基本的写要素。

要得于理方式的信, 参考: [http://www.obfuscation.org/ipf/ipf-howto.html#TOC\\_1](http://www.obfuscation.org/ipf/ipf-howto.html#TOC_1) 以及 <http://coombs.anu.edu.au/~avalon/ip-filter.html>。

IPF FAQ 可以在 <http://www.phildev.net/ipf/index.html> 到。

除此之外，你可以在 <http://marc.theaimsgroup.com/?l=ipfilter> 找到存放源代码的 IPFilter 的邮件列表存档，并进行搜索。

### 31.5.1. 启用 IPF

IPF 作为 FreeBSD 基本安装的一部分，以一个独立的内核模块的形式提供。如果在 `rc.conf` 中配置了 `ipfilter_enable="YES"`，系统就会自动地添加 IPF 内核模块。该内核模块在构建时用了日志支持，并加入了 `default pass all` 规则。如果只是需要把默认的规则置为 `block all` 的，就不需要把 IPF 加载到内核中。简单地通过把 `block all` 规则加入自己的规则集来达到同样的目的。

### 31.5.2. 内核选项

下面这些 FreeBSD 内核选项并不是启用 IPF 所必需的。这里只是作为背景知识来加以描述。如果将 IPF 加载入了内核，相应的内核模块将不被使用。

关于 IPF 相关句的内核选项配置的例子，可以在内核源代码中的 `/usr/src/sys/conf/NOTES` 找到。此列如下：

```
options IPFILTER
options IPFILTER_LOG
options IPFILTER_DEFAULT_BLOCK
```

`options IPFILTER` 用于启用 "IPFILTER" 防火墙的支持。

`options IPFILTER_LOG` 用于启用 IPF 的日志支持，所有匹配了包含 `log` 的规则包，都会被记录到 `iplog` 包缓冲区中。

`options IPFILTER_DEFAULT_BLOCK` 将改变防火墙的默认操作，然而，所有不匹配防火墙规则的包都会被阻止。

这些选项只有在重新编译并安装了上述配置的内核之后才会生效。

### 31.5.3. 可用的 rc.conf 选项

要在系统激活 IPF，需要在 `/etc/rc.conf` 中添加下面的选项：

```
ipfilter_enable="YES"      # 启用 ipf 防火墙
ipfilter_rules="/etc/ipf.rules" # 将被加载的规则定义，它是一个文本文件
ipmon_enable="YES"         # 启用 IP 流量日志
ipmon_flags="-Ds"          # D = 作为守护进程运行
                             # s = 使用 syslog 记录
                             # v = 显示 tcp 窗口大小、ack 和序号(seq)
                             # n = 将 IP 和端口映射为名字
```

如果在防火墙后面有使用了保留的私有 IP 地址的 LAN，可能需要添加下面的一些选项来启用 NAT 功能：

```
gateway_enable="YES"           # 用作 LAN 网的功能
ipnat_enable="YES"             # ipnat 功能
ipnat_rules="/etc/ipnat.rules"  # 用于 ipnat 的自定义文件
```

### 31.5.4. IPF

`ipf(8)` 命令可以用来加载自己的规则文件。 一般情况下， 可以建立一个包括自定义的规则的文件， 并使用 `ipf` 命令来替换掉正在运行的防火墙中的内部规则：

```
# ipf -Fa -f /etc/ipf.rules
```

`-Fa` 表示清除所有的内部规则表。

`-f` 用于指定将要被加载的规则文件。

这个功能使得能够修改自定义的规则文件， 通过运行上面的 IPF 命令， 可以将正在运行的防火墙刷新使用全新的规则集， 而不需要重新系统。 对于新的规则来就很方便， 因此可以任意运行上面的命令。

参考 `ipf(8)` 手册以了解这个命令提供的其它规则。

`ipf(8)` 命令假定规则文件是一个标准的文本文件。 它不能处理使用符号代码的脚本。

也有办法利用脚本的非常大的符号替换能力来构建 IPF 规则。 要了解这方面的规则， 参考 [构建采用符号替换的脚本](#)。

### 31.5.5. IPFSTAT

默认情况下， `ipfstat(8)` 会取并显示所有的累加器， 有些是防火墙以来用自定义的规则匹配的出入流量， 可以通过使用 `ipf -Z` 命令来将这些计数器清零。

参看 `ipfstat(8)` 手册以了解这方面的规则。

默认的 `ipfstat(8)` 命令输出类似于下面的样子：

```

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)

```

如果使用了 **-i** (入流量) 或者 **-o** (出流量), 该命令就只取并显示内核中所安装的过滤器器的数据。

**ipfstat -in** 以编号的形式显示入的内部表。

**ipfstat -on** 以编号的形式显示流出的内部表。

输出和下面的类似：

```

@1 pass out on xl0 from any to any
@2 block out on dc0 from any to any
@3 pass out quick on dc0 proto tcp/udp from any to any keep state

```

**ipfstat -ih** 显示内部表中的入流量，一个匹配前面会同显示匹配的次数。

**ipfstat -oh** 显示内部表中的流出流量，一个匹配前面会同显示匹配的次数。

输出和下面的类似：

```

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

```

**ipfstat** 命令的一个重要的功能可以通过指定 **-t** 参数来使用，它会以类似 **top(1)** 的显示 FreeBSD 正在运行的程序表的方式来显示数据。当你的防火墙正在受到攻击的时候，这个功能得以、，并看攻击的数据包。这个提供了希望的目的或源 IP、端口或的能力。参 [ipfstat\(8\)](#) 手册以了解信息。

### 31.5.6. IPMON

为了使 **ipmon** 能正常工作，必须打开 **IPFILTER\_LOG** 个内核。该命令提供了不同的使用模式。

内建模式是默认的模式，如果不指定 `-D` 参数，就会采用该模式。

服务器模式是持续地通过系统日志来记录的工作模式，这样，我们就可以通过查看日志来了解过去曾发生过的事情。该模式是 FreeBSD 和 IPFILTER 配合工作的模式。由于在 FreeBSD 中提供了一个内建的系统日志自举功能，因此，使用 `syslogd(8)` 比默认的将日志信息记录到一个普通文件要好。在默认的 `rc.conf` 文件中，`ipmon_flags` 语句会指定 `-Ds` 标志：

```
ipmon_flags="-Ds"           # D = 作服务器程序
                             # s = 使用 syslog
                             # v = 设置 tcp 窗口大小、ack 和序号(seq)
                             # n = 将 IP 和端口映射为名字
```

系统日志的好处是很明显的。它提供了在事后重新调查相关信息，例如某些包被拒绝，以及某些包的来源地址等等。它将攻击者提供非常有用的第一手资料。

即使用了日志机制，IPF 仍然不会对其执行任何日志记录工作。防火墙管理员可以决定收集哪些日志，并在某些包上加入 `log` 关键字。一般来，只拒绝性的包。

作为例子，通常会有一条默认的、拒绝所有网络流量的规则，并指定 `log` 关键字，作为该规则集的最后一条。这样就能看到所有没有匹配任何规则的数据包了。

### 31.5.7. IPMON 的日志

Syslogd 使用特殊的方法对日志数据进行分类。它使用称为 "facility" 和 "level" 的术语。以 `-Ds` 模式运行的 IPMON 采用 `local0` 作为默认的 "facility" 名。如果需要，可以用下列 levels 来进一步区分数据：

```
LOG_INFO - 使用 "log" 关键字指定的通过或阻止操作
LOG_NOTICE - 同通过的那些数据包
LOG_WARNING - 同阻止的数据包
LOG_ERR - 任何包含不完整的包的数据包
```

要设置 IPFILTER 来将所有数据记录到 `/var/log/ipfilter.log`，需要首先建立该文件。下面的命令可以完成该工作：

```
# touch /var/log/ipfilter.log
```

`syslogd(8)` 功能可以通过在 `/etc/syslog.conf` 文件中的语句来定义。`syslog.conf` 提供了相当多的用以控制 syslog 如何处理类似 IPF 的实用程序所产生的系统消息的方法。

需要将下列语句添加到 `/etc/syslog.conf`：

```
local0.* /var/log/ipfilter.log
```

这里的 `local0.*` 表示把所有的相关日志信息写到指定的文件中。

要使 `/etc/syslog.conf` 中的修改立即生效，可以重新引导计算机，或者通过运行 `/etc/rc.d/syslogd reload` 来

它重新取 `/etc/syslog.conf`。

不要忘了修改 `/etc/newsyslog.conf` 来新建的日志行。

### 31.5.8. 消息的格式

由 `ipmon` 生成的消息由空格分隔的数据字段成。所有的消息都包含的字段是：

1. 接到数据包的日期。
2. 接到数据包的。其格式 `HH:MM:SS.F`，分是小、分、秒，以及分秒（个数字可能有多位）。
3. 理数据包的网接口名字，例如 `dc0`。
4. 和的号，例如 `@0:17`。

可以通过 `ipfstat -in` 来看些信息。

1. 作：p 表示通，b 表示阻止，S 表示包不全，n 表示没有匹配任何，L 表示 log。示些志的序是：S, p, b, n, L。大写的 P 或 B 表示包的原因是某个全局的日志配置，而不是某个特定的。
2. 地址。上包括三部分：源地址和端口（以逗号分），一个  $\rightarrow$  符号，以及目的地址和端口，例如：  
`209.53.17.22,80  $\rightarrow$  198.73.220.17,1722`。
3. PR，后跟名称或号，例如：`PR tcp`。
4. len，后跟包的度，以及包的度，例如：`len 20 40`。

于 TCP 包，会包括一个附加的字段，由一个号始，之后是表示所置的志的一个字母。参 [ipf\(5\)](#) 机手册，以了解些字母所的志。

于 ICMP 包，在最后会有个字段。前一个是 "ICMP"，而后一个是 ICMP 消息和子消息的型，中以斜分，例如 ICMP 3/3 表示端口不可消息。

### 31.5.9. 建采用符号替的脚本

一些有的 IPF 会建包含的文件，并把它写成能与符号替脚本兼容的方式。做最大的好是能修改只修改符号名字所代表的，而在脚本行直接替掉所有的名符。作脚本，可以使用符号替来把那些常使用的直接用于多个。下面将出一个例子。

个脚本所使用的法与 [sh\(1\)](#)、[csh\(1\)](#)，以及 [tcsh\(1\)](#) 脚本。

符号替的前字段是美元符号：`$`。

符号字段不使用 `$` 前。

希望替符号字段的，必使用双引号 (") 括起来。

的文件的似：



```
##### IPF 脚本的 #####
oif="dc0"          # 外网接口的名字
odns="192.0.2.11"   # ISP 的 DNS 服务器 IP 地址
myip="192.0.2.7"    # 来自 ISP 的静 IP 地址
ks="keep state"
fks="flags S keep state"

# 可以使用个脚本来建立 /etc/ipf.rules 文件,
# 也可以 "直接地" 行它。
#
# 除个注号之一。
#
# 1) 保留下面一行, 建 /etc/ipf.rules :
#cat > /etc/ipf.rules << EOF
#
# 2) 保留下面一行,  "直接地" 行脚本 :
/sbin/ipf -Fa -f - << EOF

# 允出到我的 ISP 的域名服务器的
pass out quick on $oif proto tcp from any to $odns port = 53 $fks
pass out quick on $oif proto udp from any to $odns port = 53 $ks

# 允出未加密的 www 求
pass out quick on $oif proto tcp from $myip to any port = 80 $fks

# 允出使用 TLS SSL 加密的 https www 求
pass out quick on $oif proto tcp from $myip to any port = 443 $fks
EOF
##### IPF 脚本的束 #####
```

就是所需的全部内容。 个本身并不重要, 它主要是用于体现如何使用符号代字段, 以及如何完成的替。如果上面的例子的名字是 /etc/ipf.rules.script, 就可以通过入下面的命令来重新加:

```
# sh /etc/ipf.rules.script
```

在文件中嵌入符号有一个: IPF 无法符号替, 因此它不能直接地取的脚本。

个脚本可以使用下面方法之一来使用:

- 去掉 `cat` 之前的注, 并注掉 `/sbin/ipf` 的那一行。像其他配置一, 将 `ipfilter_enable="YES"` 放到 `/etc/rc.conf` 文件中, 并在此后立刻行脚本, 以建或更新 `/etc/ipf.rules`。
- 通把 `ipfilter_enable="NO"` (是默) 加到 `/etc/rc.conf` 中, 来禁止系脚本 IPFILTER。

在 `/usr/local/etc/rc.d/` 目中加一个似下面的脚本。 它起一个而易的名字, 例如 `ipf.loadrules.sh`。注意, `.sh` 展名是必需的。

```
#!/bin/sh
sh /etc/ipf.rules.script
```

脚本文件必须设置属于 **root**，并且属主可、可写、可执行。

```
# chmod 700 /usr/local/etc/rc.d/ipf.loadrules.sh
```

，在系，就会自动加的 IPF 了。

### 31.5.10. IPF 集

集是指一写好的依据包的决策允许通或阻止 IPF 包。包的双向交互成了一个会话交互。防火集会作用于来自于 Internet 公网的包以及由系出来回一些包的数据包。一个 TCP/IP 服（例如 telnet, www, 件等等）都由先定了其特（听）端口。到特定服的包会从源地址使用非特（高号）端口出，并到特定服在目的地址的端口。所有些参数（例如：端口和地址）都是可以防火所利用的，判断是否允许服通的准。

IPF 最初被写成使用一称作“以最后匹配的准”的理，且只能理无状的。随着代的发展，IPF 行了改，并提供了“quick”集，以及一个有状的“keep state”集。后者使理迅速地跟上了代的伐。

一中提供的一些指，是基于使用包含“quick”集和有状的“keep state”集来行述的。些是写明示允许防火集的基本要素。



当防火行操作，要谨慎行事。某些配置可能会将反在服器外面。保起，可以考在第一次行防火配置在本地控制台上，而不是程，如通 ssh 来行。

### 31.5.11. 法

里出的法已化到只理那些新式的状，并且都是“第一个匹配的”的。要了解完整的法描述，参 ipf(8) 机手册。

以 # 字符的内容会被是注。些注可以出在一行的末尾，或者独占一行。空行会被忽略。

由字成。些字必以一定的序，从左到右出在一行上。接下来的文字中字将使用粗体表示。某些字可能提供了子，些子本身可能也是字，而且可能会提供更多的子。下面的文字中，法都使用粗体的小呈，并介了其上下文。

**ACTION IN-OUT OPTIONS SELECTION STATEFUL PROTO SRC\_ADDR,DST\_ADDR OBJECT PORT\_NUM TCP\_FLAG STATEFUL**

**ACTION** = block | pass

**IN-OUT** = in | out

**OPTIONS** = log | quick | on 网接口的名字

*SELECTION* = proto 名称 | 源/目的 IP | port = 端口号 | flags 标志

*PROTO* = tcp/udp | udp | tcp | icmp

*SRC\_ADD,DST\_ADDR* = all | from 对象 to 对象

*OBJECT* = IP地址 | any

*PORT\_NUM* = port 端口号

*TCP\_FLAG* = S

*STATEFUL* = keep state

### 31.5.11.1. ACTION (动作)

动作表示匹配规则的包采取什么动作。一个规则必须包含一个动作。可以使用下面动作之一：

**block** 表示如果规则与包匹配，阻止包。

**pass** 表示如果规则与包匹配，允许包通过防火墙。

### 31.5.11.2. IN-OUT

每个过滤器都必须明确地指定是流入还是流出的。下一个关键字必须是 **in**，要是 **out**，否则将无法通过。

**in** 表示规则被用于从 Internet 公网上收到的数据包。

**out** 表示规则被用于即将发出到 Internet 的数据包。

### 31.5.11.3. OPTIONS



有些规则必须按下面指定的顺序出。

**log** 表示包被写入到 ipt 日志 (如前面 LOGGING 小节所介绍的那)，如果它与规则匹配的。

**quick** 表示如果规则出的参数与包匹配，可以提前结束，使得能“短路”掉后面的规则。提前结束于使用新式的规则是必需的。

**on** 表示将网卡的名称作规则参数的一部分。接口的名字会在 [ifconfig\(8\)](#) 的输出中显示。使用这个选项，规则只会用到某一个网卡接口上的出入数据包上。要配置新式的规则，必须使用这个选项。

当规则包，包的规则会被写入到 IPT 包日志规则中。跟 **log** 关键字，可以使用下面几个修饰符 (按照下列顺序)：

**body** 表示规则包的前 128 字节的内容。

**first** 如果 **log** 关键字和 **keep state** 同时使用，这个规则只在第一个包上触发，规则就不用再一个“keep state”包信息了。

#### 31.5.11.4. SELECTION

一个所介绍的选项字可以用于所观察的包的属性。有一个选项字主，以及一个子选项字，必须从他中选择一个。以下是一些通用的属性，它必须按下面的顺序使用：

#### 31.5.11.5. PROTO

`proto` 是一个主选项字，它必须与某个相关的子选项字配合使用。这个的作用是匹配某个特定的。要使用新式的选项理，就必须使用这个。

`tcp/udp` | `udp` | `tcp` | `icmp` 或其他在 `/etc/protocols` 中定义的。特殊的选项字 `tcp/udp` 可以用于匹配 TCP 或 UDP 包，引入这个字的作用是避免大量的重连的麻烦。

#### 31.5.11.6. SRC\_ADDR/DST\_ADDR

使用 `all` 选项，基本上相当于 "from any to any" 在没有配合其他选项字的情形。

`from src to dst`： `from` 和 `to` 选项字主要是用来匹配 IP 地址。所有的选项都必须同时给出源和目的两个参数。`any` 是一个可以用于匹配任意 IP 地址的特殊选项字。例如，可以使用 `from any to any` 或 `from 0.0.0.0/0 to any` 或 `from any to 0.0.0.0/0` 或 `from 0.0.0.0 to any` 以及 `from any to 0.0.0.0`。

如果无法使用子网掩来表示 IP 的，表示地址就会很麻烦。使用 `net-mgmt/ipcalc` port 可以帮助进行计算。参考下面的网页了解如何撰写度掩：<http://jodies.de/ipcalc>。

#### 31.5.11.7. PORT

如果源或目的指定了匹配端口，就只能用于 TCP 和 UDP 包了。当写端口比，可以指定 `/etc/services` 中所定义的名字，也可以直接用端口号来指定。如果端口号出现在源象一，被是源端口号；反之，被是目的端口号。要使用新式的选项理，就必须与 `to` 象配合使用这个。使用的例子：`from any to any port = 80`

这个端口的比可以多方式行，并可使用不同的比算符。此外，可以指定端口的。

`port "=" | "!=" | "<" | ">" | "<=" | ">=" | "eq" | "ne" | "lt" | "gt" | "le" | "ge".`

要指定端口，可以使用 `"<>"` | `"><"`。



在源和目的匹配参数之后，需要使用下面这个参数，才能使用新式的选项理。

#### 31.5.11.8. TCP\_FLAG

标志只 TCP 有用。些字母用来表示 TCP 包的标志。

新式的选项理使用 `flags S` 参数来 tcp 会初始的请求。

#### 31.5.11.9. STATEFUL

`keep state` 表示如果有一个包与匹配，其参数激活有状态的机制。



如果使用新式的理，这个参数是必需的。

### 31.5.12. 有状态

有状态将网络流量当作一对双向的包交换来处理。如果激活它，`keep-state` 会跟踪一个相关的包在双向会话交互过程中产生的内部。它能跟踪发起者和包的目的地之会话是有效的双向包交换过程的一部分。如果包与某些不符，将自地拒绝。

状态保持也使得 ICMP 包能与 TCP 或 UDP 会话相关。因此，如果在网站收到允许的状态保持匹配的 ICMP 类型 3 代码 4，某些包会被自地允许入。所有 IP 能处理的包，都可以作某会话的一部分，即使它是唯一的，也会被允许入。

所发生的事情是：

将要通入 Internet 公网的网络接口出的包，首先会跟踪状态表的。如果包与会话中期的下一个包匹配，防火墙就会允许包通过，并更新状态表中的会话的交互流信息。不属于会话的包，跟踪地交出集合。

到入 Internet 公网接口的包，也会先跟踪状态表的。如果包与会话中期的下一个包匹配，防火墙就会允许包通过，并更新状态表中的会话的交互流信息。不属于会话的包，跟踪地交入集合。

当会话结束，会话的会在状态表中删除。

有状态使得能集中于阻止/允许新的会话。一旦新会话被允许通过，所有后续的包就都被自地允许通过，而构造的包被自地拒绝。如果新的会话被阻止，后续的包也都不会被允许通过。有状态从技术角度而言，在阻止目前攻击者常用的洪水式攻击来，具有更好的抗御能力。

### 31.5.13. 明示允许集合的例子

下面的集合是如何写非常安全的明示允许防火墙集合的一个例子。明示允许防火墙只允许的服 `pass` (通)，而所有其他的都会被默地拒绝。期望用来保护其他机器的防火墙，通常也叫做 "网防火墙"，使用至少一个网络接口，并且通常只有一个接入到受信的一端 (LAN)，而一个接入不受信的一端 (Internet 公网)。另外，防火墙也可以配置只保护它所行的那个系统 - 模型称作 "主机防火墙"，通常在接入不受信网络的服务器上使用。

包括 FreeBSD 在内的所有 UNIX® 系统通常都会使用 `lo0` 和 IP 地址 `127.0.0.1` 用于操作系统中内部的通。防火墙必须允许这些包无阻碍地通过。

接入 Internet 公网的网络接口，是放置并允许将请求到 Internet 以及接收的地方。有可能是用模式的 PPP `tun0` 接口，如果的网络同 DSL 或调制解调器相关的。

如果有网络是直接接入私有网段的，某些网络接口就可能需要配置允许来自某些 LAN 的包在彼此之间，以及到外界 (Internet) 上的通的。

一般来，被三个主要的小：所有允许自由通过的接口，到公网接口的，以及入公网接口的。

一个公网接口中，常会匹配到的跟踪放置在尽可能前的位置。而最后一个跟踪是阻止包通过，并它。

下面防火墙集中，Outbound 部分是一些使用 `pass` 的，某些指定了允许的公网 Internet 服，并且指定了 `quick`、`on`、`proto`、`port`，以及 `keep state` 某些。`proto tcp` 指定了 `flag` 一个，会话的第一个包将出状态机制。

接收部分首先阻止所有不希望的包，做有四个不同的原因。其一是恶意的包可能和某些允许的流量存在部分匹配，而我希望阻止，而不是那些包与 `allow` 部分匹配就允许它进入。其二是，已确信要阻止的包被拒事件，往往并不是我需要关注的，因此只要果断地予以阻止即可。防火集中的一组部分的最后一条都是阻止并包，有助于逮捕攻击者留下法律所要求的证据。

另外一个需要注意的事情是保持不希望的数据包不做回应。无效的包被丢弃和消失。因此，攻击者便无法知道包是否到了你的系统。攻击者系统了解的越少，攻陷系统所需的包也就越多。包含 `log first` 的包只会让它第一次被触发的包，在例子中一个包被用于 `nmap OS 探测` 包。 `security/nmap` 是攻击者常用的一用于探测目标系统所用操作系统的工具。

如果你看到了 `log first` 包的日志，就使用 `ipfstat -hio` 命令来看看那个包被匹配的次数。如果数目大，则表示系统正在受到洪水式攻击。

如果包的端口号并不是你所知道的，可以在 `/etc/services` 或 [http://en.wikipedia.org/wiki/List\\_of\\_TCP\\_and\\_UDP\\_port\\_numbers](http://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers) 了解端口号通常的用途。

参考下面的网页，了解木桶使用的端口：<http://www.sans.org/security-resources/idfaq/oddports.php>。

下面是我在自己的系统中使用的完整的，非常安全的 `明示允许` 防火集中。直接使用这个集中不会造成问题，所要做的只是注掉那些不需要 `pass`(允许通过) 的服务。

如果在日志中有了希望 `阻止` 的包，只需在 `inbound` 小组中加一条阻止集中可。

必须将一个集中的 `dc0` 替系统上接入 Internet 的网络接口名称，例如，用环境下的 PPP 包是 `tun0`。

在 `/etc/ipf.rules` 中加入下面的内容：

```
#####
# No restrictions on Inside LAN Interface for private network
# Not needed unless you have LAN
#####

#pass out quick on xl0 all
#pass in quick on xl0 all

#####
# No restrictions on Loopback Interface
#####
pass in quick on lo0 all
pass out quick on lo0 all

#####
# Interface facing Public Internet (Outbound Section)
# Match session start requests originating from behind the
# firewall on the private network
# or from this gateway server destined for the public Internet.
#####

# Allow out access to my ISP's Domain name server.
# xxx must be the IP address of your ISP's DNS.
# Dup these lines if your ISP has more than one DNS server
```

```

# Get the IP addresses from /etc/resolv.conf file
pass out quick on dc0 proto tcp from any to xxx port = 53 flags S keep state
pass out quick on dc0 proto udp from any to xxx port = 53 keep state

# Allow out access to my ISP's DHCP server for cable or DSL networks.
# This rule is not needed for 'user ppp' type connection to the
# public Internet, so you can delete this whole group.
# Use the following rule and check log for IP address.
# Then put IP address in commented out rule & delete first rule
pass out log quick on dc0 proto udp from any to any port = 67 keep state
#pass out quick on dc0 proto udp from any to z.z.z.z port = 67 keep state

# Allow out non-secure standard www function
pass out quick on dc0 proto tcp from any to any port = 80 flags S keep state

# Allow out secure www function https over TLS SSL
pass out quick on dc0 proto tcp from any to any port = 443 flags S keep state

# Allow out send & get email function
pass out quick on dc0 proto tcp from any to any port = 110 flags S keep state
pass out quick on dc0 proto tcp from any to any port = 25 flags S keep state

# Allow out Time
pass out quick on dc0 proto tcp from any to any port = 37 flags S keep state

# Allow out nntp news
pass out quick on dc0 proto tcp from any to any port = 119 flags S keep state

# Allow out gateway & LAN users' non-secure FTP ( both passive & active modes)
# This function uses the IPNAT built in FTP proxy function coded in
# the nat rules file to make this single rule function correctly.
# If you want to use the pkg_add command to install application packages
# on your gateway system you need this rule.
pass out quick on dc0 proto tcp from any to any port = 21 flags S keep state

# Allow out ssh/sftp/scp (telnet/rlogin/FTP replacements)
# This function is using SSH (secure shell)
pass out quick on dc0 proto tcp from any to any port = 22 flags S keep state

# Allow out insecure Telnet
pass out quick on dc0 proto tcp from any to any port = 23 flags S keep state

# Allow out FreeBSD CVSup
pass out quick on dc0 proto tcp from any to any port = 5999 flags S keep state

# Allow out ping to public Internet
pass out quick on dc0 proto icmp from any to any icmp-type 8 keep state

# Allow out whois from LAN to public Internet
pass out quick on dc0 proto tcp from any to any port = 43 flags S keep state

```

```

# Block and log only the first occurrence of everything
# else that's trying to get out.
# This rule implements the default block
block out log first quick on dc0 all

#####
# Interface facing Public Internet (Inbound Section)
# Match packets originating from the public Internet
# destined for this gateway server or the private network.
#####

# Block all inbound traffic from non-routable or reserved address spaces
block in quick on dc0 from 192.168.0.0/16 to any      #RFC 1918 private IP
block in quick on dc0 from 172.16.0.0/12 to any      #RFC 1918 private IP
block in quick on dc0 from 10.0.0.0/8 to any         #RFC 1918 private IP
block in quick on dc0 from 127.0.0.0/8 to any        #loopback
block in quick on dc0 from 0.0.0.0/8 to any          #loopback
block in quick on dc0 from 169.254.0.0/16 to any     #DHCP auto-config
block in quick on dc0 from 192.0.2.0/24 to any       #reserved for docs
block in quick on dc0 from 204.152.64.0/23 to any    #Sun cluster interconnect
block in quick on dc0 from 224.0.0.0/3 to any       #Class D & E multicast

##### Block a bunch of different nasty things. #####
# That I do not want to see in the log

# Block frags
block in quick on dc0 all with frags

# Block short tcp packets
block in quick on dc0 proto tcp all with short

# block source routed packets
block in quick on dc0 all with opt lsrr
block in quick on dc0 all with opt ssrr

# Block nmap OS fingerprint attempts
# Log first occurrence of these so I can get their IP address
block in log first quick on dc0 proto tcp from any to any flags FUP

# Block anything with special options
block in quick on dc0 all with ipopts

# Block public pings
block in quick on dc0 proto icmp all icmp-type 8

# Block ident
block in quick on dc0 proto tcp from any to any port = 113

# Block all Netbios service. 137=name, 138=datagram, 139=session
# Netbios is MS/Windows sharing services.
# Block MS/Windows hosts2 name server requests 81

```



```

block in log first quick on dc0 proto tcp/udp from any to any port = 137
block in log first quick on dc0 proto tcp/udp from any to any port = 138
block in log first quick on dc0 proto tcp/udp from any to any port = 139
block in log first quick on dc0 proto tcp/udp from any to any port = 81

# Allow traffic in from ISP's DHCP server. This rule must contain
# the IP address of your ISP's DHCP server as it's the only
# authorized source to send this packet type. Only necessary for
# cable or DSL configurations. This rule is not needed for
# 'user ppp' type connection to the public Internet.
# This is the same IP address you captured and
# used in the outbound section.
pass in quick on dc0 proto udp from z.z.z.z to any port = 68 keep state

# Allow in standard www function because I have apache server
pass in quick on dc0 proto tcp from any to any port = 80 flags S keep state

# Allow in non-secure Telnet session from public Internet
# labeled non-secure because ID/PW passed over public Internet as clear text.
# Delete this sample group if you do not have telnet server enabled.
#pass in quick on dc0 proto tcp from any to any port = 23 flags S keep state

# Allow in secure FTP, Telnet, and SCP from public Internet
# This function is using SSH (secure shell)
pass in quick on dc0 proto tcp from any to any port = 22 flags S keep state

# Block and log only first occurrence of all remaining traffic
# coming into the firewall. The logging of only the first
# occurrence avoids filling up disk with Denial of Service logs.
# This rule implements the default block.
block in log first quick on dc0 all
##### End of rules file #####

```

### 31.5.14. NAT

NAT 是 网 址 转 换 (Network Address Translation) 的 简 写。 对 于 那 些 熟 悉 Linux® 的 人 来 讲， 这 个 概 念 叫 做 IP 伪 装 (Masquerading)； NAT 和 IP 伪 装 是 完 全 一 个 概 念。 由 IPF 的 NAT 提 供 的 一 项 功 能 是， 将 防火 墙 后 的 本 地 局 域 网 (LAN) 共 享 一 个 ISP 提 供 的 IP 地 址 来 接 入 Internet 公 网。

有 些 人 可 能 会 问， 为 什 么 需 要 这 样 做。 一 般 而 言， ISP 会 为 非 商 用 提 供 独 立 的 IP 地 址。 这 些 地 址 意 味 着 每 次 登 录 到 ISP 都 有 可 能 得 到 不 同 的 IP 地 址， 无 论 是 采 用 拨 号 登 录， 或 使 用 cable 以 及 DSL 解 调 器 的 方 式。 这 个 IP 是 与 Internet 公 网 交 互 使 用 的 身 份。

假 如 你 家 中 有 五 台 PC 需 要 接 入 Internet 的 情 形。 你 可 能 需 要 向 ISP 租 一 台 PC 所 使 用 的 独 立 的 Internet 身 份 付 费， 并 且 有 五 根 网 线。

有 了 NAT， 你 就 只 需 要 一 个 ISP 身 份， 然 后 将 其 他 四 台 PC 的 网 线 通 过 交 换 机 接 起 来， 并 通 过 这 台 FreeBSD 系 统 的 那 台 机 器 作 为 网 网 接 出 去。 NAT 会 自 动 地 将 这 台 PC 在 内 网 的 LAN IP 地 址， 在 防火 墙 后 接 入 公 网 的 IP 地 址。 此 外， 当 数 据 包 返 回 时， 也 将 进 行 逆 向 的 转 换。

在 IP 地址空间中，有一些特殊的地址是保留供 NAT 的内网 LAN IP 地址使用的。根据 RFC 1918，可以使用下面这些 IP 地址用于内网，它们不会在 Internet 公网上路由：

|                   |   |                       |
|-------------------|---|-----------------------|
| 起始 IP 10.0.0.0    | - | 结束 IP 10.255.255.255  |
| 起始 IP 172.16.0.0  | - | 结束 IP 172.31.255.255  |
| 起始 IP 192.168.0.0 | - | 结束 IP 192.168.255.255 |

### 31.5.15. IPNAT

NAT 规则是通过 `ipnat` 命令添加的。默认情况下，NAT 规则会保存在 `/etc/ipnat.rules` 文件中。参考 [ipnat\(1\)](#) 了解更多的情况。

如果在 NAT 已配置之后想要修改 NAT 规则，可以修改保存 NAT 规则的那个文件，然后在命令行 `ipnat` 命令加上 `-CF` 参数，以删除在用的 NAT 内部规则表，以及所有地址翻译表中已有的规则。

要重新添加 NAT 规则，可以使用类似下面的命令：

```
# ipnat -CF -f /etc/ipnat.rules
```

如果想要看看系统上 NAT 的规则信息，可以用下面的命令：

```
# ipnat -s
```

要列出当前的 NAT 表的映射规则，使用下面的命令：

```
# ipnat -l
```

要显示规则的信息并显示与规则相关的当前规则表：

```
# ipnat -v
```

### 31.5.16. IPNAT 规则

NAT 规则非常的灵活，能满足商业应用和家庭应用的各种不同的需求。

这里所介绍的配置方法已被简化，以适用于非商用环境中的一般情况。完整的配置方法描述，参考 [ipnat\(5\)](#) 手册中的介绍。

NAT 规则的写法与下面的例子类似：

```
map IF LAN_IP_RANGE -> PUBLIC_ADDRESS
```

规则 `map` 出口在规则的最前面。

将 *IF* 替换为网卡的接口名。

*LAN\_IP\_RANGE* 是内网中的客户机使用的地址。通常情况下，它是类似 **192.168.1.0/24** 的地址。

*PUBLIC\_ADDRESS* 既可以是外网的 IP 地址，也可以是 **0/32** 个特殊的数字，它表示分配到 *IF* 上的所有地址。

### 31.5.17. NAT 的工作原理

当包从 LAN 到防火墙，而目的地址是公网地址，它首先会通过 outbound 规则。接下来，NAT 会得到包，并按自下而上的顺序处理，而第一个匹配的规则将生效。NAT 接下来会根据包的接口名字和源 IP 地址匹配所有的规则。如果包和某个 NAT 规则匹配，它会修改包的（源 IP 地址，例如，内网的 IP 地址）是否在 NAT 规则中箭头左方指定的 IP 地址匹配。如果匹配，包的原地址将被根据用 **0/32** 数字指定的 IP 地址重写。NAT 将向它的内部 NAT 表发送此地址，然后，当包从 Internet 公网中返回，就能把地址映射回原先的内网 IP 地址，并在随后使用规则来处理。

### 31.5.18. 启用 IPNAT

要启用 IPNAT，只需在 */etc/rc.conf* 中加入下面一些语句。

使机器能在不同的网卡接口之间行包的转发，需要：

```
gateway_enable="YES"
```

再次启用自启 IPNAT：

```
ipnat_enable="YES"
```

指定 IPNAT 规则集文件：

```
ipnat_rules="/etc/ipnat.rules"
```

### 31.5.19. 大型 LAN 中的 NAT

由于在一个 LAN 中有大量 PC，以及包含多个 LAN 的情形，把所有的内网 IP 地址都映射到同一个公网 IP 上会导致源冲突的问题，因为同一个端口可能在多做了 NAT 的 LAN PC 上被多次使用，并导致冲突。有解决方法来解决这个问题。

#### 31.5.19.1. 指定使用哪些端口

普通的 NAT 规则类似于：

```
map dc0 192.168.1.0/24 -> 0/32
```

上面的规则中，包的源端口在包通过 IPNAT 规则不会个性化的。通常使用 **portmap** 数字，可以要求 IPNAT 只使用指定范围内的端口地址。比如，下面的规则将 IPNAT 把源端口改指定范围内的端口：

```
map dc0 192.168.1.0/24 -> 0/32 portmap tcp/udp 20000:60000
```

使用 **auto** 关键字可以配置得更简单一些，它会要求 IPNAT 自动地找出可用的端口并使用：

```
map dc0 192.168.1.0/24 -> 0/32 portmap tcp/udp auto
```

### 31.5.19.2. 使用公网地址池

对于很大的 LAN 而言，总有一天会遇到一个界限，此 LAN 的地址已多到了无法只用一个公网地址表的程度。如果有可用的一批公网 IP 地址，可以将这些地址作为一个“地址池”来使用，让 IPNAT 来从这些公网 IP 地址中挑选用于打包的地址，并将其与这些包建立映射关系。

例如，如果将下面一个把所有包都映射到同一公网 IP 地址的：

```
map dc0 192.168.1.0/24 -> 204.134.75.1
```

作修改，就可以用子网掩码来表 IP 地址：

```
map dc0 192.168.1.0/24 -> 204.134.75.0/255.255.255.0
```

或者用 CIDR 方法来指定的一批地址了：

```
map dc0 192.168.1.0/24 -> 204.134.75.0/24
```

### 31.5.20. 端口重定向

非常流行的一种做法是，将 web 服务器、邮件服务器、数据库服务器以及 DNS 分放到 LAN 上的不同的 PC 上。这种情况下，来自这些服务器的网络流量仍然被 NAT，但必须有办法把输入的流量到局域网的 PC 上。IPNAT 提供了 NAT 重定向机制来解决这个问题。考虑下面的情况，它的 web 服务器的 LAN 地址是 **10.0.10.25**，而它的唯一的公网 IP 地址是 **20.20.20.5**，可以写作的：

```
rdr dc0 20.20.20.5/32 port 80 -> 10.0.10.25 port 80
```

或者：

```
rdr dc0 0.0.0.0/0 port 80 -> 10.0.10.25 port 80
```

另外，也可以用 LAN 地址 **10.0.10.33** 上运行的 LAN DNS 服务器来处理公网上的 DNS 请求：

```
rdr dc0 20.20.20.5/32 port 53 -> 10.0.10.33 port 53 udp
```

### 31.5.21. FTP 和 NAT

FTP 是一个在 Internet 如今天人人所熟知之前就已出现的恐惧，那时，研究机构和大学是租用线路到一起的，而 FTP 则被用于在科研人之间共享大文件。那时，数据的安全性并不是需要考量的事情。若干年之后，FTP 则被埋在了正在形成中的 Internet 骨干，而它使用明文来交换用户名和口令的缺点，并没有随着新出的一些安全需求而得到改善。FTP 提供了两种不同的风格，即主动模式和被动模式。两者的区别在于数据通道的建立方式。被动模式相对而言要更加安全，因为数据通道是由发起 ftp 会话的一方建立的。关于 FTP 以及它所提供的不同模式，在 <http://www.slacksite.com/other/ftp.html> 进行了很好的描述。

#### 31.5.21.1. IPNAT 规则

IPNAT 提供了一个内建的 FTP 代理规则，它可以在 NAT map 规则中指定。它能处理所有外网的 FTP 主动或被动模式的会话请求，并正确地建立连接性的设备规则，只打用于数据通道的端口号。这样，就消除了 FTP 一般会防火墙带来的，需要大量地打开高端口所可能带来的安全隐患。

下面的规则可以处理来自内网的 FTP 规则：

```
map dc0 10.0.10.0/29 -> 0/32 proxy port 21 ftp/tcp
```

这个规则能处理来自网段的 FTP 规则：

```
map dc0 0.0.0.0/0 -> 0/32 proxy port 21 ftp/tcp
```

这个规则处理所有来自内网的非 FTP 网络流量：

```
map dc0 10.0.10.0/29 -> 0/32
```

FTP map 规则在普通的 map 规则之前执行。所有的包会从最上面的第一个规则开始执行。匹配的次序是网络名称，内网源 IP 地址，以及它是否是 FTP 包。如果所有规则都匹配成功，则 FTP 代理将建立一个规则的规则，以便 FTP 会话的数据包能正常出入，同时这些包执行 NAT。所有的 LAN 数据包，如果没有匹配第一条规则，则会匹配下面的规则，并最终被 NAT。

#### 31.5.21.2. IPNAT FTP 规则

如果使用了 NAT FTP 代理，则只需要 FTP 建立一个规则。

如果不使用 FTP 代理，就需要下面三个规则：

```
# Allow out LAN PC client FTP to public Internet
# Active and passive modes
pass out quick on rl0 proto tcp from any to any port = 21 flags S keep state

# Allow out passive mode data channel high order port numbers
pass out quick on rl0 proto tcp from any to any port > 1024 flags S keep state

# Active mode let data channel in from FTP server
pass in quick on rl0 proto tcp from any to any port = 20 flags S keep state
```

## 31.6. IPFW

IPFIREWALL (IPFW) 是一个由 FreeBSD 启起的防火用件，它由 FreeBSD 的志者成写和。它使用了无状和写方式，以期到状所期望的目。

准的 FreeBSD 安装中，IPFW 所出的集例（可以在 /etc/rc.firewall 和 /etc/rc.firewall6 中得到）非常，建不要不加修改地直接使用。例中没有使用状，而功能在大部分的配置中都是非常有用的，因此一并不以系自的例作基。

IPFW 的无状法，是由一提供的能力的技支持的，技超出了一般的防火安装人的知水平。IPFW 是足用，以及掌握先技的者于高的包需求而的。要完全放 IPFW 的所有的大能力，需要不同的的有深入的了解，并根据它独特的包信息来写。一的述超出了本手册的。

IPFW 由七个部分组成，其主要件是内核的防火理器，及其集成的数据包工具、日志工具、用以触 NAT 工具的 `divert` 、、高特殊用途工具、`dummynet` 流量整形机制，`fwd rule` 工具，接工具，以及 `ipstealth` 工具。IPFW 支持 IPv4 和 IPv6。

### 31.6.1. 用 IPFW

IPFW 是基本的 FreeBSD 安装的一部分，以独的可加内核模的形式提供。如果在 `rc.conf` 中加入 `firewall_enable="YES"` 句，就会自地加的的内核模。除非打算使用由它提供的 NAT 功能，一般情况下并不需要把 IPFW FreeBSD 的内核。

如果将 `firewall_enable="YES"` 加入到 `rc.conf` 中并重新系，下列信息将在程中，以高亮的白色示出来：

```
ipfw2 initialized, divert disabled, rule-based forwarding disabled, default to deny,
logging disabled
```

可加内核模在加入了日志的能力。要用日志功能，并配置日志的限制，需要在 `/etc/sysctl.conf` 中加入一些配置。些置将在重新之后生效：

```
net.inet.ip.fw.verbose=1
net.inet.ip.fw.verbose_limit=5
```

### 31.6.2. 内核

把下列在 FreeBSD 内核就加入，并不是用 IPFW 所必需的，除非需要使用 NAT 功能。这里只是将些作背景知识来介绍。

```
options    IPFIREWALL
```

这个将 IPFW 作内核的一部分来用。

```
options    IPFIREWALL_VERBOSE
```

这个将用通 IPFW 的匹配了包含 **log** 字的一个包的功能。

```
options    IPFIREWALL_VERBOSE_LIMIT=5
```

以的方式，限制通 [syslogd\(8\)](#) 的包的个数。如果在比较劣的环境下防火的活可能会需要这个。它能避免潜在的 syslog 的洪水式拒绝服务攻击。

```
options    IPFIREWALL_DEFAULT_TO_ACCEPT
```

这个默认地允许所有的包通防火，如果是第一次配置防火，使用这个将是一个不好的主意。

```
options    IPDIVERT
```

——用 NAT 功能。



如果内核中没有加入 **IPFIREWALL\_DEFAULT\_TO\_ACCEPT**，而配置使用的集中也没有明确地指定允许接入的，默认情况下，到本机和从本机出的所有包都会被阻止。

### 31.6.3. /etc/rc.conf Options

用防火：

```
firewall_enable="YES"
```

要由 FreeBSD 提供的几防火型中的一来作默认配置，需要 `/etc/rc.firewall` 文件并出合的型，然后在 `/etc/rc.conf` 中加入似下面的配置：

```
firewall_type="open"
```

可以指定下列配置之一：

- **open** - 允许所有流量通过。
- **client** - 只保护本机。
- **simple** - 保护整个网络。
- **closed** - 完全禁止除回环地址之外的全部 IP 流量。
- **UNKNOWN** - 禁止添加防火墙规则。
- **filename** - 到防火墙规则文件的路径。

有添加自定义 **ipfw** 防火墙的方法。其一是将变量 **firewall\_type** 包含在 **ipfw(8)** 命令行中的防火墙文件的完整路径。下面是一个规则集的例子：

```
add deny in
add deny out
```

除此之外，也可以将 **firewall\_script** 变量包含 **ipfw** 命令的可执行脚本，每个脚本会在规则自运行。与前面规则集文件等价的脚本如下：

**ipfw** 命令是在防火墙运行，用于在其内部表中手工逐条添加或删除防火墙的规则工具。该方法的好处在于，一旦你的计算机或停机，所有添加或删除或修改的规则也就丢失了。把所有的规则都写到一个文件中，并在每次使用这个文件来加载规则，或一次大批量地替换防火墙，那推荐使用这里介绍的方法。

**ipfw** 的一个非常实用的功能是将所有正在运行的防火墙规则显示出来。**IPFW** 的统计机制会建立一个计数器，用以统计与它匹配的包的数量。在系统的过程中，列出规则及其计数器是了解它是否工作正常的重要手段。

按顺序列出所有的规则：

```
# ipfw list
```

列出所有的规则，同时输出最后一次匹配的包：

```
# ipfw -t list
```

列出所有的规则信息、匹配规则的包的数量，以及规则本身。第一列是规则的 ID 号，随后是输出包匹配的数量，输入包的匹配数量，最后是规则本身。

```
# ipfw -a list
```

列出所有的规则和静默规则：

```
# ipfw -d list
```

同时显示已期的规则：



```
# ipfw -d -e list
```

将计数器清零：

```
# ipfw zero
```

只把编号 *NUM* 的计数器清零：

```
# ipfw zero NUM
```

### 31.6.4. IPFW 规则集

规则集是指一组写好的依据包的决策允许通过或阻止 IPFW 规则。包的双向交互成了一个会话交互。防火墙规则集会作用于来自于 Internet 公网的包以及由系统出来回一些包的数据包。每一个 TCP/IP 服务（例如 telnet, www, 邮件等等）都由规则先定义了其特征（监听）端口。到达特定服务的包会从源地址使用非特征（高编号）端口发出，并到达特定服务在目的地址的端口。所有这些参数（例如：端口和地址）都是可以防火墙规则所利用的，判断是否允许服务通过的准则。

当有数据包进入防火墙，会从规则集里的第一个规则开始行比，并自上向下地行匹配。当包与某个规则参数相匹配，将会行规则所定义的操作，并停止规则集搜索。规则策略，通常也被称作“最先匹配者”的搜索方法。如果没有任何与包相匹配的规则，那么它就会根据默认的 IPFW 策略，也就是 65535 号规则截。一般情况下这个规则是阻止包，而且不发出任何回应。



如果定义的操作是 **count**、**skipto** 或 **tee** 规则的，搜索会继续。

这里所介绍规则，都是使用了那些包含状态功能的，也就是 **keep state**、**limit**、**in**、**out** 以及 **via** 规则的规则。是编写明示允许防火墙规则集所需的基本框架。



在操作防火墙规则集要谨慎行事，如果操作不当，很容易将自己反锁在外面。

#### 31.6.4.1. 规则语法

这里所介绍的规则语法已简化了，只包括了建立标准的明示允许防火墙规则集所必需的那些。要了解完整的规则语法，请参考 [ipfw\(8\)](#) 手册。

规则是由规则字组成的：有些规则字必须以特定的顺序从左到右编写。下面的介绍中，规则字使用粗体表示。某些规则字包括了子规则，有些子规则本身可能也是规则字，有些规则字可以包含更多的子规则。

**#** 用于表示开始一段注释。它可以出现在一个规则的后面，也可以独占一行。空行会被忽略。

*CMD RULE\_NUMBER ACTION LOGGING SELECTION STATEFUL*

##### 31.6.4.1.1. CMD

每一个新的规则都可以 **add** 作前缀，它表示将规则加入内部表。

#### 31.6.4.1.2. RULE\_NUMBER

每一条规则都与一个在 1 到 65535 之间的规则号相关。

#### 31.6.4.1.3. ACTION

一个规则可以与下列的动作之一相关，所指定的动作将在输入的数据包与规则所指定的规则号相匹配时执行。

allow | accept | pass | permit

这些关键字都表示允许匹配规则的包通过防火墙，并停止规则搜索。

check-state

根据规则表数据包。如果匹配，则执行规则所指定的动作，亦即生成规则；否则，则移到下一个规则。check-state 规则没有规则号。如果规则集中没有 check-state 规则，则会在第一个 keep-state 或 limit 规则，规则表实施。

deny | drop

这两个关键字都表示匹配规则的包。同时，停止规则搜索。

#### 31.6.4.1.4. LOGGING

log or logamount

当数据包与 log 关键字的规则匹配时，将通过名为 SECURITY 的 facility 来把消息发到 syslogd(8)。只有在规则的次数没有超过 logamount 参数所指定的次数时，才会记录日志。如果没有指定 logamount，则会以 sysctl 变量 net.inet.ip.fw.verbose\_limit 所指定的限制为准。如果将限制之一指定为零，则表示不作限制。如果达到了限制数，可以通过将规则的日志数或包数清零来重新启用日志，参见 ipfw reset log 命令来了解。



日志是在所有其他匹配条件都成功之后，在规则包实施最动作 (accept, deny) 之前执行的。规则可以自行决定某些规则用日志。

#### 31.6.4.1.5. SELECTION

规则所介绍的规则关键字主要用来描述规则的包的属性，用以判断包是否与规则相匹配。

下面是一些通用的用于匹配包特征的属性，它们必须按顺序使用：

udp | tcp | icmp

也可以指定在 /etc/protocols 中所定义的规则。每个定义的是匹配的规则，在规则中必须指定它。

from src to dst

from 和 to 关键字用于匹配 IP 地址。规则中必须同时指定源和目的两个参数。如果需要匹配任意 IP 地址，可以使用特殊关键字 any。还有一个特殊关键字，即 me，用于匹配规则的 FreeBSD 系统上所有网络接口上所配置的 IP 地址，它可以用于表示网络上的其他计算机到防火墙（也就是本机），例如 from me to any 或 from any to me 或 from 0.0.0.0/0 to any 或 from any to 0.0.0.0/0 或 from 0.0.0.0 to any 或 from any to 0.0.0.0 以及 from me to 0.0.0.0。IP 地址可以通过点的 IP 地址/掩码度 (CIDR 方法)，或者一个点的 IP 地址的形式来指定。这是编写规则所必需的。使用 net-mgmt/ipcalc port 可以用来简化计算。关于

