FreeBSD 使用手册

摘要

欢迎使用 FreeBSD！ 本手册适用于安装 FreeBSD 11.2-RELEASE 和 FreeBSD 12.0-RELEASE 以及它们的日常使用。 这个手册目前由很多人 持续地 维护。 其中的内容需要不断地更新。 如果您有兴趣参加这个项目， 请发邮件到 FreeBSD 文档计划邮件列表。 此文档最新的英文原始版本可以从 FreeBSD 网站 上获得 (本手册的较早期版本可以在 http://docs.FreeBSD.org/doc/ 找到)。 由 FreeBSD 中文计划 维护的最新译本可以在 FreeBSD 中文计划 快照网站 获得， 这一译本会持续地向主站同步。 此外， 您也可以从 FreeBSD FTP 服务器 及其众多 镜像站点 取得这份文档的各种其它格式， 以及压缩形式的版本。 如果您希望得到一份印刷版本的手册， 可以从 FreeBSD Mall 购买。 除此之外， 您还可以 在手册中搜索内容。

目录

前言

预期的读者

作为 FreeBSD 的新用户，您将会在本手册第一部分找到 FreeBSD
的安装方法，同时逐渐引入概念和习俗来加强 UNIX® 基础。
阅读这部分只需要您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 中系统管理员感兴趣的所有主题的全面参考。
在阅读这些章节的内容时所需要的背景知识都注释在每一章节的大纲里面，
如果需要，可在阅读前进行预习。

要获得附加的信息来源列表，请查阅 参考文献。

相对于第三版的改动

目前的在线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
年出版的两卷第三版之后的一些重要变更：

- DTrace， DTrace， 增加了有关强大的 DTrace 性能分析工具有关的信息。
- 文件系统 Support， 文件系统支持， 增加 FreeBSD 上非原生文件系统有关的信息， 比如 Sun™ 的 ZFS。
- 安全事件审计， 安全事件审计， 增加了 FreeBSD 新的审计功能和使用方法。
- 虚拟化， 虚拟化， 增加了在虚拟化软件上安装 FreeBSD 有关的信息。

相对于第二版的改动 (2004)

您目前看到的这本手册的第三版是 FreeBSD 文档计划的成员历时两年完成的顶峰之作。
这一版的内容已经增长到需要分成两卷才能印刷出版。 第三版包含了如下的的主要变动：

- 设置和调整， 配置和优化， 进行了扩充并增加了关于 ACPI 电源和资源管理， cron 系统实用程序，以及更多的内核优化选项的相关内容。
- 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。
- 强制访问控制， 强制访问控制 (MAC) 是这一版新增的章节。 它解释了什么是 MAC，
以及这一机制如何使您的 FreeBSD 系统更安全。
- 存储， 存储， 在原有基础上增加了 USB 存储设备， 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关的内容。
- Vinum 卷管理程序，Vinum， 是这一版中的新章节。
描述了如何使用这种提供了设备无关的逻辑磁盘、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。
- 电子邮件， 电子邮件 一章中增加了关于如何使用其它的邮件传输代理、 SMTP 认证、 UUCP、 fetchmail、 procmail、 以及其它进阶内容。
- 网络服务器，网络服务， 是新版中全新的一章。 这一章包括了如何架设 Apache HTTP 服务器、 ftpd，
以及用于支持 Microsoft® Windows® 客户的 Samba。 一些段落来自原先的 高级网络， 进阶网络应用 一章。
- 高级网络，进阶网络应用 一章增加了关于在 FreeBSD 中使用 Bluetooth® 设备， 安装无线网络，
以及使用异步传输模式 (ATM) 网络的内容。
- 增加了一份词汇表， 用以说明整本书中出现的术语。
- 对于全书中图表进行了进一步的美化工作。

相对于第一版的改变 (2001)