个工具的更多信息，也可参考它的主页：<http://jodies.de/ipcalc>。

port number

这个参数主要用于那些支持端口号的协议（例如 TCP 和 UDP）。如果要通过端口号匹配某个包，就必须指定这个参数。此外，也可以通过服务的名字（根据 /etc/services）来指定服务，这会比使用数字指定端口号直接一些。

in | out

相应地，匹配输入和输出的包。这里的 **in** 和 **out** 都是关键字，在编写匹配规则时，必需作为其他条件的一部分来使用。

via IF

根据指定的网络接口的名称精确地匹配输出的包。这里的 **via** 关键字将使得接口名称成为匹配规则的一部分。

setup

要匹配 TCP 会话的发起请求，就必须使用它。

keep-state

这是一个必须使用的关键字。在生成匹配规则时，防火墙将建立一个表，其默认行是，匹配使用同一源的、从源到目的 IP/端口 的双向网络流量。

limit {src-addr | src-port | dst-addr | dst-port}

防火墙只允许匹配规则，与指定的参数相同的  $N$  个连接。可以指定至少一个源或目的地址及端口。**limit** 和 **keep-state** 不能在同一规则中同时使用。**limit** 提供了与 **keep-state** 相同的功能，并增加了一些独有的能力。

#### 31.6.4.2. 状态检测

有状态检测将网络流量当作一个双向的包交换来处理。它提供了一些额外的能力，用以检查包中的包是否来自最初的发送者，并在遵循双向包交换的规则时会。如果包与这些规则不符，将自动地拒绝它。

**check-state** 用来检查在 IPFW 规则集中的包是否符合状态检测机制的规则。如果匹配，允许包通过，此时防火墙将建立一个新的表来匹配双向交换中的下一个包。如果不匹配，将丢弃规则集中的下一个包。

状态检测机制在 SYN-flood 攻击下是脆弱的，因为这种情况会产生大量的连接，从而耗尽资源。为了抵抗攻击，从 FreeBSD 中加入了一个叫做 **limit** 的新规则。这个规则可以用来限制符合规则的会允许的连接数。如果规则表中的连接数超过 **limit** 的限制数量，包将被拒绝。

#### 31.6.4.3. 防火墙消息

日志日志的好处是显而易见的：它提供了在事后检查所发生状况的方法，例如有些包被拒绝了，有些包的来源和目的地，从而提供到攻击者所需的证据。

即使使用了日志机制，IPFW 也不会自行生成任何规则的日志。防火墙管理需要指定规则集中的某些规则日志，并在某些规则上添加 **log** 操作。一般来说，只有 deny 规则日志，例如对于输入的 ICMP ping 的 deny 规则。此外，配置“默认的 ipfw 规则 deny 规则”，并加入 **log** 操作来作为规则集的最后一条规则也是很常用的用法。这样，就能看到没有匹配任何一条规则的那些数据包。

日志是一把双刃剑，如果不谨慎地加以利用，可能会陷入过多的日志数据中，并导致磁盘被日志塞满。将磁盘填满是 DoS 攻击最老套的手法之一。由于 syslogd 除了会将日志写入磁盘之外，还会输出到 root 的控制台屏幕上，因此有太多的日志信息是很让人头痛的事情。

`IPFWALL_VERBOSE_LIMIT=5` 内核将限制同一个进程到系统日志程序 `syslogd(8)` 的消息的数量。当内核用了多少个，某一特定进程所生成的消息的数量将封顶这个数字。一般来，没有办法从 200 条一模一样的日志信息中取更多有用的信息。例如，如果同一个进程生了 5 次消息并被送到 `syslogd`，余下的相同的消息将被数，并像下面那样 `syslogd`：

```
last message repeated 45 times
```

所有的数据包消息，默认情况下会最写到 `/var/log/security` 文件中，后者在 `/etc/syslog.conf` 文件里行了定。

#### 31.6.4.4. 写脚本

大多数有用的 IPFW 用会建立一个包含的文件，并且，按能以脚本形式行的方式来写。做最大的一个好是，可以大批量地刷新防火，而无重新系就能激活它。方法在新会非常方便，因同一程在需要可以多次行。作脚本，可以使用符号替来撰写那些常需要使用的，并用同一个符号在多个中反地表示它。下面将出一个例子。

个脚本使用的法同 `sh(1)`、`csh(1)` 以及 `tcsh(1)` 脚本兼容。符号替字段使用美元符号 `$` 作前。符号字段本身并不使用 `$` 前。符号替字段的必使用“双引号”括起来。

可以使用似下面的文件：

```
##### 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 #####
```

就是所要做的全部事情了。例子中的并不重要，它主要是用来表示如何使用符号替。

如果把上面的例子保存到 `/etc/ipfw.rules` 文件中。下面的命令来会重新加。

```
# sh /etc/ipfw.rules
```

/etc/ipfw.rules 这个文件可以放到任何位置，也可以命名成随便什么的名字。

也可以手工运行下面的命令来达到类似的目的：

```
# ipfw -q -f flush
# ipfw -q add check-state
# ipfw -q add deny all from any to any frag
# ipfw -q add deny tcp from any to any established
# ipfw -q add allow tcp from any to any 80 out via tun0 setup keep-state
# ipfw -q add allow tcp from any to 192.0.2.11 53 out via tun0 setup keep-state
# ipfw -q add 00611 allow udp from any to 192.0.2.11 53 out via tun0 keep-state
```

#### 31.6.4.5. 状态规则集

以下的非-NAT 规则集，是如何编写非常安全的 '明示允许' 防火墙的一个例子。明示允许防火墙只允许匹配了 pass 规则的包通过，而默认阻止所有的其他数据包。用来保护整个网段的防火墙，至少需要有一个网口接口，并且其上必须配置规则，以便防火墙正常工作。

所有 UNIX® 操作系统，也包括 FreeBSD，都允许使用网口接口 lo0 和 IP 地址 127.0.0.1 来完成操作系统内部的通信。防火墙必须包含一些规则，使一些数据包能无障碍地收。

接入 Internet 公网的那个网口接口上，规则配置授权和控制，来限制外部的，以及来自 Internet 公网的。这个接口很可能是用的用 PPP 接口，例如 tun0，或者接在 DSL 或 modem 上的网口。

如果有至少一个网口接入了防火墙后的内网 LAN，必须有一些接口配置规则，以便一些接口之间的包能顺利地通。

所有的规则被分成三个部分，所有无阻碍地通过的，公网的出口，以及公网的接收。

公网接口相关的规则的顺序，规则是最常用到的放在尽可能前的位置，而最后一个规则，规则是阻止那个接口在那一方向上的包。

出口部分的规则只包含一些 allow 规则，允许定义的那些唯一区分规则的端口号所指定的规则通过，以允许 Internet 公网上的某些服务。所有的规则中都指定了 proto、port、in/out、via 以及 keep state 一些规则。proto tcp 规则同时指定 setup 规则，来区分开始会话的包，以触发将包放入 keep state 规则表中的操作。

接收部分首先阻止所有不希望的包，规则做有几个不同的原因。其一是恶意的包可能和某些允许规则的流量规则存在部分匹配，而我希望阻止，而不是一些包规则与 allow 规则部分匹配就允许它进入。其二是，已经信要阻止的包被拒事件，往往并不是我需要关注的，因此只要规则地予以阻止即可。防火墙规则集中的几个部分的最后一条规则都是阻止并包，有助于逮捕攻击者留下法律所要求的证据。

另外一个需要注意的事情是保持系统不希望的数据包不做回应。无效的包被丢弃和消失。规则，攻击者便无法知道包是否到了系统的系统。攻击者系统了解的越少，其攻击的程度也就越大。如果不知道端口号，可以到 /etc/services/ 或到 [http://en.wikipedia.org/wiki/List\\_of\\_TCP\\_and\\_UDP\\_port\\_numbers](http://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers) 并看一下端口号，以了解其用途。另外，也可以在网口上了解常木马所使用的端口：<http://www.sans.org/security-resources/idsfaq/oddports.php>。

#### 31.6.4.6. 明示允许规则集的例子

下面是一个非-NAT 的规则集，它是一个完整的明示允许规则集。使用它作为规则集不会有什。只需把那些不需要的服务规则的 pass 规则注释掉就可以了。如果在日志中看到消息，而且不想再看到它，

只需在接收部分加一个 `deny` 规则。可能需要把 `dc0` 改成接入公网的接口的名字。对于使用 PPP 的接口而言，它是 `tun0`。

这些规则遵循一定的模式。

- 所有请求 Internet 公网上服务的会话初始包，都使用了 `keep-state`。
- 所有来自 Internet 的授权服务请求，都采用了 `limit` 规则来防止洪水式攻击。
- 所有的规则都使用了 `in` 或者 `out` 来标明方向。
- 所有的规则都使用了 `via` 接口名来指定匹配通过一个接口的包。

这些规则都放到 `/etc/ipfw.rules`。

```
##### Start of IPFW rules file #####
# Flush out the list before we begin.
ipfw -q -f flush

# Set rules command prefix
cmd="ipfw -q add"
pif="dc0"      # public interface name of NIC
               # facing the public Internet

#####
# No restrictions on Inside LAN Interface for private network
# Not needed unless you have LAN.
# Change xl0 to your LAN NIC interface name
#####
#$cmd 00005 allow all from any to any via xl0

#####
# No restrictions on Loopback Interface
#####
$cmd 00010 allow all from any to any via lo0

#####
# Allow the packet through if it has previous been added to the
# the "dynamic" rules table by a allow keep-state statement.
#####
$cmd 00015 check-state

#####
# Interface facing Public Internet (Outbound Section)
# Interrogate session start requests originating from behind the
# firewall on the private network or from this gateway server
# destined for the public Internet.
#####

# Allow out access to my ISP's Domain name server.
# x.x.x.x must be the IP address of your ISP's DNS
# Dup these lines if your ISP has more than one DNS server
```

```

# Get the IP addresses from /etc/resolv.conf file
$cmd 00110 allow tcp from any to x.x.x.x 53 out via $pif setup keep-state
$cmd 00111 allow udp from any to x.x.x.x 53 out via $pif keep-state

# Allow out access to my ISP's DHCP server for cable/DSL configurations.
# This rule is not needed for .user ppp. connection to the public Internet.
# so you can delete this whole group.
# Use the following rule and check log for IP address.
# Then put IP address in commented out rule & delete first rule
$cmd 00120 allow log udp from any to any 67 out via $pif keep-state
#$cmd 00120 allow udp from any to x.x.x.x 67 out via $pif keep-state

# Allow out non-secure standard www function
$cmd 00200 allow tcp from any to any 80 out via $pif setup keep-state

# Allow out secure www function https over TLS SSL
$cmd 00220 allow tcp from any to any 443 out via $pif setup keep-state

# Allow out send & get email function
$cmd 00230 allow tcp from any to any 25 out via $pif setup keep-state
$cmd 00231 allow tcp from any to any 110 out via $pif setup keep-state

# Allow out FBSD (make install & CVSUP) functions
# Basically give user root "GOD" privileges.
$cmd 00240 allow tcp from me to any out via $pif setup keep-state uid root

# Allow out ping
$cmd 00250 allow icmp from any to any out via $pif keep-state

# Allow out Time
$cmd 00260 allow tcp from any to any 37 out via $pif setup keep-state

# Allow out nntp news (i.e., news groups)
$cmd 00270 allow tcp from any to any 119 out via $pif setup keep-state

# Allow out secure FTP, Telnet, and SCP
# This function is using SSH (secure shell)
$cmd 00280 allow tcp from any to any 22 out via $pif setup keep-state

# Allow out whois
$cmd 00290 allow tcp from any to any 43 out via $pif setup keep-state

# deny and log everything else that.s trying to get out.
# This rule enforces the block all by default logic.
$cmd 00299 deny log all from any to any out via $pif

#####
# Interface facing Public Internet (Inbound Section)
# Check packets originating from the public Internet
# destined for this gateway server or the private network.
#####

```

```

# Deny all inbound traffic from non-routable reserved address spaces
$cmd 00300 deny all from 192.168.0.0/16 to any in via $pif #RFC 1918 private IP
$cmd 00301 deny all from 172.16.0.0/12 to any in via $pif #RFC 1918 private IP
$cmd 00302 deny all from 10.0.0.0/8 to any in via $pif #RFC 1918 private IP
$cmd 00303 deny all from 127.0.0.0/8 to any in via $pif #loopback
$cmd 00304 deny all from 0.0.0.0/8 to any in via $pif #loopback
$cmd 00305 deny all from 169.254.0.0/16 to any in via $pif #DHCP auto-config
$cmd 00306 deny all from 192.0.2.0/24 to any in via $pif #reserved for docs
$cmd 00307 deny all from 204.152.64.0/23 to any in via $pif #Sun cluster interconnect
$cmd 00308 deny all from 224.0.0.0/3 to any in via $pif #Class D & E multicast

# Deny public pings
$cmd 00310 deny icmp from any to any in via $pif

# Deny ident
$cmd 00315 deny tcp from any to any 113 in via $pif

# Deny all Netbios service. 137=name, 138=datagram, 139=session
# Netbios is MS/Windows sharing services.
# Block MS/Windows hosts2 name server requests 81
$cmd 00320 deny tcp from any to any 137 in via $pif
$cmd 00321 deny tcp from any to any 138 in via $pif
$cmd 00322 deny tcp from any to any 139 in via $pif
$cmd 00323 deny tcp from any to any 81 in via $pif

# Deny any late arriving packets
$cmd 00330 deny all from any to any frag in via $pif

# Deny ACK packets that did not match the dynamic rule table
$cmd 00332 deny tcp from any to any established in via $pif

# Allow traffic in from ISP's DHCP server. This rule must contain
# the IP address of your ISP's DHCP server as it's the only
# authorized source to send this packet type.
# Only necessary for cable or DSL configurations.
# This rule is not needed for .user ppp. type connection to
# the public Internet. This is the same IP address you captured
# and used in the outbound section.
#$cmd 00360 allow udp from any to x.x.x.x 67 in via $pif keep-state

# Allow in standard www function because I have apache server
$cmd 00400 allow tcp from any to me 80 in via $pif setup limit src-addr 2

# Allow in secure FTP, Telnet, and SCP from public Internet
$cmd 00410 allow tcp from any to me 22 in via $pif setup limit src-addr 2

# Allow in non-secure Telnet session from public Internet
# labeled non-secure because ID & PW are passed over public
# Internet as clear text.
# Delete this sample group if you do not have telnet server enabled.

```



```
$cmd 00420 allow tcp from any to me 23 in via $pif setup limit src-addr 2

# Reject & Log all incoming connections from the outside
$cmd 00499 deny log all from any to any in via $pif

# Everything else is denied by default
# deny and log all packets that fell through to see what they are
$cmd 00999 deny log all from any to any
##### End of IPFW rules file #####
```

### 31.6.4.7. 一个 NAT 和状态表的例子

要使用 IPFW 的 NAT 功能，需要行一些外的配置。除了其他 IPFW 句之外，需要在内核配置中加上 `option IPDIVERT` 句。

在 `/etc/rc.conf` 中，除了普通的 IPFW 配置之外，需要加入：

```
natd_enable="YES"           # Enable NATD function
natd_interface="rl0"        # interface name of public Internet NIC
natd_flags="-dynamic -m"    # -m = preserve port numbers if possible
```

将状态与 `divert natd` (网地址) 会使表的写得非常。 `check-state` 的位置，以及 `divert natd` 将得非常。一来，就不再有包的序理流程了。提供了一个新的作型，称 `skipto`。要使用 `skipto` 命令，就必须一个行号，以定 `skipto` 号是希望跳到的位置。

下面出了一些未加注的例子来明如何写的，用以助理解包的理序。

理流程从文件最上的第一个开始理，并自向下地一个，直到到匹配的，且数据包从防火墙中放出止。注意号 100 101, 450, 500, 以及 510 的位置非常重要。些控制出和接收的包的地址程，它在 `keep-state` 表中的中就能与内网的 LAN IP 地址。一个需要注意的是，所有的 `allow` 和 `deny` 都指定了包的方向 (也就是 `outbound` 或 `inbound`) 以及网接口。最后，注意所有出的会求都会求 `skipto rule 500` 以完成网地址。

下面以 LAN 用使用 web 器一个 web 面例。Web 面使用 80 来完成通。当包入防火墙， 100 并不匹配，因它是出而不是收到的包。它能通 101，因是第一个包，因而它没有入状态保持表。包最到 125，并匹配。最，它会通接入 Internet 公网的网出。之前，包的源地址仍然是内网 IP 地址。一旦匹配个，就会触个作。 `keep-state` 会把个到 `keep-state` 表中，并行所指定的作。作是到表中的信息的一部分。在个例子中，个作是 `skipto rule 500`。 500 NAT 包的 IP 地址，并将其出。必牢，一非常重要。接下来，数据包将到目的地，之后返回并从集的第一条开始理。一次，它将与 100 匹配，其目的 IP 地址将被映射回的内网 LAN IP 地址。其后，它会被 `check-state` 理，而在存会表中到，并到 LAN。数据包接下来到了内网 LAN PC 上，而后者会送从程服器求下一段数据的新数据包。个包会再次由 `check-state` ，并到出的表，并行其的作，即 `skipto 500`。包跳到 500 并被 NAT 后出。

在接收一，已存在的会的数据包会被 `check-state` 自地理，并到 `divert nat` 。我需要解决的是，阻止所有的坏数据包，而只允授的服。例如在防火上行了 Apache 服，而我希望人在 Internet 公网的同，也能本地的 web 站点。新的接入始求包将匹配 100，而 IP

地址的防火墙所在的服务器而映射到了 LAN IP。此后，包会匹配所有我希望的那些令人生厌的东西，并最终匹配 425。一旦匹配，会发生一件事。数据包会被放到 keep-state 表，但此时，所有来自那个源 IP 的会话请求的数量会被限制为 2。这一做法能控制指定端口上服务器的 DoS 攻击。它作用指定了 allow 包被放到 LAN 上。包返回时，check-state 会找出包属于某一已存在的会话交互，并直接把它放到 500 做 NAT，并放到出口接口。

示例集 #1:

```
#!/bin/sh
cmd="ipfw -q add"
skip="skipto 500"
pif=rl0
ks="keep-state"
good_tcpo="22,25,37,43,53,80,443,110,119"

ipfw -q -f flush

$cmd 002 allow all from any to any via xl0 # exclude LAN traffic
$cmd 003 allow all from any to any via lo0 # exclude loopback traffic

$cmd 100 divert natd ip from any to any in via $pif
$cmd 101 check-state

# Authorized outbound packets
$cmd 120 $skip udp from any to xx.168.240.2 53 out via $pif $ks
$cmd 121 $skip udp from any to xx.168.240.5 53 out via $pif $ks
$cmd 125 $skip tcp from any to any $good_tcpo out via $pif setup $ks
$cmd 130 $skip icmp from any to any out via $pif $ks
$cmd 135 $skip udp from any to any 123 out via $pif $ks

# Deny all inbound traffic from non-routable reserved address spaces
$cmd 300 deny all from 192.168.0.0/16 to any in via $pif #RFC 1918 private IP
$cmd 301 deny all from 172.16.0.0/12 to any in via $pif #RFC 1918 private IP
$cmd 302 deny all from 10.0.0.0/8 to any in via $pif #RFC 1918 private IP
$cmd 303 deny all from 127.0.0.0/8 to any in via $pif #loopback
$cmd 304 deny all from 0.0.0.0/8 to any in via $pif #loopback
$cmd 305 deny all from 169.254.0.0/16 to any in via $pif #DHCP auto-config
$cmd 306 deny all from 192.0.2.0/24 to any in via $pif #reserved for docs
$cmd 307 deny all from 204.152.64.0/23 to any in via $pif #Sun cluster
$cmd 308 deny all from 224.0.0.0/3 to any in via $pif #Class D & E multicast

# Authorized inbound packets
$cmd 400 allow udp from xx.70.207.54 to any 68 in $ks
$cmd 420 allow tcp from any to me 80 in via $pif setup limit src-addr 1

$cmd 450 deny log ip from any to any

# This is skipto location for outbound stateful rules
$cmd 500 divert natd ip from any to any out via $pif
$cmd 510 allow ip from any to any

##### end of rules #####
```

下面的这个脚本基本上和上面一样，但使用了易于阅读的编写方式，并给出了相当多的注解，以帮助较少的IPFW 脚本写者更好地理解这些脚本到底在做什么。

示例脚本 #2：

```

#!/bin/sh
##### Start of IPFW rules file #####
# Flush out the list before we begin.
ipfw -q -f flush

# Set rules command prefix
cmd="ipfw -q add"
skip="skipto 800"
pif="rl0"      # public interface name of NIC
               # facing the public Internet

#####
# No restrictions on Inside LAN Interface for private network
# Change xl0 to your LAN NIC interface name
#####
$cmd 005 allow all from any to any via xl0

#####
# No restrictions on Loopback Interface
#####
$cmd 010 allow all from any to any via lo0

#####
# check if packet is inbound and nat address if it is
#####
$cmd 014 divert natd ip from any to any in via $pif

#####
# Allow the packet through if it has previous been added to the
# the "dynamic" rules table by a allow keep-state statement.
#####
$cmd 015 check-state

#####
# Interface facing Public Internet (Outbound Section)
# Check session start requests originating from behind the
# firewall on the private network or from this gateway server
# destined for the public Internet.
#####

# Allow out access to my ISP's Domain name server.
# x.x.x.x must be the IP address of your ISP's DNS
# Dup these lines if your ISP has more than one DNS server
# Get the IP addresses from /etc/resolv.conf file
$cmd 020 $skip tcp from any to x.x.x.x 53 out via $pif setup keep-state

# Allow out access to my ISP's DHCP server for cable/DSL configurations.
$cmd 030 $skip udp from any to x.x.x.x 67 out via $pif keep-state

# Allow out non-secure standard www function
$cmd 040 $skip tcp from any to any 80 out via $pif setup keep-state

```

```

# Allow out secure www function https over TLS SSL
$cmd 050 $skip tcp from any to any 443 out via $pif setup keep-state

# Allow out send & get email function
$cmd 060 $skip tcp from any to any 25 out via $pif setup keep-state
$cmd 061 $skip tcp from any to any 110 out via $pif setup keep-state

# Allow out FreeBSD (make install & CVSUP) functions
# Basically give user root "GOD" privileges.
$cmd 070 $skip tcp from me to any out via $pif setup keep-state uid root

# Allow out ping
$cmd 080 $skip icmp from any to any out via $pif keep-state

# Allow out Time
$cmd 090 $skip tcp from any to any 37 out via $pif setup keep-state

# Allow out nntp news (i.e., news groups)
$cmd 100 $skip tcp from any to any 119 out via $pif setup keep-state

# Allow out secure FTP, Telnet, and SCP
# This function is using SSH (secure shell)
$cmd 110 $skip tcp from any to any 22 out via $pif setup keep-state

# Allow out whois
$cmd 120 $skip tcp from any to any 43 out via $pif setup keep-state

# Allow ntp time server
$cmd 130 $skip udp from any to any 123 out via $pif keep-state

#####
# Interface facing Public Internet (Inbound Section)
# Check packets originating from the public Internet
# destined for this gateway server or the private network.
#####

# Deny all inbound traffic from non-routable reserved address spaces
$cmd 300 deny all from 192.168.0.0/16 to any in via $pif #RFC 1918 private IP
$cmd 301 deny all from 172.16.0.0/12 to any in via $pif #RFC 1918 private IP
$cmd 302 deny all from 10.0.0.0/8 to any in via $pif #RFC 1918 private IP
$cmd 303 deny all from 127.0.0.0/8 to any in via $pif #loopback
$cmd 304 deny all from 0.0.0.0/8 to any in via $pif #loopback
$cmd 305 deny all from 169.254.0.0/16 to any in via $pif #DHCP auto-config
$cmd 306 deny all from 192.0.2.0/24 to any in via $pif #reserved for docs
$cmd 307 deny all from 204.152.64.0/23 to any in via $pif #Sun cluster
$cmd 308 deny all from 224.0.0.0/3 to any in via $pif #Class D & E multicast

# Deny ident
$cmd 315 deny tcp from any to any 113 in via $pif

```

```

# Deny all Netbios service. 137=name, 138=datagram, 139=session
# Netbios is MS/Windows sharing services.
# Block MS/Windows hosts2 name server requests 81
$cmd 320 deny tcp from any to any 137 in via $pif
$cmd 321 deny tcp from any to any 138 in via $pif
$cmd 322 deny tcp from any to any 139 in via $pif
$cmd 323 deny tcp from any to any 81 in via $pif

# Deny any late arriving packets
$cmd 330 deny all from any to any frag in via $pif

# Deny ACK packets that did not match the dynamic rule table
$cmd 332 deny tcp from any to any established in via $pif

# Allow traffic in from ISP's DHCP server. This rule must contain
# the IP address of your ISP's DHCP server as it's the only
# authorized source to send this packet type.
# Only necessary for cable or DSL configurations.
# This rule is not needed for 'user ppp' type connection to
# the public Internet. This is the same IP address you captured
# and used in the outbound section.
$cmd 360 allow udp from x.x.x.x to any 68 in via $pif keep-state

# Allow in standard www function because I have Apache server
$cmd 370 allow tcp from any to me 80 in via $pif setup limit src-addr 2

# Allow in secure FTP, Telnet, and SCP from public Internet
$cmd 380 allow tcp from any to me 22 in via $pif setup limit src-addr 2

# Allow in non-secure Telnet session from public Internet
# labeled non-secure because ID & PW are passed over public
# Internet as clear text.
# Delete this sample group if you do not have telnet server enabled.
$cmd 390 allow tcp from any to me 23 in via $pif setup limit src-addr 2

# Reject & Log all unauthorized incoming connections from the public Internet
$cmd 400 deny log all from any to any in via $pif

# Reject & Log all unauthorized out going connections to the public Internet
$cmd 450 deny log all from any to any out via $pif

# This is skipto location for outbound stateful rules
$cmd 800 divert natd ip from any to any out via $pif
$cmd 801 allow ip from any to any

# Everything else is denied by default
# deny and log all packets that fell through to see what they are
$cmd 999 deny log all from any to any
##### End of IPFW rules file #####

```

# Chapter 32. 高可用网络

## 32.1. 概述

本章将就一系列与网络有关的高可用性行。

读完本章，你将了解：

- 关于网络和路由的基本知识。
- 如何配置 IEEE® 802.11 和 Bluetooth® 网络。
- 如何用 FreeBSD 做网络。
- 如何在无盘机上配置网络。
- 如何配置从网络 PXE 启动一个 NFS 根文件系统。
- 如何配置网络地址转换 (NAT)。
- 如何使用 PLIP 连接台计算机。
- 如何在运行 FreeBSD 的计算机上配置 IPv6。
- 如何配置 ATM。
- 如何利用 CARP，FreeBSD 支持的 Common Address Redundancy Protocol (共用地址冗余)。

在学本章之前，：

- 理解 /etc/rc 脚本的基本知识。
- 熟悉基本的网络。
- 了解如何配置和安装新的 FreeBSD 内核 ([配置FreeBSD的内核](#))。
- 了解如何安装第三方软件 ([安装应用程序](#)、[Packages](#) 和 [Ports](#))。

## 32.2. 网络和路由

要网络上的台计算机能相互通，就必须有一能描述如何从一台计算机到一台计算机的机制，这一机制称作路由(routing)。“路由”是一预先定的地址：“目的地(destination)”和“网关(gateway)”。一个地址所表示的意思是，通过网能完成与目的地的通信。有三种类型的目的地址：一个主机、子网、以及“默认”。如果没有可用的其它路由，就会使用“默认路由”，有默认路由的内容，将在后面的章节中进行。网也有三种类型：一个主机，网接口(也叫“链路(links)”)和以太网硬件地址(MAC 地址)。

### 32.2.1. 示例

为了说明路由的各个部分，首先来看看下面的例子。是 `netstat` 命令的输出：

```
% netstat -r
Routing tables
```

| Destination       | Gateway          | Flags | Refs | Use   | Netif  | Expire |
|-------------------|------------------|-------|------|-------|--------|--------|
| default           | outside-gw       | UGSc  | 37   | 418   | ppp0   |        |
| localhost         | localhost        | UH    | 0    | 181   | lo0    |        |
| test0             | 0:e0:b5:36:cf:4f | UHLW  | 5    | 63288 | ed0    | 77     |
| 10.20.30.255      | link#1           | UHLW  | 1    | 2421  |        |        |
| example.com       | link#1           | UC    | 0    | 0     |        |        |
| host1             | 0:e0:a8:37:8:1e  | UHLW  | 3    | 4601  | lo0    |        |
| host2             | 0:e0:a8:37:8:1e  | UHLW  | 0    | 5     | lo0 => |        |
| host2.example.com | link#1           | UC    | 0    | 0     |        |        |
| 224               | link#1           | UC    | 0    | 0     |        |        |

00行0出了当前配置中的默0路由 (将在 下一0 中0行介0) 和 localhost (本机) 路由。

0里的路由表中0出的用于 localhost 的接口 (Netif 列) 是 lo0, 也就是大家熟知的 "回000"。它表示所有以此0 "目的地" 的通信都留在本机, 而不通0 LAN 0出, 因00些流量最0会回到起点。

接着出0的是以 0:e0: 00的地址。0些是以太网硬件地址, 也称0 MAC 地址。 FreeBSD 会自000在同一个以太网中的任何主机 (如 test0), 并0其新0一个路由, 并通0那个以太网接口 - ed0 直接与它通0 (0者注: 那台主机)。与00路由表相0的也有一个超00 (Expire列), 当我0在指定00内没有收到从那个主机0来的信息, 00就派上用0了。00情况下, 到0个主机的路由就会被自00除。0些主机被使用一0叫做RIP(路由信息00—Routing Information Protocol)的机制所00, 00机制利用基于 "最短路径00 (shortest path determination)" 的0法0算出到本地主机的路由。

FreeBSD 也会0本地子网添加子网路由(10.20.30.255 是子网 10.20.30 的广播地址, 而 example.com 是0个子网相0的域名)。名称 link#1 代表主机上的第一0以太网0。0会00, 0于它0没有指定0外的接口。

00个0(本地网0主机和本地子网)的路由是由守00程 routed 自0配置的。如果它没有0行, 那就只有被静0定0 (例如, 明00入的) 的路由才存在了。

host1 行代表我0的主机, 它通0以太网地址来00。 因0我0是0送端, FreeBSD知道使用回0接口 (lo0) 而不是通0以太网接口来0行0送。

0个 host2 行是我0使用 ifconfig(8) 0名 (0看0于以太网的那部分就会知道我0什000做) 00生的一个0例。在 lo0 接口之后的 => 符号表明我0不0使用了回0 (因00个地址也0及了本地主机), 而且明0指出它是个0名。00路由只有在支持0名的主机上才能00出来。所有本地网上的其它的主机0于00路由只会000有 link#1。

最后一行 (目0子网224) 用于0理多播——它会覆0到其它的区域。

最后, 0个路由的不同属性可以在 Flags 列中看到。下0是个0于0些0志和它0的含0的一个0表:

|   |  |
|---|--|
| U | Up: 路由0于活0状0。                              |
| H | Host: 路由目0是0个主机。                           |
| G | Gateway: 所有0到目的地的网00到0一0程系0上, 并由它决定最后0到0里。 |



|   |  |
|---|--|
| S | Static: 一个路由是手工配置的，不是由系统自动生成的。                 |
| C | Clone: 生成一个新的路由，通过一个路由我可以接上一些机器。静态型的路由通常用于本地网。 |
| W | WasCloned: 指明一个路由——它是基于本地区域网（克隆）路由自动配置的。       |
| L | Link: 路由涉及到了以太网硬件。                             |

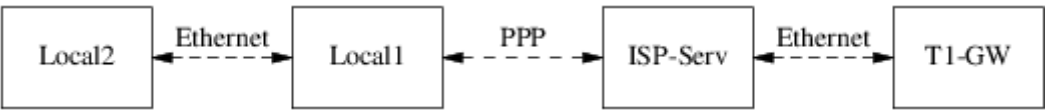
32.2.2. 默认路由

当本地系统需要与远程主机建立连接，它会查路由表以决定是否已有已知路径存在。如果远程主机属于一个我已知如何到达（克隆的路由）的子网内，那系统会查看沿着那个接口是否能连接。

如果所有已知路径都失效，系统有最后一个：“默认”路由。一个路由是特殊类型的网路路由（通常只有一个存在于系统里），并且它是在标志使用一个红色来行。对于本地区域网里的主机，一个网被置到任何与外界有直接连接的机器里（无论是通过 PPP、DSL、cable modem、T1 或其它的网接口连接）。

如果正某台本身就做网连接外界的机器配置默认路由的，那默认路由是它的“互联网服务商（ISP）”那方的网机器。

我们来看一个关于默认路由的例子。它是个很普遍的配置：



主机 Local1 和 Local2 在那。Local1 通过 PPP 号接到了 ISP。一个 PPP 服务器通过一个局域网接到一台网机器——它又通过一个外部接口接到 ISP 提供的互联网上。

它的每一台机器的默认路由是：

| Host   | Default Gateway | Interface |
|--------|-----------------|-----------|
| Local2 | Local1          | Ethernet  |
| Local1 | T1-GW           | PPP       |

一个常见的“是我什么（或谁）能将 T1-GW 置成 Local1 默认网，而不是它所接 ISP 服务器？”

住，因为 PPP 接口使用的一个地址是在 ISP 的局域网里的，用于那的连接，对于 ISP 的局域网里的其它机器，其路由会自动生。因此，我们就知道了如何到机器 T1-GW，那也就没必要中那一了——送通信 ISP 服务器。

通常使用地址 X.X.X.1 做为一个局域网的网。因此（使用相同的例子），如果本地的 C 地址空是 10.20.30，而它的 ISP 使用的是 10.9.9，那默认路由表将是：

| Host                           | Default Route       |
|--------------------------------|---------------------|
| Local2 (10.20.30.2)            | Local1 (10.20.30.1) |
| Local1 (10.20.30.1, 10.9.9.30) | T1-GW (10.9.9.1)    |

可以很轻易地通过 `/etc/rc.conf` 文件设定默认路由。在我的例子里，在主机 `Local2` 里，我在文件 `/etc/rc.conf` 里加了下内容：

```
defaultrouter="10.20.30.1"
```

也可以直接在命令行使用 `route(8)` 命令：

```
# route add default 10.20.30.1
```

要了解如何手工设置网路路由表的每一行，参考 `route(8)` 手册。

### 32.2.3. 重宿主机(Dual Homed Hosts)

有一些其它的类型的配置是我要提及的，就是一个主机属于两个不同的网。技术上，任何作网（上的例中，使用了 PPP 连接）的机器就算作是重宿主机。但这个术语上用来指那属于一个局域网之中的机器。

有一些情形，一台机器有两个网，属于各个子网都有各自的一个地址。一种情况，一台机器有一个网，但使用 `ifconfig(8)` 做了别名。如果有个独立的以太网在使用的情形就使用前者，如果只有一个物理网段，但逻辑上分成了个独立的子网，就使用后者。

两种情况都要设置路由表以便子网都知道这台主机是到其它子网的网——入站路由（inbound route）。将一台主机配置成两个子网的路由器，配置常在我需要单向或双向的包或防火被用到。

如果想主机在个接口数据包，需要激活 FreeBSD 的功能。至于怎么做，看下一部分了解更多。

### 32.2.4. 建立路由器

网路由器只是一个将数据包从一个接口到一个接口的系统。互联网标准和良好的工程实践阻止了 FreeBSD 在 FreeBSD 中把它置成默认。可以在 `rc.conf(5)` 中改下列量的值 `YES`，使个功能生效：

```
gateway_enable="YES"           # Set to YES if this host will be a gateway
```

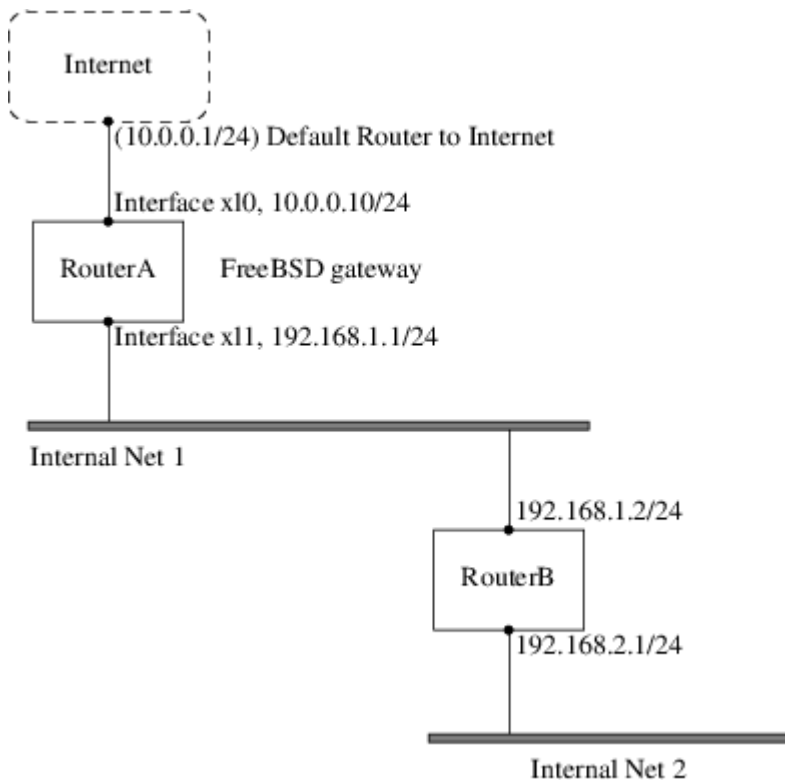
个会把 `sysctl(8)` 量——`net.inet.ip.forwarding` 置成 1。如果要地停止路由，可以把它重 0。

新的路由器需要有路由才知道将数据向何。如果网，可以使用静路由。FreeBSD 也自一个标准的 BSD 路由守护进程 `routed(8)`，称之为 RIP（version 1 和 version 2）和 IRDP。BGP v4，OSPF v2 和其它路由的支持可以从 `net/zebra` 包中得到。像 GateD® 一的商产品也提供了更的网路由解决方案。

### 32.2.5. 置静路由

#### 32.2.5.1. 手配置

假如如下一个网：



在这里，RouterA 是我的 FreeBSD 机器，它充当连接到互联网其它部分的路由器的角色。默认路由位置 10.0.0.1，它就允许与外界连接。我假定已经配置了 RouterB，并且知道如何连接到想去的任何地方。（在这个例子里很简单。只要在 RouterB 上加默认路由，使用 192.168.1.1 做网关。）

如果我看看 RouterA 的路由表，我就会看到如下一些内容：

```
% 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
```

使用当前的路由表，RouterA 是不能到达我的内网——Internal Net 2 的。它没有到 192.168.2.0/24 的路由。一种可以接受的方法是手工添加路由。以下的命令会把 Internal Net 2 网加入到 RouterA 的路由表中，使用 192.168.1.2 做下一个跳：

```
# route add -net 192.168.2.0/24 192.168.1.2
```

现在在 RouterA 就可以到达 192.168.2.0/24 网上的任何主机了。

### 32.2.5.2. 永久配置

上面的例子对于运行着的系统来配置静态路由是相当不错了。只是，有一个问题——如果重启的 FreeBSD 机器，路由信息就会消失。处理附加的静态路由的方法是把它放到系统的 /etc/rc.conf 文件里去。

```
# Add Internal Net 2 as a static route
static_routes="internalnet2"
route_internalnet2="-net 192.168.2.0/24 192.168.1.2"
```

配置变量 `static_routes` 是一串以空格隔开的字符串。一串表示一个路由名字。 在上面的例子中我有一个串在 `static_routes` 里。一个字符串中 `internalnet2`。 然后我新增一个配置变量 `route_internalnet2`， 这里我把所有 `route(8)` 命令的参数拿了来。 在上面的例子中的我使用的命令是：

```
# route add -net 192.168.2.0/24 192.168.1.2
```

因此，我需要的是 `"-net 192.168.2.0/24 192.168.1.2"`。

前面已提到， 可以把多个静态路由的名称， 放到 `static_routes` 里。 接着我就来建立多个静态路由。 下面几行所展示的， 是在一个假想的路由器上加 `192.168.0.0/24` 和 `192.168.1.0/24` 之静态路由的例子：

```
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"
```

### 32.2.6. 路由广播

我已知道了如何定向通向外界的路由， 但未及外界是如何到我这里的。

我已知道可以设置路由表， 任何指向特定地址空间（在我的例子中是一个 C 子网）的数据都会被送往网上特定的主机， 然后由这台主机向地址空间内部的数据。

当一个得到一个分配的网段的地址空间， ISP(网服务商)会设置它的路由表， 指向子网的数据就会通过 PPP 接口下到网。 但是其它跨越国界的网是如何知道将数据送到 ISP 的？

有一个系统(很像分布式 DNS 信息系统)， 它一直跟踪被分配的地址空间， 并表明它接到互联网骨干(Internet backbone)的点。 "骨干(Backbone)" 指的是全世界和跨国的网的主要干线。 一台骨干主机(backbone machine)有一主要表集的副本， 它将送特定网段的数据向相应的骨干体上(backbone carrier)， 从点往下遍服提供商， 直到数据到网的网。

服务提供商的任是向骨干网广播， 以声明它就是通向网的网点的接口点 (以及输入的路径)。 这就是路由广播。

### 32.2.7. 解答

有时候，路由广播会有一个网，一些网无法与网接。 或能看出路由是在哪里中断的最有用的命令就是 `traceroute(8)` 了。 当无法与远程主机接口， 一个命令一有用(例如 `ping(8)` 失败)。

`traceroute(8)` 命令将以想接口的主机的名字作参数运行。 不管是到了目的地， 还是因没有接口而中止， 它都会显示所经过的所有网主机。

想了解更多的信息， 看 `traceroute(8)` 的手册。

### 32.2.8. 多播路由

FreeBSD 一开始就支持多播用户件和多播路由。多播程序并不要求 FreeBSD 的任何特殊的配置，就可以工作得很好。多播路由需要支持被调入内核：

```
options MROUTING
```

另外，多播路由守护进程——[mrouted\(8\)](#) 必须通过 `/etc/mrouted.conf` 配置来打开通道和 DVMRP。更多关于多播路由配置的信息可以在 [mrouted\(8\)](#) 的手册里得到。



多播路由服务 [mrouted\(8\)](#) 用了 DVMRP 多播路由，在大多采用多播的场合，它已被 [pim\(4\)](#) 取代。[mrouted\(8\)](#) 以及相关的 [map-mbone\(8\)](#) 和 [mrinfo\(8\)](#) 工具可以在 FreeBSD 的 Ports Collection [net/mrouted](#) 中得到。

## 32.3. 无线网络

### 32.3.1. 无线网络基础

大多数无线网络都采用了 IEEE® 802.11 标准。基本的无线网络中，都包含多个以 2.4GHz 或 5GHz 频段的无线波广播的站点（不过，随所地域的不同，或者为了能够更好地通行，具体的频率会在 2.3GHz 和 4.9GHz 的范围内变化）。

802.11 网络有几种方式：在 *infrastructure* 模式中，一个通信站作主站，其他通信站都与其相连；网络称作 BSS，而主站成为无线接入点 (AP)。在 BSS 中，所有的通信都是通过 AP 来完成的；即使通信站之间要相互通信，也必须将消息通过 AP。在第二种形式的网络中，并不存在主站，通信站之间是直接通信的。这种网络形式称作 IBSS，通常也叫做 *ad-hoc* 网络。

802.11 网络最初在 2.4GHz 频段上部署，并采用了由 IEEE® 802.11 和 802.11b 标准所定义的规则。这些标准定义了采用的操作频率、包括分片和速率（通信过程中可以使用不同的速率）在内的 MAC 特性等。后来的 802.11a 标准定义了使用 5GHz 频段进行操作，以及不同的信号机制和更高的速率。其后定义的 802.11g 标准用了在 2.4GHz 上如何使用 802.11a 信号和速率机制，以提供最早的 802.11b 网络的向前兼容。

802.11 网络中采用的各种底层机制提供了不同类型的安全机制。最初的 802.11 标准定义了一种称作 WEP 的网络安全。它采用固定的密钥，并使用 RC4 加密算法来对在网络上传输的数据进行加密。全部通信站都必须采用同样的固定密钥才能通信。这一格局已被证明很容易被攻破，因此目前已很少使用了，采用这种方法只能使那些接入网络的用户迅速断网。最新的安全实践是由 IEEE® 802.11i 标准定义的，它定义了新的加密算法，并通过一种附加的密钥来通信站向无线接入点本身，并交给用于传输数据通信的密钥。更进一步，用于加密的密钥会定期地刷新，而且有机能能抵御入侵的尝试（并阻止窃听）。无线网络中一种常用的安全标准是 WPA。它是在 802.11i 之前由业界制定的一种过渡性标准。WPA 定义了 802.11i 中所定义的要求的子集，并被用来在旧式硬件上实施。特别地，WPA 要求只使用由最初 WEP 所采用的算法派生的 TKIP 加密算法。802.11i 不但允许使用 TKIP，而且要求支持更强的加密算法 AES-CCM 来用于加密数据。（在 WPA 中并没有要求使用 AES 加密算法，因为在旧式硬件上实施该算法所需的计算量性太高。）

除了前面介绍的那些标准之外，还有一种需要介绍的标准是 802.11e。它定义了用于在 802.11 网络上进行多媒体应用，如流媒体和使用 IP 传送的语音 (VoIP) 的规则。与 802.11i 类似，802.11e 也有一个前身标准，通常称作 WME（后改名为 WMM），它也是由业界定义的 802.11e 的子集，以便能在旧式硬件中使用多媒体应用。对于 802.11e 与 WME/WMM 之间的一个重要区别是，前者允许流量通过服务品质 (QoS) 规则和媒体流来安排

首先。对于某些网卡的正向，能够高速突发数据和流量分配。

FreeBSD 支持采用 802.11a, 802.11b 和 802.11g 的网卡。类似地，它 also 支持 WPA 和 802.11i 安全（与 11a、11b 和 11g 配合），而 WME/WMM 所需要的 QoS 和流量分配，它在部分无线网卡上提供了支持。

## 32.3.2. 基本安装

### 32.3.2.1. 内核配置

要使用无线网卡，需要一块无线网卡，并本地配置内核令其提供无线网卡支持。后者被分成了多个模块，因此只需配置使用所需要的部件就可以了。

首先需要的是一个无线网卡。最常用的一块无线网卡配件是 Atheros 生产的。这些网卡由 [ath\(4\)](#) 程序提供支持，需要把下面的配置加入到 `/boot/loader.conf` 文件中：

```
if_ath_load="YES"
```

Atheros 分为三个部分：一部分 ([ath\(4\)](#))、用于物理芯片有功能的支持 ([ath\\_hal\(4\)](#))，以及一个用以计算速率的算法 (`ath_rate_sample` here)。当以模块方式加载一支持，所需的其它模块会自动加载。如果你使用的不是 Atheros 网卡，看看其它的模块；例如：

```
if_wi_load="YES"
```

表示使用基于 Intersil Prism 产品的无线网卡 ([wi\(4\)](#) 模块)。



在本文余下的部分中，我将以 [ath\(4\)](#) 为例进行示范，如果要套用这些配置的，可能需要根据网卡的配置情况来修改示例中的名称。在 FreeBSD 兼容硬件说明中提供了目前可用的无线网卡，以及兼容硬件的列表。不同版本和硬件平台的说明可以在 FreeBSD 网站的 [Release Information](#) 页面找到。如果你的无线网卡没有与之匹配的 FreeBSD 驱动程序，也可以使用 [NDIS](#) 封装机制来直接使用 Windows® 网卡。

对于 FreeBSD 7.X，在配置好无线网卡之后，需要引入驱动程序所需要的 802.11 网卡支持。对于 [ath\(4\)](#) 而言，至少需要 [wlan\(4\)](#) [wlan\\_scan\\_ap](#) 和 [wlan\\_scan\\_sta](#) 模块；[wlan\(4\)](#) 模块会自动随无线网卡一同加载，剩下的模块必需要在系统引导时加载，就需要在 `/boot/loader.conf` 中加入下面的配置：

```
wlan_scan_ap_load="YES"
wlan_scan_sta_load="YES"
```

从 FreeBSD 8.0 起，这些模块成为了 [wlan\(4\)](#) 模块的基件，并会随配置器一起加载。

除此之外，还需要提供希望使用的安全所需的加密支持模块。这些模块是来自 [wlan\(4\)](#) 模块根据需要自动加载的，但目前必须手工进行配置。可以使用下面这些模块：[wlan\\_wep\(4\)](#)、[wlan\\_ccmp\(4\)](#) 和 [wlan\\_tkip\(4\)](#)。[wlan\\_ccmp\(4\)](#) 和 [wlan\\_tkip\(4\)](#) 两个模块都只有在希望采用 WPA 和/或 802.11i 安全时才需要。如果你的网卡不采用加密，就不需要 [wlan\\_wep\(4\)](#) 支持了。要在系统引导时加载这些模块，需要在 `/boot/loader.conf` 中加入下面的配置：

```
wlan_wep_load="YES"
wlan_ccmp_load="YES"
wlan_tkip_load="YES"
```

通过系引导配置文件（也就是 `/boot/loader.conf`）中的信息生效，必须重新运行 FreeBSD 的计算机。如果不想立刻重新，也可以使用 [kldload\(8\)](#) 来手工加。

如果不想加载模，也可以将些加载到内核中，方法是在内核的配置文件加入下面的配置：



```
device wlan          # 802.11 support
device wlan_wep      # 802.11 WEP support
device wlan_ccmp     # 802.11 CCMP support
device wlan_tkip     # 802.11 TKIP support
device wlan_amrr     # AMRR transmit rate control algorithm
device ath           # Atheros pci/cardbus NIC's
device ath_hal       # pci/cardbus chip support
options AH_SUPPORT_AR5416 # enable AR5416 tx/rx descriptors
device ath_rate_sample # SampleRate tx rate control for ath
```

使用 FreeBSD 7.X，需要配置下面行；FreeBSD 的其他版本不需要它。

```
device wlan_scan_ap    # 802.11 AP mode scanning
device wlan_scan_sta   # 802.11 STA mode scanning
```

将些信息写到内核配置文件中之后，需要重新内核，并重新运行 FreeBSD 的计算机。

在系之后，会在引出的信息中，到似下面于无的信息：

```
ath0: <Atheros 5212> mem 0x88000000-0x8800ffff irq 11 at device 0.0 on cardbus1
ath0: [ITHREAD]
ath0: AR2413 mac 7.9 RF2413 phy 4.5
```

### 32.3.3. Infrastructure 模式

通常的情形中使用的是 infrastructure 模式或称 BSS 模式。在模式中，有一系列无点接入了有网。个无网都会有自己的名字，个名称作网的 SSID。无客端都通无点来完成接入。

#### 32.3.3.1. FreeBSD 客机

##### 32.3.3.1.1. 如何无点

可以通过使用 `ifconfig` 命令来描网。由于系需要在操作程中切不同的无率并探可用的无点，求可能需要数分才能完成。只有超用才能描：



```
# ifconfig wlan0 create wlandev ath0
# ifconfig wlan0 up scan
SSID/MESH ID      BSSID                CHAN RATE   S:N        INT CAPS
dlinkap           00:13:46:49:41:76   11  54M  -90:96    100 EPS   WPA WME
freebsdap         00:11:95:c3:0d:ac   1   54M  -83:96    100 EPS   WPA
```



在开始扫描之前，必须将网口 up。后面的扫描就不再需要将网口 up 了。

在 FreeBSD 7.X 中，会直接配置器，例如 ath0，而不是 wlan0。因此需要把前面的命令行改：



```
# ifconfig ath0 up scan
```

在本文余下的部分中，也需要注意 FreeBSD 7.X 上的一些差异，并命令示例行类似的改。

扫描会列出所求到的所有 BSS/IBSS 网列表。除了网的名字 SSID 之外，我们还会看到 BSSID 即无点的 MAC 地址。而 CAPS 字段出了网型及其提供的功能，其中包括：

表 13. 通站功能代

| 功能代 | 含  |
|-----|--|
| E   | Extended Service Set (ESS)。表示通站是 infrastructure 网 (相对于 IBSS/ad-hoc 网) 的成。                          |
| I   | IBSS/ad-hoc 网。表示通站是 ad-hoc 网 (相对于 ESS 网) 的成。   |
| P   | 私密。在 BSS 中交的全部数据均需保数据保密性。表示 BSS 需要通站使用加密算法，例如 WEP、TKIP 或 AES-CCMP 来加密/解密与其他通站交的数据。                 |
| S   | 短前 (Short Preamble)。表示网采用的是短前 (由 802.11b High Rate/DSSS PHY 定，短前采用 56-位同字段，而不是在长前模式中所采用的 128-位字段)。 |
| s   | 短槽 (Short slot time)。表示由于不存在旧式 (802.11b) 通站，802.11g 网正使用短槽。  |

要示目前已知的网，可以使用下面的命令：

```
# ifconfig wlan0 list scan
```

些信息可能会由无配器自更新，也可使用 scan 手更新。快取存中的旧数据会自除，因此除非



行更多扫描，每个列表会逐次小。

### 32.3.3.1.2. 基本配置

在这一节中我们将展示一个例子来介绍如何配置无线网卡在 FreeBSD 中以不加密的方式工作。在熟悉了这些概念之后，我们强烈建议在它的使用中采用 [WPA](#) 来配置网络。

配置无线网络的程序可分三个基本步骤：配置无线点、配置通信站身，以及配置 IP 地址。下面的几节中将分别地介绍它们。

#### 32.3.3.1.2.1. 配置无线点

多数时候系统以内建的探测方式配置无线点就可以了。这是在将网络接口置为 up 或在 `/etc/rc.conf` 中配置 IP 地址的默认方式，例如：

```
wlans_ath0="wlan0"
ifconfig_wlan0="DHCP"
```



如前面提到的那样，FreeBSD 7.X 只需要一行配置：

```
ifconfig_ath0="DHCP"
```

如果存在多个无线点，而我们希望从中具体的一个，可以通过指定 SSID 来：

```
wlans_ath0="wlan0"
ifconfig_wlan0="ssid your_ssid_here DHCP"
```

在某些环境中，多个点可能会使用相同的 SSID（通常，我们做的目的是漫游），我们可能就需要与某个具体的点了。这种情况下，指定无线点的 BSSID（可以不指定 SSID）：

```
wlans_ath0="wlan0"
ifconfig_wlan0="ssid your_ssid_here bssid xx:xx:xx:xx:xx:xx DHCP"
```

除此之外，还有一些其它的方法能约束无线点的，例如限制系统扫描的频段，等等。如果的无线网卡支持多个频段，这样做可能会非常有用，因为扫描全部可用频段是一个十分耗时的过程。要将操作限制在某个具体的频段，可以使用 `mode` 参数；例如：

```
wlans_ath0="wlan0"
ifconfig_wlan0="mode 11g ssid your_ssid_here DHCP"
```

就会限制使用采用 2.4GHz 的 802.11g，我们在扫描的时候，就不会考虑那些 5GHz 的频段了。除此之外，我们可以通过 `channel` 参数来将操作固定在特定速率，以及通过 `chanlist` 参数来指定扫描的频段列表。对于这些参数的一些信息，可以在手册 [ifconfig\(8\)](#) 中找到。

### 32.3.3.1.2.2. 身体

一旦定义了无线点，无线站就需要完成身体，以便开始发送和接收数据。身体可以通过多种方式行，最常用的一种方式称开放，它允许任意无线站加入网络并相互通信。这种方式只在第一次配置无线网络时使用。其它的方式需要在行数据通之前，首先行密码握手；有些方式要使用先分的密码或密，要是用更一些的后台服，如 RADIUS。大多数用会使用默的开放，而第二多的是 WPA-PSK，它也称个人 WPA，在 [下面](#) 的章节中将行介。

如果使用 Apple® AirPort® Extreme 基站作无线点，可能需要同时在端配置 WEP 共享密码。可以通过在 `/etc/rc.conf` 文件中行置，或使用 [wpa\\_supplicant\(8\)](#) 程序来手工完成。如果只有一个 AirPort® 基站，可以用似下面的方法来配置：



```
wlans_ath0="wlan0"
ifconfig_wlan0="authmode shared wepmode on weptxkey 1 wepkey 01234567
DHCP"
```

一般而言，尽量避免使用共享密码的方法，因为它以非常受限的方式使用 WEP 密，使得攻者能很容易地破解密。如果必须使用 WEP (例如，为了兼容旧式的) 最好使用 WEP 配合 [open](#) 方式。于 WEP 的更多料参 [WEP](#)。

### 32.3.3.1.2.3. 通 DHCP 取 IP 地址

在定义了无线点，并配置了参数之后，必须得 IP 地址才能真正始通。多数候，会通 DHCP 来得无线 IP 地址。要到个目的，需要 `/etc/rc.conf` 并在配置中加入 [DHCP](#)：

```
wlans_ath0="wlan0"
ifconfig_wlan0="DHCP"
```

在已完成了用无线网络接口的全部准备工作了，下面的操作将用它：

```
# /etc/rc.d/netif start
```

一旦网络接口始行，就可以使用 [ifconfig](#) 来看网络接口 `ath0` 的状态了：

```
# ifconfig wlan0
wlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
ether 00:11:95:d5:43:62
inet 192.168.1.100 netmask 0xffffffff broadcast 192.168.1.255
media: IEEE 802.11 Wireless Ethernet OFDM/54Mbps mode 11g
status: associated
ssid dlinkap channel 11 (2462 Mhz 11g) bssid 00:13:46:49:41:76
country US ecm authmode OPEN privacy OFF txpower 21.5 bmiss 7
scanvalid 60 bgscan bgscanintvl 300 bgscanidle 250 roam:rssi 7
roam:rate 5 protmode CTS wme burst
```

里的 `status: associated` 表示已连接到了无线网络（在这个例子中，网络的名字是 `dlinkap`）。`ssid 00:13:46:49:41:76` 是指所用无线点的 MAC 地址；`authmode OPEN` 表示通过的内容将不加密。

#### 32.3.3.1.2.4. 静态 IP 地址

如果无法从某个 DHCP 服务器得 IP 地址，可以配置一个静态 IP 地址，方法是将前面的 `DHCP` 两字替换地址信息。必须保持其他用于连接无线点的参数：

```
wlans_ath0="wlan0"
ifconfig_wlan0="inet 192.168.1.100 netmask 255.255.255.0 ssid your_ssid_here"
```

#### 32.3.3.1.3. WPA

WPA (Wi-Fi 保护) 是与 802.11 网配合使用的安全，其目的是消除 WEP 中缺少身份能力的，以及一些其它的安全弱点。WPA 采用了 802.1X 认证，并采用从多种与 WEP 不同的加密算法中一种来保证数据保密性。WPA 支持的唯一加密算法是 TKIP (临时密钥完整性)，TKIP 是一种 WEP 所采用的基本 RC4 加密算法的扩展，除此之外提供了防止入侵的机制。TKIP 被用来与旧式硬件一同工作，只需要进行部分软件修改；它提供了一种改善安全性的折衷方案，但仍有可能受到攻击。WPA 也指定了 AES-CCMP 加密作为 TKIP 的替代品，在可能倾向于使用加密；表 32-1 的常用是 WPA2 (或 RSN)。

WPA 定义了认证和加密。通常使用两种方法之一来完成的：通过 802.1X 或类似 RADIUS 认证的后端服务器，或通过无线站和无线点之间的认证事先分配的密钥来进行最小握手。前者通常称作企业 WPA，而后者通常也叫做个人 WPA。因多数人不会在无线网配置 RADIUS 后端服务器，因此 WPA-PSK 是在 WPA 中最常用的一种。

无线连接的控制和身份工作（密钥协商或认证服务器）是通过 `wpa_supplicant(8)` 工具来完成的。该程序需要一个配置文件，`/etc/wpa_supplicant.conf`。关于该文件的更多信息，请参考手册 `wpa_supplicant.conf(5)`。

##### 32.3.3.1.3.1. WPA-PSK

WPA-PSK 也称作个人-WPA，它基于事先分配的密钥 (PSK)，该密钥是根据作为无线网上使用的主密钥的密钥生成的。它表示每个无线点都会使用相同的密钥。WPA-PSK 主要用于小型网，在网中，通常不需要或没有框架服务器。



无论如何，都使用足够长，且包括尽可能多字母和数字的口令，以免被猜出和/或攻击。

第一步是修改配置文件 `/etc/wpa_supplicant.conf`，并在其中加入在无线网上使用的 SSID 和事先分配的密钥：

```
network={
    ssid="freebsdap"
    psk="freebsdmail"
}
```

接下来，在 `/etc/rc.conf` 中，我将指定无线的配置，令其采用 WPA，并通过 DHCP 来取 IP 地址：

```
wlans_ath0="wlan0"
ifconfig_wlan0="WPA DHCP"
```

下面用无线网接口：

```
# /etc/rc.d/netif start
Starting wpa_supplicant.
DHCPDISCOVER on wlan0 to 255.255.255.255 port 67 interval 5
DHCPDISCOVER on wlan0 to 255.255.255.255 port 67 interval 6
DHCPOFFER from 192.168.0.1
DHCPREQUEST on wlan0 to 255.255.255.255 port 67
DHCPACK from 192.168.0.1
bound to 192.168.0.254 -- renewal in 300 seconds.
wlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    ether 00:11:95:d5:43:62
    inet 192.168.0.254 netmask 0xffffffff broadcast 192.168.0.255
    media: IEEE 802.11 Wireless Ethernet OFDM/36Mbps mode 11g
    status: associated
    ssid freebsdap channel 1 (2412 Mhz 11g) bssid 00:11:95:c3:0d:ac
    country US ecm authmode WPA2/802.11i privacy ON deftxkey UNDEF
    AES-CCM 3:128-bit txpower 21.5 bmiss 7 scanvalid 450 bgscan
    bgscanintvl 300 bgscanidle 250 roam:rssi 7 roam:rate 5 protmode CTS
    wme burst roaming MANUAL
```

除此之外，也可以手动地使用 [above](#) 中那个 `/etc/wpa_supplicant.conf` 来配置，方法是运行：

```
# wpa_supplicant -i wlan0 -c /etc/wpa_supplicant.conf
Trying to associate with 00:11:95:c3:0d:ac (SSID='freebsdap' freq=2412 MHz)
Associated with 00:11:95:c3:0d:ac
WPA: Key negotiation completed with 00:11:95:c3:0d:ac [PTK=CCMP GTK=CCMP]
CTRL-EVENT-CONNECTED - Connection to 00:11:95:c3:0d:ac completed (auth) [id=0 id_str=]
```

接下来的操作，是运行 `dhclient` 命令来从 DHCP 服务器取 IP：

```
# dhclient wlan0
DHCPREQUEST on wlan0 to 255.255.255.255 port 67
DHCPACK from 192.168.0.1
bound to 192.168.0.254 -- renewal in 300 seconds.
# ifconfig wlan0
wlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    ether 00:11:95:d5:43:62
    inet 192.168.0.254 netmask 0xffffffff broadcast 192.168.0.255
    media: IEEE 802.11 Wireless Ethernet OFDM/36Mbps mode 11g
    status: associated
    ssid freebsdap channel 1 (2412 Mhz 11g) bssid 00:11:95:c3:0d:ac
    country US ecm authmode WPA2/802.11i privacy ON deftxkey UNDEF
    AES-CCM 3:128-bit txpower 21.5 bmiss 7 scanvalid 450 bgscan
    bgscanintvl 300 bgscanidle 250 roam:rssi 7 roam:rate 5 protmode CTS
    wme burst roaming MANUAL
```



如果在 `/etc/rc.conf` 中把 `ifconfig_wlan0` 置成了 DHCP (像 `ifconfig_wlan0="DHCP"` 那样), 那在 `wpa_supplicant` 上了无接入点 (AP) 之后, 会自行 `dhclient`。

如果不打算使用 DHCP 或者 DHCP 不可用, 可以在 `wpa_supplicant` 通站完成了身之后, 指定静 IP 地址:

```
# ifconfig wlan0 inet 192.168.0.100 netmask 255.255.255.0
# ifconfig wlan0
wlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    ether 00:11:95:d5:43:62
    inet 192.168.0.100 netmask 0xffffffff broadcast 192.168.0.255
    media: IEEE 802.11 Wireless Ethernet OFDM/36Mbps mode 11g
    status: associated
    ssid freebsdap channel 1 (2412 Mhz 11g) bssid 00:11:95:c3:0d:ac
    country US ecm authmode WPA2/802.11i privacy ON deftxkey UNDEF
    AES-CCM 3:128-bit txpower 21.5 bmiss 7 scanvalid 450 bgscan
    bgscanintvl 300 bgscanidle 250 roam:rssi 7 roam:rate 5 protmode CTS
    wme burst roaming MANUAL
```

如果没有使用 DHCP, 需要手工配置默认网, 以及域名服务器:

```
# route add default your_default_router
# echo "nameserver your_DNS_server" >> /etc/resolv.conf
```

### 32.3.3.1.3.2. 使用 EAP-TLS 的 WPA

使用 WPA 的第二方式是使用 802.1X 后端服务器。在这个例子中, WPA 也称作 企业-WPA, 以便与安全性差、采用事先分密的个人-WPA 区分来。在企业-WPA 中, 操作是采用 EAP 完成的 (可展)。

EAP 并未附加加密方法。因此者决定将 EAP 放在加密信道中进行送。目前有多种 EAP 方法,

最常用的方法是 EAP-TLS、EAP-TTLS 和 EAP-PEAP。

EAP-TLS (最安全 的 EAP) 是在无线世界中得到了广泛支持的协议，因它是 [Wi-Fi 联盟](#) 核准的第一个 EAP 方法。EAP-TLS 需要使用三个组件：CA 证书 (在所有计算机上安装)、用以向服务器身份的服务器证书，以及一个无线客户端用于证明身份的客户端证书。在 EAP 方式中，服务器和无线客户端均通过自己的证书向对方证明身份，它们均使用方的证书是本地证书的颁发机构 (CA) 的。

与之前介绍的方法类似，配置也是通过 `/etc/wpa_supplicant.conf` 来完成的：

```
network={
    ssid="frebsdap" ①
    proto=RSN ②
    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="frebsdmailclient" ⑨
}
```

- ① 这个字段表示网络名 (SSID)。
- ② 这里，我使用 RSN (IEEE® 802.11i) 协议，也就是 WPA2。
- ③ `key_mgmt` 行表示所用的密钥管理协议。在我的例子中，它是使用 EAP 协议的 WPA：**WPA-EAP**。
- ④ 这个字段中，提到了我的设备采用 EAP 方式。
- ⑤ `identity` 字段包含了 EAP 的标识串。
- ⑥ `ca_cert` 字段指出了 CA 证书文件的路径名。在服务器端，这个文件是必需的。
- ⑦ `client_cert` 行指出了客户端证书的路径名。每个无线客户端而言，这个证书都是在全网内唯一的。
- ⑧ `private_key` 字段是客户端私钥文件的路径名。
- ⑨ `private_key_passwd` 字段是私钥的口令字。

接着，把下面的配置写入 `/etc/rc.conf`：

```
wlans_ath0="wlan0"
ifconfig_wlan0="WPA DHCP"
```

下一步是使用 `rc.d` 机制来启用网络接口：

```
# /etc/rc.d/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 0xffffffff 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
```

如前面提到的那样，也可以手工通过 `wpa_supplicant` 和 `ifconfig` 命令达到类似的目的。

### 32.3.3.1.3.3. 使用 EAP-TTLS 的 WPA

在使用 EAP-TLS 时，参与进程的服务器和客户端都需要它，而在使用 EAP-TTLS (通过安全隧道的 EAP) 时，客户端则是可选的。这种方式与某些安全 web 站点更接近，即使你没有客户端，这些 web 服务器也能建立安全的 SSL 隧道。EAP-TTLS 会使用加密的 TLS 隧道来传送信息。

对于它的配置，同是通过 `/etc/wpa_supplicant.conf` 文件来执行的：

```
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 这个字段是我所采用的 EAP 方式。
- ② `identity` 字段中是在加密 TLS 隧道中用于 EAP 的身份串。
- ③ `password` 字段中是用于 EAP 的口令字。
- ④ `ca_cert` 字段指出了 CA 文件的路径名。在服务器端，这个文件是必需的。
- ⑤ 这个字段中指出了加密 TLS 隧道中使用的认证方式。在这个例子中，我使用的是 MD5-加密口令的 EAP。"inner authentication" (注：内部认证) 通常也叫 "phase2"。

必须把下面的配置写入 `/etc/rc.conf`：

```
wlans_ath0="wlan0"
ifconfig_wlan0="WPA DHCP"
```

下一个是用网口：

```
# /etc/rc.d/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 0xffffffff broadcast 192.168.0.255
    media: IEEE 802.11 Wireless Ethernet DS/11Mbps mode 11g
    status: associated
    ssid freebsdap channel 1 (2412 Mhz 11g) bssid 00:11:95:c3:0d:ac
    country US ecm authmode WPA2/802.11i privacy ON deftxkey UNDEF
    AES-CCM 3:128-bit txpower 21.5 bmiss 7 scanvalid 450 bgscan
    bgscanintvl 300 bgscanidle 250 roam:rssi 7 roam:rate 5 protmode CTS
    wme burst roaming MANUAL
```

#### 32.3.3.1.3.4. 使用 EAP-PEAP 的 WPA



PEAPv0/EAP-MSCHAPv2 是最常用的 PEAP 方法。此文节的以下部分将使用 PEAP 指代一些方法。

PEAP (受保护的 EAP) 被用来替代 EAP-TTLS，并且是在 EAP-TLS 之后最常用的 EAP 标准。换言之，如果你的网中有多个不同的操作系统，PEAP 将是仅次于 EAP-TLS 的支持最广的标准。

PEAP 与 EAP-TTLS 很像：它使用服务器端，通过在客户端与服务器之间建立加密的 TLS 隧道来向用户身份，保护了信息的交换过程。在安全方面，EAP-TTLS 与 PEAP 的区别是 PEAP 会以明文广播用户名，只有口令是通过加密 TLS 隧道送的。而 EAP-TTLS 在送用户名和口令，都使用 TLS 隧道。

我们需要编辑 /etc/wpa\_supplicant.conf 文件，并加入与 EAP-PEAP 有关的配置：



```
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 该字段的内容是用于连接的 EAP 方式。

② **identity** 字段中是在加密 TLS 隧道中用于 EAP 的身份串。

③ **password** 字段中是用于 EAP 的口令字。

④ **ca\_cert** 字段指出了 CA 文件的路径名。在服务器端，该文件是必需的。

⑤ 该字段包含了第一阶段（TLS 隧道）的参数。随使用的服务器的不同，需要指定不同的值。多数时候，该值是 "客户端 EAP 加密"，可以通过使用 **peaplabel=0** 来指定。更多信息可以在手机手册 [wpa\\_supplicant.conf\(5\)](#) 中找到。

⑥ 该字段的内容是需要在加密的 TLS 隧道中使用的信息。对 PEAP 而言，该值是 **auth=MSCHAPV2**。

必须把下面的配置加入到 `/etc/rc.conf`：

```
wlans_ath0="wlan0"
ifconfig_wlan0="WPA DHCP"
```

下一节是用网口：

```
# /etc/rc.d/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 0xffffffff broadcast 192.168.0.255
    media: IEEE 802.11 Wireless Ethernet DS/11Mbps mode 11g
    status: associated
    ssid freebsdap channel 1 (2412 Mhz 11g) bssid 00:11:95:c3:0d:ac
    country US ecm authmode WPA2/802.11i privacy ON deftxkey UNDEF
    AES-CCM 3:128-bit txpower 21.5 bmiss 7 scanvalid 450 bgscan
    bgscanintvl 300 bgscanidle 250 roam:rssi 7 roam:rate 5 protmode CTS
    wme burst roaming MANUAL
```

#### 32.3.3.1.4. WEP

WEP（有等效的）是最初 802.11 标准的一部分。其中没有提供身份验证机制，只提供了弱认证控制，而且很容易破解。

WEP 可以通过 `ifconfig` 配置：

```
# ifconfig wlan0 create wlandev ath0
# ifconfig wlan0 inet 192.168.1.100 netmask 255.255.255.0 \
    ssid my_net wepmode on weptxkey 3 wepkey 3:0x3456789012
```

- `wepkey` 指明了使用哪个 WEP 密钥来加密数据。这里我使用第三个密钥。它必须与无线接入点的配置一致。如果不清楚哪个无线接入点，用 `1`（就是第一个密钥）来设置即可。
- `wepkey` 用于 WEP 密钥。其格式为 `index:key`，`key` 默认为 `1`；如果需要设置的密钥不是第一个，就必需指定 `index` 了。



需要将 `0x3456789012` 改在无线接入点上配置的那个。

我建议读者参考手册 `ifconfig(8)` 来了解更多的信息。

`wpa_supplicant` 机制也可以用来配置无线网使用 WEP。前面的例子也可以通过在 `/etc/wpa_supplicant.conf` 中加入下述配置来实现：

```
network={
    ssid="my_net"
    key_mgmt=NONE
    wep_key3=3456789012
    wep_tx_keyidx=3
}
```

接着：

```
# wpa_supplicant -i wlan0 -c /etc/wpa_supplicant.conf
Trying to associate with 00:13:46:49:41:76 (SSID='dlinkap' freq=2437 MHz)
Associated with 00:13:46:49:41:76
```

### 32.3.4. Ad-hoc 模式

IBSS 模式，也称 ad-hoc 模式，是点对点连接的。例如，如果希望在计算机 A 和 B 之间建立 ad-hoc 网，我只需一个 IP 地址和一个 SSID 就可以了。

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 0xffffffff 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
```

此处的 **adhoc** 参数表示无线接口以 IBSS 模式。

此时，在 B 上可能看到 A 的存在了：

```
# 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
```

在输出中的 **I** 再次显示了 A 机是以 ad-hoc 模式运行的。我只需 B 配置一不同的 IP 地址：

```
# 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 0xffffffff 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
```

此时，A 和 B 就可以交换信息了。

### 32.3.5. FreeBSD 基于主机的（无线）网络接入点

FreeBSD 可以作为一个（无线）网络接入点（AP），可以不必再去一个硬件 AP 或者使用 ad-hoc 模式的网络。当的 FreeBSD 机器作为网络接到另外一个网络的候将非常有用。

#### 32.3.5.1. 基本配置

在把的 FreeBSD 机器配置成一个 AP 以前，首先需要先在内核配置好无线网络的无线支持。当然需要加上想用的安全。想得更的信息，参 [基本安装](#)。



目前不支持使用 Windows® 和 NDIS 包装的网做 AP 使用。只有 FreeBSD 原生的无线网卡支持 AP 模式。

一旦装了无线网的支持，您就可以看一下看看的无线网卡是否支持基于主机的无线接入模式（通常也被称为 hostap 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>
```

这段输出显示了网所支持的各项功能；其中的关键字 **HOSTAP** 表示网可以作无线接入点来使用。此外，这里还会输出所支持的加密算法：WEP、TKIP、AES，等等。这些信息对于知道在接入点上使用何安全非常重要。

只有建网后才能配置无线网卡是否以 hostap 模式运行，如果之前已存在了相关的网，需要首先将其：

```
# ifconfig wlan0 destroy
```

接着，在配置其它参数前，以正好的网重新生成：

```
# 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** 网 wlan0 网接口的状态：

```
# 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 0xffffffff broadcast 192.168.0.255
media: IEEE 802.11 Wireless Ethernet autoselect mode 11g <hostap>
status: running
ssid freebsdap channel 1 (2412 Mhz 11g) bssid 00:11:95:c3:0d:ac
country US ecm authmode OPEN privacy OFF txpower 21.5 scanvalid 60
protmode CTS wme burst dtimperiod 1 -dfs
```

**hostap** 参数指定了接口以主机接入点的方式运行。

通常在 `/etc/rc.conf` 中加入下面的配置，也可以在系统启动的过程中自动完成关于网接口的配置：

```
wlans_ath0="wlan0"
create_args_wlan0="wlanmode hostap"
ifconfig_wlan0="inet 192.168.0.1 netmask 255.255.255.0 ssid freebsdap mode 11g channel 1"
```

### 32.3.5.2. 不使用 WPA 或加密的（无 WPA）无线接入点

尽管我不推荐运行一个不使用任何 WPA 或加密的 AP，但它是一个非常简单的无线 AP 是否正常工作的方法。无线配置对于客户端也非常重要。

一旦 AP 被配置成了我前面所展示的那样，就可以在另外一台无线机器上初始化一次扫描来得到这个 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
```

在客户端机上能看到已连接上了（无 WPA）无线接入点：

```
# 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 0xffffffff broadcast 192.168.0.255
media: IEEE 802.11 Wireless Ethernet OFDM/54Mbps mode 11g
status: associated
ssid freebsdap channel 1 (2412 Mhz 11g) bssid 00:11:95:c3:0d:ac
country US ecm authmode OPEN privacy OFF txpower 21.5 bmiss 7
scanvalid 60 bgscan bgscanintvl 300 bgscanidle 250 roam:rssi 7
roam:rate 5 protmode CTS wme burst
```

### 32.3.5.3. 使用 WPA 的（无 WPA）无线接入点

一段将注重介绍在 FreeBSD（无 WPA）无线接入点上配置使用 WPA 安全。更多有关 WPA 和配置基于 WPA 无客户端的无线参考 [WPA](#)。

hostapd 守护进程将被用于管理与客户端的无线和在无线 WPA（无 WPA）无线接入点上的密钥管理。

接下来，所有的配置操作都将在作为 AP 的 FreeBSD 机器上完成。一旦 AP 能正常工作了，便把如下行加入 /etc/rc.conf 使得 hostapd 能在机器启动的时候自行运行：

```
hostapd_enable="YES"
```

在配置 hostapd 以前，确保已完成了基本配置中所介绍的 [基本配置](#)。

### 32.3.5.3.1. WPA-PSK

WPA-PSK 旨在没有服务器的无服务器的小型网络而设计的。

配置文件 `/etc/hostapd.conf` file :

```
interface=wlan0 ①
debug=1 ②
ctrl_interface=/var/run/hostapd ③
ctrl_interface_group=wheel ④
ssid=freebsdap ⑤
wpa=1 ⑥
wpa_passphrase=freebsdmail ⑦
wpa_key_mgmt=WPA-PSK ⑧
wpa_pairwise=CCMP TKIP ⑨
```

- ① 指定了无线接入点所使用的无线接口。
- ② 指定了运行 hostapd 时显示相关信息的程度。1 表示最小的。
- ③ `ctrl_interface` 指定了 hostapd 存储与其他外部程序（比如 `hostapd_cli(8)`）通信的域套接口文件路径。这里使用了默认。
- ④ `ctrl_interface_group` 指定了允许控制界面文件的属性（这里我使用了 `wheel`）。
- ⑤ 指定了无线网络的名称。
- ⑥ `wpa` 表示用了 WPA 而且指明要使用何种 WPA 模式。1 表示 AP 将使用 WPA-PSK。
- ⑦ `wpa_passphrase` 包含用于 WPA 模式的 ASCII 密码。
- ⑧ `wpa_key_mgmt` 行表明了我所使用的密钥管理。在这个例子中是 WPA-PSK。
- ⑨ `wpa_pairwise` 表示（无线）无线接入点所接受的加密算法。在这个例子中，TKIP(WPA) 和 CCMP(WPA2) 密码都会被接受。CCMP 密码是除 TKIP 外的唯一选项，CCMP 一般作为首选密码；只有在 CCMP 不能被使用的环境中才用 TKIP。

接下来的一步就是运行 hostapd :

```
# /etc/rc.d/hostapd forrestart
```

```
# ifconfig wlan0
wlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 2290
  inet 192.168.0.1 netmask 0xfffff000 broadcast 192.168.0.255
  inet6 fe80::211:95ff:fec3:dac%ath0 prefixlen 64 scopeid 0x4
  ether 00:11:95:c3:0d:ac
  media: IEEE 802.11 Wireless Ethernet autoselect mode 11g <hostap>
  status: associated
  ssid freebsdap channel 1 bssid 00:11:95:c3:0d:ac
  authmode WPA2/802.11i privacy MIXED deftxkey 2 TKIP 2:128-bit txpowmax 36
  protmode CTS dtimperiod 1 bintval 100
```

在客户端能接上行的（无）接入点了，更多可以参 [WPA](#)。看有些客接上了 AP 可以行命令 `ifconfig wlan0 list sta`。

#### 32.3.5.4. 使用 WEP 的（无）接入点

我不推荐使用 WEP 来置一个（无）接入点，因没有的机制并容易被破解。一些史留下的无网支持 WEP 作安全，些网允搭建不含或 WEP 的 AP。

在置了正的 SSID 和 IP 地址后，无就可以入 hostap 模式了：

```
# ifconfig wlan0 create wlandev ath0 wlanmode hostap
# ifconfig wlan0 inet 192.168.0.1 netmask 255.255.255.0 \
    ssid freebsdap wepmode on weptxkey 3 wepkey 3:0x3456789012 mode 11g
```

- `wepkey` 表示中使用一个 WEP 密。个例子中用了第3把密（注意密的号从 1 始）。个参数必置以用来加密数据。
- `wepkey` 表示置所使用的 WEP 密。它符合 `index:key` 的格式。如果没有指定 index，那默 1。就是如果我使用了除第一把以外的密，那就需要指定 index。

再使用一次 `ifconfig` 命令看 wlan0 接口的状：

```
# 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 0xfffff00 broadcast 192.168.0.255
    media: IEEE 802.11 Wireless Ethernet autoselect mode 11g <hostap>
    status: running
    ssid freebsdap channel 4 (2427 Mhz 11g) bssid 00:11:95:c3:0d:ac
    country US ecm authmode OPEN privacy ON deftxkey 3 wepkey 3:40-bit
    txpower 21.5 scanvalid 60 protmode CTS wme burst dtimperiod 1 -dfs
```

在可以从外一台无机器上初始化一次描来到个 AP 了：

```
# ifconfig wlan0 create wlandev ath0
# ifconfig wlan0 up scan
```

| SSID      | BSSID             | CHAN | RATE | S:N  | INT | CAPS |
|-----------|-------------------|------|------|------|-----|------|
| freebsdap | 00:11:95:c3:0d:ac | 1    | 54M  | 22:1 | 100 | EPS  |

在客机能使用正的参数（密等）到并上（无）接入点了，更多参 [WEP](#)。

#### 32.3.6. 同使用有和无接

一般而言，有网的速度更快而且更可，而无网提供更好的活及机性，使用本的用，往往会希望合者的点，并能在必接之无切。

在 FreeBSD 上可以将多个网接口合并到一起，并以“故障移”的方式自切，也就是，一网信口有一定的先序，而操作系在路状生化自自行切，例如当同存在有和无接的时候

先使用有线网，而当有线网断开，则自动切换到无线网。

我将在后面的 [路由聚合与故障转移](#) 中介入路由聚合和故障转移，并在 [有线网和无线网接口间的自动切换](#) 中配置方式行示例。

### 32.3.7. 故障排除

如果在无线网中遇到了麻烦，此章提供了一系列用以帮助排除故障的步骤。

- 如果在列表中找不到无线点，可能没有将无线配置使用有限的一段。
- 如果无法连接到无线点，无线点的通信站配置与无线点的配置一致。包括模式以及安全。尽可能简化配置。如果正使用类似 WPA 或 WEP 的安全，将无线点配置开放和不采用安全措施，并测试是否数据能通信。
- 一旦能连接到无线点之后，就可以使用工具如 [ping\(8\)](#) 来断安全配置了。

`wpa_supplicant` 提供了多种支持；手工运行它，在指定 `-dd`，并察看输出结果。

- 除此之外有多种其它的底层工具。可以使用 `/usr/src/tools/tools/net80211` 中的 `wldebug` 命令来调用 802.11 支持的功能。例如：

```
# wldebug -i ath0 +scan+auth+debug+assoc
net.wlan.0.debug: 0 => 0xc80000<assoc,auth,scan>
```

可以用来调用与扫描无线点和 802.11 在安排通信与握手有关的控制台信息。

有多种有用的信息是由 802.11 提供的；`wlanstats` 工具可以显示些信息。些数据能指出由 802.11 输出的。注意某些可能是由内核在 802.11 之下输出的，因此些可能并不显示。要断与有关的，需要参考内核程序的文档。

如果上述信息没能帮助到具体的问题所在，提交报告，并在其中附上些工具的输出。

## 32.4. 蓝牙

### 32.4.1. 简介

Bluetooth (蓝牙) 是一种无线技术，用于建立 2.4GHz，波长 10 米的私有网。网一般是由便携式设备，比如手机 (cellular phone)，掌上设备 (handhelds) 和膝上设备 (laptops) 以 ad-hoc 形式组成。不像其它流行的无线技术——Wi-Fi，Bluetooth 提供了更高的服务面，像 FTP 的文件服务、文件推送 (file pushing)、语音送、串行模式等等。

在 FreeBSD 里，蓝牙 (Bluetooth stack) 通常使用 Netgraph 框架 (看 [netgraph\(4\)](#)) 来的。大量的“Bluetooth USB dongle”由 [ng\\_ubt\(4\)](#) 程序支持。基于 Broadcom BCM2033 芯片的 Bluetooth 可以通过 [ubtbcmfw\(4\)](#) 和 [ng\\_ubt\(4\)](#) 程序支持。3Com Bluetooth PC 卡 3CRWB60-A 由 [ng\\_bt3c\(4\)](#) 程序支持。基于 Serial 和 UART 的蓝牙由 [sio\(4\)](#)、[ng\\_h4\(4\)](#) 和 [hcseriald\(8\)](#)。本章介绍 USB Bluetooth dongle 的使用。



### 32.4.2. 加载

默认的 Bluetooth 模块程序已存在于内核模块里。接入前，需要将模块程序加载入内核：

```
# kldload ng_ubt
```

如果系统 Bluetooth 模块已存在于系统里，那从 /boot/loader.conf 里加一个模块：

```
ng_ubt_load="YES"
```

插入 USB dongle。控制台(console)(或syslog中)会输出类似如下的信息：

```
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
```

脚本 /etc/rc.d/bluetooth 是用来启动和停止 Bluetooth stack (蓝牙)的。最好在拔出前停止 stack(stack)，当然也不是非做不可。蓝牙 stack ( )，会得到如下的输出：

```
# /etc/rc.d/bluetooth start ubt0
BD_ADDR: 00:02:72:00:d4:1a
Features: 0xff 0xff 0xf 00 00 00 00 00
<3-Slot> <5-Slot> <Encryption> <Slot offset>
<Timing accuracy> <Switch> <Hold mode> <Sniff mode>
<Park mode> <RSSI> <Channel quality> <SCO link>
<HV2 packets> <HV3 packets> <u-law log> <A-law log> <CVSD>
<Paging scheme> <Power control> <Transparent SCO data>
Max. ACL packet size: 192 bytes
Number of ACL packets: 8
Max. SCO packet size: 64 bytes
Number of SCO packets: 8
```

### 32.4.3. 主控制器接口 (HCI)

主控制器接口 (HCI) 提供了通向基带控制器和接口管理器的命令接口及硬件状态字和控制寄存器的通道。该接口提供了蓝牙基带 (Bluetooth baseband) 功能的一种方式。主机上的 HCI 与蓝牙硬件上的 HCI 固件交换数据和命令。主控制器的固件 (如物理层) 驱动程序提供一个 HCI 交换信息的能力。

每个蓝牙 (Bluetooth) 设备建立一个 hci 类型的 Netgraph 点。HCI 点一般连接蓝牙设备的点 (下行流) 和 L2CAP 点 (上行流)。所有的 HCI 操作必须在 HCI 点上执行而不是设备点。HCI 点的默认名是 "devicehci"。更多参考 [ng\\_hci\(4\)](#) 的司机手册。

最常见的任务是 RF proximity 中的蓝牙 (Bluetooth) 设备。这个就叫做 探测 (inquiry)。探测及 HCI 相关的操作可以由 [hccontrol\(8\)](#) 工具来完成。以下的例子展示如何输出设备内的蓝牙设备。在几秒内可以得到一个设备列表。注意程序主机只有被置于 discoverable(可发现) 模式才能回答。

```
% 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** 是蓝牙的特定地址，类似于网络的 MAC 地址。需要用此地址与某个蓝牙设备通信。可以分配 BD\_ADDR 分配由人可读的名字 (human readable name)。文件 `/etc/bluetooth/hosts` 包含已知蓝牙主机的信息。下面的例子展示如何为设备分配程序的可读名。

```
% hccontrol -n ubt0hci remote_name_request 00:80:37:29:19:a4
BD_ADDR: 00:80:37:29:19:a4
Name: Pav's T39
```

如果在程序蓝牙上运行，设备会认为的计算机是 "your.host.name (ubt0)"。分配本地设备的名字可随更改。

蓝牙提供点对点连接（只有两个蓝牙设备参与）和点对多点连接。在点对多点连接中，连接由多个蓝牙设备共享。以下的例子展示如何取得本地设备的活基带 (baseband) 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
```

`connection handle` (连接柄) 在需要终止基带连接时有用。注意：一般不需要手动完成。栈 (stack) 会自行终止不活动的基带连接。

```
# hccontrol -n ubt0hci disconnect 41
Connection handle: 41
Reason: Connection terminated by local host [0x16]
```

参考 `hccontrol help` 获取完整的 HCI 命令列表。大部分 HCI 命令不需要超出使用限制。

#### 32.4.4. 连接控制和配置 (L2CAP)

连接控制和配置 (L2CAP) 层上提供面向连接和无连接的数据服务，并提供多种功能和分割重用操作。L2CAP 充层上服务和用户程序和接收最大宽度 64K 的 L2CAP 数据包。

L2CAP 基于通道 (channel) 的概念。通道 (Channel) 是位于基带 (baseband) 连接之上的连接。一个通道以多一的方式定义一个单一协议 (single protocol)。多个通道可以定义同一个设备，但一个通道不可以定义多个设备。一个在通道里接收到的 L2CAP 数据包被送到相应的上层。多个通道可共享同一个基带连接。

一个蓝牙 (Bluetooth) 可以建立一个 *l2cap* 型的 Netgraph 点。L2CAP 点一般连接 HCI 点(下行流)和蓝牙的 点(上行流)。L2CAP 点的默认名是 "device12cap"。更多参考 [ng\\_l2cap\(4\)](#) 的司机手册。

一个有用的命令是 [l2ping\(8\)](#)，它可以用来 ping 其它点。一些蓝牙可能不会返回所有发送它的数据，所以下例中的 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\(8\)](#) 工具用于在 L2CAP 上行多操作。以下例子展示如何取得本地点的连接 (通道) 和基本连接的列表：

```
% 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 0           0 OPEN
```

一个断工具是 [btsockstat\(1\)](#)。它完成与 [netstat\(1\)](#) 类似的操作，只是用了蓝牙网相关的数据。以下例子显示与 [l2control\(8\)](#) 相同的连接。

```
% btsockstat
Active L2CAP sockets
PCB      Recv-Q Send-Q Local address/PSM      Foreign address  CID   State
c2afe900  0        0 00:02:72:00:d4:1a/3    00:07:e0:00:0b:ca 66    OPEN
Active RFCOMM sessions
L2PCB    PCB      Flag MTU   Out-Q DLCs State
c2afe900 c2b53380 1    127    0    Yes  OPEN
Active RFCOMM sockets
PCB      Recv-Q Send-Q Local address      Foreign address  Chan DLCI State
c2e8bc80  0      250 00:02:72:00:d4:1a 00:07:e0:00:0b:ca 3     6    OPEN
```

### 32.4.5. RFCOMM

RFCOMM 提供基于 L2CAP 的串行端口模式。它基于 ETSI TS 07.10 标准。RFCOMM 是一个的，附加了模式 9 的 RS-232(EIA/TIA-232-E) 串行端口的定义。RFCOMM 最多支持 60 个并连接 (RFCOMM通道)。

除了 RFCOMM，运行于不同上的应用程序建立起一条于它之通信段的通信路径。RFCOMM上用于使用串行端口的用户程序。通信段是一个到一个的蓝牙连接 (直接连接)。

RFCOMM 关心的只是直接接口之间的连接，或在网里一个与 modem 之间的连接。RFCOMM 能支持其它的配置，比如在一端通过蓝牙技术通而在另一端使用有接口。

在FreeBSD, RFCOMM 在蓝牙套接字 (Bluetooth sockets layer) 。

### 32.4.6. 配对(Pairing of Devices)

默认情况下，蓝牙通信是不需要配对的，任何设备可与其它任何设备。一个设备（比如手机）可以提供某种特殊服务（比如号码服务）。设备一般使用 PIN(PIN codes)。一个 PIN 是最多 16 个字符的 ASCII 字符串。用设备需要在设备中输入相同的PIN。用设备输入了 PIN 后，设备会生成一个连接密钥(link key)。接着连接密钥可以存储在设备或存储器中。设备会使用先前生成的连接密钥。以上介绍的程序被称为配对(pairing)。注意如果任何一方丢失了连接密钥，必须重新配对。

守护进程 [hcsecd\(8\)](#) 管理所有蓝牙请求。默认的配置文件是 /etc/bluetooth/hcsecd.conf。下面的例子显示一个手机的 PIN 被设置为"1234"：

```
device {
    bdaddr 00:80:37:29:19:a4;
    name    "Pav's T39";
    key     nokey;
    pin     "1234";
}
```

PIN 没有限制(除了长度)。有些设备(例如蓝牙耳机)会有一个设置的 PIN 。-d 选项控制 [hcsecd\(8\)](#) 守护进程于前台，因此很容易看清发生了什么。设置设备准备接收配对(pairing)，然后设备连接到设备。设备返回接收了设备并请求PIN。设备与 hcsecd.conf 中一设备的 PIN 。设备在个人计算机已与设备连接了。设备也可以在设备上初始点。

可以通过在 /etc/rc.conf 文件中添加下面的行，以便 hcsecd 在系统启动时运行：

```
hcsecd_enable="YES"
```

以下是设备的 hcsecd 服务输出本：

```
hcsecd[16484]: Got Link_Key_Request event from 'ubt0hci', remote bdaddr
0:80:37:29:19:a4
hcsecd[16484]: Found matching entry, remote bdaddr 0:80:37:29:19:a4, name 'Pav's T39',
link key 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
```

### 32.4.7. 服务发现 (SDP)

服务发现 (SDP) 提供了一种客户端—服务器方法，它能够通过服务器提供的服务及属性。服务的属性包括所提供服务的类型或名称，使用服务所需要的机制或协议。

SDP 包括 SDP 服务器和 SDP 客户端之间的通信。服务器维护一个服务列表，它介绍服务器上服务的特性。每个服务包含关于该服务的信息。通过发出 SDP 请求，客户端会得到服务列表的信息。如果客户端（或者客户端上的应用程序）决定使用一个服务，它使用一个服务时必须与服务提供者都建立一个独立的连接。SDP 提供了服务及其属性的机制，但它并不提供使用某些服务的机制。

一般地，SDP 客户端按照服务的某些期望特征来搜索服务。但是，即使没有任何关于由 SDP 服务器提供的服务的信息，有时也能令人满意地找到它的服务列表里所描述的是哪些服务类型。服务提供者提供的名称称为 *浏览* (browsing)。

在 FreeBSD 中，SDP 服务器 `sdpd(8)` 和命令行客户端 `sdpcontrol(8)` 都包括在了标准的 FreeBSD 安装里。下面的例子展示如何进行 SDP 浏览。

```
% 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
```

等等。注意一个服务有一个属性（比如 RFCOMM 通道）列表。根据服务可能需要一些属性做个注释。有些“蓝牙实现” (Bluetooth implementation) 不支持服务发现，可能会返回一个空列表。在这种情况下，可以搜索指定的服务。下面的例子展示如何搜索 OBEX Object Push (OPUSH) 服务：

```
% sdpcontrol -a 00:01:03:fc:6e:ec search OPUSH
```

要在 FreeBSD 里让蓝牙客户端提供服务，可以使用 `sdpd(8)` 服务。可以通过在 `/etc/rc.conf` 中加入下面的行：

```
sdpd_enable="YES"
```

然后用下面的命令来启动 `sdpd` 服务：

```
# /etc/rc.d/sdpd start
```

需要客户端提供蓝牙服务的本地的服务程序会使用本地 `SDP` 进程注册服务。像这样的程序就有 [rfcomm\\_pppd\(8\)](#)。一旦启动它，就会使用本地 `SDP` 进程注册蓝牙 LAN 服务。

使用本地 `SDP` 进程注册的服务列表，可以通过本地控制通道输出 `SDP` 信息得到：

```
# sdpcontrol -l browse
```

### 32.4.8. 号码网 (DUN) 和使用 PPP(LAN) 的网接入

号码网 (DUN) 配置通常与 modem 和手机一起使用。如下是一配置所涉及的内容：

- 计算机使用手机或 modem 作为无 modem 来连接号码因特网接入设备，或者使用其它的号码网；
- 计算机使用手机或 modem 接收数据请求。

使用 PPP(LAN) 的网接入常使用在如下情形：

- 一个蓝牙的局域网接入；
- 多个蓝牙的局域网接入；
- PC 到 PC (使用基于串行模式的 PPP 网)。

在 FreeBSD 中，一个方面使用 [ppp\(8\)](#) 和 [rfcomm\\_pppd\(8\)](#) (一个封装器，可以将 RFCOMM 蓝牙接口 PPP 可操作的设备) 来连接。在使用任何方面之前，一个新的 PPP 配置必须在 `/etc/ppp/ppp.conf` 中建立。想要示例参考 [rfcomm\\_pppd\(8\)](#)。

在下面的例子中，[rfcomm\\_pppd\(8\)](#) 用来在 NUN RFCOMM 通道上打开一个到 BD\_ADDR 为 00:80:37:29:19:a4 的设备 RFCOMM 连接。具体的 RFCOMM 通道号要通过 `SDP` 从设备得到。也可以手动指定通 RFCOMM，这种情况下 [rfcomm\\_pppd\(8\)](#) 将不能进行 `SDP` 操作。使用 [sdpcontrol\(8\)](#) 来从设备上的 RFCOMM 通道。

```
# rfcomm_pppd -a 00:80:37:29:19:a4 -c -C dun -l rfcomm-dialup
```

为了提供 PPP(LAN) 网接入服务，必须进行 [sdpd\(8\)](#) 服务。一个新的 LAN 客户端条目必须在 `/etc/ppp/ppp.conf` 文件中建立。想要示例参考 [rfcomm\\_pppd\(8\)](#)。最后，在有效地通道号上启动 RFCOMM PPP 服务。RFCOMM PPP 服务会使用本地 `SDP` 进程自行注册蓝牙 LAN 服务。下面的例子展示如何启动 RFCOMM PPP 服务。

```
# rfcomm_pppd -s -C 7 -l rfcomm-server
```

### 32.4.9. OBEX 对象推送 (OBEX Object Push - OPUSH) 界面

OBEX 被广泛地用于移动设备之间的文件传输。它的主要用途是在无线通信域，被用于笔记本或手持设备之间的一般文件传输。

OBEX 服务器和客户端由第三方软件包 `obexapp`，它可以从 [comms/obexapp](https://comms.obexapp.org) port 安装。

OBEX 客户端用于向 OBEX 服务器推入或接出对象。一个对象可以是(举个例子)商业卡片或约会。OBEX 客户端能通过 SDP 从远程取得 RFCOMM 通道号。可以通过指定服务器名代替 RFCOMM 通道号来完成。支持的服务器名是：IrMC、FTRN 和 OPUSH。也可以用数字来指定 RFCOMM 通道号。下面是一个 OBEX 会话的例子，一个信息对象从手机中被拉出，一个新的对象被推入手机的目录。

```
% 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 OBEX 推入服务，`sdpd(8)` 必须处于运行状态。必须建立一个根目录用于存放所有接入的对象。根文件的默认路径是 `/var/spool/obex`。最后，在有效的 RFCOMM 通道号上启动 OBEX 服务。OBEX 服务会使用 SDP 程序自己注册 OBEX 对象推送 (OBEX Object Push) 服务。下面的例子展示如何启动 OBEX 服务。