本手册的第二版是 FreeBSD 文档计划的成员历时两年完成的顶峰之作。 第二版包含了如下的的主要变动：

- 添加了完整的索引。
- 用图形替换了以前所有用 ASCII 插图。
- 每个章节添加了标准大纲，列出了该章所包含的信息和读者所应该了解的知识。
- 内容逻辑地分成三个部分： "起步"，"系统管理"和"附录"。
- 安装 FreeBSD ("安装 FreeBSD") 新版本中使用了抓屏图片，使新用户更容易的领会正文。
- UNIX 基础 ("UNIX® 基础")扩充了进程、守护进程和信号的附加信息。
- 安装应用程序. Packages 和 Ports ("安装应用程序") 扩充了二进制包管理的附加信息。
- X Window 系统 ("X Window 系统") 新版本中着重介绍使用现代桌面技术例如 XFree86™ 4.x 上的 KDE 和 GNOME
- FreeBSD 引导过程 ("FreeBSD 启动过程") 对第一版内容进行扩充。
- 存储 ("存储") 由第一版中两个单独的章节"磁盘"和"备份"合并而成。
  我们认为这两部分作为一个整体比较容易理解。同时 RAID (包括硬件和软件 RAID) 部分也被添加进来。
- 串口通讯 ("串口通信") 对第一版进行完善，并为 FreeBSD 4.x/5.x 做了更新。
- PPP 和 SLIP ("PPP 和 SLIP") 全部更新。
- 许多新的内容被添加到 高级网络 ("高级网络").
- 电子邮件 ("电子邮件")增加了关于配置 sendmail 的信息。
- Linux® 二进制兼容模式("Linux® 兼容性") 增加了关于安装 Oracle® 和 SAP® R/3® 的信息。
- 第二版中也涵盖了下列主题：
  - 配置和调整(设置和调整)。
  - 多媒体(多媒体)

本手册的组织

这本手册分成了五个逻辑清晰的部分。第一部分 起步 涵盖了 FreeBSD 的安装和基本使用方法。
读者可根据自己的情况按顺序或者跳过一些熟悉的主题来阅读。第二部分 常用操作 涵盖了 FreeBSD
常用的功能， 这部分可以不按顺序阅读。每个部分由一个简明的大纲开始，
这个大纲描述本章节涵盖的内容和读者应该已经知道的知识。
这主要是让读者可以更好的选择感兴趣的章节阅读。第三部分 系统管理 涵盖了 FreeBSD
高级用户所感兴趣的广泛的话题。 第四部分 网络通讯 包括了网络和服务的话题，
而第五部分则是资源信息的附录。

介绍， 介绍
: 向新用户介绍 FreeBSD。它描述了 FreeBSD 计划的历史、目标和开发模式。

安装 FreeBSD， 安装
: 本章将会带领用户完成安装过程。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 操作系统基础命令和功能。 如果熟悉 Linux® 或者其他类 UNIX®
操作系统，则可以跳过这章。

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" 和标准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 系统并详细地介绍了如何在 FreeBSD 上使用它。
此外他也描述了常用的桌面环境， 例如 KDE 和 GNOME。

桌面应用， 桌面应用
列出了一些常用的桌面应用程序，比如 web 浏览器和办公套件， 描述了在 FreeBSD 上如何安装它们。

多媒体， 多媒体
展示了如何为您的系统设置声卡和视频回放支持。 也描述了一些简单的音频和视频应用程序。

配置FreeBSD的内核， 配置 FreeBSD 内核
解释了为什么需要配置一个新内核并提供了配置、 编译、安装自定义内核的详细说明。

打印， 打印
描绘了 FreeBSD 上打印机管理， 包括横幅页、打印统计，还有初始的设置。

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® 应用程序的详细的安装说明， 比如 Oracle® 和 Mathematica®。

设置和调整， 配置和调整
本章描述了管理员调整 FreeBSD 系统以优化性能时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 引导过程， 启动过程
本章描述 FreeBSD 的启动过程并且解释了如何用配置选项来控制这个过程。