```
# obexapp -s -C 10
```

### 32.4.10. 串口(SP)界面

串口(SP)界面允许通过完成 RS232 (或类似) 串口的仿真。这个界面所涉及到的情形是，通过虚拟串口使用字节代替原来处理以前的程序。

工具 `rfcomm_sppd(1)` 来仿真串口。"Pseudo tty" 用来作虚拟的串口。下面的例子展示如何连接程序的串口服务。注意不必指定 RFCOMM 通道——`rfcomm_sppd(1)` 能通过 SDP 从远端那里得到。如果想代替它的，可以在命令行里指定 RFCOMM 通道来：

```
# rfcomm_sppd -a 00:07:E0:00:0B:CA -t /dev/tty6
rfcomm_sppd[94692]: Starting on /dev/tty6...
```

一旦连接上，"pseudo tty"就可以充当串口了：

```
# cu -l tty6
```

### 32.4.11. 问题解答

### 32.4.11.1. 不能直接端

一些老的蓝牙并不支持角色 (role switching)。默认情况下, FreeBSD 接受一个新的直接, 它会进行角色并成主控端 (master)。不支持角色的将无法直接。注意角色是在新直接建立行的, 因此如果程序不支持角色, 就不可能向它出求。一个 HCI 用来在本地端禁用角色。

```
# hccontrol -n ubt0hci write_node_role_switch 0
```

### 32.4.11.2. 如果有, 能否知道到底正在生什?

可以。需要借助第三方软件包 hcidump, 它可以通过 [comms/hcidump](#) port 来安装。hcidump 工具和 [tcpdump\(1\)](#) 非常相像。它可以用来示蓝牙数据包的内容, 并将其到文件中。

## 32.5. 接

### 32.5.1. 介

有, 会有需要将一个物理网分成个独立的网段, 而不是建新的 IP 子网, 并将其通路由器相。以这种方式接个网的称 "网 (bridge)"。有个网接口的 FreeBSD 系可以作网来使用。

网通学个网接口上的 MAC 地址 (以太网地址) 工作。只当数据包的源地址和目地址于不同的网, 网才行。

在很多方面, 网就像一个有很少端口的以太网交换机。

### 32.5.2. 合接的情况

合使用网的, 有多不同的情况。

#### 32.5.2.1. 使多个网相互通

网的基本操作是将个或多个网段接在一起。由于各式各的原因, 人会希望使用一台真正的计算机, 而不是网来充任网的角色, 常的原因包括的限制、需要进行防火, 或虚拟机网接口接虚网。网也可以将无网以 hostap 模式接入有网。

#### 32.5.2.2. 网/数据整形防火

使用防火的常情形是无需行路由或网地址的情况 (NAT)。

例来, 一家通 DSL 或 ISDN 接到 ISP 的小公司, 有 13 个 ISP 分配的全局 IP 地址和 10 台 PC。在这种情况下, 由于分子网的, 采用路由来防火会比困。

基于网的防火可以串接在 DSL/ISDN 路由器的后面, 而无需考 IP 制的。

#### 32.5.2.3. 网

网可以用于接个不同的网段, 并用于往返的以太网。可以通过在网接口上使用 [bpf\(4\)/tcpdump\(1\)](#), 或通将全部以太网制到一个网接口 (span 口) 来。



#### 32.5.2.4. 2 VPN

通过 IP 连接的网，可以利用 EtherIP 隧道或基于 [tap\(4\)](#) 的解决方案，如 OpenVPN 可以将两个以太网连接到一起。

#### 32.5.2.5. 2 冗余

网可以通过多条路连接在一起，并使用生成树协议 (Spanning Tree Protocol) 来阻止多余的通路。使以太网能正常工作，每个网之间只有一条激活通路，而生成树能阻塞路，并将多余的路置为阻断状态。当激活通路断，网能计算外一，并重新激活阻断的通路，以恢复到网各点的连通性。

### 32.5.3. 内核配置

本节主要介绍 [if\\_bridge\(4\)](#) 网。除此之外，还有一个基于 netgraph 的网，如欲了解一，参看手册 [ng\\_bridge\(4\)](#)。

网是一个内核模块，并会随使用 [ifconfig\(8\)](#) 创建网接口而加载。也可以将 `device if_bridge` 加入到内核配置文件中，以便将其静置内核。

包可以通过使用了 [pf\(9\)](#) 框架的任意一防火墙包来完成。些防火墙可以以模块形式加载，也可以静置内核。

通过配合 [altq\(4\)](#) 和 [dummynet\(4\)](#)，网也可以用于流量控制。

#### 32.5.4. 用网

网是通过接口制来建的。可以使用 [ifconfig\(8\)](#) 来建网接口，如果内核不包括网，它会自行将其入。

```
# ifconfig bridge create
bridge0
# ifconfig bridge0
bridge0: flags=8802<BROADCAST,SIMPLEX,MULTICAST> metric 0 mtu 1500
        ether 96:3d:4b:f1:79:7a
        id 00:00:00:00:00:00 priority 32768 hellotime 2 fwddelay 15
        maxage 20 holdcnt 6 proto rstp maxaddr 100 timeout 1200
        root id 00:00:00:00:00:00 priority 0 ifcost 0 port 0
```

如此就建立了一个网接口，并为其随机分配了以太网地址。 `maxaddr` 和 `timeout` 参数能控制网在表中保存多少个 MAC 地址，以及表中主机的期。其他参数控制生成网的方式。

将成网接口加入网。了网能所有网成接口包，网接口和所有成接口都需要于用状态：

```
# ifconfig bridge0 addm fxp0 addm fxp1 up
# ifconfig fxp0 up
# ifconfig fxp1 up
```

网会在 `fxp0` 和 `fxp1` 之间以太网。等效的 `/etc/rc.conf` 配置如下，如此配置将在系建同的网

□。

```
cloned_interfaces="bridge0"
ifconfig_bridge0="addm fxp0 addm fxp1 up"
ifconfig_fxp0="up"
ifconfig_fxp1="up"
```

如果网□主机需要 IP 地址，□□将其□在网□□□本身，而不是某个成□□□上。□□可以通□静□□置或 DHCP 来完成：

```
# ifconfig bridge0 inet 192.168.0.1/24
```

除此之外，也可以□网□接口指定 IPv6 地址。

### 32.5.5. 防火□

当□用包□□□，通□网□的包可以分□在□入的网□接口、网□接口和□出的网□接口上□行□□。□些□段均可禁用。当包的流向很重要□，最好在成□接口而非网□接口上配置防火□。

网□上可以□行□多配置以决定非 IP 及 ARP 包能否通□，以及通□ IPFW □□二□防火□。□参□ [if\\_bridge\(4\)](#) □机手册以了解□一□的□□。

### 32.5.6. 生成□

网□□□□□了快速生成□□ (RSTP 或 802.1w)，并与□早的生成□□ (STP) 兼容。生成□□可以用来在网□拓□中□□并消除□路。RSTP 提供了比□□ STP 更快的生成□覆□速度，□□□□会在相□的交□机之□交□信息，以迅速□入□□状□，并避免□生□路。FreeBSD 支持以 RSTP 和 STP 模式□行，而 RSTP 是默□模式。

使用 **stp** 命令可以在成□接口上□用生成□。□包含 fxp0 和 fxp1 的网□，可以用下列命令□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
```

网□的生成□ ID □ 00:01:02:4b:d4:50 而□先□□ 32768。其中 **root id** 与生成□□相同，表示□是作□生成□根的网□。

□一个网□也□用了生成□：

```
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
```

里的 `root id 00:01:02:4b:d4:50 priority 32768 ifcost 400000 port 4` 表示根网口是前面的 `00:01:02:4b:d4:50`，而从此网口出发的通路代价 `400000`，此通路到根网口是通 `port 4` 即 `fxp0` 口的。

## 32.5.7. 网口的高口用法

### 32.5.7.1. 重建流量流

网口支持 `ifconfig` 模式，在 `bpf(4)` 口理之后会将包口，而不是口口理或口口。口可以用于将口个或多个接口上的口入口化口一个 `bpf(4)` 流。在将口个独立的接口上的口的 RX/TX 信号重整口一个口，口会非常有用。

如果希望将四个网口接口上的口入口成一个流：

```
# ifconfig bridge0 addm fxp0 addm fxp1 addm fxp2 addm fxp3 monitor up
# tcpdump -i bridge0
```

### 32.5.7.2. 像口 (Span port)

网口收到的口个以太网口都可以口到像口上。网口上的像口数量没有限制，如果一个接口已口被配置口像口，口它就不能再作网口的成口口来使用。口用法主要是口与网口像口相口的口听机配合使用。

如果希望将所有口到名 `fxp4` 的接口上：

```
# ifconfig bridge0 span fxp4
```

### 32.5.7.3. 口用接口 (Private interface)

口用接口不会口流量到除口用接口之外的其他端口。口些流量会无条件地阻断，因此包括 ARP 在内的以太网口均不会被口。如果需要口性地阻断流量，口使用防火口。

### 32.5.7.4. 自学口接口 (Sticky Interfaces)

如果网口的成口接口口口自学口，口口口口的地址口一旦口入口快取口存，即被口口是静口口。自学口口不会从快取口存中口期或替口掉，即使地址在口一接口上出口也是如此。口使得不必事先口布口表，口也能根据口口果得到静口的有点，但在口些网段被网口看到的客口机，就不能漫游至口一网段了。

一种用法是将网桥与 VLAN 功能结合，每个客户网桥会被隔离在一个，而不会浪费 IP 地址空间。考虑 CustomerA 在 `vlan100` 上，而 CustomerB 在 `vlan101` 上。网桥地址 `192.168.0.1`，同时作为 internet 路由器使用。

```
# ifconfig bridge0 addm vlan100 sticky vlan100 addm vlan101 sticky vlan101
# ifconfig bridge0 inet 192.168.0.1/24
```

两台主机均将 `192.168.0.1` 作为默认网桥，由于网桥快速存取是自学式的，因而它无法构造 MAC 地址来截取其他客户的网桥流量。

在 VLAN 之间的通信可以通过禁用接口 (或防火墙) 来阻断：

```
# ifconfig bridge0 private vlan100 private vlan101
```

这样些客户机就完全相互隔离了。可以使用整个的 `/24` 地址空间，而无需子网。

### 32.5.7.5. 地址限制

接口后的源 MAC 地址数量是可以控制的。一旦达到了限制未知源地址的包将会被丢弃，直至有缓存中的一期或被移除。

下面的例子是设置 CustomerA 在 `vlan100` 上可接收的以太网帧最大为 10。

```
# ifconfig bridge0 ifmaxaddr vlan100 10
```

### 32.5.7.6. SNMP 管理

网桥接口和 STP 参数能够由 FreeBSD 基本系统的 SNMP 守护进程管理。输出的网桥 MIB 符合 IETF 标准，所以任何 SNMP 客户端或管理包都可以被用来接收数据。

在网桥机器上从 `/etc/snmp.config` 文件中去掉以下行的注释 `begemotSnmppModulePath."bridge" = "/usr/lib/snmp_bridge.so"` 并启动 `bsnmppd` 守护进程。其他的配置如 `community names` 和 `access lists` 可能也需要修改。参看 [bsnmppd\(1\)](#) 和 [snmp\\_bridge\(3\)](#) 获取更多信息。

以下的例子中使用了 Net-SNMP 套件 ([net-mgmt/net-snmp](#)) 来启动一个网桥，当然同时也能使用 `port net-mgmt/bsnmptools`。在 SNMP 客户端 Net-SNMP 的配置文件 `$HOME/.snmp/snmp.conf` 中加入以下几行来加入网桥的 MIB 定义：

```
mibdirs +/usr/shared/snmp/mibs
mibs +BRIDGE-MIB:RSTP-MIB:BEGEMOT-MIB:BEGEMOT-BRIDGE-MIB
```

通过 IETF BRIDGE-MIB(RFC4188) 启动一个独立的网桥

```
% 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` 的 2 意味着 STP 网拓改了 2 次，拓改表示 1 个或多个网中的接口或失效并且有一个新口生成。`dot1dStpTimeSinceTopologyChange.0` 的 0:31:39.59 表示是何时改的。

多个网接口可以使用 private 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
```

通过 `mib-2.dot1dBridge` 子树正在被哪个网接口：

```
% snmpset -v 2c -c private bridge1.example.com
BEGEMOT-BRIDGE-MIB::begemotBridgeDefaultBridgeIf.0 s bridge2
```

## 32.6. 链路聚合与故障转移

### 32.6.1. 简介

使用 `lagg(4)` 接口，能够将多个网口接口聚合为一个虚接口，以提供容量和高速连接的能力。

### 32.6.2. 运行模式

#### Failover (故障转移)

只通过主网口收发数据。如果主网口不可用，则使用下一个激活的网口。在配置里加入的第一个网口便会被设为主网口；此后加入的其他网口，则会被设成故障转移的备用网口。如果主网口发生故障转移之后，原先的网口又恢复了可用状态，则它仍会作为主网口使用。

#### Cisco® Fast EtherChannel®

Cisco® Fast EtherChannel® (FEC) 是一种静态配置，并不依赖于点-to-point 或交叉以太网来控制链路情况。如果交换机支持 LACP，则使用后者而非静态配置。

FEC 将输出流量在激活的网口之间以散列信息为依据分拆，并接收来自任意激活网口的入流量。散列信息包含以太网源地址、目的地址，以及（如果有的话）VLAN tag 和 IPv4/IPv6 源地址及目的地址信息。

#### LACP

支持 IEEE® 802.3ad 链路聚合控制协议 (LACP) 和静态配置。LACP 能够在一点与若干链路聚合之点之间协商链路。一个链路聚合 (LAG) 由一组相同速度、以全双工模式运行的网口组成。流量在 LAG 中的网口之间，会以速度最大的原网口进行分拆。当物理链路发生变化时，链路聚合会迅速重新形成新的配置。

LACP 也是将输出流量在激活的网口之间以散列信息为依据分拆，并接收来自任意激活网口的入流量。散列信息包含以太网源地址、目的地址，以及（如果有的话）VLAN tag 和 IPv4/IPv6 源地址及目的地址信息。

#### Loadbalance (负载均衡)

它是 FEC 模式的别名。

#### Round-robin (轮询)

将输出流量以轮询方式在所有激活端口之间度，并从任意激活端口接收入流量。轮询模式反了以太网排序原则，因此要小心使用。

### 32.6.3. 例子

在这个例子中，我们将 FreeBSD 的两个网口作为一个负载均衡和故障转移链路聚合接口接到交换机上。在此基础上，可以增加更多的网口，以提高吞吐量和故障容忍能力。由于以太网链路上端点的顺序是制性的，因此两个点之间的连接速度，会取决于一个网口的最大速度。该算法会尽量采用更多的信息，以便将不同的网口流量分配到不同的网口接口上，并平衡不同网口的负载。

在 Cisco® 交换机上将 *FastEthernet0/1* 和 *FastEthernet0/2* 两个网口添加到 *channel-group 1*：

```
interface FastEthernet0/1
  channel-group 1 mode active
  channel-protocol lacp
!
interface FastEthernet0/2
  channel-group 1 mode active
  channel-protocol lacp
```

使用 *fxp0* 和 *fxp1* 创建 *lagg(4)* 接口，并用该接口并配置 IP 地址 *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
```

用下面的命令查看接口状态：

```
# ifconfig lagg0
```

其中 *ACTIVE* 的接口是激活聚合的部分，表示它已完成与对端交换机的协商，同时，流量将通过这些接口来收发。在 *ifconfig(8)* 的输出中会输出 LAG 的信息。

```
lagg0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
  options=8<VLAN_MTU>
  ether 00:05:5d:71:8d:b8
  media: Ethernet autoselect
  status: active
  laggproto lacp
  laggport: fxp1 flags=1c<ACTIVE,COLLECTING,DISTRIBUTING>
  laggport: fxp0 flags=1c<ACTIVE,COLLECTING,DISTRIBUTING>
```

如果需要查看交换机上的端口状态，可以使用 *show lacp neighbor* 命令：

```
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:

| Port  | Flags | LACP port<br>Priority | Dev ID         | Age | Oper<br>Key | Port<br>Number | Port<br>State |
|-------|-------|-----------------------|----------------|-----|-------------|----------------|---------------|
| Fa0/1 | SA    | 32768                 | 0005.5d71.8db8 | 29s | 0x146       | 0x3            | 0x3D          |
| Fa0/2 | SA    | 32768                 | 0005.5d71.8db8 | 29s | 0x146       | 0x4            | 0x3D          |

如欲查看一口的情况，需要使用 `show lacp neighbor detail` 命令。

如果希望在系重启保持这些配置，可在 `/etc/rc.conf` 中添加如下配置：

```
ifconfig_fxp0="up"
ifconfig_fxp1="up"
cloned_interfaces="lagg0"
ifconfig_lagg0="laggproto lacp laggport fxp0 laggport fxp1 10.0.0.3/24"
```



#### 例 41. 故障切换模式

故障切换模式中，当主链路产生故障，会自切换到备用端口。首先用成接口，接着是配置 `lagg(4)` 接口，其中，使用 `fxp0` 作主接口，`fxp1` 作备用接口，并在整个接口上配置 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
```

创建成功之后，接口状态会是似下面，主要的区别是 MAC 地址和名称：

```
# ifconfig lagg0
lagg0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
    options=8<VLAN_MTU>
    ether 00:05:5d:71:8d:b8
    inet 10.0.0.15 netmask 0xffffffff broadcast 10.0.0.255
    media: Ethernet autoselect
    status: active
    laggproto failover
    laggport: fxp1 flags=0<>
    laggport: fxp0 flags=5<MASTER,ACTIVE>
```

系统将在 `fxp0` 上执行流量的接收。如果 `fxp0` 的接口中断，`fxp1` 会自成为激活接口。如果主端口的接口恢复，它又会成为激活接口。

如果希望在系统重启保持这些设置，在 `/etc/rc.conf` 中添加如下配置：

```
ifconfig_fxp0="up"
ifconfig_fxp1="up"
cloned_interfaces="lagg0"
ifconfig_lagg0="laggproto failover laggport fxp0 laggport fxp1 10.0.0.15/24"
```

#### 例 42. 有网口和无网口接口的自切

由于使用本机的用网来，通常会希望使用无网口接口作备用接口，以便在有网口不可用保持网口接。通过使用 `lagg(4)`，我可以只使用一个 IP 地址的情况下，先使用性能 and 安全性都更好的有网口，同时保持通过无网口接口来数据的能力。

要实现的目的，就需要将用于接无网口的物理接口的 MAC 地址修改与所配置的 `lagg(4)` 一致，后者是从主网口接口，也就是有网口接口，继承而来。

在这个配置中，我将先使用有网口接口 `bge0` 作主网口接口，而将无网口接口 `wlan0` 作备用网口接口。里面的 `wlan0` 使用的物理口是 `iwn0`，我需要将它的 MAC 地址修改与有网口接口一致。到了到这个目的首先要得到有网口接口上的 MAC 地址：

```
# ifconfig bge0
bge0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
    options=19b<RXCSUM,TXCSUM,VLAN_MTU,VLAN_HWTAGGING,VLAN_HWCSUM,TSO4>
    ether 00:21:70:da:ae:37
    inet6 fe80::221:70ff:feda:ae37%bge0 prefixlen 64 scopeid 0x2
    nd6 options=29<PERFORMNUD,IFDISABLED,AUTO_LINKLOCAL>
    media: Ethernet autoselect (1000baseT <full-duplex>)
    status: active
```

可能需要将 `bge0` 改回系上使用的接口，并从输出结果中的 `ether` 行输出有网口的 MAC 地址。接着是修改物理的无网口接口，`iwn0`：

```
# ifconfig iwn0 ether 00:21:70:da:ae:37
```

用无网口接口，但不在其上配置 IP 地址：

```
# ifconfig wlan0 create wlandev iwn0 ssid my_router up
```

用 `bge0` 接口。建 `lagg(4)` 接口，其中 `bge0` 作主网口接口，而以 `wlan0` 作备用接口：

```
# ifconfig bge0 up
# ifconfig lagg0 create
# ifconfig lagg0 up laggproto failover laggport bge0 laggport wlan0
```

新建的接口的状态如下，系上的 MAC 地址和名字等可能会有所不同：

```
# ifconfig lagg0
lagg0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
options=8<VLAN_MTU>
ether 00:21:70:da:ae:37
media: Ethernet autoselect
status: active
laggproto failover
laggport: wlan0 flags=0<>
laggport: bge0 flags=5<MASTER,ACTIVE>
```

接着用 DHCP 客户端来取 IP 地址：

```
# dhclient lagg0
```

如果希望在系重启保持些置，在 `/etc/rc.conf` 中加如下配置：

```
ifconfig_bge0="up"
ifconfig_iwn0="ether 00:21:70:da:ae:37"
wlans_iwn0="wlan0"
ifconfig_wlan0="WPA"
cloned_interfaces="lagg0"
ifconfig_lagg0="laggproto failover laggport bge0 laggport wlan0 DHCP"
```

## 32.7. 无操作

FreeBSD 主机可以从网而无需本地磁就可操作，使用的是从 NFS 服务器装的文件系。除了准的配置文件，无需任何的系修改。很容易置的系因所有必要的元素都很容易得到：

- 至少有可能的方法从网加内核：
  - PXE：Intel® 的先行环境 (Preboot eXecution Environment) 系是一活的引 ROM 模式，个 ROM 内建在一些网或主板的中。看 [pxeboot\(8\)](#) 以取更多。
  - Etherboot port ([net/etherboot](#)) 生通网加内核的可 ROM 代。些代可以入网上的 PROM 上，或从本地 (或硬) 器加，或从行着的 MS-DOS® 系加。它支持多网。
- 一个板脚本 (`/usr/shared/examples/diskless/clone_root`) 化了服务器上工作站根文件系的建和。个脚本需要少量的自定，但能很快的熟悉它。
- `/etc` 存在准的系文件用于和支持无的系。
- 可以向 NFS 文件或本地磁行交(如果需要的)。

置无工作站有多方法。有很多相的元素大部分可以自定以合本地情况。以下将介一个完整系的安装，的是性和与准 FreeBSD 脚本的兼容。介的系有以下特性：

- 无工作站使用一个共享的只 / 文件系和一个共享的只 `/usr`。

root 文件系统是一标准的 FreeBSD 根文件系统（一般是服务器的），只是一些配置文件被特定于无盘操作的配置文件覆。

root 文件系统必须可写的部分被 md(4) 文件系统覆。任何的改写在重启后都会失。

- 内核由 etherboot 或 PXE 送和加，有些情况可能会指定使用其中之一。



如上所述，个系是不安全的。它于网的受保区域并不被其它主机信任。

部分所有的信息均在 5.2.1-RELEASE 上。

### 32.7.1. 背景信息

置无盘工作站相要而又易出。有分析一些原因是很的。例如：

- 在行可能生不同的行。
- 出信息常是加密了的或根本就没有。

在里，及到的一些背景知于可能出的的解决是很有助的。

要成功地引系有些操作需要做。

- 机子需要取初始的参数，如它的 IP 地址、行文件、服务器名、根路径。个可以使用 或 BOOTP 来完成。DHCP 是 BOOTP 的兼容展，并使用相同的端口和基本包格式。

只使用 BOOTP 来配置系也是可行的。bootpd(8) 服程序被包含在基本的 FreeBSD 系里。

不，DHCP 相比 BOOTP 有几个好（更好的配置文件，使用 PXE 的可能性，以及多其它并不直接相关的无盘操作），接着我会要描述一个 DHCP 配置，可能的会利用与使用 bootpd(8) 相同的例子。个板配置会使用 ISC DHCP 件包 (3.0.1.r12 行版安装在服务器上)。

- 机子需要送一个或多个程序到本地内存。TFTP 或 NFS 会被使用。TFTP 是 NFS 需要在几个地方的"里置。通常的源是文件名指定了的：TFTP 通常从服务器里的一个目送所有文件，并需要相个目的文件名。NFS 需要的是文件路径。
- 介于程序和内核之的可能部分需要被初始化并行。在部分有几个重要的量：
  - PXE 会装入 pxeboot(8)——它是 FreeBSD 第三段装器的修改版。loader(8) 会得多参数用于系，并在送控制之前把它留在内核境里。在情况下，使用 GENERIC 内核就可能了。
  - Etherboot 会做很少的准直接装内核。要使用指定的建立 (build) 内核。

PXE 和 Etherboot 工作得的一的好。不，因一般情况下内核希望 loader(8) 做了更多的事情，PXE 是推的方法。

如果的 BIOS 和网都支持 PXE，就使用它。

- 最后，机子需要它的文件系统。NFS 使用在所有的情况下。

看 diskless(8) 手册。

## 32.7.2. 安装说明

### 32.7.2.1. 配置使用ISC DHCP

ISC DHCP 服务器可以回答 BOOTP 和 DHCP 的请求。

ISC DHCP 4.2 并不属于基本系统。首先需要安装 [net/isc-dhcp42-server](#) port 或相应的“包”。

一旦安装了 ISC DHCP，需要一个配置文件才能运行（通常名叫 `/usr/local/etc/dhcpd.conf`）。它有个注释的例子，里面主机 `margaux` 使用 Etherboot，而主机 `corbieres` 使用 PXE：

```
default-lease-time 600;
max-lease-time 7200;
authoritative;

option domain-name "example.com";
option domain-name-servers 192.168.4.1;
option routers 192.168.4.1;

subnet 192.168.4.0 netmask 255.255.255.0 {
    use-host-decl-names on; ①
    option subnet-mask 255.255.255.0;
    option broadcast-address 192.168.4.255;

    host margaux {
        hardware ethernet 01:23:45:67:89:ab;
        fixed-address margaux.example.com;
        next-server 192.168.4.4; ②
        filename "/data/misc/kernel.diskless"; ③
        option root-path "192.168.4.4:/data/misc/diskless"; ④
    }
    host corbieres {
        hardware ethernet 00:02:b3:27:62:df;
        fixed-address corbieres.example.com;
        next-server 192.168.4.4;
        filename "pxeboot";
        option root-path "192.168.4.4:/data/misc/diskless";
    }
}
```

- ① 这个选项告诉 `dhcpd` 发送 `host` 里声明的用于无盘主机的主机名的。另外可能会添加一个 `option host-name margaux` 到 `host` 声明里。
- ② `next-server` 正式指定 TFTP 或 NFS 服务器用于装入服务器或内核文件（默认使用的是相同的主机作为 DHCP 服务器）。
- ③ `filename` 正式定义要装的文件——etherboot 或 PXE 的下一行将装它。根据使用的方式，它必须指定。Etherboot 可以被用来使用 NFS 或 TFTP。FreeBSD port 默认配置了 NFS。PXE 使用 TFTP，它就是什么在里使用相同文件名（可能依赖于 TFTP 服务器配置，不会相当典型）。同时，PXE 会装 `pxeboot`，而不是内核。另外有几个很有意思的可能，如从 FreeBSD CD-ROM 的 `/boot` 目录装 `pxeboot`（因 `pxeboot(8)` 能装 GENERIC 内核，这就使得可以使用 PXE 从程序的 CD-ROM 里面）。

④ **root-path** 指定到根 (root) 文件系的路径，通常是 NFS 符号。当使用 PXE 时，只要不用内核里的 BOOTP 时，可以不管主机的IP。NFS 服务器然后就如同 TFTP 一样。

### 32.7.2.2. 配置使用BOOTP

这里跟的是一个等效的 bootpd 配置 (至少到一个客户端)。它可以在 /etc/bootptab 里找到。

注意：为了使用BOOTP，etherboot 必须使用非默认 **NO\_DHCP\_SUPPORT** 来运行，而且 PXE 需要 DHCP。bootpd 的唯一可取的好处是它存在于基本系统中。

```
.def100:\
:hn:ht=1:sa=192.168.4.4:vm=rfc1048:\
:sm=255.255.255.0:\
:ds=192.168.4.1:\
:gw=192.168.4.1:\
:hd="/tftpboot":\
:bf="/kernel.diskless":\
:rp="192.168.4.4:/data/misc/diskless":
```

```
margaux:ha=0123456789ab:tc=.def100
```

### 32.7.2.3. 使用Etherboot准备程序

[Etherboot 的网站](#) 包含有[更多的文档](#)——主要指的是 Linux 系，但无疑包含有有用的信息。如下列出的是关于在 FreeBSD 系里使用 Etherboot。

首先必须安装 [net/etherboot](#) 包或 port。

可以改变 Etherboot 的配置 (如使用 TFTP 来代替 NFS)，方法是修改 Config 文件——在 Etherboot 源目里。

关于我的配置，我要使用一个。关于其它的方法(PROM，或 MS-DOS®程序)，请参考 Etherboot 文档。

想要使用它，先输入一个到安装有 Etherboot 的机器的驱动器里，然后把当前路径改到 src 目录——在 Etherboot 下，接着输入：

```
# gmake bin32/devicetype.fd0
```

devicetype 依赖于无工作站上的以太网卡的类型。参考在同一个目录下的 NIC 文件正确的 devicetype。

### 32.7.2.4. 使用PXE

默认地，[pxeboot\(8\)](#) 装进通过 NFS 装进内核。它可以用来使用 TFTP——通过在文件 /etc/make.conf 里指定 **LOADER\_TFTP\_SUPPORT** 来代替。参看 /usr/shared/examples/etc/make.conf 里的注释了解如何配置。

除此之外还有一个未明确的 **make.conf** 选项——它可能用于置一系列控制台无机器会有用：**BOOT\_PXELDR\_PROBE\_KEYBOARD**和 **BOOT\_PXELDR\_ALWAYS\_SERIAL**。

当机器里，要使用 PXE，通常需要 **Boot from network** 选项——在 BIOS 设置里，或者在 PC 初始化的

候加入一个功能 (function key)。

### 32.7.2.5. 配置 TFTP 和 NFS 服务器

如果你正在使用 PXE 或 Etherboot——配置使用了 TFTP，那你需要在文件服务器上运行 tftpd：

1. 建立一个目录——从那里 tftpd 可以提供文件服务，如 /tftpboot。
2. 把一行加入到 /etc/inetd.conf 里：

```
tftp      dgram    udp wait   root      /usr/libexec/tftpd  tftpd -l -s /tftpboot
```



好像有一些版本的 PXE 需要 TCP 版本的 TFTP。在这种情况下，加入第二行，使用 **stream tcp** 来代替 **dgram udp**。

3. 让 inetd 重读其配置文件。要正运行这个命令，在 /etc/rc.conf 文件中必须加入 **inetd\_enable="YES"**：

```
# /etc/rc.d/inetd restart
```

你可以把 tftpboot 目录放到服务器上的任何地方。它的位置要在 inetd.conf 和 dhcpd.conf 里。

在所有的情况下，你都需要运行 NFS，并且 NFS 服务器上出现相应的文件系统。

1. 把一行加入到 /etc/rc.conf 里：

```
nfs_server_enable="YES"
```

2. 通往 /etc/exports 里加入下面几行(完整输入点"列，并且使用无工作站的名称替换 *margaux corbieres*)，输出文件系统——无根目录存在于此：

```
/data/misc -alldirs -ro margaux corbieres
```

3. 让 mountd 重读它的配置文件。如果你真的需要运行第一行的 /etc/rc.conf 里 NFS，你可能就要重启系统了。

```
# /etc/rc.d/mountd restart
```

### 32.7.2.6. 建立无内核

如果你在使用 Etherboot，你需要无客户端建立内核配置文件，使用如下(除了常使用的外)：

```
options      BOOTP          # Use BOOTP to obtain IP address/hostname
options      BOOTP_NFSROOT  # NFS mount root filesystem using BOOTP info
```

可能也想使用 `BOOTP_NFSV3`，`BOOT_COMPAT` 和 `BOOTP_WIRED_TO` (参考 NOTES 文件)。

些名字具有历史性，并且有些有些，因它上用了内核里 (它可能限制 BOOTP 或 DHCP 的使用)，与 DHCP 和 BOOTP 的无的。

内核(参考[配置FreeBSD的内核](#))，然后将它制到 `dhcpcd.conf` 里指定的地方。



当使用 PXE 里，使用以上建立内核并不做格要求(尽管建做)。用它会在内核引起更多的 DHCP 提及的求，来的小小的是在有些特殊情况下新和由 [pxeboot\(8\)](#) 取回的之的不一致性。使用它的好是主机名会被附置。否，就需要使用其它的方法来置主机名，如在客户端指定的 `rc.conf` 文件里。



了使有 Etherboot 的内核可引，就需要把提示 (device hint) 去。通常要在配置文件(看 NOTES 配置注文件) 里置下列：

```
hints      "GENERIC.hints"
```

### 32.7.2.7. 准根(root)文件系统

需要无工作站建立根文件系统，它就是 `dhcpcd.conf` 里的 `root-path` 所指定的目。

#### 32.7.2.7.1. 使用 `make world` 来制根文件系统

方法可以迅速安装一个底干的系 (不根文件系统) 到 `DESTDIR`。要做的就是地行下面的脚本：

```
#!/bin/sh
export DESTDIR=/data/misc/diskless
mkdir -p ${DESTDIR}
cd /usr/src; make buildworld && make buildkernel
make installworld && make installkernel
cd /usr/src/etc; make distribution
```

一旦完成，可能需要定制 `/etc/rc.conf` 和 `/etc/fstab`——根据的需要放到 `DESTDIR`里。

### 32.7.2.8. 配置 swap(交)

如果需要，位于服务器上的交文件可以通过 NFS 来。

#### 32.7.2.8.1. NFS 交区

内核并不支持在引用 NFS 交区。交区必通脚本用，其程是挂接一个可写的文件系统，并在其上建并用交文件。要建立尺寸合的交文件，可以这样做：



```
# dd if=/dev/zero of=/path/to/swapfile bs=1k count=1 oseek=100000
```

要用它，要把下面几行加到 `rc.conf` 里：

```
swapfile=/path/to/swapfile
```

### 32.7.2.9. 用户

#### 32.7.2.9.1. 用户 `/usr` 是只在

如果无工作站是配置来支持 X，那就必须整 XDM 配置文件，因为它默认把信息写到 `/usr`。

#### 32.7.2.9.2. 使用非 FreeBSD 服务器

当用作根文件系统的服务器行的是非 FreeBSD，要在 FreeBSD 机器上建立根文件系统，然后把它复制到它的目的地，使用的命令可以是 `tar` 或 `cpio`。

在这种情况下，有于 `/dev` 里的一些特殊的文件会有，原因就是不同的 "最大/最小" 整数大小。一解决的方法就是从非 FreeBSD 服务器里出一个目录，并把它入 FreeBSD 到机子上，并使用 `devfs(5)` 来用透明地分派点。

## 32.8. 从 PXE 启动一个 NFS 根文件系统

Intel® 启动环境 (PXE) 能操作系统从网启动。通常由近代主板的 BIOS 提供 PXE 支持，它可以通过在 BIOS 设置里从网启动。一个功能完整的 PXE 配置需要正确地设置 DHCP 和 TFTP 服务。

当计算机启动的时候，通过 DHCP 取得于从 TFTP 得到引导加载器 (boot loader) 的信息。在计算机接受此信息以后，便通过 TFTP 下载并运行引导加载器。这些于 [启动环境 \(PXE\) 的 2.2.1 章](#) 中。在 FreeBSD 中，在 PXE 过程中取得的引导加载器 `/boot/pxeboot`。在 `/boot/pxeboot` 运行之后，FreeBSD 的内核被加载，接着是其他的 FreeBSD 相关部分依次被运行。更多关于 FreeBSD 启动过程的信息参 [FreeBSD 引导程序](#)。

### 32.8.1. 配置用于 NFS 根文件系统的 `chroot` 环境

1. Choose a directory which will have a FreeBSD installation which will be NFS mountable. For example, a directory such as `/b/tftpboot/FreeBSD/install` can be used.

□□一个可被用□ NFS 挂□并安装有 FreeBSD 的目□。 比如可以使用像 `/b/tftpboot/FreeBSD/install` □□的一个目□。

```
# export NFSROOTDIR=/b/tftpboot/FreeBSD/install
# mkdir -p ${NFSROOTDIR}
```

2. 使用如下的命令□□ NFS 服□ [配置NFS](#).
3. 将下面□行加入 `/etc/exports` 用以通□ NFS □出此目□：

```
/b -ro -alldirs
```

4. 重起 NFS 服□：

```
# /etc/rc.d/nfsd restart
```

5. 按照 [□置](#) 中□明的□□用 [inetd\(8\)](#)。

6. 将如下□行加入到 `/etc/inetd.conf`：

```
tftp dgram udp wait root /usr/libexec/tftpd tftpd -l -s /b/tftpboot
```

7. 重□ inetd：

```
# /etc/rc.d/inetd restart
```

8. [重新□□ FreeBSD 内核和用□□](#)：

```
# cd /usr/src
# make buildworld
# make buildkernel
```

9. 把 FreeBSD 安装到 NFS 挂□目□：

```
# make installworld DESTDIR=${NFSROOTDIR}
# make installkernel DESTDIR=${NFSROOTDIR}
# make distribution DESTDIR=${NFSROOTDIR}
```

10. □□ TFTP 服□是否能下□将从 PXE □取的引□加□器：

```
# tftp localhost
tftp> get FreeBSD/install/boot/pxeboot
Received 264951 bytes in 0.1 seconds
```

11. 将 `${NFSROOTDIR}/etc/fstab` 并加入以下一行挂载 NFS 根文件系统：

| # Device                                       | Mountpoint | FSType | Options |
|--|------------|--------|---------|
| Dump Pass                                      |            |        |         |
| myhost.example.com:/b/tftpboot/FreeBSD/install | /          | nfs    | ro      |
| 0 0  |            |        |         |

用你的 NFS 服务器主机名或者 IP 地址替换 `myhost.example.com`。在此例中，根文件系统是以“只读”的方式挂载用来防止 NFS 客户端可能意外删除根文件系统上的文件。

12. 设置 `chroot(8)` 环境中的 root 密码。

```
# chroot ${NFSROOTDIR}
# passwd
```

此设置从 PXE 环境的客户端的 root 密码。

13. 允许 `ssh root` 登录从 PXE 环境的客户端，将 `${NFSROOTDIR}/etc/ssh/sshd_config` 并添加 `PermitRootLogin`。于此设置的说明参考 `sshd_config(5)`。
14. 将 `${NFSROOTDIR}` 的 `chroot(8)` 环境做些其他的定制。可以是像使用 `pkg_add(1)` 安装二进制包，使用 `vipw(8)` 修改密码，或者将 `amd.conf(8)` 映射自挂载等。例如：

```
# chroot ${NFSROOTDIR}
# pkg_add -r bash
```

### 32.8.2. 配置 `/etc/rc.initdiskless` 中用到的内存文件系统

如果你从一个 NFS 根卷启动，`/etc/rc` 如果启动是从 NFS 启动便会运行 `/etc/rc.initdiskless` 脚本。看看此脚本中的注释部分以便了解到底发生了什么。我需把 `/etc` 和 `/var` 做成内存文件系统的原因是它们需要能被写入，但 NFS 根文件系统是只读的。

```
# 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
```

当系统启动的时候，`/etc` 和 `/var` 内存文件系统就会被创建并挂载，`cpio.gz` 就会被复制到去。

### 32.8.3. 配置 DHCP 服务

PXE 需要配置一个 TFTP 服务器和一个 DHCP 服务器。DHCP 服务并不要求与 TFTP 服务在同一台机器上，但是必须能从网络的网口到它。

1. 按照此文[安装和配置 DHCP 服务器](#) 方法安装 DHCP 服务。确保 `/etc/rc.conf` 和 `/usr/local/etc/dhcpd.conf` 都配置正确。
2. 在 `/usr/local/etc/dhcpd.conf` 中配置 `next-server`, `filename`, `option root-path` 指向的 TFTP 服务器的 IP 地址，以及 TFTP 上 `/boot/pxeboot` 文件的路径，和 NFS 根文件系统的路径。这里是一个 `dhcpd.conf` 示例：

```
subnet 192.168.0.0 netmask 255.255.255.0 {
    range 192.168.0.2 192.168.0.3 ;
    option subnet-mask 255.255.255.0 ;
    option routers 192.168.0.1 ;
    option broadcast-address 192.168.0.255 ;
    option domain-name-server 192.168.35.35, 192.168.35.36 ;
    option domain-name "example.com";

    # IP address of TFTP server
    next-server 192.168.0.1 ;

    # path of boot loader obtained
    # via tftp
    filename "FreeBSD/install/boot/pxeboot" ;

    # pxeboot boot loader will try to NFS mount this directory for root FS
    option root-path "192.168.0.1:/b/tftpboot/FreeBSD/install/" ;

}
```

### 32.8.4. 配置 PXE 客户端与网络接口

1. 当客户端的时候， 入 BIOS 配置菜单。 置 BIOS 从网。 如果之前所有的配置都正的， 那所有部分能 "正常工作"。
2. 使用 [net/wireshark](#) port 看 DHCP 和 TFTP 的网流量来各。
3. 保 pxeboot 能从 TFTP 取。 在的 TFTP 服务器上 /var/log/xferlog 日志保 pxeboot 被从正的位置取。 可以上面例子 dhcpd.conf 中所置的：

```
# tftp 192.168.0.1
tftp> get FreeBSD/install/boot/pxeboot
Received 264951 bytes in 0.1 seconds
```

和 [tftpd\(8\)](#) 和 [tftp\(1\)](#)。 其中的 [BUGS](#) 列出了 TFTP 的一些限制。

4. 保根文件系统能从 NFS 挂。 可以上面例子 dhcpd.conf 中所置的：

```
# mount -t nfs 192.168.0.1:/b/tftpboot/FreeBSD/install /mnt
```

5. 在 `src/sys/boot/i386/libi386/pxe.c` 中的代码以了解 pxeboot 加器如何置如 `boot.nfsroot.server` 和 `boot.nfsroot.path` 之的量。 些量被用在了 `src/sys/nfsclient/nfs_diskless.c` 的 NFS 无根挂代中。
6. Read [pxeboot\(8\)](#) and [loader\(8\)](#).

## 32.9. ISDN

于 ISDN 技术和硬件的一个好的源是 [Dan Kegel 的 ISDN 主](#)。

一个快速的到 ISDN 的路如下：

- 如果住在欧洲，可能要看一下 ISDN 部分。
- 如果正首要地使用 ISDN 基于号非用路接到有提供商的互网， 可能要了解一下端配器。 如果更改提供商的， 会来最大的活性、最小的麻。
- 如果接了个局域网 (LAN)，或使用了用的 ISDN 接到互网，可能要考独的路由器/网。

在决定一方案的候，价格是个很的因素。 下面列有从不算到最的：

### 32.9.1. ISDN

FreeBSD 的 ISDN 工具通被 (passive card) 支持 DSS1/Q.931(或 Euro-ISDN) 准。 此外也支持一些 active card， 它的固件也支持其它信号， 其中包括最先得到支持的 "Primary Rate (PRI) ISDN"。

isdn4bsd 件允接到其它 ISDN 路由器，使用的是原始的 HDLC 上的 IP 或利用同 PPP：使用有 `isppp` (一个修改的 `sppp(4)` 程序的 PPP 内核，或使用用区 (userland) `ppp(8)`。 通使用 userland `ppp(8)`， 个或更多 ISDN 的 B 通道得可能。 除了多如 300 波特 (Baud) 的 modem 一的工具外， 可以答机用。

在 FreeBSD 里，正有更多的 PC ISDN 被支持；广告显示在整个欧洲及世界的其它多地区可以成功使用。

被支持的主型 ISDN 主要是有 Infineon (以前的 Siemens) ISAC/HSCX/IPAC ISDN 芯片，外有有 Cologne (只有 ISA 卡) 芯片的 ISDN 卡、有 Winbond W6692 芯片的 PCI 卡、一部分有 Tiger300/320/ISAC 芯片的卡以及有一些商家有的芯片的卡 (如 AVM FritzCard PCI V.1.0 和 the AVM FritzCard PnP)。

当前卡的支持的 ISDN 卡有 AVM B1 (ISA 和 PCI) BRI 卡和 AVM T1 PCI PRI 卡。

关于 isdn4bsd 的文章，看看 [isdn4bsd 的主页](#)，那里也有提示、勘误表以及更多的文章 (如 [isdn4bsd 手册](#))。

要是你对加不同 ISDN 卡的支持，当前不支持的 ISDN PC 卡的支持或想看看 isdn4bsd 的性能，联系 Hellmuth Michaelis <[hm@FreeBSD.org](mailto:hm@FreeBSD.org)>。

关于安装、配置以及 isdn4bsd 故障排除的，可以利用 [freebsd-isdn](#) 邮件列表。

### 32.9.2. ISDN 终端适配器

终端适配器 (TA) 关于 ISDN 就好比 modem 关于常。

多 TA 使用标准的 Hayes modem AT 命令集，并且可以降低来代替 modem。

TA 基本的工作同 modem 一样，不同之处是接口和整个速度更比老 modem 更快。同 modem 的安装一样，也需要配置 PPP。卡的串口速度已足够高。

使用 TA 接口互联网提供商的主要好处是可以做 PPP。由于 IP 地址空得越来越，多提供商都不再提供静态 IP。多的独立的路由器是不支持 IP 分配的。

TA 完全依赖于正在运行的 PPP 进程，以完成它的功能和接口。可以在 FreeBSD 机子里易地从使用 modem 升到 ISDN，要是已安装了 PPP 的。只是，在使用 PPP 程序所涉及到任何相同也存在。

如果想要最大的稳定性，使用 PPP 内核，而不要使用 userland PPP。

下面的 TA 就可以同 FreeBSD 一起工作：

- Motorola BitSurfer 和 Bitsurfer Pro
- Adtran

大部分其它的 TA 也可能工作，TA 提供商他的产品可以接受大部分的标准 modem AT 命令集。

关于外置 TA 的是：象 modem 要一样，机器需要有一个好的串行口。

想要更深入地理解串行以及和同串口不同的点，就要看看 [FreeBSD 串行硬件教程](#)了。

TA 将标准的 PC 串口 (同的) 限制到了 115.2 Kbs，即使有 128 Kbs 的接口。想要完全利用 ISDN 有能力达到的 128 Kbs，就需要把 TA 移到同串行口上。

当心被去一个内置的 TA 以及自可以避免同/。内置的 TA 只是地将一标准 PC 串口芯片内建在里。所做的些只是省去一根串行口以及省去一个空的孔。

有 TA 的同至少和一个独立的路由器同一快地，而且使用一个的 386 FreeBSD 盒它。

同TA 是独立的路由器，是个要高度慎的。在件列表里有些相的。我建去搜索一下于完整的。

### 32.9.3. 独的 ISDN /路由器

ISDN 或路由器根本就没有指定要 FreeBSD 或其它任何的操作系。更多完整的于路由和接技的描述，参考网指南的籍。

部分的内容里，路由器和接个将会交替地使用。

随着 ISDN 路由器/的价格下滑，它的也会得越来越流行。ISDN 路由器是一个小盒子，可以直接地接入的本地以太网，并且自我管理到其它/路由器的接。它有个内建的件用于与通信——通 PPP 和其它流行的。

路由器有比准 TA 更快的吞吐量，因它会使用完全同的 ISDN 接。

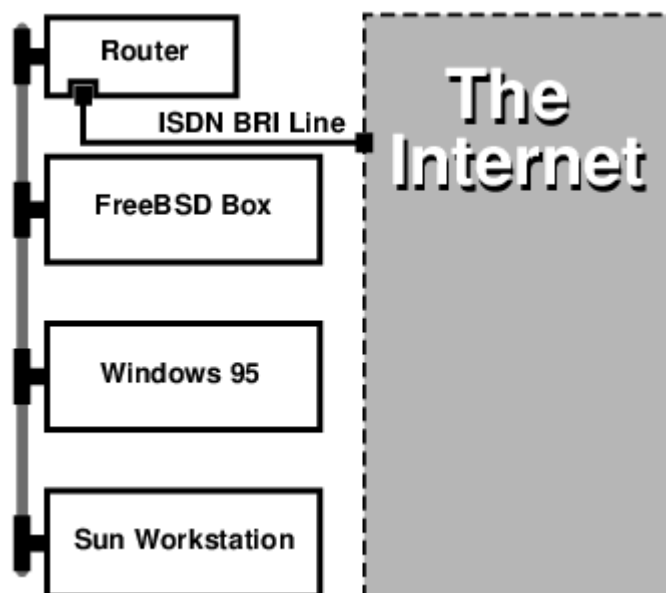
使用 ISDN 路由器和的主要是个生商之的同性仍存在。如果接到互网提供商，跟他行交。

如果接个局域网网段，如的家庭网和公网，将是最最低的解决方案。因的用于接的，可以保接一定会成功。

例如接到家里的计算机，或者是公网里的一个分支接到公主网，那下面的置就可能用到：

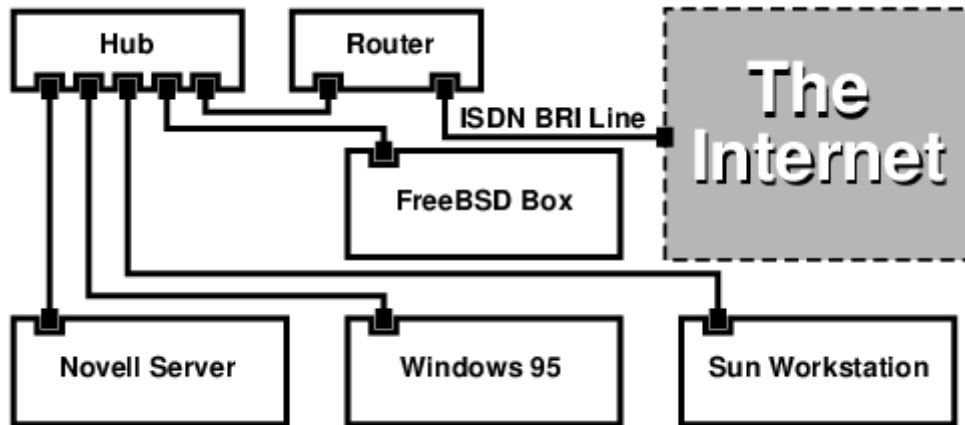
例 43. 公室局部或家庭网

网使用基于拓的 10 base 2 以太网 ("瘦网(thinnet)")。如果有必要，用网接路由器和 AUI/10BT 收器。



如果的家里或公室支部里只有一台计算机，可以使用一根交叉的双直接接那台独立路由器。

网使用的是星形拓扑的 10 base T 以太网("双")。



大部分路由器/网有一大好处就是，它允许在同一网，有每个分独立的 PPP 连接到每个分点上。点在多的 TA 上是不支持的，除非有每个串口的特定模式(通常都很)。不要把它与通道接、MPP 等相混。

是个非常有用的功能，例如，如果在办公室里有每个有的 ISDN 接，而且想接入到里，但休想一根 ISDN 也能工作。办公室里的路由器能管理每个有的 B 通道接到互网 (64 Kbps) 以及使用一个通道 B 来完成独的数据接。第二个 B 通道可以用于、出或与第一个 B 通道行接 (MPP 等)，以取更大。

以太网也允许的不 IP 通信。也可以送 IPX/SPX 或其它任何所使用的。

## 32.10. 网地址

### 32.10.1. 概要

FreeBSD 的网地址服，通常也被叫做 [natd\(8\)](#)，是一个能接收入的未理 IP 包，将源地址修改本地址然后重新将些包注入到出 IP 包流中。[natd\(8\)](#) 同修改源地址和端口，当接收到数据，它作逆向以便把数据回原先的求者。

NAT 最常的用途是人所熟知的 Internet 接共享。

### 32.10.2. 安装

随着 IPv4 的 IP 地址空的日益枯竭，以及使用如 DSL 和等高速接的用的逐多，越来越多的人始需要 Internet 接共享的解决方案。由于能将多计算机通一个外的 IP 地址行接入，[natd\(8\)](#) 成了一个理想的。

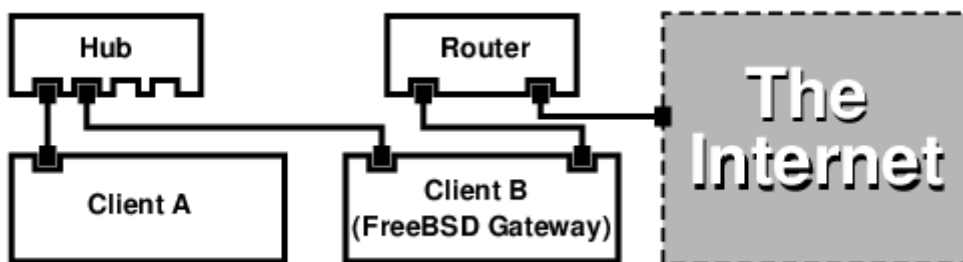
更常的情况，一个用通或者 DSL 路接入，并有一个 IP 地址，同，希望通台接入 Internet 的计算机来 LAN 上更多的计算机提供接入服。

了完成一任，接入 Internet 的 FreeBSD 机器必扮演网的角色。台网必有网 - 一用于接 Internet 路由器，一用来接 LAN。所有 LAN 上的机器通 Hub 或交换机行接。





有多种方法能通 FreeBSD 网将 LAN 接入 Internet。 个例子只介了有至少网的网。



上述配置被广泛地用于共享 Internet 接。 LAN 中的一台机器接到 Internet 中。 其余的计算机通那台“网”机来接 Internet。

### 32.10.3. 引加器配置

在默的 GENERIC 内核中， 并没有用通 `natd(8)` 行网址翻的功能， 不， 一功能可以通在 `/boot/loader.conf` 中添加配置来在引自予以加：

```
ipfw_load="YES"
ipdivert_load="YES"
```

此外， 可以将引加器量 `net.inet.ip.fw.default_to_accept` 1：

```
net.inet.ip.fw.default_to_accept="1"
```



在始配置防火和 NAT 网， 加个配置是个好主意。 默的 `ipfw(8)` 将是 `allow ip from any to any` 而不是默的 `deny ip from any to any`， ， 在系重， 也就不太容易被反在外面。

### 32.10.4. 内核配置

当不能使用内核模， 或更希望将全部需要的功能内核， 可以在内核配置中添加下面的置来：

```
options IPFIREWALL
options IPDIVERT
```

此外， 下列是一些可的：

```
options IPFIREWALL_DEFAULT_TO_ACCEPT
options IPFIREWALL_VERBOSE
```

### 32.10.5. 系统引导的配置

如果希望在系统引导过程中启用防火墙和 NAT 支持，请在 `/etc/rc.conf` 中添加下列配置：

```
gateway_enable="YES" ①
firewall_enable="YES" ②
firewall_type="OPEN" ③
natd_enable="YES"
natd_interface="fxp0" ④
natd_flags="" ⑤
```

- ① 将机器配置为网桥。运行 `sysctl net.inet.ip.forwarding=1` 效果相同。
- ② 在系统中用 `/etc/rc.firewall` 中的防火墙规则。
- ③ 指定一个特定的允许所有包通过的防火墙规则集。参看 `/etc/rc.firewall` 以了解其他类型的规则集。
- ④ 指定通过哪个网络接口包（接入 Internet 的那一个）。
- ⑤ 其他希望在 `natd(8)` 的参数。

在 `/etc/rc.conf` 中加入上述配置将在系统启动时运行 `natd -interface fxp0`。同一工作也可以手工完成。

当有太多配置需要时，也可以使用一个 `natd(8)` 的配置文件来完成。这种情况下，每个配置文件必须通过在 `/etc/rc.conf` 里添加下面内容来定义：

```
natd_flags="-f /etc/natd.conf"
```



`/etc/natd.conf` 文件会包含一个配置规则列表，每行一个。在跟部分的例子里将使用下面的文件：

```
redirect_port tcp 192.168.0.2:6667 6667
redirect_port tcp 192.168.0.3:80 80
```

对于配置文件的更多信息，参考 `natd(8)` 手册中关于 `-f` 的那一部分。

在 LAN 后面的每一台机器和接口都被分配私有地址空间（由 [RFC 1918](#) 定义）里的 IP 地址，并且默认网桥或 `natd` 机器的内网 IP 地址。

例如：客户端 A 和 B 在 LAN 后面，IP 地址是 `192.168.0.2` 和 `192.168.0.3`，同一台 `natd` 机器的 LAN 接口上的 IP 地址是 `192.168.0.1`。客户端 A 和 B 的默认网关必须设置成 `natd` 机器的 IP——`192.168.0.1`。`natd` 机器外部，或互联网接口不需要了 `natd(8)` 而做任何特殊的修改就可工作。

### 32.10.6. 端口重定向

使用 `natd(8)` 的缺点就是 LAN 客户端不能从互联网访问。LAN 上的客户端可以连接到外面的连接，而不能接收进来的连接。如果想在 LAN 的客户端机器上运行互联网服务，就会有问题。此问题的解决方法是在 `natd` 机器上重定向指定的互联网端口到 LAN 客户端。

例如：在客户端 A 上行 IRC 服，而在客户端 B 上行 web 服。想要正的工作，在端口 6667 (IRC) 和 80 (web) 上接收到的接就必须重定向到相应的机子上。

`-redirect_port` 需要使用适当的送 `natd(8)`。法如下：

```
-redirect_port proto targetIP:targetPORT[-targetPORT]
                [aliasIP:]aliasPORT[-aliasPORT]
                [remoteIP[:remotePORT[-remotePORT]]]
```

在上面的例子中，参数是：

```
-redirect_port tcp 192.168.0.2:6667 6667
-redirect_port tcp 192.168.0.3:80 80
```

就会重定向到的 `tcp` 端口到 LAN 上的客户端机子。

`-redirect_port` 参数可以用来指出端口用来代替个端口。例如，`tcp 192.168.0.2:2000-3000 2000-3000` 就会把所有在端口 2000 到 3000 上接收到的接重定向到主机 A 上的端口 2000 到 3000。

当直接行 `natd(8)`，就可以使用些，把它放到 `/etc/rc.conf` 里的 `natd_flags=""` 上，或通一个配置文件行送。

想要更多配置，参考 `natd(8)`。

### 32.10.7. 地址重定向

如果有几个 IP 地址提供，那地址重定向就会很有用，然而他必在一个机子上。使用它，`natd(8)` 就可以分配一个 LAN 客户端它自己的外部 IP 地址。`natd(8)` 然后会使用到的部 IP 地址重写从 LAN 客户端外出的数据包，以及重定向所有来的数据包——一定的 IP 地址回到特定的 LAN 客户端。也叫做静 NAT。例如，IP 地址 128.1.1.1、128.1.1.2 和 128.1.1.3 属于 `natd` 网机子。128.1.1.1 可以用来作 `natd` 网机子的外 IP 地址，而 128.1.1.2 和 128.1.1.3 用来回 LAN 客户端 A 和 B。

`-redirect_address` 法如下：

```
-redirect_address localIP publicIP
```

|          |                      |
|----------|----------------------|
| localIP  | LAN 客户端的内部 IP 地址。    |
| publicIP | 相应 LAN 客户端的外部 IP 地址。 |

在个例子里，参数是：

```
-redirect_address 192.168.0.2 128.1.1.2 -redirect_address 192.168.0.3 128.1.1.3
```

象 `-redirect_port` 一，些参数也是放在 `/etc/rc.conf` 里的 `natd_flags=""` 上，或通一个配置文件送它。使用地址重定向，就没有必要用端口重定向了，因所有在某个 IP 地址上收到的数据都被重定向了。

在 natd 架子上的外部 IP 地址必须激活并且名到 (aliased) 外接口。要这样做就看看 [rc.conf\(5\)](#)。

## 32.11. 并行 IP (PLIP)

PLIP 允许我在一个并行口运行 TCP/IP。 在使用笔记本， 或没有网络的计算机， 它会非常有用。 其中， 我将：

- 制作用于并口的 (laplink) 。
- 使用 PLIP 接口台计算机。

### 32.11.1. 制作并行口。

可以在多计算机供应商店里买到并行口。 如果找不到， 或者希望自行制作， 可以参看下面的表格， 它介绍了如何利用普通的打印机并行口来改制：

表 14. 用于网络接口的并行口接口方式

| A-name                       | A 端                  | B 端                  | 描述          | Post/Bit                      |
|------------------------------|----------------------|----------------------|-------------|-------------------------------|
| ....<br>DATA0 -ERROR<br>.... | ....<br>2 15<br>.... | ....<br>15 2<br>.... | 数据          | ....<br>0/0x01 1/0x08<br>.... |
| ....<br>DATA1 +SLCT<br>....  | ....<br>3 13<br>.... | ....<br>13 3<br>.... | 数据          | ....<br>0/0x02 1/0x10<br>.... |
| ....<br>DATA2 +PE<br>....    | ....<br>4 12<br>.... | ....<br>12 4<br>.... | 数据          | ....<br>0/0x04 1/0x20<br>.... |
| ....<br>DATA3 -ACK<br>....   | ....<br>5 10<br>.... | ....<br>10 5<br>.... | 脉冲 (Strobe) | ....<br>0/0x08 1/0x40<br>.... |
| ....<br>DATA4 BUSY<br>....   | ....<br>6 11<br>.... | ....<br>11 6<br>.... | 数据          | ....<br>0/0x10 1/0x80<br>.... |
| GND                          | 18-25                | 18-25                | GND         | -                             |

### 32.11.2. 配置 PLIP

首先，需要一根 laplink 线。然后， 每台计算机的内核都有 [lpt\(4\)](#) 程序的支持：

```
# grep lp /var/run/dmesg.boot
lpt0: Printer on ppbus0
lpt0: Interrupt-driven port
```

并行口必须是一个中断驱动的端口， 在 `/boot/device.hints` 文件中配置：

```
hint.ppc.0.at="isa"
hint.ppc.0.irq="7"
```

然后在内核配置文件中是否有一行 `device plip` 或加上了 `plip.ko` 内核模块。在这种情况下，在使用 `ifconfig(8)` 命令时都会显示并口型的网络接口，类似：

```
# ifconfig plip0
plip0: flags=8810<POINTOPOINT,SIMPLEX,MULTICAST> mtu 1500
```

用 `laplink` 接通两台计算机的并口。

在可以 `root` 身份配置网络参数。例如，如果希望将 `host1` 通过一台机器 `host2` 连接：

|            | host1    | ----- | host2    |
|------------|----------|-------|----------|
| IP Address | 10.0.0.1 |       | 10.0.0.2 |

配置 `host1` 上的网络接口，照此做：

```
# ifconfig plip0 10.0.0.1 10.0.0.2
```

配置 `host2` 上的网络接口，照此做：

```
# ifconfig plip0 10.0.0.2 10.0.0.1
```

现在你有工作的连接了。想要更多的信息，看看 `lp(4)` 和 `lpt(4)` 手册。

你还可以加个主机到 `/etc/hosts`：

|           |                     |           |
|-----------|---------------------|-----------|
| 127.0.0.1 | localhost.my.domain | localhost |
| 10.0.0.1  | host1.my.domain     | host1     |
| 10.0.0.2  | host2.my.domain     | host2     |

要连接是否工作，可以到一台机子上，然后 `ping` 外一台。例如，在 `host1` 上：

```
# ifconfig plip0
plip0: flags=8851<UP,POINTOPOINT,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    inet 10.0.0.1 --> 10.0.0.2 netmask 0xff000000
# netstat -r
Routing tables

Internet:
Destination            Gateway                Flags      Refs      Use      Netif Expire
host2                   host1                  UH          0          0          plip0

# ping -c 4 host2
PING host2 (10.0.0.2): 56 data bytes
64 bytes from 10.0.0.2: icmp_seq=0 ttl=255 time=2.774 ms
64 bytes from 10.0.0.2: icmp_seq=1 ttl=255 time=2.530 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=255 time=2.556 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=255 time=2.714 ms

--- host2 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 2.530/2.643/2.774/0.103 ms
```

## 32.12. IPv6

IPv6 (也被称作 IPng "下一代 IP") 是众所周知的 IP 协议 (也叫 IPv4) 的新版本。和其他平台的 \*BSD 系统一样，FreeBSD 包含了 KAME 的 IPv6 参考实现。因此，它的 FreeBSD 系统包含了所有 IPv6 所需要的工具。它主要集中在如何配置和使用 IPv6。

在 1990 年代早期，人们开始担心可用的 IPv4 地址空间在不断地缩小。随着 Internet 的爆炸式发展，主要的担心是：

- 用尽所有的地址。当然在某个时候已不再那么尖锐，因为 RFC1918 私有地址空间 (10.0.0.0/8、172.16.0.0/12，以及 192.168.0.0/16) 和网络地址转换 (NAT) 技术已被广泛采用。
- 路由表条目变得太大。这点今天仍然是焦点。

IPv6 解决这些问题和其它问题的方法：

- 128 位地址空间。简单地说，理论上 340,282,366,920,938,463,374,607,431,768,211,456 个地址可以使用。这意味着在我们的星球上每平方米大约有  $6.67 * 10^{27}$  个 IPv6 地址。
- 路由器在它们的路由表里存放网络地址集，这就减少路由表的平均空间到 8192 个条目。

IPv6 有其它很多有用的功能，如：

- 地址自动配置 (RFC2462)
- Anycast (任意播) 地址("一多")
- 限制的多播地址
- IPsec (IP 安全)
- 更多的...

- 移动的 (Mobile) IP
- IPv6 到 IPv4 的隧道机制

要更多信息，看看：

- IPv6 概览，在[playground.sun.com](http://playground.sun.com)
- [KAME.net](http://KAME.net)

### 32.12.1. 关于 IPv6 地址的背景知识

有几不同模型的 IPv6 地址：Unicast, Anycast 和 Multicast。

Unicast 地址是人们所熟知的地址。一个被送到 unicast 地址的包会上会到属于该地址的接口。

Anycast 地址基本上与 unicast 地址没有差别，只是它只有一个接口。指定该 anycast 地址的包会到最近的 (以路由距离) 接口。Anycast 地址可能只被路由器使用。

Multicast 地址只有一个接口。指定该 multicast 地址的包会到属于 multicast 的所有的接口。



IPv4 广播地址 (通常 `xxx.xxx.xxx.255`) 由 IPv6 的 multicast 地址来表示。

表 15. 保留的 IPv6 地址

| IPv6 地址                       | 固定度 (bits) | 描述               | 备注  |
|-------------------------------|------------|------------------|---|
| <code>::</code>               | 128 bits   | 未指定              | 类似 IPv4 中的 <code>0.0.0.0</code>           |
| <code>::1</code>              | 128 bits   | 回路地址             | 类似 IPv4 中的 <code>127.0.0.1</code>         |
| <code>::00:xx:xx:xx:xx</code> | 96 bits    | 嵌入的 IPv4         | 低 32 bits 是 IPv4 地址。也称为 "IPv4 兼容 IPv6 地址" |
| <code>::ff:xx:xx:xx:xx</code> | 96 bits    | IPv4 映射的 IPv6 地址 | 低的 32 bits 是 IPv4 地址。用于那些不支持 IPv6 的主机。    |
| <code>fe80:: - feb::</code>   | 10 bits    | 链路本地             | 类似 IPv4 的回路地址。                            |
| <code>fec0:: - fef::</code>   | 10 bits    | 站点本地             |   |
| <code>ff::</code>             | 8 bits     | 多播               |   |
| <code>001</code> (base 2)     | 3 bits     | 全球多播             | 所有的全球多播地址都指定到该地址池中。前三个二进制位是 "001"。        |

### 32.12.2. IPv6 地址的写法

该形式被描述为：`x:x:x:x:x:x:x`， 一个"x"就是一个 16 位的 16 进制数。当然， 一个十六进制数可以三个"0"开始也可以省略。如 `FEBc:A574:382B:23C1:AA49:4592:4EFE:9982`

通常一个地址会有很长的子串全部为零， 因此该地址的短子串常被缩写为 "::"。 例如：`fe80::1` 的完整形式是 `fe80:0000:0000:0000:0000:0000:0000:0001`。

第三种形式是以人所周知的用点"."作分隔符的十进制 IPv4 形式, 写出最后 32 Bit 的部分。例如 `2002::10.0.0.1` 的十进制正表方式是 `2002:0000:0000:0000:0000:0a00:0001` 它也相当于写成 `2002::a00:1`。

到现在, 读者能理解下面的内容了:

```
# ifconfig
```

```
rl0: flags=8943UP,BROADCAST,RUNNING,PROMISC,SIMPLEX,MULTICAST mtu 1500
    inet 10.0.0.10 netmask 0xffffffff broadcast 10.0.0.255
    inet6 fe80::200:21ff:fe03:8e1%rl0 prefixlen 64 scopeid 0x1
    ether 00:00:21:03:08:e1
    media: Ethernet autoselect (100baseTX )
    status: active
```

`fe80::200:21ff:fe03:8e1%rl0` 是一个自配置的回路地址。它作为自配置的一部分由 MAC 生成。

关于 IPv6 地址的更多信息, 参看 [RFC3513](#)。

### 32.12.3. 行接

目前, 有四种方式可以连接到其它 IPv6 主机和网络:

- 客户的互联网服务提供商是否提供 IPv6。
- [SixXS](#) 向全球提供通道。
- 使用 6-to-4 通道 ([RFC3068](#))
- 如果使用的是号接, 可以使用 [net/freenet6](#) port。

### 32.12.4. IPv6 世界里的 DNS

关于 IPv6 有新型的 DNS : IETF 已宣布 A6 是准; 行的准是 AAAA 。

使用 AAAA 是很的。通加下面内容, 的主机分配才接收到的新的 IPv6 地址:

```
MYHOSTNAME          AAAA      MYIPv6ADDR
```

到的主域 DNS 文件里, 就可以完成。要是自己没有 DNS 域服, 可以到的 DNS 提供商。目前的 bind 版本 (version 8.3 与 9) 和 [dns/djbdns](#)(含 IPv6) 支持 AAAA 。

### 32.12.5. 在 /etc/rc.conf 中行所需的修改

#### 32.12.5.1. IPv6 客机置

些置将助把一台 LAN 上的机器配置一台客机, 而不是路由器。要 [rtsol\(8\)](#) 在自配置的网, 只需添加:



```
ipv6_enable="YES"
```

要自⼰地靜⼰指定 IP 地址，例如 **2001:471:1f11:251:290:27ff:fee0:2093**，到 fxp0 上，⼰写上：

```
ipv6_ifconfig_fxp0="2001:471:1f11:251:290:27ff:fee0:2093"
```

要指定 **2001:471:1f11:251::1** 作⼰默⼰路由，需要在 /etc/rc.conf 中加入：

```
ipv6_defaultrouter="2001:471:1f11:251::1"
```

### 32.12.5.2. IPv6 路由器/网⼰配置

⼰将⼰助⼰从隧道提供商那里取得必要的⼰料，并将⼰些⼰料⼰化⼰在重⼰能⼰保持住的⼰置。要在⼰⼰恢⼰的隧道，需要在 /etc/rc.conf 中⼰加：

列出要配置的通用隧道接口，例如 gif0：

```
gif_interfaces="gif0"
```

配置⼰接口使用本地端地址 *MY\_IPv4\_ADDR* 和⼰程端地址 *REMOTE\_IPv4\_ADDR*：

```
gifconfig_gif0="MY_IPv4_ADDR REMOTE_IPv4_ADDR"
```

⼰用分配⼰用于 IPv6 隧道⼰端的 IPv6 地址，需要⼰加：

```
ipv6_ifconfig_gif0="MY_ASSIGNED_IPv6_TUNNEL_ENDPOINT_ADDR"
```

此后⼰置 IPv6 的默⼰路由。⼰是 IPv6 隧道的⼰一端：

```
ipv6_defaultrouter="MY_IPv6_REMOTE_TUNNEL_ENDPOINT_ADDR"
```

### 32.12.5.3. IPv6 隧道配置

如果服⼰器将⼰的网⼰通⼰ IPv6 路由到世界的其他角落，⼰需要在 /etc/rc.conf 中添加下面的配置：

```
ipv6_gateway_enable="YES"
```

## 32.12.6. 路由宣告和主机自⼰配置

⼰将⼰助⼰配置 [rtadvd\(8\)](#) 来宣示默⼰的 IPv6 路由。

要用 `rtadvd(8)` 需要在 `/etc/rc.conf` 中添加：

```
rtadvd_enable="YES"
```

指定由哪个网口来完成 IPv6 路由请求非常重要。例如，`rtadvd(8)` 使用 `fxp0`：

```
rtadvd_interfaces="fxp0"
```

接下来我需要创建配置文件，`/etc/rtadvd.conf`。示例如下：

```
fxp0:\
:addr#1:addr="2001:471:1f11:246::":prefixlen#64:tc=ether:
```

将 `fxp0` 改成打算使用的接口名。

接下来，将 `2001:471:1f11:246::` 改成分配的地址前缀。

如果有用的 `/64` 子网，不需要修改其他配置。反之，需要把 `prefixlen#` 改成正的。

## 32.13. 桥接模式 (ATM)

### 32.13.1. 配置 classical IP over ATM (PVCs)

Classical IP over ATM (CLIP) 是最常用的使用 IP 的 ATM 的方法。该方法可以用在交互式连接 (SVC) 和永久连接 (PVC) 上。部分描述的就是配置基于 PVC 的网络。

#### 32.13.1.1. 完全互连的配置

第一使用 PVC 来配置 CLIP 的方式就是通过用的 PVC 网络里的一台机器都互连在一起。尽管配置起来很复杂，但对于数量更多一点的机器来就有些不切实际了。例如我有四台机器在网络里，一台都使用一个 ATM 适配器连接到 ATM 网络。第一就是配置 IP 地址和机器的 ATM 连接。我使用下面的：

| 主机    | IP 地址         |
|-------|---------------|
| hostA | 192.168.173.1 |
| hostB | 192.168.173.2 |
| hostC | 192.168.173.3 |
| hostD | 192.168.173.4 |

为了建造完全互连的网络，我需要在第一台机器有一个 ATM 连接：

| 机器            | VPI.VCI 号 |
|---------------|-----------|
| hostA - hostB | 0.100     |
| hostA - hostC | 0.101     |
| hostA - hostD | 0.102     |

| 机器            | VPI.VCI |
|---------------|---------|
| hostB - hostC | 0.103   |
| hostB - hostD | 0.104   |
| hostC - hostD | 0.105   |

在一个接口端 VPI 和 VCI 的值都可能会不同，只是为了图起，我们假定它是一致的。下面我们还需要配置一个主机上的 ATM 接口：

```
hostA# ifconfig hatm0 192.168.173.1 up
hostB# ifconfig hatm0 192.168.173.2 up
hostC# ifconfig hatm0 192.168.173.3 up
hostD# ifconfig hatm0 192.168.173.4 up
```

假定所有主机上的 ATM 接口都是 hatm0。在 PVC 需要配置到 hostA 上 (我们假定它都已配置在了 ATM 交换机上，至于怎么做的，就需要参考一下交换机的手册了)。

```
hostA# atmconfig natm add 192.168.173.2 hatm0 0 100 llc/snap ubr
hostA# atmconfig natm add 192.168.173.3 hatm0 0 101 llc/snap ubr
hostA# atmconfig natm add 192.168.173.4 hatm0 0 102 llc/snap ubr

hostB# atmconfig natm add 192.168.173.1 hatm0 0 100 llc/snap ubr
hostB# atmconfig natm add 192.168.173.3 hatm0 0 103 llc/snap ubr
hostB# atmconfig natm add 192.168.173.4 hatm0 0 104 llc/snap ubr

hostC# atmconfig natm add 192.168.173.1 hatm0 0 101 llc/snap ubr
hostC# atmconfig natm add 192.168.173.2 hatm0 0 103 llc/snap ubr
hostC# atmconfig natm add 192.168.173.4 hatm0 0 105 llc/snap ubr

hostD# atmconfig natm add 192.168.173.1 hatm0 0 102 llc/snap ubr
hostD# atmconfig natm add 192.168.173.2 hatm0 0 104 llc/snap ubr
hostD# atmconfig natm add 192.168.173.3 hatm0 0 105 llc/snap ubr
```

当然，除 UBR 外其它的通信定也可 ATM 配置器支持。此情况下，通信定的名字要跟人通信参数后面。工具 [atmconfig\(8\)](#) 的帮助可以得到：

```
# atmconfig help natm add
```

或者在 [atmconfig\(8\)](#) 手册里得到。

相同的配置也可以通 /etc/rc.conf 来完成。于 hostA，看起来就象：

```
network_interfaces="lo0 hatm0"
ifconfig_hatm0="inet 192.168.173.1 up"
natm_static_routes="hostB hostC hostD"
route_hostB="192.168.173.2 hatm0 0 100 llc/snap ubr"
route_hostC="192.168.173.3 hatm0 0 101 llc/snap ubr"
route_hostD="192.168.173.4 hatm0 0 102 llc/snap ubr"
```

所有 CLIP 路由的当前状␣可以使用如下命令␣得：

```
hostA# atmconfig natm show
```

### 32.14. Common Address Redundancy Protocol (CARP, 共用地址冗余␣␣)

Common Address Redundancy Protocol, 或␣称 CARP 能␣使多台主机共享同一 IP 地址。在某些配置中, ␣␣做可以提高可用性, 或␣␣␣␣均衡。下面的例子中, ␣些主机也可以同␣使用其他的不同的 IP 地址。

要␣用 CARP 支持, 必␣在 FreeBSD 内核配置中␣加下列␣␣, 并按照 [配置FreeBSD的内核](#) 章␣介␣的方法重新␣␣内核：

```
device carp
```

␣外的一个方法是在␣␣␣加␣ if\_carp.ko 模␣。把如下的␣行加入到 /boot/loader.conf：

```
if_carp_load="YES"
```

␣␣就可以使用 CARP 功能了, 一些具体的参数, 可以通␣一系列 `sysctl`OID 来␣整。

| OID   | 描述  |
|---|---|
| <code>net.inet.carp.allow</code>            | 接受␣来的 CARP 包。默␣␣用。  |
| <code>net.inet.carp.preempt</code>          | 当主机中有一个 CARP 网␣接口失去␣␣␣, ␣个␣␣将停止␣台主机上所有的 CARP 接口。默␣禁用。                             |
| <code>net.inet.carp.log</code>              | 当␣␣ 0 表示禁止␣␣所有日志。␣␣ 1 表示␣␣␣坏的 CARP 包。任何大于 1 表示␣␣ CARP 网␣接口的状␣␣化。默␣␣␣ 1。           |
| <code>net.inet.carp.arpbalance</code>       | 使用 ARP 均衡本地网␣流量。默␣禁用。   |
| <code>net.inet.carp.suppress_preempt</code> | 此只␣ OID ␣示抑制␣占的状␣。如果一个接口上的␣接口失去␣␣, ␣␣占会被抑制。当␣个␣量的␣␣ 0 ␣, 表示␣占未被抑制。任何␣␣都会使 OID ␣␣。 |

CARP ␣␣可以通␣ `ifconfig` 命令来␣建。

```
# ifconfig carp0 create
```

在真⬡境中，⬡些接口需要一个称作 VHID 的⬡⬡号。⬡个 VHID 或 Virtual Host Identification (虚⬡主机⬡) 用于在网⬡上区分主机。

### 32.14.1. 使用 CARP 来改善服⬡的可用性 (CARP)

如前面提到的那⬡，CARP 的作用之一是改善服⬡的可用性。⬡个例子中，将⬡三台主机提供故障⬡移服⬡，⬡三台服⬡器各自有独立的 IP 地址，并提供完全一⬡的 web 内容。三台机器以 DNS ⬡⬡的方式提供服⬡。用于故障⬡移的机器有⬡个 CARP 接口，分⬡配置⬡外⬡台服⬡器的 IP 地址。当有服⬡器⬡生故障⬡，⬡台机器会自⬡得到故障机的 IP 地址。⬡⬡以来，用⬡就完全感⬡不到⬡生了故障。故障⬡移的服⬡器提供的内容和服⬡，⬡与其⬡之提供⬡⬡的服⬡器一致。

⬡台机器的配置，除了主机名和 VHID 之外⬡完全一致。在我⬡的例子中，⬡⬡台机器的主机名分⬡是 `hosta.example.org` 和 `hostb.example.org`。首先，需要将 CARP 配置加入到 `rc.conf`。⬡于 `hosta.example.org` 而言，`rc.conf` 文件中⬡包含下列配置：

```
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，⬡⬡的 `rc.conf` 配置⬡是：

```
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"
```



在⬡台机器上由 `ifconfig` 的 `pass` ⬡⬡指定的密⬡必⬡是一致的，⬡一点非常重要。 `carp` ⬡⬡只会⬡听和接受来自持有正⬡密⬡的机器的公告。此外，不同虚⬡主机的 VHID 必⬡不同。

第三台机器， `provider.example.org` 需要⬡行配置，以便在⬡外⬡台机器出⬡⬡⬡接管。⬡台机器需要⬡个 `carp` ⬡⬡，分⬡理⬡个机器。⬡⬡的 `rc.conf` 配置⬡似下面⬡⬡：

```
hostname="provider.example.org"
ifconfig_fxp0="inet 192.168.1.5 netmask 255.255.255.0"
cloned_interfaces="carp0 carp1"
ifconfig_carp0="vhid 1 advskew 100 pass testpass 192.168.1.50/24"
ifconfig_carp1="vhid 2 advskew 100 pass testpass 192.168.1.51/24"
```

配置⬡个 `carp` ⬡⬡，能⬡⬡ `provider.example.org` 在⬡台机器中的任何一个停止⬡⬡，立即接管其 IP 地址。



默认的 FreeBSD 内核可能占用了主机名。如果是这样的话，[provider.example.org](https://provider.example.org) 可能在正式的内容服务器上恢复时不释放 IP 地址。此时，管理员必须手工控制 IP 回到原来内容服务器。具体做法是在 [provider.example.org](https://provider.example.org) 上使用下面的命令：

```
# ifconfig carp0 down ifconfig carp0 up
```

这个操作需要在与出主机的这个 carp 接口上行。

现在已完成了 CARP 的配置，并可以开始了。过程中，可以随时重新或切断这台机器的网。

如欲了解更多，请参考 [carp\(4\)](#) 手册。

## Part V: 附口

# 附录 A: 获取 FreeBSD

## A.1. CDROM 和 DVD 发行商

### A.1.1. 零售盒装产品

可以从下面几个零售商那里得到 FreeBSD 的盒装产品 (FreeBSD CD, 附加配件, 印刷文档) :

- CompUSA  
WWW: <http://www.compusa.com/>
- Frys Electronics  
WWW: <http://www.frys.com/>

### A.1.2. CD 和 DVD 光盘

FreeBSD CD 和 DVD 光盘可以从许多在零售商那里得到 :

- FreeBSD Mall, Inc.  
700 Harvest Park Ste F  
Brentwood, CA 94513  
USA  
Phone: +1 925 240-6652  
Fax: +1 925 674-0821  
Email: <[info@freebsdmail.com](mailto:info@freebsdmail.com)>  
WWW: <http://www.freebsdmail.com/>
- Dr. Hinner EDV  
St. Augustinus-Str. 10  
D-81825 München  
Germany  
Phone: (089) 428 419  
WWW: <http://www.hinner.de/linux/freebsd.html>
- Ikarios  
22-24 rue Voltaire  
92000 Nanterre  
France  
WWW: <http://ikarios.com/form/#freebsd>
- JMC Software  
Ireland  
Phone: 353 1 6291282  
WWW: <http://www.thelinuxmall.com>
- The Linux Emporium  
Hilliard House, Lester Way  
Wallingford  
OX10 9TA  
United Kingdom



Phone: +44 1491 837010

Fax: +44 1491 837016

WWW: <http://www.linuxemporium.co.uk/products/bsd/>

- Linux+ DVD Magazine  
Lewartowskiego 6  
Warsaw  
00-190  
Poland  
Phone: +48 22 860 18 18  
Email: <[editors@lpmagazine.org](mailto:editors@lpmagazine.org)>  
WWW: <http://www.lpmagazine.org/>
- Linux System Labs Australia  
21 Ray Drive  
Balwyn North  
VIC - 3104  
Australia  
Phone: +61 3 9857 5918  
Fax: +61 3 9857 8974  
WWW: <http://www.lsl.com.au>
- LinuxCenter.Ru  
Galernaya Street, 55  
Saint-Petersburg  
190000  
Russia  
Phone: +7-812-3125208  
Email: <[info@linuxcenter.ru](mailto:info@linuxcenter.ru)>  
WWW: <http://linuxcenter.ru/shop/freebsd>

### A.1.3. 行人

如果您是售商并且想售 FreeBSD CDROM 品， 和行人系：

- Cylogistics  
809B Cuesta Dr., #2149  
Mountain View, CA 94040  
USA  
Phone: +1 650 694-4949  
Fax: +1 650 694-4953  
Email: <[sales@cylogistics.com](mailto:sales@cylogistics.com)>  
WWW: <http://www.cylogistics.com/>
- Ingram Micro  
1600 E. St. Andrew Place  
Santa Ana, CA 92705-4926  
USA  
Phone: 1 (800) 456-8000  
WWW: <http://www.ingrammicro.com/>

- Kudzu, LLC  
7375 Washington Ave. S.  
Edina, MN 55439  
USA  
Phone: +1 952 947-0822  
Fax: +1 952 947-0876  
Email: <[sales@kudzuenterpises.com](mailto:sales@kudzuenterpises.com)>
- LinuxCenter.Kz  
Ust-Kamenogorsk  
Kazakhstan  
Phone: +7-705-501-6001  
Email: <[info@linuxcenter.kz](mailto:info@linuxcenter.kz)>  
WWW: <http://linuxcenter.kz/page.php?page=fr>
- LinuxCenter.Ru  
Galernaya Street, 55  
Saint-Petersburg  
190000  
Russia  
Phone: +7-812-3125208  
Email: <[info@linuxcenter.ru](mailto:info@linuxcenter.ru)>  
WWW: <http://linuxcenter.ru/freebsd>
- Navarre Corp  
7400 49th Ave South  
New Hope, MN 55428  
USA  
Phone: +1 763 535-8333  
Fax: +1 763 535-0341  
WWW: <http://www.navarre.com/>

## A.2. FTP 站点

官方的 FreeBSD 源代可以从遍布全球的镜像站点通过匿名 FTP 下。站点 <ftp://ftp.FreeBSD.org/pub/FreeBSD/> 有着良好的网连接并且允大量的并连接，但是 或更想一个 "更近的" 镜像站点 (特是当想行某形式的像的候)。

FreeBSD 可以从下面些镜像站点通过匿名 FTP 下。如果了通匿名 FTP 取 FreeBSD，尽量使用比近的站点。被列 "主镜像站点" 的镜像站点一般都有完整的 FreeBSD 文件 (体系的所有当前可用的版本)，或从所在的国家或地区的站点下会得到更快的下速度。个站点提供了最流行的体系的最近的版本而有可能不提供完整的 FreeBSD 存。所有的站点都提供匿名 FTP 而有些站点也提供其他的的方式。个站点可用的的方式 在其主机名后有所明。

[Central Servers](#), [Primary Mirror Sites](#), [Armenia](#), [Australia](#), [Austria](#), [Brazil](#), [Czech Republic](#), [Denmark](#), [Estonia](#), [Finland](#), [France](#), [Germany](#), [Greece](#), [Hong Kong](#), [Ireland](#), [Japan](#), [Korea](#), [Latvia](#), [Lithuania](#), [Netherlands](#), [New Zealand](#), [Norway](#), [Poland](#), [Russia](#), [Saudi Arabia](#), [Slovenia](#), [South Africa](#), [Spain](#), [Sweden](#), [Switzerland](#), [Taiwan](#), [Ukraine](#), [United Kingdom](#), [United States of America](#).

(as of UTC)

## Central Servers

<ftp://ftp.FreeBSD.org/pub/FreeBSD/> (ftp / ftpv6 / <http://ftp.FreeBSD.org/pub/FreeBSD/> / <http://ftp.FreeBSD.org/pub/FreeBSD/>)

## Primary Mirror Sites

In case of problems, please contact the hostmaster <[mirror-admin@FreeBSD.org](mailto:mirror-admin@FreeBSD.org)> for this domain.

- <ftp://ftp1.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp2.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp3.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp4.FreeBSD.org/pub/FreeBSD/> (ftp / ftpv6 / <http://ftp4.FreeBSD.org/pub/FreeBSD/> / <http://ftp4.FreeBSD.org/pub/FreeBSD/>)
- <ftp://ftp5.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp6.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp7.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp10.FreeBSD.org/pub/FreeBSD/> (ftp / ftpv6 / <http://ftp10.FreeBSD.org/pub/FreeBSD/> / <http://ftp10.FreeBSD.org/pub/FreeBSD/>)
- <ftp://ftp11.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp13.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp14.FreeBSD.org/pub/FreeBSD/> (ftp / <http://ftp14.FreeBSD.org/pub/FreeBSD/>)

## Armenia

In case of problems, please contact the hostmaster <[hostmaster@am.FreeBSD.org](mailto:hostmaster@am.FreeBSD.org)> for this domain.

- <ftp://ftp1.am.FreeBSD.org/pub/FreeBSD/> (ftp / <http://ftp1.am.FreeBSD.org/pub/FreeBSD/> / rsync)

## Australia

In case of problems, please contact the hostmaster <[hostmaster@au.FreeBSD.org](mailto:hostmaster@au.FreeBSD.org)> for this domain.

- <ftp://ftp.au.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp2.au.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp3.au.FreeBSD.org/pub/FreeBSD/> (ftp)

## Austria

In case of problems, please contact the hostmaster <[hostmaster@at.FreeBSD.org](mailto:hostmaster@at.FreeBSD.org)> for this domain.

- <ftp://ftp.at.FreeBSD.org/pub/FreeBSD/> (ftp / ftpv6 / <http://ftp.at.FreeBSD.org/pub/FreeBSD/> / <http://ftp.at.FreeBSD.org/pub/FreeBSD/>)

## Brazil

In case of problems, please contact the hostmaster <[hostmaster@br.FreeBSD.org](mailto:hostmaster@br.FreeBSD.org)> for this domain.

- <ftp://ftp2.br.FreeBSD.org/FreeBSD/> (ftp / <http://ftp2.br.FreeBSD.org/>)
- <ftp://ftp3.br.FreeBSD.org/pub/FreeBSD/> (ftp / rsync)
- <ftp://ftp4.br.FreeBSD.org/pub/FreeBSD/> (ftp)

## Czech Republic

In case of problems, please contact the hostmaster <[hostmaster@cz.FreeBSD.org](mailto:hostmaster@cz.FreeBSD.org)> for this domain.

- <ftp://ftp.cz.FreeBSD.org/pub/FreeBSD/> (ftp / <ftp://ftp.cz.FreeBSD.org/pub/FreeBSD/> / <http://ftp.cz.FreeBSD.org/pub/FreeBSD/> / rsync / rsyncv6)
- <ftp://ftp2.cz.FreeBSD.org/pub/FreeBSD/> (ftp / <http://ftp2.cz.FreeBSD.org/pub/FreeBSD/>)

## Denmark

In case of problems, please contact the hostmaster <[staff@dotsrc.org](mailto:staff@dotsrc.org)> for this domain.

- <ftp://ftp.dk.FreeBSD.org/pub/FreeBSD/> (ftp / ftpv6 / <http://ftp.dk.FreeBSD.org/pub/FreeBSD/> / <http://ftp.dk.FreeBSD.org/pub/FreeBSD/>)

## Estonia

In case of problems, please contact the hostmaster <[hostmaster@ee.FreeBSD.org](mailto:hostmaster@ee.FreeBSD.org)> for this domain.

- <ftp://ftp.ee.FreeBSD.org/pub/FreeBSD/> (ftp)

## Finland

In case of problems, please contact the hostmaster <[hostmaster@fi.FreeBSD.org](mailto:hostmaster@fi.FreeBSD.org)> for this domain.

- <ftp://ftp.fi.FreeBSD.org/pub/FreeBSD/> (ftp)

## France

In case of problems, please contact the hostmaster <[hostmaster@fr.FreeBSD.org](mailto:hostmaster@fr.FreeBSD.org)> for this domain.

- <ftp://ftp.fr.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp1.fr.FreeBSD.org/pub/FreeBSD/> (ftp / <http://ftp1.fr.FreeBSD.org/pub/FreeBSD/> / rsync)
- <ftp://ftp3.fr.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp6.fr.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp6.fr.FreeBSD.org/pub/FreeBSD/> (ftp / rsync)
- <ftp://ftp7.fr.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp8.fr.FreeBSD.org/pub/FreeBSD/> (ftp)

## Germany

In case of problems, please contact the hostmaster <[de-bsd-hubs@de.FreeBSD.org](mailto:de-bsd-hubs@de.FreeBSD.org)> for this domain.

- <ftp://ftp.de.FreeBSD.org/pub/FreeBSD/> (ftp)

- <ftp://ftp1.de.FreeBSD.org/freebsd/> (ftp / <http://www1.de.FreeBSD.org/freebsd/> / <rsync://rsync3.de.FreeBSD.org/freebsd/>)
- <ftp://ftp2.de.FreeBSD.org/pub/FreeBSD/> (ftp / <http://ftp2.de.FreeBSD.org/pub/FreeBSD/> / rsync)
- <ftp://ftp4.de.FreeBSD.org/FreeBSD/> (ftp / <http://ftp4.de.FreeBSD.org/pub/FreeBSD/>)
- <ftp://ftp5.de.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp7.de.FreeBSD.org/pub/FreeBSD/> (ftp / <http://ftp7.de.FreeBSD.org/pub/FreeBSD/>)

## Greece

In case of problems, please contact the hostmaster <[hostmaster@gr.FreeBSD.org](mailto:hostmaster@gr.FreeBSD.org)> for this domain.

- <ftp://ftp.gr.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp2.gr.FreeBSD.org/pub/FreeBSD/> (ftp)

## Hong Kong

<ftp://ftp.hk.FreeBSD.org/pub/FreeBSD/> (ftp)

## Ireland

In case of problems, please contact the hostmaster <[hostmaster@ie.FreeBSD.org](mailto:hostmaster@ie.FreeBSD.org)> for this domain.

- <ftp://ftp3.ie.FreeBSD.org/pub/FreeBSD/> (ftp / rsync)

## Japan

In case of problems, please contact the hostmaster <[hostmaster@jp.FreeBSD.org](mailto:hostmaster@jp.FreeBSD.org)> for this domain.

- <ftp://ftp.jp.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp2.jp.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp3.jp.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp4.jp.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp5.jp.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp6.jp.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp7.jp.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp8.jp.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp9.jp.FreeBSD.org/pub/FreeBSD/> (ftp)

## Korea

In case of problems, please contact the hostmaster <[hostmaster@kr.FreeBSD.org](mailto:hostmaster@kr.FreeBSD.org)> for this domain.

- <ftp://ftp.kr.FreeBSD.org/pub/FreeBSD/> (ftp / rsync)
- <ftp://ftp2.kr.FreeBSD.org/pub/FreeBSD/> (ftp / <http://ftp2.kr.FreeBSD.org/pub/FreeBSD/>)

## Latvia

In case of problems, please contact the hostmaster <[hostmaster@lv.FreeBSD.org](mailto:hostmaster@lv.FreeBSD.org)> for this domain.

- <ftp://ftp.lv.FreeBSD.org/pub/FreeBSD/> (ftp / <http://ftp.lv.FreeBSD.org/pub/FreeBSD/>)

## **Lithuania**

In case of problems, please contact the hostmaster <[hostmaster@lt.FreeBSD.org](mailto:hostmaster@lt.FreeBSD.org)> for this domain.

- <ftp://ftp.lt.FreeBSD.org/pub/FreeBSD/> (ftp / <http://ftp.lt.FreeBSD.org/pub/FreeBSD/>)

## **Netherlands**

In case of problems, please contact the hostmaster <[hostmaster@nl.FreeBSD.org](mailto:hostmaster@nl.FreeBSD.org)> for this domain.

- <ftp://ftp.nl.FreeBSD.org/pub/FreeBSD/> (ftp / <http://ftp.nl.FreeBSD.org/os/FreeBSD/> / rsync)
- <ftp://ftp2.nl.FreeBSD.org/pub/FreeBSD/> (ftp)

## **New Zealand**

- <ftp://ftp.nz.FreeBSD.org/pub/FreeBSD/> (ftp / <http://ftp.nz.FreeBSD.org/pub/FreeBSD/>)

## **Norway**

In case of problems, please contact the hostmaster <[hostmaster@no.FreeBSD.org](mailto:hostmaster@no.FreeBSD.org)> for this domain.

- <ftp://ftp.no.FreeBSD.org/pub/FreeBSD/> (ftp / rsync)

## **Poland**

In case of problems, please contact the hostmaster <[hostmaster@pl.FreeBSD.org](mailto:hostmaster@pl.FreeBSD.org)> for this domain.

- <ftp://ftp.pl.FreeBSD.org/pub/FreeBSD/> (ftp)
- [ftp2.pl.FreeBSD.org](ftp://ftp2.pl.FreeBSD.org)

## **Russia**

In case of problems, please contact the hostmaster <[hostmaster@ru.FreeBSD.org](mailto:hostmaster@ru.FreeBSD.org)> for this domain.

- <ftp://ftp.ru.FreeBSD.org/pub/FreeBSD/> (ftp / <http://ftp.ru.FreeBSD.org/FreeBSD/> / rsync)
- <ftp://ftp2.ru.FreeBSD.org/pub/FreeBSD/> (ftp / <http://ftp2.ru.FreeBSD.org/pub/FreeBSD/> / rsync)
- <ftp://ftp5.ru.FreeBSD.org/pub/FreeBSD/> (ftp / <http://ftp5.ru.FreeBSD.org/pub/FreeBSD/> / rsync)
- <ftp://ftp6.ru.FreeBSD.org/pub/FreeBSD/> (ftp)

## **Saudi Arabia**

In case of problems, please contact the hostmaster <[ftpadmin@isu.net.sa](mailto:ftpadmin@isu.net.sa)> for this domain.

- <ftp://ftp.isu.net.sa/pub/ftp.freebsd.org> (ftp)

## **Slovenia**

In case of problems, please contact the hostmaster <[hostmaster@si.FreeBSD.org](mailto:hostmaster@si.FreeBSD.org)> for this domain.

- <ftp://ftp.si.FreeBSD.org/pub/FreeBSD/> (ftp)

## South Africa

In case of problems, please contact the hostmaster <[hostmaster@za.FreeBSD.org](mailto:hostmaster@za.FreeBSD.org)> for this domain.

- <ftp://ftp.za.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp2.za.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp4.za.FreeBSD.org/pub/FreeBSD/> (ftp)

## Spain

In case of problems, please contact the hostmaster <[hostmaster@es.FreeBSD.org](mailto:hostmaster@es.FreeBSD.org)> for this domain.

- <ftp://ftp.es.FreeBSD.org/pub/FreeBSD/> (ftp / <http://ftp.es.FreeBSD.org/pub/FreeBSD/>)
- <ftp://ftp3.es.FreeBSD.org/pub/FreeBSD/> (ftp)

## Sweden

In case of problems, please contact the hostmaster <[hostmaster@se.FreeBSD.org](mailto:hostmaster@se.FreeBSD.org)> for this domain.

- <ftp://ftp.se.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp2.se.FreeBSD.org/pub/FreeBSD/> (ftp / <rsync://ftp2.se.FreeBSD.org/>)
- <ftp://ftp3.se.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp4.se.FreeBSD.org/pub/FreeBSD/> (ftp / <ftp://ftp4.se.FreeBSD.org/pub/FreeBSD/> / <http://ftp4.se.FreeBSD.org/pub/FreeBSD/> / <http://ftp4.se.FreeBSD.org/pub/FreeBSD/> / <rsync://ftp4.se.FreeBSD.org/pub/FreeBSD/> / <rsync://ftp4.se.FreeBSD.org/pub/FreeBSD/>)
- <ftp://ftp6.se.FreeBSD.org/pub/FreeBSD/> (ftp / <http://ftp6.se.FreeBSD.org/pub/FreeBSD/>)

## Switzerland

In case of problems, please contact the hostmaster <[hostmaster@ch.FreeBSD.org](mailto:hostmaster@ch.FreeBSD.org)> for this domain.

- <ftp://ftp.ch.FreeBSD.org/pub/FreeBSD/> (ftp / <http://ftp.ch.FreeBSD.org/pub/FreeBSD/>)

## Taiwan

In case of problems, please contact the hostmaster <[hostmaster@tw.FreeBSD.org](mailto:hostmaster@tw.FreeBSD.org)> for this domain.

- <ftp://ftp.ch.FreeBSD.org/pub/FreeBSD/> (ftp / <ftp://ftp.tw.FreeBSD.org/pub/FreeBSD/> / [rsync / rsyncv6](rsync://ftp.tw.FreeBSD.org/pub/FreeBSD/))
- <ftp://ftp2.tw.FreeBSD.org/pub/FreeBSD/> (ftp / <ftp://ftp2.tw.FreeBSD.org/pub/FreeBSD/> / <http://ftp2.tw.FreeBSD.org/pub/FreeBSD/> / <http://ftp2.tw.FreeBSD.org/pub/FreeBSD/> / [rsync / rsyncv6](rsync://ftp2.tw.FreeBSD.org/pub/FreeBSD/))
- <ftp://ftp4.tw.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp5.tw.FreeBSD.org/pub/FreeBSD/> (ftp)

- <ftp://ftp6.tw.FreeBSD.org/pub/FreeBSD/> (ftp / <http://ftp6.tw.FreeBSD.org/> / rsync)
- <ftp://ftp7.tw.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp8.tw.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp11.tw.FreeBSD.org/pub/FreeBSD/> (ftp / <http://ftp11.tw.FreeBSD.org/FreeBSD/>)
- <ftp://ftp12.tw.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp13.tw.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp14.tw.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp15.tw.FreeBSD.org/pub/FreeBSD/> (ftp)

## Ukraine

- <ftp://ftp.ua.FreeBSD.org/pub/FreeBSD/> (ftp / <http://ftp.ua.FreeBSD.org/pub/FreeBSD/>)
- <ftp://ftp6.ua.FreeBSD.org/pub/FreeBSD/> (ftp / [http://ftp6.ua.FreeBSD.org/pub/FreeBSD](http://ftp6.ua.FreeBSD.org/pub/FreeBSD/) / rsync://ftp6.ua.FreeBSD.org/FreeBSD/)
- <ftp://ftp7.ua.FreeBSD.org/pub/FreeBSD/> (ftp)

## United Kingdom

In case of problems, please contact the hostmaster <[hostmaster@uk.FreeBSD.org](mailto:hostmaster@uk.FreeBSD.org)> for this domain.

- <ftp://ftp.uk.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp2.uk.FreeBSD.org/pub/FreeBSD/> (ftp / <http://ftp2.uk.FreeBSD.org/pub/FreeBSD/> / rsync://ftp2.uk.FreeBSD.org/ftp.freebsd.org/pub/FreeBSD/)
- <ftp://ftp3.uk.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp4.uk.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp5.uk.FreeBSD.org/pub/FreeBSD/> (ftp)

## United States of America

In case of problems, please contact the hostmaster <[hostmaster@us.FreeBSD.org](mailto:hostmaster@us.FreeBSD.org)> for this domain.

- <ftp://ftp1.us.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp2.us.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp3.us.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp4.us.FreeBSD.org/pub/FreeBSD/> (ftp / ftpv6 / <http://ftp4.us.FreeBSD.org/pub/FreeBSD/> / <http://ftp4.us.FreeBSD.org/pub/FreeBSD/>)
- <ftp://ftp5.us.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp6.us.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp8.us.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp10.us.FreeBSD.org/pub/FreeBSD/> (ftp)
- <ftp://ftp11.us.FreeBSD.org/pub/FreeBSD/> (ftp)



- <ftp://ftp13.us.FreeBSD.org/pub/FreeBSD/> (ftp / <http://ftp13.us.FreeBSD.org/pub/FreeBSD/> / rsync)
- <ftp://ftp14.us.FreeBSD.org/pub/FreeBSD/> (ftp / <http://ftp14.us.FreeBSD.org/pub/FreeBSD/>)
- <ftp://ftp15.us.FreeBSD.org/pub/FreeBSD/> (ftp)

## A.3. 匿名 CVS

### A.3.1. 概述

匿名 CVS(或人常称的 *anoncvs*)是由和 FreeBSD 附带的 CVS 工具提供的用于和远程的 CVS 代相同的一个特性。尤其是，它允许 FreeBSD 用户不需要特殊的权限访问任何一台 FreeBSD 项目的官方 *anoncvs* 服务器进行只读的 CVS 操作。要使用它，用户的位置 *CVSROOT* 环境变量指向适当的 *anoncvs* 服务器，输入 *cvs login* 命令并提供广为人知的密码“anoncvs”，然后使用 *cvs(1)* 命令像访问任何本地一样来访问它。



*cvs login* 命令把用来登录 CVS 服务器的密码存在用户的 *HOME* 目录中一个叫 *.cvspass* 的文件里。如果这个文件不存在，第一次使用 *cvs login* 的时候可能会出错。创建一个空的 *.cvspass* 文件，然后重新登录。

也可以看看 *CVSup* 和 *anoncvs* 服务器本地上提供了同样的功能，但是有各种各样的不同的场合可以影响使用相同的方式的访问。例如，*CVSup* 在网络资源利用方面更加有效，而且是到目前为止在用户之中技术上更成熟的除了成本方面。要使用 *CVSup*，在下载任何软件之前必须首先安装配置特定的客户端，而且只能用于下载相当大件的 *CVSup* 称作 *collections*。

相比之下，*anoncvs* 可以通过 CVS 模式名来从任何文件里取出任何软件并运行特定的程序（比如 *ls* 或者 *grep*）。当然，*anoncvs* 也只用于只读的 CVS 操作，所以如果用户是想用和 FreeBSD 项目共享的提供本地访问的，*CVSup* 几乎是唯一的访问。

### A.3.2. 使用匿名 CVS

配置 *cvs(1)* 使用匿名 CVS 用户可以访问的固定 *CVSROOT* 环境变量指向 FreeBSD 项目的 *anoncvs* 服务器之一。到此写作为止，下面的服务器都是可用的：

- 法国: *:pserver:anoncvs@anoncvs.fr.FreeBSD.org:/home/ncvs* (使用 *pserver* 模式，用 *cvs login* 配合口令“anoncvs”来登录。如果使用的是 *ssh*，不需要口令。)
- 台湾地区: *:pserver:anoncvs@anoncvs.tw.FreeBSD.org:/home/ncvs* (使用 *pserver* 模式，用 *cvs login* 配合口令“anoncvs”来登录。如果使用的是 *ssh*，不需要口令。)

```
SSH2 HostKey: 1024 02:ed:1b:17:d6:97:2b:58:5e:5c:e2:da:3b:89:88:26
/etc/ssh/ssh_host_rsa_key.pub
SSH2 HostKey: 1024 e8:3b:29:7b:ca:9f:ac:e9:45:cb:c8:17:ae:9b:eb:55
/etc/ssh/ssh_host_dsa_key.pub
```

- 美国: *anoncvs@anoncvs1.FreeBSD.org:/home/ncvs* (使用 *ssh*，使用版本 2，不需要口令。)

```
SSH2 HostKey: 2048 53:1f:15:a3:72:5c:43:f6:44:0e:6a:e9:bb:f8:01:62
/etc/ssh/ssh_host_dsa_key.pub
```

因 CVS 库上允许“检出”曾经存在的（或者，某些情况下将会存在）FreeBSD 源代码的任何版本，您需要熟悉 `cvs(1)` 的版本 (`-r`) 参数，以及在 FreeBSD 代码库中可用的。

有主干，修修补补和分支。修修补补特指一个特定的修补版本。含始是不的。分支，一方面，指代特定分支的最新修，因分支不及特定的修版本，它明天所代表的含就可能和今天的不同。

CVS 包括了用可能感兴趣的修。注意，些并不用于 Ports Collection，因它并不包含多个分支。

当您指定一个分支，通常会得到那个分支的文件的最新版本。如果您希望得到一些旧的版本，可以用 `-D date` 制定一个日期。察看 `cvs(1)` 手册了解更多。

### A.3.3. 示例

在之前烈建通 `cvs(1)` 的手册，里有一些的例子来展示如何使用匿名 CVS：

例 45. 从 `-CURRENT` 出些西 (`ls(1)`)：

```
% setenv CVSROOT :pserver:anoncvs@anoncvs.tw.FreeBSD.org:/home/ncvs
% cvs login
>在提示符，入任意密 "password".
% cvs co ls
```

例 46. 通 SSH 出整个 `src/` 代码：

```
% cvs -d anoncvs@anoncvs1.FreeBSD.org:/home/ncvs co src
The authenticity of host 'anoncvs1.freebsd.org (216.87.78.137)' can't be
established.
DSA key fingerprint is 53:1f:15:a3:72:5c:43:f6:44:0e:6a:e9:bb:f8:01:62.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'anoncvs1.freebsd.org' (DSA) to the list of known
hosts.
```

例 47. 出 `8-STABLE` 分支中的 `ls(1)` 版本：

```
% setenv CVSROOT :pserver:anoncvs@anoncvs.tw.FreeBSD.org:/home/ncvs
% cvs login
在提示符，入任意密 "password".
% cvs co -rRELENG_8 ls
```

例 48. 创建 `ls(1)` 的快照列表(用标准的 `diff`)

```
% setenv CVSROOT :pserver:anoncvs@anoncvs.tw.FreeBSD.org:/home/ncvs
% cvs login
在提示符, 输入任意密码 "password"。
% cvs rdiff -u -rRELENG_8_0_0_RELEASE -rRELENG_8_1_0_RELEASE ls
```

例 49. 列出可以使用的其它的模块名：

```
% setenv CVSROOT :pserver:anoncvs@anoncvs.tw.FreeBSD.org:/home/ncvs
% cvs login
在提示符, 输入任意密码 "password"。
% cvs co modules
% more modules/modules
```

### A.3.4. 其他资源

下面附加的资源可能帮助你学习 CVS 有帮助：

- [CVS 教程](#)，来自加州州立理工大学。
- [CVS 主页](#)，CVS 用户和支持社区。
- [CVSweb](#) 是 FreeBSD 官方的 CVS web 界面。

## A.4. 使用 CTM

CTM 是保持程序目录和中央服务器目录同步的一种方法。它被用于 FreeBSD 的源代码，当然其他人随着推移也会将它用于其他目的。当前几乎没有，也或者只有很少的文档描述构建 deltas 的，所以如果你希望使用 CTM 去做其它事情，请系 [ctm-users-desc](#) 邮件列表了解更多信息。

### A.4.1. 为什么我要使用 CTM?

CTM 会提供一个 FreeBSD 源代码的本地副本。代码有很多的 "flavors" 可用。不管你是希望跟踪完整的 CVS 还是一个分支，CTM 都会提供信息。如果你是 FreeBSD 上的一个活跃的开发者，但是缺乏或者不存在 TCP/IP 连接，或者只是希望把快照自给送，CTM 就是适合的。对于最新的分支，将会每天得到三个以上的 deltas。然而，通常考虑通邮件来自给送。升级的大小是保持尽可能的小。通常小于 5K，也偶尔(十分之一可能)会有 10-50K，也不见得有个大的 100K+ 甚至更大的。

你也需要自己了解直接和代码而不是发行版本打交道的各种警告。情况会很显著，如果你有了 "current" 代码的。强烈建议和 [FreeBSD 保持同步](#)。

### A.4.2. 使用 CTM 我需要做什么？

你需要知道：CTM 程序，有初始的 deltas 来 feed it(到 "current" 邮件)。

CTM 程序从版本 2.0 发布以来一直是 FreeBSD 的一部分了，如果你安装了源代码副本的，它位于 `/usr/src/usr.sbin/ctm`。

CTM 的 "deltas" 可以有多种方式，FTP 或者 email。如果你有普通的 Internet 的 FTP 限制，那下面的 FTP 站点支持 CTM：

<ftp://ftp.FreeBSD.org/pub/FreeBSD/CTM/>

或者看看一小映像。

FTP 相关的目录并取得 README 文件，从那里开始。

如果你希望通过 email 得到 deltas：

一个 CTM 分发列表。[ctm-src-cur-desc](#) 支持最新的分支。[ctm-src-7-desc](#) 支持 7.X 行分支，等等。(如果你不知道如何邮件列表，点上面的列表名或者到 <https://lists.freebsd.org> 点你想要的列表。列表包含了所有必要的指引。)

当你开始接收到邮件中的 CTM 升级，你可以使用 `ctm_rmail` 程序来解并用它。事实上如果你想程序以全自行的形式运行的，你可以通过在 `/etc/aliases` 中配置直接使用 `ctm_rmail` 程序。看 `ctm_rmail` 手册了解更多。



不管使用什么方法得到 CTM deltas，你都看看 [ctm-announce-desc](#) 邮件列表。以后会有你独的地方提交有关 CTM 系统的操作的公告。点上面的邮件列表名并按照指示邮件列表。

### A.4.3. 第一次使用 CTM

在你开始使用 CTM delta 之前，你需要得一个起始点。

首先你得确定你已经有了什么。任何人都可以从一个"空"目开始。你必须用一个初始的 "空的" delta 来开始的 CTM 支持。曾有的便利些 "起始" deltas 被有意的通过 CD 来进行，然而你在已不做了。

因时代有数十兆字节，更喜从手上已有的西开始。如果你有一 `-RELEASE` CD 光盘，你可以从里面复制或解一个初始代出来。这会省非常多的数据量。

你会看到些 "初始的" deltas 名字的数字后面都有个 X (比如 `src-cur.3210XEmpty.gz`)。后面加一个 X 的符合的初始 "seed" 的由来。Empty 是一个空目。通常一个基本的从 Empty 开始的由 100 个 deltas 成。便一下，他都很大！70 到 80 兆字节的 gzip 的数据于 XEmpty deltas 是很平常的。

一旦你已经定了一个基本的 delta 开始，就需要比个数高的所有的 delta。

### A.4.4. 在你的日常生活中使用 CTM

要用 deltas，你的输入：

```
# cd /where/ever/you/want/the/stuff
# ctm -v -v /where/you/store/your/deltas/src-xxx.*
```

CTM 能理解被 gzip 的 deltas，所以你不需要先 gunzip 他，可以省磁盘空。

除非你得整个工程非常可靠，CTM 不会涉及到你的代码的。你也可以使用 `-c` 来校验 `delta`，CTM 就不会涉及到；它会只校验 `delta` 的完整性看看是否可以安全的用于你的当前代码。

CTM 有其他的一些参数，看手册或者源代码了解更多信息。

真的就是全部的事情了。每次得到一个新的 `delta`，就通过 CTM 运行它来保证你的代码是最新的。

如果某些 `deltas` 很重新下载的不要删除它。有些东西坏掉的时候你会想到保留它的。即使只有你，也考虑使用 `fdwrite` 来做一份副本。

#### A.4.5. 支持本地的

作为一名爱好者喜欢，改变代码中的文件。CTM 用一种受限的方式支持本地修改：再文件 `foo` 存在之前，首先 `foo.ctm`。如果这个文件存在，CTM 会操作它而不是 `foo`。

运行我提供了一种的方式来支持本地的改：只要控制修改的文件并用 `.ctm` 的后重新命名。然后就可以自由的修改代码了，CTM 会更新 `.ctm` 文件到最新版本。

#### A.4.6. 其他有趣的 CTM

##### A.4.6.1. 正的输出些将被更新

可以指定列表，CTM 可以做到，在你的代码上使用 CTM 的 `-l`。

很有用如果你想保存改变的日志，`pre-` 或者 `post-` 用各种格式处理修改的文件的，或者你是想感受一下孩子般的狂。

##### A.4.6.2. 在升前制作

有可能想将要被 CTM 升级所改的所有文件。

指定 `-B backup-file` 会导致 CTM 将要被指定的 CTM `delta` 改的所有文件到 `backup-file`。

##### A.4.6.3. 限定受升级影响的文件

有可能限定一个指定的 CTM 升级的感兴趣，也有可能想知道从一系列 `deltas` 中解一部分文件。

可以通过使用 `-e` 和 `-x` 指定列表式来控制 CTM 即将之操作的文件列表。

例如，要从保存的 CTM `deltas` 集里解出一个最新的 `lib/libc/Makefile` 文件，运行个命令：

```
# cd /where/ever/you/want/to/extract/it/
# ctm -e '^lib/libc/Makefile' ~ctm/src-xxx.*
```

于一个在 CTM `delta` 中指定的文件，`-e` 和 `-x` 按照命令行指定的顺序用。文件只有在所有的 `-e` 和 `-x` 被用之后合格之后才能被 CTM 操作。

#### A.4.7. CTM 未来的

其中几：

- 在 CTM 中使用一些方式，来允许冒充的 CTM 丁。
- 整理 CTM 的，它得乱而反直了。

#### A.4.8. 镜像

也有一系列的 **ports** collection 的 deltas，但是人它的致没有那高。

#### A.4.9. CTM 镜像

CTM/FreeBSD 可以在下面的镜像站点通匿名 FTP 下。如果通匿名 FTP 取 CTM，着使用一个近的站点。

如果有，系 [ctm-users-desc](#) 件列表。

加利福尼亚州，旧金山湾区，官方源代

- <ftp://ftp.FreeBSD.org/pub/FreeBSD/development/CTM/>

南非，旧的 **deltas** 的服器

- <ftp://ftp.za.FreeBSD.org/pub/FreeBSD/CTM/>

中国台湾

- <ftp://ctm.tw.FreeBSD.org/pub/FreeBSD/development/CTM/>
- <ftp://ctm2.tw.FreeBSD.org/pub/FreeBSD/development/CTM/>
- <ftp://ctm3.tw.FreeBSD.org/pub/FreeBSD/development/CTM/>

如果在附近不到像或者像不完整，着使用索引比如 [alltheweb](#)。

## A.5. 使用 CVSup

### A.5.1. 概述

CVSup 是一个用于从程服器主机上的主 CVS 布和升源代的件包。FreeBSD 的源代在加利福尼亚州一台主服器的 CVS 里。有了 CVSup，FreeBSD 用可以很容易的保持他自己的源代更新。

CVSup 使用所的升 *pull* 模式。在 *pull* 模式下，客端在需要的时候向服器端求更新。服器被的等待客端的升求。因此所有的升都是客端起的。服器决不会送未求的升。用必手行 CVSup 客端取更新，或者置一个 **cron** 作来它以固定的律自行。

CVSup 用大写字母写正是表示，代表了完整的件包。它的主要件是行在个用机器上的客端 **cvsup**，和行在个 FreeBSD 像站点上的服器端 **cvsupd**。

当 FreeBSD 文和件列表，可能会看 **sup**。Sup 是 CVSup 的前身，有着相似的目的。CVSup 使用很多和 **sup** 相同的方式，而且，它是用使用和 **sup** 的兼容的配置文件。Sup 已不再被 FreeBSD 目使用了，因 CVSup 既快又有更好的活性。



csup 是用 C 语言写的 CVSup 库的重写。它最大的好处是，这个程序更快一些，并且也不需要依赖于 Modula-3 语言，因此也就不需要安装后者。另外，你可以直接使用它，因为它基本是系统的一部分。假如你决定使用 csup，你可以跳过安装 CVSup 库，并在文章中余下部分提到的 CVSup 改用 csup。

## A.5.2. 安装

安装 CVSup 最简的方式就是使用 FreeBSD [packages collection](#) 中提供的 [net/cvsup](#) 包。如果你想从源代码构建 CVSup，你可以使用 [net/cvsup](#) port。但是先警告一下：[net/cvsup](#) port 依赖于 Modula-3 系统，会花相当的磁盘空间来下载。



如果想在没有安装 Xorg 的计算机，例如服务器上使用 CVSup，只能使用不包含 CVSupGUI 的 [net/cvsup-without-gui](#)。

## A.5.3. CVSup 配置

CVSup 的操作被一个叫做 `supfile` 的配置文件所控制。在目录 [/usr/shared/examples/cvsup/](#) 下面有一些示例的 `supfiles`。

`supfile` 中的信息解答了 CVSup 下面的几个问题：

- 你想接收哪些文件？
- 你想要它的哪个版本？
- 你想从哪取它？
- 你想把它放在自己机器的什么地方？
- 你想把它的状态文件放在哪？

在下面的章节里，我将通顺依次回答这些问题来构建一个典型的 `supfile` 文件。首先，我将描述一下 `supfile` 的整体构成。

`supfile` 是个文本文件。注应用 `#` 符号，至行尾有效。空行和只包含注释的行会被忽略。

每个保留行描述一批用希望接收的文件。行以 `"collection"`，由服务器端定义的合理的文件分，的名字。 `collection` 的名字告诉服务器想要的文件。 `collection` 名字结束或者有更多的字段，用空格分隔。这些字段回答了上面列出的问题。字段类型有：字符串字段和数字字段。字符串字段由独立的字符串组成，比如， `delete` 或者 `compress`。数字字段也用字符串，数字后面跟 `=` 和第二个数字而没有空格。例如， `release=cvs` 是一个数字字段。

一个典型的 `supfile` 往往接收多于一个的 `collection`。构建 `supfile` 的一种方式明确的每个 `collection` 指定相关的字段。然而，这使得 `supfile` 的行变得特，很不方便，因为 `supfile` 中的所有 `collection` 的大部分字段都是相同的。CVSup 提供了一个默认机制来避免这些问题。用特定的 `collection` 名 `*default` 的行可以被用来设置和 `supfile` 中随后的 `collection` 中的默认。默认可以通过每个 `collection` 自身指定不同的值来覆盖，也可以在 `mid-supfile` 中通附加的 `*default` 行改变或补充。

知道了这些，我可以在就可以开始构建一个用于接收和升 [FreeBSD-CURRENT](#) 主源代码的 `supfile` 文件了。

- 你想接收哪些文件？



通过 CVSup 可用的文件可能叫做 "collections" 的名称。一些可用的 collection 在 [随后的章节](#) 中描述。在这个例子里，我希望接收 FreeBSD 系的完整的主代码。有一个独特的大 collection `src-all` 我完成它。建我的 supfile 的第一行，我列出一些 collection，每一行（在这个例子里，只有一行）：

```
src-all
```

- 想要他的哪个版本？

通过 CVSup，你可以接收曾经存在的源代码的任何版本。是有可能的，因为 cvsupd 服务器直接通过 CVS 工作，那包含了所有的版本。可以用 `tag=` 和 `date=` 字段指定一个想要的版本。



仔细的正指定任何 `tag=` 字段。有一些 tag 只特定的 collection 文件合法。如果你指定了一个不正的或者写的 tag，CVSup 会删除可能不想删除的文件。特地，`ports-*` collection 只使用 `tag=.`。

`tag=` 字段在中表示一个符号。有，修和分支。修代表一个特定的修版本。它的含是一成不变的。分支，一方面，代表特定上特定的最新修。因分支不代表一个特定的修版本，它明天的含就可能和今天的有所不同。

CVS 包含了用可能感兴趣的分支。当在 CVSup 的配置文件中指定时，必用 `tag=` (`RELENG_8` 会成 `tag=RELENG_8`)。住只有 `tag=.` 可以用于 Ports Collection。



注意像看到的那正的正入名。CVSup 不能辨合法和不法。如果你写了名，CVSup 会像指定了一个没有任何文件的合法一工作，那会删除已存在的代。

当你指定一个分支的时候，通常会收到上文件的最新版本。如果你希望接收一些的版本，可以通过用 `date=` 字段指定一个日期来做到。 [cvsup\(1\)](#) 手册解了如何做。

于我的示例来，我希望接收 FreeBSD-CURRENT。我在我自己的 supfile 的添加行：

```
*default tag=.
```

有一个重要的特例，如果既没指定 `tag=` 字段也没指定 `date=` 字段的情况。情况下，会收到直接来自于服务器 CVS 的真 RCS 文件，而不是某一特定版本。人一般喜操作模式。通过在他的系上一自身的副本，他可以修史以及文件去的版本。然而，个好是以大量的磁盘空代价的。

- 想从里取他？

我使用 `host=` 字段来告诉 cvsup 从里取更新。任何一个 CVSup 像站点都可以，然一个比近的站点。在这个例子里我将使用一个虚的 FreeBSD 布站点， `cvsup99.FreeBSD.org`：

```
*default host=cvsup99.FreeBSD.org
```



需要在运行 CVSup 之前把该个改成一个已经存在的站点。在任何 `cvsup` 运行的特定时刻，都可以在命令行上使用 `-h hostname` 来覆盖主机配置。

- 想把它放在自己机器的什么地方？

`prefix=` 字段告诉 `cvsup` 把接收的文件放在哪里。在这个例子中，我把源代码文件直接放在我的主源代码目录，`/usr/src`。该目录已包含在我接收的 collection 里了，所以正确的写法是：

```
*default prefix=/usr
```

- `cvsup` 在哪里存储它的状态文件？

CVSup 客户端在被叫做 "base" 的目录中存储了几个状态文件。这些文件帮助 CVSup 更有效的工作，通过跟踪已经接收到哪些更新的方式。我将使用标准的 base 目录，`/var/db`：

```
*default base=/var/db
```

如果该的 base 目录不存在，请在最好创建它。如果 base 目录不存在，`cvsup` 客户端会拒绝工作。

- 其他的 supfile 配置：

在 supfile 中有一些其他可能需要介绍一下：

```
*default release= cvs delete use-rel-suffix compress
```

`release=cvs` 指示服务器从 FreeBSD 的主 CVS 库中获取信息。事实上是的，但是也有可能超出该库的范围。

`delete` 让 CVSup 限制删除文件。该是指定一个，让 CVSup 可以保证的源代码完全更新。CVSup 很小心的只删除那些不再依赖的文件。该有任何额外的文件会被严格的保留。

`use-rel-suffix` 是 ... 不可思议的。如果你真的想了解它，请看 [cvsup\(1\)](#) 手册。否则，就指定而不用担心该。

`compress` 用 gzip 格式的信道。如果你的网络接口是 T1 或者更快，可能不想使用该。否则，它非常有帮助。

- 把它放在一起：

这是我的示例的完整 supfile 文件：

```
*default tag=.
*default host=cvsup99.FreeBSD.org
*default prefix=/usr
*default base=/var/db
*default release=cvs delete use-rel-suffix compress

src-all
```

#### A.5.3.1. refuse 文件

像上面提到的，CVSup 使用一种 *pull* 方法。基本上，这意味着要连接到 CVSup 服务器，服务器说：“有些可能下载的……”，然后客户端反问：“好，我要一个，一个，一个，一个。”在默认的配置中，CVSup 客户端会取回在配置文件中指定的 collection 和它的文件。然而，并不是你想要的，尤其是在同 doc, ports, 或者 www 中 - 大部分人都不能阅读四或者五语言，因此他不需要下载特定语言的文件。如果在 CVSup Ports Collection, 可以通过单独指定一个 collection 来避免一个 (比如, *ports-astrology*, *ports-biology*, 等等取代它的说明 *ports-all*)。然而，因 doc 和 www 中没有特定语言的 collection, 必须使用 CVSup 许多好的特性之一：refuse 文件。

refuse 文件本上是告诉 CVSup 它不能从 collection 中取得某些文件；简单说，它告诉客户端拒绝来自服务器的特定的文件。refuse 文件可以在 base/sup/ 中找到(或者，如果没有，创建一个)。base 在 supfile 中定义；默认情况下，base 就是 /var/db, 这意味着默认的 refuse 文件就是 /var/db/sup/refuse。

refuse 文件的格式很简单；它包含不希望下载的文件和目录名。例如，如果你除了英语和德语之外不会任何其他语言，而且也不打算阅读德文的文档翻译版本，你可以把下面一些放在你的 refuse 文件里：

```
doc/bn_*
doc/da_*
doc/de_*
doc/el_*
doc/es_*
doc/fr_*
doc/hu_*
doc/it_*
doc/ja_*
doc/mn_*
doc/nl_*
doc/no_*
doc/pl_*
doc/pt_*
doc/ru_*
doc/sr_*
doc/tr_*
doc/zh_*
```

等等其他语言(可以通过 [FreeBSD CVS](#) 得到完整的列表)。

有一个非常有用的特性，那些慢速连接或者要他通过 Internet 连接支付的用户就可以节省宝贵的时间因为他不再需要下载那些从来不用文件。要了解 refuse 文件的更多信息以及其它 CVSup 的优雅的特性，看看它的

手册。

### A.5.4. 运行 CVSup

现在准备升了。命令很：

```
# cvsup supfile
```

supfile 的位置当然就是建的 supfile 文件名。如果在 X11 下面行，cvsup 会示一个有一些可以做平常事情的按的 GUI 口。按 **[go]** 按，然后看着它行。

在这个例子里将要升目前的 /usr/src，将需要用 root 来行程序，cvsup 有需要的限来更新的文件。建了配置文件，又从来没有使用个程序，不安是可以理解的。有一个的方法不改当前的文件来做一次性的行。只要在方便的地方建一个空目，并在命令行上作一个外的参数明：

```
# mkdir /var/tmp/dest
# cvsup supfile /var/tmp/dest
```

指定的目会作所有文件更新的目的路径。CVSup 会在 /usr/src 中的文件，但是不会修改或删除。任何文件更新都会被放到 /var/tmp/dest/usr/src 里了。在方式下行 CVSup 也会把它的 base 目状文件保持原。些文件的新版本 会被写到指定的目。因有 /usr/src 目的限，所以行一次性的行甚至不需要使用 root 用。

如果没有行 X11 或者不喜 GUI，当行 cvsup 的时候需要在命令行添加个：

```
# cvsup -g -L 2 supfile
```

**-g** 告 CVSup 不要使用 GUI。如果没在行 X11 个是自的，否则必指定它。

**-L 2** 告 CVSup 出所有正在升的文件的。有三个等可以，从 **-L 0** 到 **-L 2**。默是 0，意味着除了消息 什么都不出。

有好多其它的可用。想要一个短列表，入 **cvsup -H**。要看更的描述，看手册。

一旦升工作的方式意了，就可以使用 [cron\(8\)](#) 来安排的行 CVSup。很然的，不 CVSup 通 [cron\(8\)](#) 行的候使用它的 GUI。

### A.5.5. CVSup 文件 collection

CVSup 可用的文件 collection 是分。有几个大的 collection，然后它有分成更小的子 collection。接收一个大的 collection 等同于接收它的一个子 collection。collection 的等系在下面列表中通的使用 反映出来。

最常用的 collection 是 **src-all**，和 **ports-all**。其它的 collection 只被有着特定目的的小部分人使用，有些站点可能不全部支持。

### cvsvall release=cvsv

FreeBSD 主 CVS 库，包含 密 系 的代 。

### distrib release=cvsv

FreeBSD 行版本和像相 的文件。

### doc-all release=cvsv

FreeBSD 使用手册和其它文 的源代 。其中不包含 FreeBSD web 站点的文件。

### ports-all release=cvsv

FreeBSD Ports Collection。



如果 不想升 全部的 **ports-all**(整个 ports 库)，而只是使用下面列出的一个子集， 保 是升 了 **ports-base** 子 collection！无 何 在 ports 库建下 造有所改 的 候都会通 **ports-base** 表 出来，事 上某些 改 会很快的被" "的 ports 使用，因此，如果 只升 了 " "的 ports 而他 使用了一些新的特性， 就有 大的可能 会因一些神秘的 信息而失 。 情况下非常快速的要做的事情 就是 保 的 **ports-base** 子 collection 更新到 最新。



要自行 建 ports/INDEX， 必 接受 **ports-all** (完整的 ports tree)。在部分 ports tree 上 建 ports/INDEX 是不被支持的。 参 [FAQ](#)。

### ports-accessibility release=cvsv

用以 助残疾用 的 件。

### ports-arabic release=cvsv

阿拉伯 支持。

### ports-archivers release=cvsv

存 工具。

### ports-astro release=cvsv

天文相 的 ports。

### ports-audio release=cvsv

声音支持。

### ports-base release=cvsv

Ports Collection 库建下部 造 - 位于 /usr/ports 的 Mk/ 和 Tools/ 子目 的各 各 的文件。



看**重要警告**： 是更新 个 子 collection，无 更新 FreeBSD Ports Collection 的任何部分的 候！

### ports-benchmarks release=cvsv

基准。

**ports-biology release=cvs**

生物学。

**ports-cad release=cvs**

计算机辅助工具。

**ports-chinese release=cvs**

中文语言支持。

**ports-comms release=cvs**

通信软件。

**ports-converters release=cvs**

字符转换。

**ports-databases release=cvs**

数据库

**ports-deskutils release=cvs**

计算机前常出在面上的西。

**ports-devel release=cvs**

工具。

**ports-dns release=cvs**

DNS 相关件。

**ports-editors release=cvs**

器

**ports-emulators release=cvs**

其它操作系的模拟器

**ports-finance release=cvs**

，金融相关用程序。

**ports-ftp release=cvs**

FTP 客户端和服务端工具。

**ports-games release=cvs**

游

**ports-german release=cvs**

德支持。

**ports-graphics release=cvs**

形像工具。

**ports-hebrew release=cv**s

希伯来语支持。

**ports-hungarian release=cv**s

匈牙利语支持。

**ports-irc release=cv**s

Internet 多用户交互(IRC)工具。

**ports-japanese release=cv**s

日语支持。

**ports-java release=cv**s

Java™ 工具。

**ports-korean release=cv**s

韩国语支持。

**ports-lang release=cv**s

编程语言。

**ports-mail release=cv**s

邮件软件。

**ports-math release=cv**s

数学软件。

**ports-misc release=cv**s

杂项工具。

**ports-multimedia release=cv**s

多媒体软件。

**ports-net release=cv**s

网络软件。

**ports-net-im release=cv**s

即时消息软件。

**ports-net-mgmt release=cv**s

网管软件。

**ports-net-p2p release=cv**s

对等网 (peer to peer network) 应用。

**ports-news release=cv**s

USENET 新闻软件。

**ports-palm release=cvs**

Palm™ 系列软件支持。

**ports-polish release=cvs**

波兰语支持。

**ports-ports-mgmt release=cvs**

用于管理 ports 和软件包的工具。

**ports-portuguese release=cvs**

葡萄牙语支持。

**ports-print release=cvs**

打印软件。

**ports-russian release=cvs**

俄语支持。

**ports-science release=cvs**

科学计算。

**ports-security release=cvs**

安全工具。

**ports-shells release=cvs**

命令行 shell。

**ports-sysutils release=cvs**

系统实用工具。

**ports-textproc release=cvs**

文本处理工具(不包含页面出版)。

**ports-ukrainian release=cvs**

乌克兰语支持。

**ports-vietnamese release=cvs**

越南语支持。

**ports-www release=cvs**

万维网(WWW)相关软件。

**ports-x11 release=cvs**

支持 X window 系统的 ports。

**ports-x11-clocks release=cvs**

X11 时钟。

**ports-x11-drivers release=cvs**

X11 窗口程序。

**ports-x11-fm release=cvs**

X11 文件管理器。

**ports-x11-fonts release=cvs**

X11 字体和字体工具。

**ports-x11-toolkits release=cvs**

X11 工具包。

**ports-x11-servers release=cvs**

X11 服务器。

**ports-x11-themes release=cvs**

X11 主题。

**ports-x11-wm release=cvs**

X11 窗口管理器。

**projects-all release=cvs**

FreeBSD 内部项目的代码。

**src-all release=cvs**

FreeBSD 主代码，包含密码系统的代码。

**src-base release=cvs**

/usr/src 中的各式各样的文件。

**src-bin release=cvs**

实用模式下可能用到的实用工具 (/usr/src/bin)。

**src-cddl release=cvs**

采用了 CDDL 授权的实用工具和函数 (/usr/src/cddl)。

**src-contrib release=cvs**

FreeBSD 项目之外的工具和库，通常在 FreeBSD 中不作修改 (/usr/src/contrib)。

**src-crypto release=cvs**

FreeBSD 项目之外的密码系统工具和库，通常在 FreeBSD 中不作修改 (/usr/src/crypto)。

**src-eBones release=cvs**

Kerberos 和 DES (/usr/src/eBones)。目前的 FreeBSD 中不再使用使用。

**src-etc release=cvs**

系统配置文件 (/usr/src/etc)。



**src-games release=cvs**

游戏 (/usr/src/games)。

**src-gnu release=cvs**

GNU 公共可执行的工具 (/usr/src/gnu)。

**src-include release=cvs**

头文件 (/usr/src/include)。

**src-kerberos5 release=cvs**

Kerberos5 安全包 (/usr/src/kerberos5)。

**src-kerberosIV release=cvs**

KerberosIV 安全包 (/usr/src/kerberosIV)。

**src-lib release=cvs**

库 (/usr/src/lib)。

**src-libexec release=cvs**

通常被其它程序调用的系统程序 (/usr/src/libexec)。

**src-release release=cvs**

生成 FreeBSD 版本必需的文件 (/usr/src/release)。

**src-rescue release=cvs**

用于紧急修复的静态链接的程序；参见 [rescue\(8\)](#) (/usr/src/rescue)。

**src-sbin release=cvs**

系统模式的系统工具 (/usr/src/sbin)。

**src-secure release=cvs**

密码相关和命令 (/usr/src/secure)。

**src-share release=cvs**

跨多个平台的共享的文件 (/usr/src/share)。

**src-sys release=cvs**

内核 (/usr/src/sys)。

**src-sys-crypto release=cvs**

内核密码系统代码 (/usr/src/sys/crypto)。

**src-tools release=cvs**

FreeBSD 的各自各异的工具 (/usr/src/tools)。

**src-usrbin release=cvs**

用户工具 (/usr/src/usr.bin)。

**src-usrsbin release=cvs**

系统工具 (/usr/src/usr.sbin)。

**www release=cvs**

FreeBSD WWW 站点的源代码。

**distrib release=self**

CVSup 服务器的配置文件。用于 CVSup 镜像站点。

**gnats release=current**

GNATS bug 跟踪数据。

**mail-archive release=current**

FreeBSD 邮件列表存档。

**www release=current**

管理的 FreeBSD WWW 站点文件(不是源文件)。用于 WWW 镜像站点。

## A.5.6. 更多信息

CVSup FAQ 以及关于 CVSup 的其他信息，看看 [CVSup 主页](#)。

多数与 FreeBSD 有关的 CVSup 项目会在 [FreeBSD 技术邮件列表](#) 进行。每个项目的新版本会在那里和 [FreeBSD 公告邮件列表](#) 公布。

如果你对 CVSup 有任何疑问，或希望提交 bug 报告，请参考 [CVSup FAQ](#)。

## A.5.7. CVSup 站点

FreeBSD 的 [CVSup](#) 服务器运行于下列站点：

## A.6. CVS 简介

当使用 `cvs` 或者 `CVSup` 获取和升级源代码的时候，必须指定一个修订号。修订号代表 FreeBSD 的一个特定分支，或者一个特定的点。第一个叫做 "分支号"，第二个叫做 "版本号"。

### A.6.1. 分支号

所有项目，除了 **HEAD** (它是个合法点) 以外，只用于 `src/`、`ports/`、`doc/`，和 `www/` 没有分支。

#### HEAD

主分支的符号名，或者 FreeBSD-CURRENT。当没有指定修订版本的的时候也是默认的。

在 CVSup 里，它通常通一个 `.` 来反映出来(不是点，而是一个 `.` 字符)。



在 CVS 里，当没有修订号指定它是默认的。在一台 STABLE 机器上检出或者升级到 CURRENT 源代码通常不是一个好主意，除非它是你的本意。

## **RELENG\_8**

是 FreeBSD-8.X 的分支，也被称作 FreeBSD 8-STABLE。

## **RELENG\_8\_2**

是 FreeBSD-8.2 的行版分支，只用于安全公告，以及其他重要更新。

## **RELENG\_8\_1**

FreeBSD-8.1 的行版分支，只用于安全公告，以及其他重要更新。

## **RELENG\_8\_0**

FreeBSD-8.0 的行版分支，只用于安全公告，以及其他重要更新。

## **RELENG\_7**

是 FreeBSD-7.X 的分支，也被称作 FreeBSD 7-STABLE。

## **RELENG\_7\_4**

FreeBSD-7.4 的行版分支，只用于安全公告，以及其他重要更新。

## **RELENG\_7\_3**

FreeBSD-7.3 的行版分支，只用于安全公告，以及其他重要更新。

## **RELENG\_7\_2**

FreeBSD-7.2 的行版分支，只用于安全公告，以及其他重要更新。

## **RELENG\_7\_1**

FreeBSD-7.1 的行版分支，只用于安全公告，以及其他重要更新。

## **RELENG\_7\_0**

FreeBSD-7.0 的行版分支，只用于安全公告，以及其他重要更新。

## **RELENG\_6**

是 FreeBSD-6.X 的分支，也被称作 FreeBSD 6-STABLE。

## **RELENG\_6\_4**

FreeBSD-6.4 的行版分支，只用于安全公告，以及其他重要更新。

## **RELENG\_6\_3**

FreeBSD-6.3 的行版分支，只用于安全公告，以及其他重要更新。

## **RELENG\_6\_2**

FreeBSD-6.2 的行版分支，只用于安全公告，以及其他重要更新。

## **RELENG\_6\_1**

FreeBSD-6.1 的行版分支，只用于安全公告，以及其他重要更新。

## **RELENG\_6\_0**

FreeBSD-6.0 的行版分支，只用于安全公告，以及其他重要更新。

## **RELENG\_5**

是 FreeBSD-5.X 的分支， 也被称作 FreeBSD 5-STABLE。

## **RELENG\_5\_5**

FreeBSD-5.5 安全分支。 只被安全公告和其它重要更新使用。

## **RELENG\_5\_4**

FreeBSD-5.4 安全分支。 只被安全公告和其它重要更新使用。

## **RELENG\_5\_3**

FreeBSD-5.3 安全分支。 只被安全公告和其它重要更新使用。

## **RELENG\_5\_2**

FreeBSD-5.2 和 FreeBSD-5.2.1 的安全分支， 只被安全公告和其它重要更新使用。

## **RELENG\_5\_1**

FreeBSD-5.1 的分支， 只被安全公告和其它重要更新使用。

## **RELENG\_5\_0**

FreeBSD-5.0 的分支， 只被安全公告和其它重要更新使用。

## **RELENG\_4**

FreeBSD-4.X 分支， 也被叫做 FreeBSD-STABLE。

## **RELENG\_4\_11**

FreeBSD-4.11 安全分支。 只被安全公告和其它重要更新使用。

## **RELENG\_4\_10**

FreeBSD-4.10 安全分支。 只被安全公告和其它重要更新使用。

## **RELENG\_4\_9**

FreeBSD-4.9 安全分支。 只被安全公告和其它重要更新使用。

## **RELENG\_4\_8**

FreeBSD-4.8 安全分支。 只被安全公告和其它重要更新使用。

## **RELENG\_4\_7**

FreeBSD-4.7 安全分支。 只被安全公告和其它重要更新使用。

## **RELENG\_4\_6**

FreeBSD-4.6 和 4.6.2 的安全分支。 只被安全公告和其它重要更新使用。

## **RELENG\_4\_5**

FreeBSD-4.5 安全分支。 只被安全公告和其它重要更新使用。

## **RELENG\_4\_4**

FreeBSD-4.4 安全分支。 只被安全公告和其它重要更新使用。

## RELENG\_4\_3

FreeBSD-4.3 安全分支。 只被安全公告和其它重要更新使用。

## RELENG\_3

FreeBSD-3.X 的分支， 也被叫做 3.X-STABLE。

## RELENG\_2\_2

FreeBSD-2.2.X 的分支， 也被叫做 2.2-STABLE。 这个分支基本上已死了。

## A.6.2. 版本

当一个特定的 FreeBSD 版本发行， 一些代表了一个指定的点。 发布工程在 [Release Engineering Information](#) 和 [Release Process](#) 文中被描述。 src 使用以 **RELENG\_** 的。 ports 和 doc 使用以 **RELEASE** 的。 最后， www 上不会有任何特定行版的。

### RELENG\_8\_2\_0\_RELEASE

FreeBSD 8.2

### RELENG\_8\_1\_0\_RELEASE

FreeBSD 8.1

### RELENG\_8\_0\_0\_RELEASE

FreeBSD 8.0

### RELENG\_7\_4\_0\_RELEASE

FreeBSD 7.4

### RELENG\_7\_3\_0\_RELEASE

FreeBSD 7.3

### RELENG\_7\_2\_0\_RELEASE

FreeBSD 7.2

### RELENG\_7\_1\_0\_RELEASE

FreeBSD 7.1

### RELENG\_7\_0\_0\_RELEASE

FreeBSD 7.0

### RELENG\_6\_4\_0\_RELEASE

FreeBSD 6.4

### RELENG\_6\_3\_0\_RELEASE

FreeBSD 6.3

### RELENG\_6\_2\_0\_RELEASE

FreeBSD 6.2

**RELENG\_6\_1\_0\_RELEASE**

FreeBSD 6.1

**RELENG\_6\_0\_0\_RELEASE**

FreeBSD 6.0

**RELENG\_5\_5\_0\_RELEASE**

FreeBSD 5.5

**RELENG\_5\_4\_0\_RELEASE**

FreeBSD 5.4

**RELENG\_4\_11\_0\_RELEASE**

FreeBSD 4.11

**RELENG\_5\_3\_0\_RELEASE**

FreeBSD 5.3

**RELENG\_4\_10\_0\_RELEASE**

FreeBSD 4.10

**RELENG\_5\_2\_1\_RELEASE**

FreeBSD 5.2.1

**RELENG\_5\_2\_0\_RELEASE**

FreeBSD 5.2

**RELENG\_4\_9\_0\_RELEASE**

FreeBSD 4.9

**RELENG\_5\_1\_0\_RELEASE**

FreeBSD 5.1

**RELENG\_4\_8\_0\_RELEASE**

FreeBSD 4.8

**RELENG\_5\_0\_0\_RELEASE**

FreeBSD 5.0

**RELENG\_4\_7\_0\_RELEASE**

FreeBSD 4.7

**RELENG\_4\_6\_2\_RELEASE**

FreeBSD 4.6.2

**RELENG\_4\_6\_1\_RELEASE**

FreeBSD 4.6.1

**RELENG\_4\_6\_0\_RELEASE**

FreeBSD 4.6

**RELENG\_4\_5\_0\_RELEASE**

FreeBSD 4.5

**RELENG\_4\_4\_0\_RELEASE**

FreeBSD 4.4

**RELENG\_4\_3\_0\_RELEASE**

FreeBSD 4.3

**RELENG\_4\_2\_0\_RELEASE**

FreeBSD 4.2

**RELENG\_4\_1\_1\_RELEASE**

FreeBSD 4.1.1

**RELENG\_4\_1\_0\_RELEASE**

FreeBSD 4.1

**RELENG\_4\_0\_0\_RELEASE**

FreeBSD 4.0

**RELENG\_3\_5\_0\_RELEASE**

FreeBSD-3.5

**RELENG\_3\_4\_0\_RELEASE**

FreeBSD-3.4

**RELENG\_3\_3\_0\_RELEASE**

FreeBSD-3.3

**RELENG\_3\_2\_0\_RELEASE**

FreeBSD-3.2

**RELENG\_3\_1\_0\_RELEASE**

FreeBSD-3.1

**RELENG\_3\_0\_0\_RELEASE**

FreeBSD-3.0

**RELENG\_2\_2\_8\_RELEASE**

FreeBSD-2.2.8

**RELENG\_2\_2\_7\_RELEASE**

FreeBSD-2.2.7

## RELENG\_2\_2\_6\_RELEASE

FreeBSD-2.2.6

## RELENG\_2\_2\_5\_RELEASE

FreeBSD-2.2.5

## RELENG\_2\_2\_2\_RELEASE

FreeBSD-2.2.2

## RELENG\_2\_2\_1\_RELEASE

FreeBSD-2.2.1

## RELENG\_2\_2\_0\_RELEASE

FreeBSD-2.2.0

## A.7. AFS 站点

FreeBSD 的 AFS 服务器运行于下面的站点：

瑞典

文件的路径是：`/afs/stacken.kth.se/ftp/pub/FreeBSD/`

|                              |   |
|------------------------------|---|
| <code>stacken.kth.se</code>  | <code># Stacken Computer Club, KTH, Sweden</code> |
| <code>130.237.234.43</code>  | <code>#hot.stacken.kth.se</code>                  |
| <code>130.237.237.230</code> | <code>#fishburger.stacken.kth.se</code>           |
| <code>130.237.234.3</code>   | <code>#milko.stacken.kth.se</code>                |

用户 `ftp@stacken.kth.se`

## A.8. rsync 站点

下面的站点 FreeBSD 可以通过 rsync 下载。rsync 程序和 [rcp\(1\)](#) 的工作方式很相像，但是有更多的选项，使用 rsync 程序更新只需要文件的差异，因此能大幅度的提高网络同步速率。如果是 FreeBSD FTP 服务器或者 CVS 的镜像站点，这一点非常有用。rsync 套件可以工作在多种操作系统上，在 FreeBSD 上，请看 [net/rsync](#) port 或者使用 package。

捷克共和国

`rsync://ftp.cz.FreeBSD.org/`

可用的 collection：

- `ftp`：FreeBSD FTP 服务器的部分镜像。
- `FreeBSD`：FreeBSD FTP 服务器的完整镜像。

荷兰

`rsync://ftp.nl.FreeBSD.org/`



可用的 collection :

- FreeBSD: 关于 FreeBSD FTP 服务器的完整镜像。

俄罗斯

rsync://ftp.mtu.ru/

可用的 collections:

- FreeBSD: 完整的 FreeBSD FTP 服务器镜像。
- FreeBSD-gnats: GNATS 追踪数据。
- FreeBSD-Archive: FreeBSD 档案的 FTP 服务器镜像。

瑞典

rsync://ftp4.se.freebsd.org/

可用的 collections :

- FreeBSD : FreeBSD FTP 服务器的完整镜像。

台湾地区 (中国)

rsync://ftp.tw.FreeBSD.org/

rsync://ftp2.tw.FreeBSD.org/

rsync://ftp6.tw.FreeBSD.org/

可用的 collection :

- FreeBSD : FreeBSD FTP 服务器的完整镜像。

英国

rsync://rsync.mirrorservice.org/

可用的 collection :

- sites/ftp.FreeBSD.org: FreeBSD FTP 服务器的完整镜像。

美国

rsync://ftp-master.FreeBSD.org/

服务器只供 FreeBSD 主镜像站点使用。

可用的 collection :

- FreeBSD : FreeBSD FTP 服务器的主要存档。
- acl : FreeBSD 主 ACL 列表。

rsync://ftp13.FreeBSD.org/

可用的 collection :

- FreeBSD : FreeBSD FTP 服务器的完整映像。

## 附录 B: 参考文献

尽管手册可能提供关于 FreeBSD 操作系统最权威的参考资料，但它有却不能告诉我如何整个系统很好地运行起来。因此，一本关于 UNIX® 系统管理的好书，以及一本好的用户手册是不可或缺的。

### B.1. 关于 FreeBSD 的书籍与杂志

非英文的书籍和杂志：

- [FreeBSD 入门与应用](#) (繁体中文)，出版商：[Drmaster](#)，1997. ISBN 9-578-39435-7.
- FreeBSD 技术内幕 (简体中文版本)，[机械工业出版社](#)。ISBN 7-111-10201-0。
- FreeBSD 使用大全 第一版 (简体中文)，机械工业出版社。ISBN 7-111-07482-3。
- FreeBSD 使用大全 第二版 (简体中文)，机械工业出版社。ISBN 7-111-10286-X。
- FreeBSD Handbook (第二版简体中文版本)，[人民邮电出版社](#)。ISBN 7-115-10541-3。
- FreeBSD 3.x Internet 高服务器的架设与管理 (简体中文)，[清华大学出版社](#)。ISBN 7-900625-66-6。
- FreeBSD & Windows 集成网络 (简体中文)，[中国铁道出版社](#)。ISBN 7-113-03845-X。
- FreeBSD 网站架 (简体中文)，中国铁道出版社。ISBN 7-113-03423-3。
- FreeBSD for PC 98'ers (日文，出版商：SHUWA System Co, LTD. ISBN 4-87966-468-5 C3055 P2900E。
- FreeBSD (日文，出版商：CUTT. ISBN 4-906391-22-2 C3055 P2400E。
- [Complete Introduction to FreeBSD](#) (日文)，出版商：[Shoeisha Co., Ltd.](#) ISBN 4-88135-473-6 P3600E。
- [Personal UNIX Starter Kit FreeBSD](#) (日文)，出版商：[ASCII](#)。ISBN 4-7561-1733-3 P3000E。
- FreeBSD Handbook (日文版本)，出版商：[ASCII](#)。ISBN 4-7561-1580-2 P3800E。
- FreeBSD mit Methode (德文)，出版商：[Computer und Literatur Verlag/Vertrieb Hanser](#), 1998. ISBN 3-932311-31-0。
- [FreeBSD 4 - Installieren, Konfigurieren, Administrieren](#) (德文)，出版商：[Computer und Literatur Verlag](#), 2001. ISBN 3-932311-88-4。
- [FreeBSD 5 - Installieren, Konfigurieren, Administrieren](#) (德文)，出版商：[Computer und Literatur Verlag](#), 2003. ISBN 3-936546-06-1。
- [FreeBSD de Luxe](#) (德文)，出版商：[Verlag Moderne Industrie](#), 2003. ISBN 3-8266-1343-0。
- [FreeBSD Install and Utilization Manual](#) (日文)，出版商：[Mainichi Communications Inc.](#), 1998. ISBN 4-8399-0112-0。
- Onno W Purbo, Dodi Maryanto, Syahrial Hubbany, Widjil Widodo [Building Internet Server with FreeBSD](#) (印尼文)，出版商：[Elex Media Komputindo](#)。
- Absolute BSD: The Ultimate Guide to FreeBSD (繁体中文) 出版商：[GrandTech Press](#), 2003. ISBN 986-7944-92-5。
- [The FreeBSD 6.0 Book](#) (繁体中文)，出版商：[Drmaster](#), 2006. ISBN 9-575-27878-X。

英文版的书籍和杂志：

- [Absolute FreeBSD, 2nd Edition: The Complete Guide to FreeBSD](#), 出版商： [No Starch Press](#), 2007. ISBN: 978-1-59327-151-0
- [The Complete FreeBSD](#), 出版商： [O'Reilly](#), 2003. ISBN: 0596005164
- [The FreeBSD Corporate Networker's Guide](#), 出版商： [Addison-Wesley](#), 2000. ISBN: 0201704811
- [FreeBSD: An Open-Source Operating System for Your Personal Computer](#), 出版商： [The Bit Tree Press](#), 2001. ISBN: 0971204500
- [Teach Yourself FreeBSD in 24 Hours](#), 出版商： [Sams](#), 2002. ISBN: 0672324245
- [FreeBSD 6 Unleashed](#), 出版商： [Sams](#), 2006. ISBN: 0672328755
- [FreeBSD: The Complete Reference](#), 出版商： [McGrawHill](#), 2003. ISBN: 0072224096
- [BSD Magazine](#), 出版商： [Software Press Sp. z o.o. SK](#). ISSN 1898-9144

## B.2. 用□指南

- Computer Systems Research Group, UC Berkeley. *4.4BSD User's Reference Manual*. O'Reilly & Associates, Inc., 1994. ISBN 1-56592-075-9
- Computer Systems Research Group, UC Berkeley. *4.4BSD User's Supplementary Documents*. O'Reilly & Associates, Inc., 1994. ISBN 1-56592-076-7
- *UNIX in a Nutshell*. O'Reilly & Associates, Inc., 1990. ISBN 093717520X
- Mui, Linda. *What You Need To Know When You Can't Find Your UNIX System Administrator*. O'Reilly & Associates, Inc., 1995. ISBN 1-56592-104-6
- Ohio State University □写了一□ [UNIX 入门教程](#) 并提供了在□的 HTML 和 PostScript 格式的版本。
- □□文□的意大利文 翻□ 是 FreeBSD Italian Documentation Project 的一部分。
- [Jpman Project, Japan FreeBSD Users Group](#). [FreeBSD User's Reference Manual](#) (日文□本). [Mainichi Communications Inc.](#), 1998. ISBN4-8399-0088-4 P3800E.
- [Edinburgh University](#) has written an [Online Guide](#) for newcomers to the UNIX environment.

## B.3. 管理□指南

- Albitz, Paul and Liu, Cricket. *DNS and BIND*, 4th Ed. O'Reilly & Associates, Inc., 2001. ISBN 1-59600-158-4
- Computer Systems Research Group, UC Berkeley. *4.4BSD System Manager's Manual*. O'Reilly & Associates, Inc., 1994. ISBN 1-56592-080-5
- Costales, Brian, et al. *Sendmail*, 2nd Ed. O'Reilly & Associates, Inc., 1997. ISBN 1-56592-222-0
- Frisch, Aileen. *Essential System Administration*, 2nd Ed. O'Reilly & Associates, Inc., 1995. ISBN 1-56592-127-5
- Hunt, Craig. *TCP/IP Network Administration*, 2nd Ed. O'Reilly & Associates, Inc., 1997. ISBN 1-56592-322-7
- Nemeth, Evi. *UNIX System Administration Handbook*. 3rd Ed. Prentice Hall, 2000. ISBN 0-13-020601-6

- Stern, Hal *Managing NFS and NIS* O'Reilly & Associates, Inc., 1991. ISBN 0-937175-75-7
- [Jpman Project](#), [Japan FreeBSD Users Group](#). [FreeBSD System Administrator's Manual](#) (日文<sup>3</sup>本). [Mainichi Communications Inc.](#), 1998. ISBN4-8399-0109-0 P3300E.
- Dreyfus, Emmanuel. [Cahiers de l'Admin: BSD](#) 2nd Ed. (in French), Eyrolles, 2004. ISBN 2-212-11463-X

## B.4. <sup>3</sup>指南

- Asente, Paul, Converse, Diana, and Swick, Ralph. *X Window System Toolkit*. Digital Press, 1998. ISBN 1-55558-178-1
- Computer Systems Research Group, UC Berkeley. *4.4BSD Programmer's Reference Manual*. O'Reilly & Associates, Inc., 1994. ISBN 1-56592-078-3
- Computer Systems Research Group, UC Berkeley. *4.4BSD Programmer's Supplementary Documents*. O'Reilly & Associates, Inc., 1994. ISBN 1-56592-079-1
- Harbison, Samuel P. and Steele, Guy L. Jr. *C: A Reference Manual*. 4th ed. Prentice Hall, 1995. ISBN 0-13-326224-3
- Kernighan, Brian and Dennis M. Ritchie. *The C Programming Language*. 2nd Ed. PTR Prentice Hall, 1988. ISBN 0-13-110362-8
- Lehey, Greg. *Porting UNIX Software*. O'Reilly & Associates, Inc., 1995. ISBN 1-56592-126-7
- Plauger, P. J. *The Standard C Library*. Prentice Hall, 1992. ISBN 0-13-131509-9
- Spinellis, Diomidis. [Code Reading: The Open Source Perspective](#). Addison-Wesley, 2003. ISBN 0-201-79940-5
- Spinellis, Diomidis. [Code Quality: The Open Source Perspective](#). Addison-Wesley, 2006. ISBN 0-321-16607-8
- Stevens, W. Richard and Stephen A. Rago. *Advanced Programming in the UNIX Environment*. 2nd Ed. Reading, Mass. : Addison-Wesley, 2005. ISBN 0-201-43307-9
- Stevens, W. Richard. *UNIX Network Programming*. 2nd Ed, PTR Prentice Hall, 1998. ISBN 0-13-490012-X
- Wells, Bill. "Writing Serial Drivers for UNIX". *Dr. Dobbs's Journal*. 19(15), December 1994. pp68-71, 97-99.

## B.5. 操作系<sup>3</sup>原理

- Andleigh, Prabhat K. *UNIX System Architecture*. Prentice-Hall, Inc., 1990. ISBN 0-13-949843-5
- Jolitz, William. "Porting UNIX to the 386". *Dr. Dobbs's Journal*. 1991年1月 - 1992年6月
- Leffler, Samuel J., Marshall Kirk McKusick, Michael J Karels and John Quarterman *The Design and Implementation of the 4.3BSD UNIX Operating System*. Reading, Mass. : Addison-Wesley, 1989. ISBN 0-201-06196-1
- Leffler, Samuel J., Marshall Kirk McKusick, *The Design and Implementation of the 4.3BSD UNIX Operating System: Answer Book*. Reading, Mass. : Addison-Wesley, 1991. ISBN 0-201-54629-9

- McKusick, Marshall Kirk, Keith Bostic, Michael J Karels, and John Quarterman. *The Design and Implementation of the 4.4BSD Operating System*. Reading, Mass. : Addison-Wesley, 1996. ISBN 0-201-54979-4

([本书](#)的第二章的 [在线版本](#) 是 FreeBSD Documentation Project 的一部分。)

- Marshall Kirk McKusick, George V. Neville-Neil *The Design and Implementation of the FreeBSD Operating System*. Boston, Mass. : Addison-Wesley, 2004. ISBN 0-201-70245-2
- Stevens, W. Richard. *TCP/IP Illustrated, Volume 1: The Protocols*. Reading, Mass. : Addison-Wesley, 1996. ISBN 0-201-63346-9
- Schimmel, Curt. *Unix Systems for Modern Architectures*. Reading, Mass. : Addison-Wesley, 1994. ISBN 0-201-63338-8
- Stevens, W. Richard. *TCP/IP Illustrated, Volume 3: TCP for Transactions, HTTP, NNTP and the UNIX Domain Protocols*. Reading, Mass. : Addison-Wesley, 1996. ISBN 0-201-63495-3
- Vahalia, Uresh. *UNIX Internals — The New Frontiers*. Prentice Hall, 1996. ISBN 0-13-101908-2
- Wright, Gary R. and W. Richard Stevens. *TCP/IP Illustrated, Volume 2: The Implementation*. Reading, Mass. : Addison-Wesley, 1995. ISBN 0-201-63354-X

## B.6. 安全方面的参考文献

- Cheswick, William R. and Steven M. Bellovin. *Firewalls and Internet Security: Repelling the Wily Hacker*. Reading, Mass. : Addison-Wesley, 1995. ISBN 0-201-63357-4
- Garfinkel, Simson and Gene Spafford. *Practical UNIX & Internet Security*. 2nd Ed. O'Reilly & Associates, Inc., 1996. ISBN 1-56592-148-8
- Garfinkel, Simson. *PGP Pretty Good Privacy* O'Reilly & Associates, Inc., 1995. ISBN 1-56592-098-8

## B.7. 硬件参考

- Anderson, Don and Tom Shanley. *Pentium Processor System Architecture*. 2nd Ed. Reading, Mass. : Addison-Wesley, 1995. ISBN 0-201-40992-5
- Ferraro, Richard F. *Programmer's Guide to the EGA, VGA, and Super VGA Cards*. 3rd ed. Reading, Mass. : Addison-Wesley, 1995. ISBN 0-201-62490-7
- Intel 公司在他[的 网站](#)上, 提供了[关于他](#)的 CPU, 芯片, 以及[准](#)的文。多数是PDF文件。
- Shanley, Tom. *80486 System Architecture*. 3rd ed. Reading, Mass. : Addison-Wesley, 1995. ISBN 0-201-40994-1
- Shanley, Tom. *ISA System Architecture*. 3rd ed. Reading, Mass. : Addison-Wesley, 1995. ISBN 0-201-40996-8
- Shanley, Tom. *PCI System Architecture*. 4th ed. Reading, Mass. : Addison-Wesley, 1999. ISBN 0-201-30974-2
- Van Gilluwe, Frank. *The Undocumented PC*, 2nd Ed. Reading, Mass: Addison-Wesley Pub. Co., 1996. ISBN 0-201-47950-8
- Messmer, Hans-Peter. *The Indispensable PC Hardware Book*, 4th Ed. Reading, Mass: Addison-

## B.8. 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. 它也被称作 [Jargon File](#)
- 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](#).
- Don Libes, Sandy Ressler *Life with UNIX* - special edition. Prentice-Hall, Inc., 1989. ISBN 0-13-536657-7
- BSD 族。 <http://www.FreeBSD.org/cgi/cvsweb.cgi/src/shared/misc/bsd-family-tree> 或在 FreeBSD 机器上的 [/usr/shared/misc/bsd-family-tree](#)。
- *Networked Computer Science Technical Reports Library*. <http://www.ncstrl.org/>
- *Old BSD releases from the Computer Systems Research group (CSRG)*. <http://www.mckusick.com/csrg/>: The 4CD set covers all BSD versions from 1BSD to 4.4BSD and 4.4BSD-Lite2 (but not 2.11BSD, unfortunately). The last disk also holds the final sources plus the SCCS files.

## B.9. 各期刊

- *The C/C++ Users Journal*. R&D Publications Inc. ISSN 1075-2838
- *Sys Admin - The Journal for UNIX System Administrators* Miller Freeman, Inc., ISSN 1061-2688
- *freeX - Das Magazin für Linux - BSD - UNIX* (德文) Computer- und Literaturverlag GmbH, ISSN 1436-7033

# 附录 C: Internet 上的资源

发展迅猛的 FreeBSD 使得有的印刷、平面媒体跟不上它的发展度！而电子版的也是最好的，通常是唯一一个可以跟上最新发展方向的。FreeBSD 来自于志愿者的成果，社区通常也扮演着“技术支持部”的角色。通过电子，Web 邮件和 USENET 新闻可以很快的得到他。

以下列出了尽量多的联系 FreeBSD 用社区的方式。如果还有其他的资源没有被包括在这儿，请告诉 FreeBSD 文档邮件列表，以便将它加入到里。

## C.1. 邮件列表

邮件列表通常是提或是起有 FreeBSD 某一方面的技术最直接的途径。有多用于不同 FreeBSD 的邮件列表。把的送到最合的邮件列表通常能得更加快速准确的回。

本文的最后出了各个不同的邮件列表的使用。在任何一个列表之前，先使用条文。在有些邮件列表的人天都会收到上百封于 FreeBSD 的信件。立列表的使用条文有助于减少量。否则些的列表将失去其意。



如果想要送一封信件到 FreeBSD 邮件列表，可以把件往 [FreeBSD 邮件列表](#)。不要往其他的列表送件。

如果不知道个邮件列表合于送的，参[如何从 FreeBSD-questions 邮件列表中更快地得到答案](#)。

在列表中送任何之前，首先学使用邮件列表的最佳方式，例如如何通[邮件列表常见问题回答集 \(FAQ\)](#)文，来避免常重的。

全部的邮件列表都可以在[FreeBSD World Wide Web服务器](#)上到。此服务器提供了很棒的搜索功能，可得到FAQ的解答。而在邮件列表上提之前，先搜索是否已有答案。注意意味着所有往 FreeBSD 邮件列表的消息都会被永久保存。当及到私保的，可以考使用一个可使用后的子邮件地址并只送公的信息。

### C.1.1. 列表摘要

一般性的列表: 以下的列表都是一般性的，而且可以自由地加入，鼓励大家加入他：

| 目                                    | 用途                      |
|--------------------------------------|-------------------------|
| <a href="#">freebsd-advocacy</a>     | FreeBSD鼓吹者              |
| <a href="#">FreeBSD 公告邮件列表</a>       | 重要的事件和里程碑               |
| <a href="#">freebsd-arch</a>         | 架构和的                    |
| <a href="#">freebsd-bugbusters</a>   | 与FreeBSD告数据和有工具相的       |
| <a href="#">freebsd-bugs</a>         | 告FreeBSD的Bug            |
| <a href="#">freebsd-chat</a>         | 和技无的FreeBSD区            |
| <a href="#">freebsd-chromium</a>     | FreeBSD Chromium 相的     |
| <a href="#">FreeBSD-CURRENT 邮件列表</a> | 使用 FreeBSD-CURRENT 有的一些 |



| 目                                   | 用途                     |
|-------------------------------------|------------------------|
| <a href="#">freebsd-isp</a>         | ISP使用FreeBSD的          |
| <a href="#">freebsd-jobs</a>        | 与FreeBSD有的工作机会         |
| <a href="#">freebsd-questions</a>   | 用和技支持                  |
| <a href="#">FreeBSD 安全通知件列表</a>     | 安全通知                   |
| <a href="#">FreeBSD-STABLE; 件列表</a> | 使用 FreeBSD-STABLE 有的一些 |
| <a href="#">FreeBSD 件列表</a>         | 在真正送一个件到件列表之前可以先送到里    |

技性的件列表： 以下的件列表是用来技性的。 在加入及之前必真个列表主，因他的内容都是格地被限制着的。

| 目                                 | 用途   |
|-----------------------------------|--|
| <a href="#">FreeBSD ACPI 件列表</a>  | ACPI 和源管理的                                 |
| <a href="#">freebsd-afs</a>       | 将 AFS 移植到 FreeBSD                          |
| <a href="#">freebsd-aic7xxx</a>   | Adaptec® AIC 7xxx                          |
| <a href="#">freebsd-amd64</a>     | 将 FreeBSD 移植到 AMD64 系                      |
| <a href="#">freebsd-apache</a>    | 于与 Apache 有的 ports 的                       |
| <a href="#">freebsd-arm</a>       | 将 FreeBSD 移植到 ARM® 理器                      |
| <a href="#">freebsd-atm</a>       | 在 FreeBSD 上使用 ATM 网                        |
| <a href="#">freebsd-bluetooth</a> | 在 FreeBSD 上使用 Bluetooth® 技                 |
| <a href="#">freebsd-cluster</a>   | 在集群境中使用 FreeBSD                            |
| <a href="#">freebsd-cvsw eb</a>   | CVSweb                                     |
| <a href="#">freebsd-database</a>  | FreeBSD 下和使用数据                             |
| <a href="#">freebsd-doc</a>       | 建 FreeBSD 相文                               |
| <a href="#">freebsd-drivers</a>   | FreeBSD 撰写                                 |
| <a href="#">freebsd-eclipse</a>   | FreeBSD 上的 Eclipse IDE、工具、富客用，以及 ports 的用。 |
| <a href="#">freebsd-embedded</a>  | 在嵌入式用中使用 FreeBSD                           |
| <a href="#">freebsd-eol</a>       | 于与 FreeBSD 有，但已不再 FreeBSD Project 所件的互助支持。 |
| <a href="#">freebsd-emulation</a> | 在 FreeBSD 上模其它系，如 Linux/MS-DOS®/Windows®   |
| <a href="#">freebsd-firewire</a>  | FreeBSD 的 FireWire® (iLink, IEEE 1394) 技   |
| <a href="#">freebsd-fs</a>        | 文件系  |
| <a href="#">freebsd-gecko</a>     | Gecko 索引 issues                            |
| <a href="#">freebsd-geom</a>      | GEOM 的和                                    |
| <a href="#">freebsd-gnome</a>     | 移植 GNOME 和 GNOME 用程序                       |

| 目                                   | 用途   |
|-------------------------------------|--|
| <a href="#">freebsd-hackers</a>     | 一般性的技  |
| <a href="#">freebsd-hardware</a>    | 一般性的支持 FreeBSD 的硬件的  |
| <a href="#">freebsd-i18n</a>        | FreeBSD 的国际化   |
| <a href="#">freebsd-ia32</a>        | 在 IA-32 (Intel® x86) 平台上行 FreeBSD                            |
| <a href="#">freebsd-ia64</a>        | 将 FreeBSD 移植到 Intel® 即将推出的 IA64 系                            |
| <a href="#">freebsd-ipfw</a>        | 于 IP 防火代再的技性   |
| <a href="#">freebsd-isdn</a>        | ISDN 人   |
| <a href="#">freebsd-jail</a>        | 于 <a href="#">jail(8)</a> 机制的                                |
| <a href="#">freebsd-java</a>        | Java™ 人以及移植 JDK™s 到 FreeBSD 的人                               |
| <a href="#">freebsd-kde</a>         | 移植 KDE 和 KDE 用程序   |
| <a href="#">freebsd-lfs</a>         | 移植 LFS 到 FreeBSD 上   |
| <a href="#">freebsd-mips</a>        | 移植 FreeBSD 到 MIPS®   |
| <a href="#">freebsd-mobile</a>      | 于便携式计算机的   |
| <a href="#">freebsd-mono</a>        | FreeBSD 上的 Mono 和 C# 用                                       |
| <a href="#">FreeBSD 多媒体用件列表</a>     | 多媒体用程序   |
| <a href="#">freebsd-new-bus</a>     | 技于架  |
| <a href="#">freebsd-net</a>         | 网子系和 TCP/IP 源代的  |
| <a href="#">freebsd-office</a>      | FreeBSD 上的公套件  |
| <a href="#">freebsd-performance</a> | 高性能、下安装后的性能整   |
| <a href="#">freebsd-perl</a>        | 多与 perl 相的 ports 的   |
| <a href="#">freebsd-pf</a>          | 于 packet filter 防火系的   |
| <a href="#">freebsd-platforms</a>   | 于向非 Intel® 架的平台上移植的  |
| <a href="#">freebsd-ports</a>       | 于 Ports Collection 的   |
| <a href="#">freebsd-ports-bugs</a>  | ports bugs/PRs   |
| <a href="#">freebsd-ppc</a>         | 移植 FreeBSD 到 PowerPC®  |
| <a href="#">freebsd-proliant</a>    | 于 FreeBSD 在 HP ProLiant 服器平台上的技                              |
| <a href="#">freebsd-python</a>      | FreeBSD 属的 Python  |
| <a href="#">freebsd-rc</a>          | 于 rc.d 系及其的  |
| <a href="#">freebsd-realtime</a>    | FreeBSD 展的   |
| <a href="#">freebsd-ruby</a>        | 于 FreeBSD 上 Ruby 的   |
| <a href="#">freebsd-scsi</a>        | SCSI 子系  |
| <a href="#">FreeBSD 安全件列表</a>       | 系安全  |
| <a href="#">freebsd-small</a>       | 在嵌入式系上使用 FreeBSD (已； 使用 <a href="#">freebsd-embedded</a> 代替) |

| 目                                      | 用途                              |
|--|---------------------------------|
| <a href="#">freebsd-sparc64</a>        | 移植 FreeBSD 到 SPARC® 系           |
| <a href="#">freebsd-standards</a>      | FreeBSD 的 C99 以及 POSIX® 准       |
| <a href="#">freebsd-sysinstall</a>     | <a href="#">sysinstall(8)</a> 的 |
| <a href="#">freebsd-threads</a>        | 程                               |
| <a href="#">freebsd-testing</a>        | FreeBSD 性能和定性                   |
| <a href="#">freebsd-tilera</a>         | 将 FreeBSD 移植到 Tilera 系列 CPU     |
| <a href="#">freebsd-tokenring</a>      | 在 FreeBSD 中支持 Token Ring        |
| <a href="#">freebsd-toolchain</a>      | 在 FreeBSD 中集成的工具集               |
| <a href="#">freebsd-usb</a>            | 于 FreeBSD 的 USB 支持的             |
| <a href="#">freebsd-virtualization</a> | 各 FreeBSD 支持的虚拟化技               |
| <a href="#">freebsd-vuxml</a>          | 于 VuXML 的                       |
| <a href="#">freebsd-x11</a>            | 和支持在 FreeBSD 上行的 X11            |
| <a href="#">freebsd-xen</a>            | FreeBSD Xen™ 上的移植 - 和使用         |
| <a href="#">freebsd-xfce</a>           | FreeBSD 上 XFCE 的移植和             |

限制列表: 以下的列表是某些特定的者而的, 而且并不合被当成是一般公区。最好在某一技区参与后再有些 限制列表, 因可以了解到在些区言所需要的礼。

| 目                                   | 用途                                |
|-------------------------------------|-----------------------------------|
| <a href="#">freebsd-hubs</a>        | 行象站点的成(支持基本服)                     |
| <a href="#">freebsd-user-groups</a> | 用整                                |
| <a href="#">freebsd-vendors</a>     | 商家在布之前的整                          |
| <a href="#">freebsd-wip-status</a>  | FreeBSD 目度状                       |
| <a href="#">freebsd-wireless</a>    | 802.11 , 工具和                      |
| <a href="#">freebsd-www</a>         | <a href="#">www.FreeBSD.org</a> 的 |

分列表: 所有以上的列表在一个分格式里面是可利用的。一旦了一个列表, 可以在的号里面置的分。

CVS 和 SVN 列表: 以下的件是FreeBSD源代的更有趣的人看的, 而且它是只的件列表, 不能Email他。

| 列表                           | 源位置                      | 描述                           |
|------------------------------|--------------------------|------------------------------|
| <a href="#">cvs-all</a>      | /usr/(CVSROOT doc ports) | 所有源代的改 (其他 CVS commit 列表的超集) |
| <a href="#">cvs-doc</a>      | /usr/(doc www)           | 所有 doc 和 www 源代的改            |
| <a href="#">cvs-ports</a>    | /usr/ports               | 所有 ports 源代的改                |
| <a href="#">cvs-projects</a> | /usr/projects            | 所有 projects 源代的改             |

| 列表   | 源位置           | 描述   |
|--|---------------|--|
| <a href="#">cvs-src</a>  | /usr/src      | 所有 CVS 源代码的更改 (由 svn-to-cvs 提交程序生成)                                |
| <a href="#">SVN 整个 src 的修信息 (除了 "user" 与 "projects")</a>           | /usr/src      | 所有 Subversion 的更改 (除了 user 和 projects)                             |
| <a href="#">SVN src 的 head/-current 分支的修信息</a>                     | /usr/src      | 所有 Subversion 的 "head" 分支的更改 (FreeBSD-CURRENT 分支)                  |
| <a href="#">svn-src-projects</a>                                   | /usr/projects | 所有 Subversion 源中有 projects 部分的更改                                   |
| <a href="#">svn-src-release</a>                                    | /usr/src      | 所有 Subversion 源中有 releases 部分的更改                                   |
| <a href="#">svn-src-releng</a>                                     | /usr/src      | 所有 Subversion 源中有 releng 部分的更改 (security / release engineering 分支) |
| <a href="#">svn-src-stable</a>                                     | /usr/src      | 所有 Subversion 源中有 stable 分支的更改                                     |
| <a href="#">svn-src-stable-6</a>                                   | /usr/src      | 所有 Subversion 源中有 stable/6 分支的更改                                   |
| <a href="#">svn-src-stable-7</a>                                   | /usr/src      | 所有 Subversion 源中有 stable/7 分支的更改                                   |
| <a href="#">svn-src-stable-8</a>                                   | /usr/src      | 所有 Subversion 源中有 stable/8 分支的更改                                   |
| <a href="#">SVN commit messages for only the 9-stable src tree</a> | /usr/src      | 所有 Subversion 源中有 stable/9 分支的更改                                   |
| <a href="#">svn-src-stable-other</a>                               | /usr/src      | 所有 Subversion 源中早期 stable 分支的更改                                    |
| <a href="#">svn-src-svnadmin</a>                                   | /usr/src      | 所有 Subversion 源中管理用脚本, hook 和其他配置数据的更改                             |
| <a href="#">svn-src-user</a>                                       | /usr/src      | 所有 Subversion 源中有 user 部分的更改                                       |
| <a href="#">svn-src-vendor</a>                                     | /usr/src      | 所有 Subversion 源中有 vendor 部分的更改                                     |

### C.1.2. 如何

一个列表, 点上面的列表名字或到 <https://lists.freebsd.org> 并点入感兴趣的列表, 个列表的面包含了所必需的的操作指南。

其只需送件到 [列表名@FreeBSD.org](mailto:列表名@FreeBSD.org)。它将被再次到全世界的个件列表的成。

点上面的 [URL](#)，在列表的底部可以从该列表中退出。也可以发送一个邮件到 [unsubscribe@FreeBSD.org](mailto:unsubscribe@FreeBSD.org) 来退。

此外，我要求你必须保持在技术性的邮件列表中只是技术。如果只是一些重要的公告感兴趣，建议加入 [FreeBSD 公告邮件列表](#)，它的通信量比低。

### C.1.3. 列表章节

所有 FreeBSD 的邮件列表都有同的基本，所有人必须按照来做。相反一些，FreeBSD Postmaster [postmaster@FreeBSD.org](mailto:postmaster@FreeBSD.org) 会在前次发送警告，如果第三次违反，FreeBSD Postmaster 将从所有 FreeBSD 的邮件列表中除的人，并来自信人之后的所有邮件。我很遗憾必须要遵守的，但今天的互联网是一个很混乱的环境，它上面的很多约束机制，都相当脆弱。

具体：

- 任何列表的主都必须当附合基本的列表概况。例如，如果列表是有技术的，那该表的文章包含技术。不要把不相的放在一起。于没有主的自由形式的，可以使用 [FreeBSD-chat](mailto:freebsd-chat@FreeBSD.org) [freebsd-chat@FreeBSD.org](mailto:freebsd-chat@FreeBSD.org)。
- 不要将同一个发送到超个的邮件列表上，当有一个清晰和明确的必要 表到个列表的要求，也只能是个。于大多数的列表，已有相当多的了，除了一些比深奥的(如"-stable & -scsi")，没有必要同将一个到多个列表上。如果一个信息以方式（多个邮件列表在Cc行出）被送，那Cc行在把它再送出去之前也将被整理。无是最初表者，都会致自己的交叉送。
- 不容许行人身攻击和（在前后的争中），包括用和私人。当遵守最起码的网礼，象需要征得同意才可以引用或私人邮件等。然而，也有非常少的情况下，的内容会符合列表章，因此，它会在最初予警告（或禁止）。
- 格的禁止非FreeBSD相关产品或服务的广告，一旦将上取。

独立的列表章：

#### FreeBSD ACPI 邮件列表

ACPI和电源管理

#### freebsd-afs

Andrew文件系统

个列表是用来porting和从CMU/Transarc使用AFS。

#### FreeBSD 公告邮件列表

重要事件/里程碑

是一个布FreeBSD重大事件的邮件列表。包括有snapshots和其他版本的公告，新的 FreeBSD的性能的公告，可以用于指派志者等等。个列表比小。

#### freebsd-arch

架构和

个列表是FreeBSD的架构。本上保内容的技术性。例如主是：

- 如何重新构建系统使其同时有几个自己构造的系统运行。
- 需要什么才能修复VFS来使Heidemann工作。
- 我如何改变系统程序接口以便能在多系统和体系结构上使用相同的程序。
- 如何写一个网络。

## freebsd-bluetooth

*FreeBSD 上的 Bluetooth®*

是一个 FreeBSD 的 Bluetooth® 用户聚集的社区。这里欢迎于用户、用户、补丁、公告、月度公告，功能需求以及其他与 Bluetooth® 相关的。

## freebsd-bugbusters

同等公告管理

这个列表的目的是作为一个整合和系统来服务于Bug列表的创建，Bugbuster列表创建和其他任何的PR数据真正有趣的创建。这个列表不是为了关于Bug，补丁或PRs。

## freebsd-bugs

*Bug公告*

是一个公告FreeBSD的Bug的列表。可以随通过 [send-pr\(1\)](#) 命令或[WEB页面](#)来提交Bug。

## freebsd-chat

*与FreeBSD社区相关的非技术性项目*

这个列表超出了其他有非技术、社会信息的内容。包括Jordan看起来是否像一个机敏的侦探，是否句首的字母要大写，喝了很多，儿子的酒造的最好，在他的地下室里造了酒等等。由于偶然宣布重大的事件（例如：将要行的聚会，婚礼，生日，新工作等等）也能使用技术列表，除上述列之外任何事情都可以布在-chat列表上。

## freebsd-chromium

*FreeBSD 上的 Chromium*

是一个 FreeBSD 上 Chromium 相关的列表。是一个和安装 Chromium 的技术列表。

## Core Team

*FreeBSD核心*

是一个只供核心成员内部使用的列表，只有当一个与FreeBSD相关的重要的事情需要裁决或格核，才能送消息到这个列表。

## FreeBSD-CURRENT 列表

于使用FreeBSD-CURRENT版的

是一个FreeBSD-CURRENT用的列表。它包括一些可能影响用的新特性的警告，使用FreeBSD-current的一些指。任何行"CURRENT"的人必同意个列表，是一个技术的列表。

## freebsd-cvswweb

*FreeBSD CVSweb*

关于FreeBSD-CVSweb的使用， 和 的技 性。

## freebsd-doc

文

个 件列表是与FreeBSD 建的文的 的出版和 的。 个 件列表的成 都会提交到"The FreeBSD Documentation Project"。它是一个 放的列表，可以自由地加入和做 献！

## freebsd-drivers

FreeBSD 撰写

是关于 FreeBSD 上的 的技 。它主要供 写 的 人 提出 于如何使用 FreeBSD 内核提供的 API 来 写 程序的。

## freebsd-eclipse

FreeBSD 上的 Eclipse IDE、工具、 富客 用，以及 ports 的用 。

个 件列表的目的，是在 FreeBSD 平台上 、安装、使用、 和 Eclipse IDE、工具、 富客 用的用，提供互助式支持，以及 将 Eclipse IDE 和 件移植到 FreeBSD 境中提供 助。

一个目的是建立一个在 Eclipse 社区和 FreeBSD 社区之 的交流管道，以 到互惠互利。

尽管 个列表主要 注的是 Eclipse 用的 求，它也 使用 Eclipse 框架 FreeBSD 用的 用提供了 。

## freebsd-embedded

在嵌入式 用中使用 FreeBSD

个列表 于在嵌入式系 中如何使用 FreeBSD 的。 是一个技 性的 件列表，其主要内容是技 。 一 件列表， 我 将嵌入式系 定 那些不作 面系 、 只完成某些 一任 的 算 。 些 例包括路由器交 机和 PBX 的网 、 程 量 、 PDA、PoS 系 ，等等。

## freebsd-emulation

模 其他系 ，例如 Linux/MS-DOS®/Windows®

是一个 于如何在 FreeBSD 上 行 其他操作系 所撰写的程序的 。

## freebsd-eol

于与 FreeBSD 有 ，但已不再 FreeBSD Project 所 的 件的互助支持。

个 件列表主要用于那些有 趣提供或使用 已不再 FreeBSD Project 官方所支持 (例如，以安全更新或 丁的形式) 的 FreeBSD 相 件的用 或公司 。

## freebsd-firewire

FireWire® (iLink, IEEE 1394)

个 件列表是关于FreeBSD子系FireWire® (aka IEEE 1394 aka iLink)的 和 行。相 特定的主 包括 准， 和他 的 ， 配器板/ /芯片 置，及他 的正 的代 的 和 施。

## freebsd-fs

文件系统

关于FreeBSD文件系统的页面。它是一个技术的组件列表。

## freebsd-gecko

Gecko 渲染引擎

是一个关于 FreeBSD 上 Gecko 有的应用程序的组件列表。

关于 FreeBSD 上 Gecko Ports 应用程序的页面，以及它的安装，页面和支持。

## freebsd-geom

GEOM

关于GEOM和相关行的页面。它是一个技术的组件列表。

## freebsd-gnome

GNOME

关于在FreeBSD系统上的GNOME桌面环境 它是一个技术的组件列表。

## freebsd-ipfw

IP防火墙

是关于在FreeBSD里重新配置IP防火墙代码的技术页面。

## freebsd-ia64

移植FreeBSD到IA64

是一个有关于将FreeBSD移植到Intel® IA64架构上的技术页面列表， 提供一些相关的页面与解决方案。也欢迎一些感兴趣的页面者。

## freebsd-isdn

ISDN通信

是一个FreeBSD支持的ISDN系统的组件列表。

## freebsd-java

Java™

是一个关于Java™ 应用和JDK™s的porting与页面的组件列表。

## freebsd-jobs

工作的提供和

这个页面是关于与 FreeBSD 相关的雇佣信息和个人页面， 比如： 如果你想一个与 FreeBSD 相关的工作或有一个工作需要 FreeBSD 它是一个来广告的好地方。 它不是一般性雇佣的组件列表， 这个页面已经有了足多的页面。



注意一个列表，像其他的 [FreeBSD.org](http://FreeBSD.org) 的件列表一样是会分给全世界的作者的。因此，需要明白由于位置和地域，固定之是容易联系和可合作的。

Email最好使用 - 文本格式，不基本的PDF,HTML和 很少其他的能被更多作者接受的格式也是可以的。Microsoft® Word (.doc) 格式是被件列表服务器拒的。

## freebsd-kde

*KDE*

关于在FreeBSD系上使用KDE。是一个技术的件列表。

## freebsd-hackers

技术

是一个与FreeBSD相关的技术，是一个主要的技术性件列表。他是的工作在FreeBSD上的人来提出或相关的解决方案, 也欢迎些感兴趣的个人的作者。是一个技术的件列表。

## freebsd-hardware

*FreeBSD* 硬件的普通

有FreeBSD行的硬件型的普通，包括是否的一些和建。

## freebsd-hubs

象站点

人行FreeBSD的象站点的公告和。

## freebsd-isp

*ISP* 供商

是一个使用FreeBSD的ISP供商的件列表。是一个技术的件列表。

## freebsd-mono

*FreeBSD* 上的 Mono 和 C# 用

是一个 FreeBSD 上的 Mono 框架的件列表。是一个技术的件列表。它是将 Mono 或 C# 用移植到 FreeBSD，以及提出及其他解决方案的人准的。此外，也欢迎有趣参与的其他人。

## freebsd-office

*FreeBSD* 上的公套件用

于公套件用，它的安装、和 FreeBSD 支持的中心。

## freebsd-performance

关于整及高速行*FreeBSD*

个件列表提供了一个客，管理和有的体去与FreeBSD性能相关的主的空。可以在里行包括在任意高下，体版下或者是有限制的条件下安装FreeBSD。非常鼓励自地了改FreeBSD性能的相体去个列表。是个高技术含量的列表理上合有富的FreeBSD用，客，或FreeBSD的速度、性能、升感兴趣的管理。不是一个答式的列表，于些去相关文

，但他是一个可以投稿的地方，或者了解关于待解决的与性能相关的主。

## freebsd-pf

关于 *packet filter* 防火系统的相关和

关于 FreeBSD 环境下 *packet filter* (pf) 防火系统的相关。 这里迎接技术， 以及一般的应用。 此外， 这里也是 ALTQ QoS 框架的合适所。

## freebsd-platforms

移植到非 Intel® 平台上

跨平台的 FreeBSD 相关， 关于非 Intel® FreeBSD 移植版本的相关和提。 是一个技术性的文件列表， 其内容严格限制技术。

## freebsd-ports

"ports"的相关

关于FreeBSD的"ports collection" (/usr/ports)的相关， ports的基础构造和调整的ports相关。 是一个技术的文件列表。

## freebsd-ports-bugs

"ports" bugs的相关

关于FreeBSD的"ports collection" (/usr/ports),报告 ports建， 或者ports的修正。 是一个技术的文件列表。

## freebsd-proliant

关于 FreeBSD 在 HP ProLiant 服务器平台上的技术

一个文件列表用来在 HP ProLiant 服务器上使用 FreeBSD， 包括 ProLiant 用的、 管理、 配置工具， 以及 BIOS 更新等。 同时， 这里也是 hpsmcli、 hpsmcli， 以及 hpacucli 模块的主要所。

## freebsd-python

FreeBSD 上的 Python

是一个关于如何在 FreeBSD 上改善 Python 支持的文件列表。 是一个技术的文件列表。 它是那些移植 Python、 其第三方模块， 以及 Zope 相关件到 FreeBSD 上的人准的。 这里也欢迎参与技术的人。

## freebsd-questions

用

是一个有FreeBSD相关的文件列表。 不当送"how to" 技术列表， 除非是个是非常可的技术。

## freebsd-ruby

有 FreeBSD 上 Ruby 的相关

是一个关于 Ruby 在 FreeBSD 上支持的文件列表。 是一个技术的文件列表。 它是那些移植 Ruby、 第三方以及各 framework 准的。

里也欢迎参与技术的人。

## freebsd-scsi

SCSI子系

是一个FreeBSD的SCSI子系的件列表。是一个技术的列表。

## FreeBSD 安全件列表

安全

FreeBSD的计算机安全（DES,Kerberos,已知的安全漏洞和修等）。是一个技术的件列表。注意：  
不是一个和答的列表，但是同出 和答案到FAQ是欢迎的。

## FreeBSD 安全通知件列表

安全通知

FreeBSD安全修的通知。不是一个列表，的列表当是FreeBSD-security

## freebsd-small

在嵌入式用程序中使用FreeBSD

个列表了与小的和嵌入的FreeBSD安装的。是一个技术的列表。



一列表已被 [freebsd-embedded](#) 代替。

## FreeBSD-STABLE; 件列表

于FreeBSD-STABLE版的使用

是一个FreeBSD-STABLE用的件列表。它包括-STABLE的新特性可能会影响的警告。任何  
行"STABLE"的人当常注个列表。是一个技术的列表。

## freebsd-standards

C99 & POSIX一致

是于FreeBSD C99和POSIX准的技术。

## freebsd-toolchain

FreeBSD 中集成的工具集

是于FreeBSD 中集成的工具集的。里有包括 Clang 和 GCC，以及其他似器、接口和  
器等件的。

## freebsd-usb

FreeBSD 的 USB 支持

个件列表是于 FreeBSD 上的 USB 支持的技术性。

## freebsd-user-groups

用整列表

这个邮件列表从各地的使用群体到彼此相互学习和从核心中指定个人。这个邮件列表被限制到大和用 的之内。

## freebsd-vendors

商家

FreeBSD 和 FreeBSD 硬件商家的。

## freebsd-virtualization

各 FreeBSD 支持的虚拟化技术

FreeBSD 所支持的各虚拟化技术的邮件列表。在注重基本功能，加入新特性的同时，也提供了一个求助和其他的使用的所。

## freebsd-wip-status

FreeBSD 项目度状

这个邮件列表是用来布 FreeBSD 相关目的建设和工作度的。至这个列表的消息将会先被核。通常建设把消息用 "To:" 一个更典型的 FreeBSD 列表，而只用 "BCC:" 这个列表。的工作度就能在典型的列表上，因为这个列表是不允许的。

看一下中含的消息作例子。

可能每隔几个月，会从这个列表中的消息中提取出一个性的消息摘要到 FreeBSD 网站做状公告的一部分。也能从那里到更多的例子和以往的公告。

## freebsd-wireless

802.11 工具

FreeBSD-wireless 邮件列表集中 802.11 (sys/net80211)，程序和工具的。

## freebsd-xen

FreeBSD 有 Xen™ 上的移植 - 和使用

这个邮件列表集中 FreeBSD 的 Xen™ 移植。期的流量会很小，所以这个列表旨在同 与的技术和管理部属 提供一个的所。

## freebsd-xfce

XFCE

是用于向 FreeBSD 移植 XFCE 的。是一个技术性的邮件列表。其成是目前正活地行 FreeBSD XFCE 移植的人，主要用于提出或其他解决方法。此外，也迎希望注相关技术的其他人士。

### C.1.4. 邮件列表

FreeBSD 邮件列表是使用了多方法去消除件、病毒和其他没用的子件。

部分所描述的并不包括所有常用的保邮件列表的消除方法。

邮件列表只包含一些允的附件型。所有在列表中有 MIME 型的附件的子件在 邮件列表中被之前将被掉。

- 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



一些附件列表可以允许附件其他MIME类型，但是以上列出的 被多数的附件列表所采用。

如果一个子附件包含HTML和文本形式，HTML的形式将被删除。  
HTML形式，他将被转换为文本格式。

如果一个子附件内容只是

## C.2. Usenet新组

除了FreeBSD这个特殊的新组，还有很多FreeBSD或与FreeBSD用相关的其他组。一些新组的搜索方案是可以使用的，有什么可以与Warren Toomey [wkt@cs.adfa.edu.au](mailto:wkt@cs.adfa.edu.au)联系。

一些新组的搜索

### C.2.1. BSD特殊的新组

- [comp.unix.bsd.freebsd.announce](#)
- [comp.unix.bsd.freebsd.misc](#)
- [de.comp.os.unix.bsd](#) (德国)
- [fr.comp.os.bsd](#) (法国)
- [it.comp.os.freebsd](#) (意大利)
- [tw.bbs.comp.386bsd](#) (繁体中文)

### C.2.2. Internet上其他的UNIX®新组

- [comp.unix](#)
- [comp.unix.questions](#)
- [comp.unix.admin](#)
- [comp.unix.programmer](#)
- [comp.unix.shell](#)
- [comp.unix.user-friendly](#)

- [comp.security.unix](#)
- [comp.sources.unix](#)
- [comp.unix.advocacy](#)
- [comp.unix.misc](#)
- [comp.bugs.4bsd](#)
- [comp.bugs.4bsd.ucb-fixes](#)
- [comp.unix.bsd](#)

### C.2.3. X Window系

- [comp.windows.x.i386unix](#)
- [comp.windows.x](#)
- [comp.windows.x.apps](#)
- [comp.windows.x.announce](#)
- [comp.windows.x.intrinsics](#)
- [comp.windows.x.motif](#)
- [comp.windows.x.pex](#)
- [comp.emulators.ms-windows.wine](#)

## C.3. World Wide Web服务器

### C.3.1. 论坛， 部落格， 社会性网

- [The FreeBSD Forums](#) 提供了一个基于 web 的论坛用以 FreeBSD 相关讨论与技巧。
- [Planet FreeBSD](#) 提供了许多由 FreeBSD 用户部落格摘要的集合。 很多的用户都在上面发表有其他工作需要的帖子， 新的补丁和工作进度。
- [The BSDConferences YouTube Channel](#) 提供了一系列世界各地 BSD 峰会的高质量视频。 它是一个不错的观看重要用户展示最新 FreeBSD 有成果的方法。

### C.3.2. Official Mirrors

[Central Servers](#), [Armenia](#), [Australia](#), [Austria](#), [Czech Republic](#), [Denmark](#), [Finland](#), [France](#), [Germany](#), [Hong Kong](#), [Ireland](#), [Japan](#), [Latvia](#), [Lithuania](#), [Netherlands](#), [Norway](#), [Russia](#), [Slovenia](#), [South Africa](#), [Spain](#), [Sweden](#), [Switzerland](#), [Taiwan](#), [United Kingdom](#), [United States of America](#).

(as of UTC)

#### Central Servers

- <https://www.FreeBSD.org/>

#### Armenia

- <http://www.at.FreeBSD.org/> (IPv6)

## **Australia**

- <http://www.au.FreeBSD.org/>
- <http://www2.au.FreeBSD.org/>

## **Austria**

- <http://www.at.FreeBSD.org/> (IPv6)

## **Czech Republic**

- <http://www.cz.FreeBSD.org/> (IPv6)

## **Denmark**

- <http://www.dk.FreeBSD.org/> (IPv6)

## **Finland**

- <http://www.fi.FreeBSD.org/>

## **France**

- <http://www1.fr.FreeBSD.org/>

## **Germany**

- <http://www.de.FreeBSD.org/>

## **Hong Kong**

- <http://www.hk.FreeBSD.org/>

## **Ireland**

- <http://www.ie.FreeBSD.org/>

## **Japan**

- <http://www.jp.FreeBSD.org/www.FreeBSD.org/> (IPv6)

## **Latvia**

- <http://www.lv.FreeBSD.org/>

## **Lithuania**

- <http://www.lt.FreeBSD.org/>

## **Netherlands**

- <http://www.nl.FreeBSD.org/>

## **Norway**

- <http://www.no.FreeBSD.org/>

## **Russia**

- <http://www.ru.FreeBSD.org/> (IPv6)

## **Slovenia**

- <http://www.si.FreeBSD.org/>

## **South Africa**

- <http://www.za.FreeBSD.org/>

## **Spain**

- <http://www.es.FreeBSD.org/>
- <http://www2.es.FreeBSD.org/>

## **Sweden**

- <http://www.se.FreeBSD.org/>

## **Switzerland**

- <http://www.ch.FreeBSD.org/> (IPv6)
- <http://www2.ch.FreeBSD.org/> (IPv6)

## **Taiwan**

- <http://www.tw.FreeBSD.org/>
- <http://www2.tw.FreeBSD.org/>
- <http://www4.tw.FreeBSD.org/>
- <http://www5.tw.FreeBSD.org/> (IPv6)

## **United Kingdom**

- [http://www1.uk.FreeBSD.org](http://www1.uk.FreeBSD.org/)
- <http://www3.uk.FreeBSD.org/>

## **United States of America**

- <http://www5.us.FreeBSD.org/> (IPv6)



## C.4. Email地址

下面的用□□提供了与FreeBSD相□的□件地址。如果他被□用的□， □个列表的管理□有收回的□利。

| 域                   | 工具              | 用□□  | 管理□  |
|---------------------|-----------------|--|--|
| ukug.uk.FreeBSD.org | Forwarding only | <a href="mailto:ukfreebsd@uk.FreeBSD.org">ukfreebsd@uk.FreeBSD.org</a> | Lee Johnston<br><a href="mailto:lee@uk.FreeBSD.org">lee@uk.FreeBSD.org</a> |

## 附 D: PGP 公

有些时候, 您可能需要校验名或者送加密的件官或者者, FreeBSD.org 用密可以在 [pgpkeyring.txt](#) 下。

里了方便而提供了一些密。完整的

### D.1. Officers

#### D.1.1. Security Officer <[security-officer@FreeBSD.org](mailto: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/6RK09
K8DaEvSIqD9b0/xgK02KgLSdp0Bucq8HLDfYUk3McFa6Z3YwjobNCWkxc72ipvVl
uA0GN4H6fuoY0peg4cLk1H9pktUIrZONTCixaZzc/Bu6X+aX4ywGeCfsuu8g5v03
fLCPBLLgf3Bm5wsyZ6ZaGmsmILrWzd+d/rbr35Mcc5BekdgywUI4R191qo1bdrw9
mEJP1V7Ik3jpEx0sNnuhMTvm50QMeCtfUvVE0tBU15Qtbt+1LXF5FI0gML0LwS5v
RHZN+5w/xvzSnEULpj24UuMKLds/u9rj8U/zET8QaE+oG7m/mr4jJWZEmdX8HKd0
Wrpnvj6UAppk72qdBIEfLsOW2xB/NOjJpppbCQH3+sw7DRYA2UnKE9Mptj/KKiE4
cs4c8Cupo2WSu931EZDC5rCrULpT21FeEXnRYLC/5oIgY5w9sFide9VI4CzHkkWX
Z2NPW/i1w3mFhoXjvnNLG0YMFAMKPxsRC2/Bn3bY0IhKvuIZ4rAeu7FTmKDDqFKQ
YEcrUOW74ZVng17AB29xzjWr4zNJVvp/CybFiUb8JoKkwtVWRqAVZIEgenAjU40d
G5+W4e+ccL0mfTQfEBbXRjnL2BL2tnaoBR42cTfbZGRucPHz7Mr1KBEEZQARAQAB
tDdGcmVlQlNEIFNlY3VyaXR5IE9mZmljZXIgaPHNlY3VyaXR5LW9mZmljZXJARnJl
ZUJTRC5vcmc+iQJUBBMBCgA+FiEE/A6HiuWv54gCjWNv05eS9J6n5cIFAlmT2+AC
GwMFCQoek4AFCwkIBwMFFQoJCAasFFgIDAQAChgECF4AACgkQ05eS9J6n5cKd9A/9
Fz3uGjNy28D0ALT1d/JJGzdQ2R3YwspHk9KHBrl1ePkog9wf1WRalwCeNtPmA+g5
cn24psuz0eh1tRElImTZ2eE2ENPZ9XzK/J0ok0nK42MvmIwmMCyz+CaWv9GXW+FK
0oXnFmHi4YaQUVN3p+45TGkd9T+05biVww7P47n/NnWsTfhLx0bzC7LyjPKXINai
/LgPgt1c0gY65/YhW/qhADCKoU7qMp9is41jMjTu1WB30BPJkUkNpHfu6r15y8FN
Wqsk7K4W60br/WQ6VKGGXgh/a5mTcaEoFGM016uHiJAY4nXeb2HGZ1BKxgmPH9Ur
aT4A9Pz/n+rIRMrK+rs+msFPemQHhNBxy+x99uBpRBNT2Su6GouZIxu5J16aIM
V0ZyOy/dy7m/uJ4sMhJPqKkd8a+MoQs/2L1M1y1EAzs0/QZqIrKrCluaftNN9k/B
qU0XC1SDqB6sRMF7HFzYqb+f+M6cwSL/3Cp1Yx4rZ/onEE/MdWp64+3R87dETTXd
5tWXQw04q0hfPri5cBTI7r3t/qM01iNXCgSG5RJbGkas6N6t6Mj83L4ItjI8doLf
aSIWZjj1XP3/me2hfJ6h2G5y5A+kh04ZwhC0ATFSq1fYbVGHw5AtfthIgNn8FoWu
+Sb8h7/RqTr7F6LgWagAoAh0GtVj02SVABZjcNZz/AKJAjceEAKEACEWIQQc9/9v
rfXKn74bjLLtZ+zWXc9q5wUCWZPcTAMFAngACgkQ7Wfs1L3PauflkRAAgYcaBX0Y
ic4btXKoP/eOVpgUciOPPKehDCiloQDyf4XQnZFDomfjgcHpbLTBZ6kiAz2UzDGr
fJ4yUqrD+xfixUfCd5YpwzsaSpCGzDzSx0BcP/SpuAFhe40awSOIf5MrUQar9Mlf
33Jys1DLULXXeewAq2pcGk0/Wrr0ragI6Cs2vPGy9XP96VvLxyhjrWjLKmn0+//w
UF8oIO5hhKoqbt0xxlcqJgsWVYhCh0mnPzvr6GWwoPhFXocnh1oPdbLjX1AwmGm9
ltEYMge4QxONIX1XJR0TvuDuJOaLNVtOC30I8L97fdBcZS7eNJR65FAYR5Ft3ISf
```

```
KJowIsSLGDt/cYApqpyP2pv7FpCvnwHgXHYar7/q4zhngCFRQ2DPuX1cIJQ3Bgh
HZoLKyK1X7XE5ZVdfZ3s3gcHSVKS89p1pgHHZnr4sSm0anA8rXHcyHS4o2zSi1ie
r4iBwnOk6cCd6UNzEIiq0y/XhP/sc7xeL0mn3wDuV7jDBP9sp65sexL1qtIAfnzL
pLQevm0z41ifrUH5nNeL6RdbXpaoXc8M4PJJeQKJD04KzLcQpZdUdCJsbS6Q09w
srWR8enQXPEhz2C04L77bM9TgY029222jTqEPcbXcmx/k1x01rpsTTHUnHHi1Z
LUGYCbZPt+laTJ2YPHTjUtN1Jw85vSKCEuJATMEEAEKAB0WIS7KNQLNg7uk2rt
FW/l97zLo73d+AUCWjSYRWAKCRDl97zLo73d+JKyB/9N5Ytao12nD5QzMLvceGh5
otCLN99TUryYiDVL0nKBivq3jHQA/hOX2rweueFq0+LF8/2DnglJuUICntCxIzL
WXXf/Hr5iWBUQ0JxYNPQzzjdMSXGE0WmWYVpAbCGxHpIsetKLdHUCwneYhaywe3I
KzmRJSJDGV1IJB0sAfoFtgybZXHgIR61jQjtnNmmYXliYCd0wmIhXQDFN91tzzG
+EzdJ3Fao9JsMC+x55j06EOLVysZGRF5E8vCeKUWemQciKFC7EhKcljILPYAA21u
NmHCAgRHKWU9JmDFK0w9lQuN2HQaNFkahjarTNM/Q6LwxY0dLG0vVYife085WFAf
uQINBFmT2+ABEACxi39m5nQZexzY3c9sg/w5mUYCD89ZNSkj427gduQMYYGn7YW6
jSPfVJ/V3+PDK824c0a0XasyDapQFY1CPTZYrReRPoyjb8tJjsSVGXCTFpJZlFU
br6kS9mgcx58Sypke2PMV73+W1N1Yco+nahfTECRuM2/T2zHHr0AdKuBPF28U+H
TxyLatKoIgQwHds4E/f4ZTbAoHvu3PixA17XHvXCgz0cHaLhRljXizbZDXngOdGm
lqdfLAIpL6/l8E3m1Er0m3IfFo6qSzWRHg/KaBGIL4YKetJ6ACjlkCe5qbatDpmk
gWlg3Ux4RBVjyCK834Xh7eZpEcNf2iwpM28glWh7XMHGUpLTHkU3PWQ4vGfNxB8
HB0d9r02/cHL6MiHwhCAfIzZGVtqR0i9Ira57TMDXTPJWNXUcgsCMsi/Bg2a+hsn
aiYlRZc18uNL5nq0qsqKG3c1TcmeN7nbxVgnrNST4Ajteu1khmB9p8tNOXA3u979
000T5LPwdqIpobdZ0lfw4URnAGw4Wd4Sm9PtRw0RvuAk2M2e5KXNyxPWAuMVkoRR
a7wG6h/R8pki54Gexyc+Jkfb4Zc0rzHNLurw6DhxroyfRs8WEgX0wNIgmJvCXSBG
54jb5w9qudYwzIg4YPfVUx8sfeY8MTNha13rF0tvVloGj3l709wlaWlBYwARAQAB
iQI8BBgBCgAmFieE/A6HiuWv54gCjWNV05eS9J6n5cIFA1mT2+ACGwwFCQoek4AA
CgkQ05eS9J6n5cKhWw/+PT0R4r2gPAxi8ESEe380BY0mneNAH24MFOgWXqWCj4zX
Uz992BVnW2aL5nH405d822LGeCrYUC7SCpQvliFdHZHjobgtizLTWuu40bc3gS0z
cxWlx2jKfx3Ezn6Qz2mhhK6fZ1A000biQxQq25ldURep95L78E/C8XkCe11YlUR
ng3wQKeHM7awZWRw/QBC92haHuVtU3cx7At+zQL7jTBKSZqd34zss0uoXIhk2h94
007MMDZ8z8MeU337vdL+RKYtD2bljLwpf7/kqg1D/q44RJ4ZpZcha9G0GvtLaQg2
+MAP1Lg1vOWZ8wOTLaQHm+uzYRpkqXkIV80uVd4UikCd8t3VNjNG5rG/YRNIAX0A
UEzs6oMF5Y0FE8LmykesBUHAbC07Vcb0AsT5u3XKixDiIpPdnYSwG1kvo0VVLdeh
q/aXlK9V8BpViG5+a8xP2fdF1eMqdnrKAsi04GEiq193PN/FA049VeIs3fd0izAa
x7+ag1MGtoF5Pij5iTVJm6phH5SUd1P3FY30mcLxWj/MbL4ba/G/6FWcy5NXxdw9
L1bRqaM2KEHJ67aF6NzZ7UmlDwExAWzFbUon1LUpKysAukxVf0EnntydbEVOQ+JO
HdqEpirrVLMpxPttUB2xxbo947nmj7/Bnme2gvb0vxaC9xSGVxrpW9cg5iCwSdc=
=8rds
-----END PGP PUBLIC KEY BLOCK-----
```

### D.1.2. Security Team Secretary <[secteam-secretary@FreeBSD.org](mailto: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  
NAZr20x1BaSgX7106Psa6v9si8nxo0tMc5BCM/ps/fmedFU48YtqOTGF+utxvACg  
Ou6SKintEMUa1eoPcww1jzDZ3mxx49bQaNAJLjVxeiAZoYHe9loTe1fxsprCONnx  
Era1hrI+YA2KjMWDORcwa0sSXRCI3V+b4PUnbMUOQa3fFVUr iM4Qj jUBU6hW0Ub0  
GDPcZq45nd7PoPPtb3/EauaYfk/zdx8Xt00muKti9/vMkvB09AEUyShbyzoebaKH  
dKtXlzyAPCZoh9dihFM67rhUg4umckFLc8vc5P2tNblwYrnHgLyUa0IjZB/f0i  
Z20ZLVCiDeHNjjK3VZ6jLAiPyiYT61Hrk9E8NaZDeUgIb9X/K06JXVBQIKNSGfX5  
LLp/j2wr+Kbg3QtEBkcStlUGB0zfcbbKpE2nySnuIyspfDb/6JbhD/qYqMJerX0T  
d5ekk1JtXtM6aX2iTXgZ8cqv+5gyouEF5akrkLi1ySgZetQfjm+zhy/1x/NjGd0u  
35QbUye7sTbfSimwzCXKIIPy06zIO4iNA0P/vgG4v7yDjMvXsW8FRULSecDT19Gq  
xOZGfSPVrSRSAhgNxHzwUivxJbr05NNdwhJSbx9m57naXouLfvVPAMeJYwARAQAB  
tD9GcmVlQlNEIFNlY3VyaXR5IFRlYW0gU2VjcmlvYXJ5IDxzZWNoZWZtLXNlY3Jl  
dGFyeUBGcmVlQlNELm9yZz6JAj0EEwEKACcFAlJBjIICGwMFCQGh7b8FCwkIBwMF  
FQoJCAsFFgIDAQAChGECFAAACgkQPLlQ/MPWxmYt8Q/+IfFhPIbqglh4rwFzgR58  
8YonMZcq+50p3qiUBh6tE6yRz6VEqBqTahyCQGik4xGzrHSIOIj2e6gEk5a4zYtf  
0jNjprk3pxu20g05USJmd8lPSbyBF20FVm5W0dhWMKHagL5dGS8zInlwRYxr6mMi  
UuJjj+2Hm3PoUNGAWL1SH2BV0eAeudtzu80vAlbRlujYVmjiDn/dWVjqnWgEBNHT  
SD+WpA3yW4mBJyxWil0sAJQbTlt5EM/XPORVZ2tvETxJIrXea/Sda9mFwvJ02pJn  
gHi6TGyOYydmubu0ob9Ma9AvUrLxv8V9eN7eZUtvNa6n+IT8WEJj2+snJl04SpHL  
D3Z+l7zwfYeM8F0dzGZdVFgxeYBU7t3AnPjYfHmoneqgLcC00nJDKq/98ohz5T9i  
FbNR/vtLaEiYFBeX3C9E96pP6BU26BXhw+dRSnFeyIhD+4g+/AZ0XJ1CPF19D+5  
z0oJanJkh7LZn4JL+V6+mF1e0ExiGrydIi i SXDA/p5FhavMMu80m4S0sn5iaQ2aX  
wRUv2SUKhbHDqhIILLeQKlB3X26obx1Vg0nRhy47qNqn/xc9oSWLAQSV0gsShQeC  
6DSzrKIBdKB3V8uW0muM7LWAoCP53bDRW+XIOu9wfpSaXN2VTyqzU7zpTq5BHX1a  
+XRw8KNHZGnCSAOCofZWnKyJAhwEEAEKAAYFAlJBjYgACgkQ7Wfs1L3PaudFcQ//  
Uim7EXsIHLwHxez32TzA/0uNMPWFHQN4Ezzg4PKB6Cc4amva5qbgbhoeCPuP+XPI  
2ELfRviahbmyZ/zIqqpLDC4nmyisMoKlpK0Yo1w4qbix9EVVZr2ztL8F43qN3Xe/  
NUSMTBgt/Jio7l5lYyhuVS3JQCfDlYgbq6NPk0xfYoYOMOZASoPhEquCxM5D4D0Z  
3J3CBeAjyVzdF37HUw9rVQe2IRLxGn1YAyMb5EpR2Ij612GFad8c/5ikzDh5q6JD  
tB9ApdvLkr0czTBucDljChSpFJ7ENPjAgZuH9N5Dmx2rRUj2mdBmi7HKqxAN9Kdm  
+pg/6vZ3vM18rBlXmw1poQdc3srAL+6MHmI fHhRq49oksLyHweL8T6B04d4nTZU  
xObP7PLAeWrdrd1Sb3EWLZJ9HB/m2UL9w90m1c6cb6X2DoCzQAStVypAE6SQCMBK  
pxkWRj90L41BS62snja+B1ZTELUuLTHULRkQW3fFkUxLDSMUn96QksWlwZLcxCv  
hKxJXOX+pHAiUuMIImaPQ0TBDBWWf5d8zOQlNPsyhSGFR5Skwzlg+m9ErQ+jy7Uz  
UmNCNztlyGrKeckXuvr73seoKoNXHrn7vWQ6qB1IRURj2bfpshqlmYuITmcBhfFS  
Dw0fdYXSDXrmG9wad98g49g4HwCJhPA10j55f93gHLGIRgQQEQoABgUCUKG05gAK  
CRAV1ogEymzfsoL4AKCI7rOnptuoXgwYx2Z9HkUKuugSRwCgkyW9pxa5EovDijEF  
j1jG/cdxT0aJAhwEEAEKAAYFAlJBkdUACgkQkshDRW2mpm6aLxAAzpWNHMZVFt7e  
wQnCJnf/FMLTjduGTEhVFnVCKEtI+YKarveE6pcLqKJfSRFDxruZ6PHGG2CDfMig  
J6mdDdmXCkN//TbILRGowVgsxpIRg4jQVh4S3D0Nz50h+Zb7CHbjp6WAPVoWZz7b  
Myp+pN7qx/miJJweiw22Eet4Hjj1QymKwjWyY146V928BV/wDBS/xiWfg3xIVPZr  
RqtiOGN/AGpMGeGQKKp1keITY7AXiAd+mL4H/eNf8b+o0Ce2Z9oSxSsGPF3DzMTL  
kIX7sWD3rjy3Xe2BM20stIDrJS2a1fbnIwFvqsZS3Z3sF5bLc6W0iyPJdtbQ0pt6  
nekRl9nboAdUs0R+n/6QNYBkj4AcSh3jpZKe82NwnD/6WyzHwTc0SDRTVkcQWXPW  
EaWlmv8VqfzdBiw6aLcxlmXQSAr0cUA6zo6/bMQZosKwiCfG13tR4Pbwgvybyjoi  
pF+ZXfz7rWWUqZ2C79hy3YTytwILVMOnp3MyOV+9ub0sFhLuRDxAksIMaRTs07ii  
5J4z1d+jzWMW4g1B50CoQ8W+FyAfVp/8qGwzvGN7wxN8P1iR+DZjtpCt7J+Xb9Pt  
L+LRKS0/a0gOfDksyt2fEKY4yEWdzq9A3VkRo1HCdUQY6SJ/qt7IyQHumxvL90F6  
vbB3edrR/fVGeJsz4vE10hzy7kI1QT65Ag0EUKGMggEQAMTsvyKEdUsgEehymKz9  
MRn9wiwfHEX5CLmpJAvnX9MITgcsTX8MKiPyrTBnyY/QzA0rh+yyhzkY/y55yxMP  
INdpl5xgJCS1SHyJK85H0dN77uKDCkwHfphlWYGLBPuaXyxkiWYXJTVUggSju04b

```

jeKwDqF1/4Xc0XeZNqWVjqHtKF91wwgdXXgAzUL1/nwN3IglxiIR31y10GQd0QEG
4T3ufx6gv73+qbFc0RzgZUQiJykQ3tZK1+Gw6aDirgjQY0c90o2Je0RJHjd0byZQ
aQc4PTZ2D07CElFEt2EHJCXLyP/taeLq+IdpKe6sLPckwakqtbqwunWVoPTbgkx0
Q1eCMzgrkRu23B2TJaY9zbZAFP3cpL65vQAVJVQISqJvDL8K5hvAWJ3vi92qfBcz
jqydAcbhjkzJUI9t44v63cIXTI0+QyqTQhqkvEJhHZkbb8MYoimebDVxFVtQ3I1p
Eyn0YPfn4IMvaItLFbkgZpR/zjHYau5snErR9NC4A0IfNFpxM+ffFJQ7W88JP3cG
JLl9dcRGERq28PDU/CTDH9r1k1kZ0xZpRDkJijKdNfIXt2ajijVOZx7l2jPL1njx
s4xa1jK0/39kh6XnrCgK49WQsJM5If1VR2JAi8BLi2q/e0NQ62pgn0QL695Sqbbp
NbrrJGRcRJD9sUkQTPmsLLQTABEBAAGJAiUEGAEKAA8FALJBjIICGwwFCQgH7b8A
CgkQPLLq/MPWxmZAew//et/LToMVR3q6/qP/pf9ob/QwQ3MgejkC0DY3Md7JBRl/
6GWfySYn00Vm5IoJofcv1hbhc/y30eZTvK4s+BOQsNokYe34mCxZG4dypNaepkQi
x0mLuJeU/n4Y0p0LTljhGLVdKina2dM9HmllgYr4KumT58g6eGjxs2oZD6z5ty0L
viU5tx3lz3o0c3I9soH2RN2zNHVjXNW0EvWJwFLxFeLJbk/Y3UY1/kXCtcyMzLua
S5L5012eU0EvaZr5iYDKjy+w0xY4SUCNYf0GPmSej8CBbwHOF2XCwXytSzm6hNb3
5TRgCGb0SFTIy9MxfV5lppdQcdzjmuF5L8LySkL2yuJxjLI7uKNDN+NlFODIPMg
rdH0hBSyKci6Uz7Nz/Up3qdE+aISq68k+Hk1fiKJG1UcBRJidheds29FCzj3hoyZ
VDmf60L60hL0YI1/4GjIkJyetlPzjMp8J7K3GweOUkfHcFihYZlbiMe7z+oIWEc7
0fNScrAGF/+JN3L6mjXKB6Pv+ER5ztzpfuhBJ/j7AV5BaNMmDXAV04aTphWL7Dje
iecenUGTpkK8Ugv5cMJC4QJaWDkj/9sACc0EFgigPo68KjegvKg5R8jUPwb8E7T6
LIjBtlclVhaUrE2uLx/yTz2Apbm+GAmD8M0dQ7IYsOF1ZNBW9zjgLLCtWDW+p1A=
=5gJ7
-----END PGP PUBLIC KEY BLOCK-----

```

### D.1.3. 核心秘密 <core-secretary@FreeBSD.org>

```

pub   rsa4096/D8C8C83B49F26F17 2020-06-26 [SC] [expires: 2022-06-30]
       Key fingerprint = 4B64 E9E0 BDE9 B3EC C06B 5C66 D8C8 C83B 49F2 6F17
uid           FreeBSD Core Team Secretary <core-
secretary@freebsd.org>
sub   rsa4096/377C937536E4821B 2020-06-26 [E] [expires: 2022-06-30]

```

-----BEGIN PGP PUBLIC KEY BLOCK-----

```

mQINBF72HwABEAC5h14kfH8DyRpp0WE5rwnuS+wQ51EVTGs1vLho80Z2XruzlQT
AezCnKLSqMgD/UEaBcn9kbKoeqp2sIwuEUX+P79KhRc4C8RJ8TMfDH00tC091QVp
MYWbIsVZYC004K+rN1Dbk2En3B0JVgTowqbZzR3hPvzeU2/P+Y3zMtpQGea2DB5d
24Q/tIuPMh89evEX0x0K5eM/4P2awSmA3J+h+r09UYjKejJ50BUJQsMervWAHgCA
TxJQH0Pxw+ZKpJB3dzyHKTmukVZhdCjK6Zt2tih/rO/CHDsitMgYRIl3w2X6pDfV
J0pv0Blzg7nooIw94v6Uxr2y/JWg0Gh2qy07u4qE//y6uS155s+Vq5TrFr79VSwB
GhY9As/0Dk1lyFisKp1/yiet2W7Pu4c99Z5dsrQPSTLFvkvonVRX8wgxRZwk6gWA
LEYklwoR0NXiqLrpBT10Tsnsa4aoUvZW6eyOWZrKsdsVn05sgRmvlfpqBbwqlDJ
0EeF/MztPuhmq4Hgn+DmmYnx/P85pZpThcfJx16VxS8nB7ExYljeC9LF8V8/1d7e
tfgAj8ezzntr2TXSZ5gb1QtYlJkdgBiBZqsxHPYHzfG8Zx3eYs2Myklf9p4lt7nv
atTroDt8pUGXfhGfoqSHSLXODfYA09/7DOPqTy5Pan4i7aWBPP+gfK0kgQARAQAB
tDhGcmVlQlNEIENvcmUgVGh5bSBtZWNYZXRhcncgPGNvcmUtc2VjcmV0YXJ5QGZy
ZWVlc2Qub3JnPokCVAQTAQoAPhYhBEtk6eC96bPswGtcZtjIyDtJ8m8XBQJe9h8A
AhsDBQKdX60ABQsJCAcDBRUKCQgLBRyDAgEAAh4BAheAAAoJENjIyDtJ8m8XQFwP
/RqHPMSsLlTcq5NfK2MAVGmdtpL5wf84bchVWtcXUUEwXW1wI2cdDwu9SoqudDbP

```

2LrbMpxWeUWAgCpPCF/vCVo4Nzd0zb1cEGKRKFZe/4EQ8dfvqr03YyupSQvx6+P  
oY+8y3kl7iHJKBkwrASraB2p+N9XDAJDgqz+1M2Xbo7rcJx64wBOCyPAxd9JWsge  
d8mXyAqZlRlihsTjLbhuYbJxpKM5YjGubVaQZaNI DxUduqc8Pt9VgHvWJBc9VPPA  
3B6E9/PUFZYZeZQSROkYniN9NE7keitxj/rvZkpzcaXfAoDMC7CSolBz1P+CJZ+i  
Kk7IWz4JpxiYkE/IY4VvMMYms9tRP8fVv0+R7r7yKEA9SSlH+e9qC++0oWg4b+wV  
OrWtVIWvaJctj5ZAPCut6ZxBdvXEBHd/Gv6uCzG86n4huz23U+Y4iLzoAlVeInQs  
Hqu1wSAUBNpplyeZ1TvrGg2pufxLh8iXfh0npDP/6J+u0GUfeX4JoAzvxlatXMYI  
fBmqmcZI6ShJN8qQtCUa50Mqbnieo7Fmpf8BsLegjAsQ+8w21ATD2boinStntLzF  
/yoL/z9WYxmoOdHYcQ8bil djCvtbAKrZie8sI4SgWQz2UX6KX9sc/WOmWUEtjdqB  
WfGratZNoxQLUvEDftt7r9ts1jKVUL3dMPTCfU4wcj5iQIzBBABCgAdFiEEVbCT  
pybDiFVxIrrVNqQMg7DW754FA172J74ACgkQNqQMg7DW756LaA//Z3CCF5fQ08tx  
RLeqHNs55xCYS97TjZxY6xAMBjebkS+ABkgdbedSH+YNGfdaGSD/SMTvMAmNx55t  
18DDdA4pqC5x2USaHjXFdbDdxKuKMAoSatOpipVASVmW0FkZI5C5FDe3MF8+mfGb  
EPHVPwKbo7R5tk4jUPyX8wUa0AyUX9fyQnwDxN+zThvKwnX/+qwpokaY2N4ZOI0w  
rOF1kkczibbfwvjVYcpPovGALmTccnWo1Xvpkh1lg93Y21mH+T2Ub/BK3GhvgJQi  
WwiDtMweUnPLp4W14510U10yGzeT/XwuMPH9dsKz5Iw4/g1zqQEtZj2Gc0DP5we  
HM50doTn+dVIF+WCFLhPYm0RSf8Zj8ngbX/HV2UYLB5k+uNT9YTnBVEdKVydx7Cp  
Ip1C7XAPJEFtUk7w17YC6n5P5Yo1C7DSJlwcAjxdbffXLowBhgy0q+EJJgnqerZl  
r4db58h2epIHRKgnS15z4KoAGW105dFShBz1UYPj4cZdeE+twpcgEg3/7LMzPzF/  
xQAQZ89axxXBaCPL+YVsuMJSerbNdPp1SjCs9e8Vev91tLFmt/sY4IpbvPHZavG1  
/4ealh8E1zPgflVW9TPRUY6mjN/uDI2y39tk2EofZ0cSQhLEM6gRW8uV4q92cWM  
V55hu7Vs2RrKA7fve9y+YBi3DdTwWHSJATMEEA EKAB0WISfAoNvUNOtWrDaxYgM  
tAPk6VuW7AUCXvY98wAKCRAMtAPk6VuW7CDLB/9PSUSMV/pnC+X4ougpjpqfSjF8  
5bozjkKSkNqXZmt2vJVImc/oSK13awq46FC4rAhk591T3kaH6EKvDHQ5G8Twi07u  
VotCdtfMjXgPV6RLmo6Hps0E1nzmbsum6xeemRDf3D3n1kAdUteXNBxHTIdAbeY  
p4Wxu46CC/SqD6HbnUF2o+/6dXXyV11TnViIj6m5eFD20Q4Jdq7GPsSjSS2XL4f9  
jHZUOUJyyA0aFwjJ+SczMKXSunyiOC14uUHDcgivLIRyZ/giWoQpr8sAgHXCh82h  
T3BmbHgmcmgMh+wNxH87IPwUU0CKRd2dL5k0SZVCFuMnFsc9eIie5kMEJwPuQIN  
BF72HwABEADT9L4GIYiFaYg2QbQ3wsmmFnP/pAZiHDxXI6wL6xCKj6o2sc1/b5j3  
ILEiAqZ5ZenXX6T7Epja10AskfsGo/n3vF18grSudIkXJPQXcb61fXU7xfmGAEU  
HWABQG+OD/HTvUPAITVck14LxVFkz3oqRnq13rxDk1XZYvLVWeBn8vWF4/gLz9k  
etfLw71Pk9f86BuNb0vCPnWpOpZa0xKlabdGpMKDD+1RYC/L+ZEwKiLBfgXTzK3g  
IWAX3kTrQjKBZzsQ0s5TFWkm+z80GVUq8HK1XU0uF8s7cX+KXGU2kYcC8DQrxPdL  
jYm6N8ax0n4RR8eP5ZFA0W7qMieFSHAjqCs4srdN1bGC3nS0zGsQCvtTRBbu0nen  
06uwzWQgtZWVfV+dqaEH2crnhn5CUI0A8jdbFBGDIBbWJz/QfRray1CEc8q+hZFM  
OLBsVXRdVe6hUXTVeG9cXanXC+0o3nnc7WhWr1caTbbhnzLEbME8u2oLif7rkhc7  
FanuQEYKa76J1zou08ZeLK/pUFXTbRCoyUEVL+VIxLESCWi1ptkDpiZey3l6fe0Q  
WWRMLFmpbu3WTN121bEwfRL03+fP1q+yGAV5hyJv/EMl dd76v577dAoLIsth+aDP  
PMJ7mJ5NwOuic20HI1cjuVT5A2pBIzFfraZY/v4dzoaOpXZjEz9wIwARAQABiQI8  
BBgBCgAmFieES2tP4L3ps+zAa1xm2MjIO0nybxcFA172HwACGwwFCQPhrQAACgkQ  
2MjIO0nybxcflQ/9FYvM/LBSzy4VF0jNsUkRtjmPtyw2dJmQOCbWoSHmibRCG26a  
Upt5lp1n4LG/qEtDlus5mDETL+/TnYhCG+hhnHADc87goLwBwL37yK1NAYv0y2rm  
TddjDT5vZW0yzHjHqIJ1NlxQ40jMi/XjyHIzb0PGNayFVi3XkLVxWZI+lwON1btWk  
gpFfEgqRqQbJxM2cSEqimkfrre+b2/M4cGX9rThpTtpfpbyHjTsS6juo4/eIdnBA  
UXpKce4Q9LB5zxDaakKoDVxxkc9R0HAAoIH4u+Fu8az+CuH2sJcVJWK7Nxt++N8  
Xhj+FUS+Ay8siu+ScQjs0HOHRwr6a+6NT58eylwR5hwo tmnzJHLZReqknoAjLEGT  
d33jzKM/y60qPe/oPGj2b13RkA2vRnCPm33+T57sLMonNe6hh1Xs9VTgXxSAzfMa  
cmV0dP+nxUsoc3MtqjE2z2BcI9WmmJFeEgE2BOj703CQuot+8jcZFXGUW+i6V1a  
k7dZEMDSbALNzxaRNgeJC6HiM1+dXFGLNHEIgBLGwdvFAxTfNaUVK0p7skDWEx44  
giaUjZYpQ21+SHjVKTUnFQiiIDORvs3jdZDaxK/Y/vSolRUiLBiHZWa6mxQY4uc6  
5nAzLZB2BiBRfdL8fE0154nWjAZBLbKhK+ke2DBoPvSWubLPJqZyh+GmZAE=



```
=3AI7
```

```
-----END PGP PUBLIC KEY BLOCK-----
```

#### D.1.4. Ports 管理□□秘□ <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-----
```

```
mQENBFA0zqYBCACYd+KGv0/DduIRpSEKWZG2yfDILStzWfdaQMD+8zdWihB0x7dd
JDBUpV0o0Ixzt9mvu5CHybx+9l0HeFRhZshFXc+bIJ0Pyi+JrSs100o7Lo6jg6+c
Si2vME0ixG4x9YjCi8DisXIGJ1kZiDXhmVWwCvL+vLInpeXrtJnK8yFkmszC0r4Y
Q3GXuvdU0BF2tL/Wo/eCbSf+3U9syopVS2L2wKcP76bbYU0io035Y503rJEK6R5G
TchwYvYjSXuhv4ec7N1/j3thrMC9GNpoqjVninTynOk2kn+YZuMp03c6b/pfoNcq
MxoizG1Tu8VT400/SF1y520kKjpAsENbFaNTABEBAAG0R0ZyZWVU0QgUG9ydHMg
TWFuYWdlbWVudCBUZWFtIFNlY3JldGFyeSA8cG9ydG1nci1zZWNYZXRhcn1ARnJl
ZUJTRC5vcmc+iQE4BBMBAgAiBQJQDs6mAhsDBgsJCAcDAgYVCAIJCgsEFgIDAQIe
AQIXgAAKCRDYKU7Du8TX1QW2B/0coHe8utbTfGKpeM4BY9IyC+PFgkE58Hq50o8d
shoB9gfommcUaK9PNwJPxTEJNlwiKPZy+VoKs/+d08gahovchbRdSyP1ejn3CFy+
H8pol0hDDU4n7Ldc50q54GLuZijdcJZq1g0LoZqWOYtXfK1KPZjdUvYN8KHAntgf
u361rwM4DZ40HngYY9fdGc4SbXurGA5m+vLAURLzPv+QRQqHfaI1DZF6gzMgY49x
qS1JBF4kPoicpgvs3o6CuX8MD9ewGFSAMM3EdzV6ZdC8pnpXC8+8Q+p6FjNqmtjk
GpW39Zq/p8SJVg1RortCH6qWLe7dW7TaFYov7gF1V/DYwDN5iEYEEBECAAYFA1N2
WksACgkQtzkaJjShBftuMwCg0MXdQTcGMM0ma7LC3L5b4MEoZ+wAn0WyUHPHwHnn
pn2oYD1fAbwTl0WiIQEiBBABAgAMBQJQGiE/BQMAEnUAAAOJEJcQuJvKV618AhwH
/ie5fi+wL0aapUHHs454xUv8xtDPfKpA35U4R2ZaVZ6wTCWW13by+i81YIiGFVmQ
uQkkms1vRFhY6fVYzOxQR3VhWTTfexgL1LdI88eoBlzlsIvv7/4bNT9dUcg2TeYS
Ah3TzZsmVbqXIg8XvrCBD/WhG1cXGonHKszP6RcyFSDDD+bQogONqujM94dIcuoZ
7VCqbUvFKJ+rI3uXA1XFZSgFI9cDtnKYQqpGjCDEH2VeSfX/4XFuTg64g93AcCvE
qHANGquSWBfJGAy9fqV51LrRp7wB1GLf00Vr0HSpn148wXDBSdi5T1AdfiBb8END
1fUfI/QfxHS5klkGE/akQNqJAhwEEwEKAAYFAldVzKcACgkQrbv4YQo3ibdgsg//
SS3vgpNtwiZHkUhfkAvuEhTLHUgJno0s55JT8JhApX3wAWHf5HJDtU+rpvthVHF7
iWUBNZyilDb2fKQVsLJtUQx7Mq6Cov1wddGtU3cf675VPPMrW3i31Ai86Aq3+qZD
EFqF0v96kunpVdFsdPQ7n5Pl4w9bI2tsLrLIRvMvtVggOSYz962XQ/TVXc60Q4JK
LU0VH7mDdSySDem5Q4B4+xVT+8Fu/kdHGRaDojbCyQgNBTFEosLPfFMrI2sC4iGP
XUyc+l1QNA4AdE2Vjt5log4yE3iq3RnKnzzYiYZr1CgZJ902KUzzX5KE0/6f6uQo
WJrtLmxnsktRCyo5pJSfg88p32S87512FzYK0XJHrqhK1DcTNCet7/JZn4Q0UThb
3qh1nadFzMV1ZkeWzLp0pXYORGBKC8kGzfkCD5w0wBDxqwrX9LfOEUMTUHH+e+v1
gW2X83kngBZqQRQnYnLGejI8LcI6+PqNeZ9UDE2XKNTWk6n/L18M5fcDOGmLYNTp
BeM/B3R7sudAhwnreJk+XiYMGVHxb13WzKR16axCNGPTji+GjIpwtXoBd3V7f9J+
e4ughPhNpN1CTzbzK7adB2gxJymfHH5AVBAC5FGIucMKp7R1diUYrKqEZx3byU+y
JpRzHu1mibInwSv0+D506GioPD+itU6Yfhci8VC2mI2JAhwEEgEIAAYFA1QYNs8A
```

CgkQ6rA8WL/cR49ZJRAA12CXRwt87Hce8A7CbLbVncpHGaNf1VC1HTdAQF09py1  
dLS5+LhYxEw0jIR2Vcc8CA03iwfJKshKXH2Z6FW6iSFCc9bCVzHTrcuwsGc5WZQM  
JlGiHxYmsyWNad/fCW8mH0q1SjTqcPcApg2+1j/XQa4rcL42i87vB0LI57TtEv+Y  
WvxA301N6qHAVE42wrRkfS6gJjx0F/PpYhdhH2TBKcXd6yVrGL9sv50qUoAvzKUH  
sJ8q+n/iUP+/NVKGQej6cQsX7uA+sZTsnREQVIPhXwwFmEfil/gxjmpUZVMmks/D  
b8ZfP29CJvwEicfWn6shGcSA6Y0ds7fIybd8Mdt2D/BEBLIMBfKSK498BUgUht2v  
3LJIgTVCIQoSEfYUK1T1GoTf48ewxgBsZPymvarERmEx30q/Y401EPobX3hQUQ0d  
J5GsjK0f1pP3AHAS1Wz5QacL580934fsC3BdxKwW8eY6fgaBXnvA+gE31L0MTHIF  
Wwz1JIDLmtxLSjFTtdI4SKF6FVfkoPRmFmZBMtbTkSp9M7HdmYFwtrESTpubbQkn  
WE0HXWzNQPY40rYkceIREtehgsL5C/D9jPfqsiH40MYx1N93/6/M6fQ8jU9VZa7  
cIRcbTqDHZq7G2Dgk42uF4NeISTYUWnNJBge24+sPe6fLjK0E9z5UjLaMq0cq025  
AQ0EUA70pgEIAMzcSWKX+T66lfKXUV5J9+hXu0auu9WLW2dlq0+w9A1WfGfgGbmZ  
z/7YmNUqzNjHqQngNsQQIShrFgVqNWkZ0Zn2TpLCrIl7Itl3gVYXWZQpKzjCgU1Y  
mp+yNZR+fWZzSDF9DZdLZY9y9SI311jb33o+N1EVfNUvRJksKY21m6Gra9gPtaCZ  
LuZu5zw/SSDnJBNU8wbMoce8yFDfc0jCq+4BcPAws8qcFCsmICmE6FZvEJZE/y9w  
dH7kB1oU+kThpOKPEgu9YmubXNU81MxhGdQEDvG2525iSwQyr3lhwqGngJEcyDc6  
A/cAJ/c+KZ4pc3ufULSgtXDq1JLE7wUg4esAEQEAAYkBHWQYAIACQUCUA70pgIb  
DAAKCRDYKU7Du8TX1Th6B/4uVcoIFXcKaj1/GAZjeZl0M4mGTUKIjjfCITrMaV+F  
tp1wo+ZujAAlepyQ1VesM1B/P1bw0AvZx2qwnaGjbb1yc9B2pmLeYBydBtW2LgX  
LiafxRIy2uPFxS0510IyR5/K5cGTIyvz1o9ZcNlmYnpl4rs0xAx/HN2ErRiUmvoG  
HvBMqt1yVQHtLUDoSJqFoBg0d7Vn4K8mvyHRBW/45ox0huV1H2yNLzcKpX/N9gak  
kCjxv2YseDJgSYLWm+Ee9Tx/FgZm5jRYw3pcVcn8/NPojIBG/Wi/dTH3gTQtigda  
RZOpdZkE7Iqu2o8FS1SSvDaZRwJHS5pAQjvDIDI8xoZRuQINBFJPLygBEAC5Ao/e  
SZr5n4MrTsZ0dcPMZ0Ab1MbWMX/gVsK1NCHzjAcQJlw0X233vQXNww2coJo62dEQ  
g3FSeDpgzUXP776E7Lf09LKOPXTNYdoul1IYQ+B8nDY5MdMACMmfY14nU9y8gIQ  
k0n/xo7aXdkQr5gLF3CR84YWyeXjNkgZF5jXZ+DHgv4Qm31H8zeuH5rBkeLSiiJT  
z2sskGkL/3Tj96HxIeElkXPCrYxA0rImh87VjFhCUCjf+50nlbsA6VSS/ZeM1Bgd  
E6pStKf53nKEJDqJZx1F+aCuxJop/U4z2AhqmABFrjNQW114mCHYr5RJt2VPr6qu  
gm0sP0ig2dmbiuFMwxkJ1u1snLG16mdfGnsIdZnX/ufmAcXl2s0Sb3xWqQf1r7Va  
ovOCzloqK9SATDINzwsGygo4R8F76FeVeJN5bYa51YMogk63i170yzYCCkEBVL8F  
tibHLSGtx4QYJIZCQUgj5qVgCGfXgNUboLxUPPd14bfl8vs991qsYFM4Q/GaJygg  
RGWtvWY+0niGF3GZLZsQm0beD2JIBaKu1Qvvd6rV7jmjrwtG/apJjY+RKR6M0be  
kjb9LA3v00E9C2nEekEvT6v2DIwLPgqN4MNI sDgJVb6YmX6BG+qbq39E89If1EUR  
1GZ4BAAbZ0XImcbgoPYPxmg/KP/j19J8zlh+DwARAQABiQM+BBgBAgAJBQJST5WI  
AhsCAikJENgTs07xNfVw0gBBkBAgAGBQJST5WIAAoJEMogMoV3Bk63xfYP/Aui  
Ybn8sZ2Dta5j1EQhVdHpt1o13MUJIUGtN7v7FPDhTcqdlDgfi8SN1wLBnuEZ3JTx  
du9yuRuTn1ACdoY1hRqzKlxjS5bLibwyQLJnR5yOnLMNUbzp9psjM1Ek6sT0Mp5L  
57boIaosl2xkGqMUCVVo67oaIHCJFWLzk+JzBlwspPUPrqKCzcPMac7GQcPN851o  
Y483prvwWzn412396nSvTVjJAX3GkfbA4UQPmGOG3K06Ey1xX4d4DDdaak7gDPHW  
xnNP20uefMjULhWifnF95W0i18+aImyYupLnci/RPwpFbzBVky3EEFgj0uaytHXL  
aHVtcfCZ8FZ2hVNLaxSMZkPsJOEteegnWME7/9S21hinFc/ToNXvcCC9i7FWiw/T  
iEZ8eBvHS6mSc+KZhhBEuwl+q6ybkqV8TTo0D7YYj7NM/6vXXqISiZFIi+rZZ9Dg  
oCPNtNxxki6txUwBXq9lUvptVzJANppNVr+rWXsmDJ2AGQgc3NehdszIOXVSp1d  
deGx5kU19G+UEDQeyVNwt+JHU2oIy6+Hc/+QYV4KH84z/R4FUCgqeyb03VE+zdMv  
5PpkTJpyvL51nWzmR+PCDhCw4TJFOUCIih6CWl7M08Fa0vC7UkCLXyKAP5uhDkRm  
cVfGov/BvpASujRBMUdXZQFvjPn0YczYKR6qC63H0eMIAJEBqtzNv4RnVP2dY+gg  
yhLwNf4kkaZwZ31TPCLeUtUKxHLZiKNWME108efoB+ZXw1I1NnCJQ/h4H8RESsJ  
foW9imPTsi6tgFIYX/KwLxtnVPXYw6e01iLxd8HsS5YruAuhFnMJZxHnAqZD1GL+  
/zym9IdMQmd9LJm+nc5H3duJoImUyPxo00dUMpdQwsATPZID0YBIZ1d7pprA9yj1  
3+VMMxz7JYa1kztadojC+HsZ15ZJy+5iSo4+Hd4MouS8VVUs7dsYuTMgpBYHxZE  
nyLYoXEaR3c8bAVvu2FsNRC5XCjHjpGUHMGZt1aRP2Q+A0Cus8Bn50MvFkuw0UiR



```
Iya5Ag0EUk+VvwEQAMG0+JaBKCu2WPH87fIcoXQYEm6VxVgAesnshg2KfbnBuMko
5X2SAHsPxDsyEf+vKSohsyBNPpMS05b1Q3CaKln664KJqJ3RitqMit+ajrGDpWnu
rbGZu9yJybhkgvn2dVb9BwSJ/CWaSVxVe8+BaLWhuYmig3WQo8WsFRFg5e89Mw0i
yoDy7SuLkrBFbECIT60WC0qk+5gp5XgqnTbt0uHLP7V2mZpkmZ3796p5IjzTTUmo
GazFcT+PX1Qnb5yeISg2zABIG8WVcBHgeBxYnI8ADVEEztaknPofG8JKzf0YGk9G
c952pPgyTjyqkVeVvlnpseDstqE9PEqjEW4Yqf73iC761fqyh/pW/bsadL2Fp0b1
qHTpRJWtDXgc38+kr/GSng/sSydutJiAeH+AVGAgYkg7vFCkw332G2vudCPUYFff
rpidlk3Kfzw0f+fTPXukdhd3XDNEHsvIVQ3H/qz2NrYV8pkv0SB3D2rg4746R6cd
cb4aBLtoL9MicAQ1F/8VVsB/5Xouw09HHdXQdaSoj8zb1kqD6es4anAerVLx0vQi
0c6LZm9jogJBTkXhFZHAey/mxvTwTdM1rUlybgki+VgG/rMkcwRsCw0/okDmNMLw
mfhLZegRvK7MkJAUcUxuY18wyLE5doZihhZL09sXHD3jcYW6tbRjBnw1VZpABEB
AAGJAR8EGAECaAkFALJP1b8CGwwACgkQ2C10w7vE19XBDgf9FsF98hfzAvLqSNGA
G44bfihuHC4F0qqNcnWYEMh6QH/HLbc6v7gg7uMCxINAg5q3k16IjUfVtFC4ZJ9S
xZGUComdrkh/8KkFCaxOD/Tq0tU1wpSM74waBLn5Et24N8LgCok0HA9V8ck4VBt6
muJ2x1byouHlLrm6cyaH+0A++S7nR5i53m0JrQjQGYJoBFgNLLZ9B1jdUDX6MtT
Rni7dJM2tFRY5upeinjHnVFC2Wc7YNCm7hZkCK3jfSR0Gya66l/Hr+e0qhLWp6D8
l+C18VSRRJk5+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  
yfB62KqbYuz16bmQKaA9zHN4FGfiTvR6t166LVHm1s/5HPiLv8sP14GsruLro9zN  
v72d07a9i68bMw+jarP0nu9dGiDFEI0dAC0kdCGEYKEUapQeNpmWRRQ46BeXyFwF  
JcNx76bJJUkwk6fWC0W63D762e61CEX6ndoaPjjLBnFvtx13heNGUc8RukBwe2mA  
U5pSGHj47J05bdWiRSwZaXa8PcW+20zTWaP755w7zWe4h60GANY70sT9nuQqsioJ  
QonxTrJuZweKRV8fNQ1EfDws3HZr7/7iXv03ABEBAAG0PEZyZWVU0QgRG9jZW5n  
IFRlYW0gU2VjcmV0YXJ5IDxkb2Nlbmctc2VjcmV0YXJ5QGZyZWVic2Qub3JnPokB  
VAQTAQoAPhYhBPJNezK4ZGJeVUGg50HANYCutF5YBQJduxRXAhsDBQkFo5qABQsJ  
CAcDBRUKCQgLBRYDAgEAAh4BAheAAAJEOHANYCutF5YB2IALw+EPYmOz9qlqIn  
oTFmk/5MrCdzC5iLEfxubbF6TopDWsWPi0h5mAuvfEmROS6f6ctvdYe9UtQV3VNY  
KeyskeFrIBOf02KG/dFqKPAWef6IfhbW3HWDWo5u0Bg01jHzQ/pB1n6SMKiXfsM  
idL9wN+UQKx3Y7S/bVrZTV0isRUo109+8kQeSYT/NMojVM0H2fWrTP/TaNEW4fY  
JBDA15hsktZd18sdbNqdC0GiX3xb4GvgVzGGQELagsxjfuXk6Pf0yn6Wx2d+yRcI  
FrKojmhihBp5VGFQkntBIXQkaW0xhW+WBGxwXdaAl0drQLZ3W+edgd01705x73kf  
Uw3Fh2a5AQ0EXbsUVWEIANEPAsltM4vFj2pi5xEuHEcZIrIX/ZJhoaBtZkqvKB+H  
4pu3/eQHK5hg0Dw12ugffPMz8mi57iGNI9TXd8ZYMJxAdvEZSDHCKZTX9G+FcxWa  
/AzKNiG25uSISzz7rMB/lV1gofCdGtpHFRFTiNxFcoacugTd1YDiscgJZMJSG/hC  
GXBdEKXR5WRAGAgandcL8l1CTo0t11ZE0kd5vJM861w6evgDhAZ2HGhRuG8/NDxG  
r4UtlNygUCFof/Q4oPNbDJzmZXF+80QyTncEpVD3leEOWG1Uv5XWS2XKVHcHZZ++  
ISo/B5Q60i3SJFCVV9f+g09YF+PgFP/mVMBgIf2fT20AEQEAAYkBPAYQAQoAJhYh  
BPJNezK4ZGJeVUGg50HANYCutF5YBQJduxRXAhsMBQkFo5qAAAJOHANYCutF5Y  
kecIAMTh2VHQqjXHTszQMsy3NjiTVVITI3z+pzY0u2EYmLyTXQ2pZMzLHMcklmub  
5po0X4EvL6bZiJcLMI2mSr0s0Gp8P3hyMI40IkqoLMp7VA2LF1PgIJ7K5W4oVwf8  
khY6lw7qg2L69APm/MM3xAyiL4p6MU8tpvWg5AncZ6lxyy27rxVfLzEtCrKQuG/a  
oVa01MjH3uxv0K6IIXlhwWD0nKs/e2h2HIAZ+ILE6ytS5ZEg2GXuigoQZdEnv71L  
xyvE9JANwGZLkDxnS5pgN2ikfkQYlFpJEkrNTQleCOHIIIP8vgJngEaP51x0IbQM  
CiG/y3cmKQ/ZfH7BBv1ZVtZKQsI=  
=MQKT

-----END PGP PUBLIC KEY BLOCK-----