用户和基本的帐户管理， 用户和基本帐号管理
本章描述了如何创建和操作用户帐号， 同样也论述了设置用户资源限制和其他账号管理任务的方法。

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

Jails， Jail
介绍了 jail 框架， 以及 jail 相对于 FreeBSD 中传统的 chroot 支持的改进。

强制访问控制， 强制访问控制
解释了何谓强制访问控制 (MAC) 以及如何利用这一机制来加强 FreeBSD 系统的安全。

安全事件审计， 安全事件审计
介绍了 FreeBSD 事件审计是什么， 以及如何安装、 配置它， 并检查或监视审计记帐信息。

存储， 存储
本章描述了怎样用 FreeBSD 来管理存储介质和文件系统，包括物理磁盘、RAID 阵列、光学和磁带媒体、后备存储磁盘以及网络文件系统。

GEOM. 模块化磁盘变换框架， 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级 RAID。

文件系统 Support, 文件系统支持
探讨了 FreeBSD 对非原生文件系统的支持，比如 Sun™ 的 Z 文件系统。

Vinum 卷管理程序, Vinum
本章描述了怎样使用逻辑卷管理器 Vinum。它提供了设备无关的逻辑磁盘和软件 RAID-0、RAID-1 以及 RAID-5。

虚拟化， 虚拟化
介绍了虚拟化系统提供的功能， 以及如何配合 FreeBSD 使用它们。

本地化－I18N/L10N使用和设置， 本地化
本章描述了如何在 FreeBSD 上使用非英语语言。它涵盖了系统和应用程序级的本地化。

更新与升级 FreeBSD， 更新与升级 FreeBSD
介绍了 FreeBSD-STABLE、 FreeBSD-CURRENT 以及 FreeBSD 发行版本之间的差异。
描述了一般用户如何紧跟开发过程并从中受益。
涵盖了如何更新用户的系统至发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.
动态跟踪可以通过实时的系统分析，帮助找出系统性能瓶颈。

串口通讯， 串行通信
本章解释了如何连接终端和调制解调器到 FreeBSD 系统，包括拨入和拨出连接。

PPP 和 SLIP， PPP 和 SLIP
本章描述了如何用 FreeBSD 通过使用 PPP，SLIP 或者基于以太网的 PPP (PPPoE) 来连接远程系统。

电子邮件， 电子邮件
本章解释了一个 email 服务器的不同组成部分并且简单讨论了关于最流行的 mail 服务器软件 sendmail 的配置。

网络服务， 网络服务
提供了详细的指引和示范配置文件以说明如何将一台 FreeBSD 机器作为网络文件系统服务器，
域名服务器， 网络信息服务器或时间同步服务器来使用的方法。

防火墙， 防火墙
解释了基于软件的防火墙的原理， 并提供了关于配置 FreeBSD 上的几种防火墙的详细说明。

高级网络应用， 高级网络应用
描述了许多关于网络的主题，包括如何在您的局域网中共享 Internet 连接， 高级路由话题， 无线网络，
Bluetooth®， ATM， IPv6以及许多高级话题。

获取 FreeBSD， 获取 FreeBSD
列出了获得 FreeBSD 安装 CDROM 或 DVDROM 的不同资源，也提供了允许您自由下载 FreeBSD 的不同 Internet 站点。

参考书目， 参考书目
由于本手册触及到了很多不同的主题，因而可能引发您想要获取更多详细的讲解。
参考书目列出了很多写作这本书时参考的好书。

Internet 上的资源， Internet 上的资源
讲述了很多对 FreeBSD 用户有用的能够提出问题并进行技术交流的关于 FreeBSD 的论坛。

PGP 公钥， PGP 公钥
列出了一些 FreeBSD 开发者的 PGP 签名公钥。

本书中使用的一些约定

为了使本书保持一致性和易读性特做了以下约定：

排版约定

斜体
斜体 字用来表示文件名、URLs、强调文字和术语的主流用法。

等宽
等宽 字体用来表示错误信息、 命令、 环境变量、 port 的名字、 主机名、 用户名、 组名、 设备名、 变量名， 以及代码片断。

粗体
粗体 字用来表示应用程序、命令和关键字。

用户输入

按键用粗体来突出于其他文本。 组合键意味着字用+连接时，同时的按下它们，例如：

Ctrl + Alt + Del

表示您应该同时按下 Ctrl， Alt 和 Del 键。

按顺序依次键入的关键字通常是用逗号隔开，例如：

Ctrl + X, Ctrl + S

这意味着用户应该同时按Ctrl 和 X，然后同时按 Ctrl 和 S。

示例

以 E:\> 开头的例子代表一个 MS-DOS® 命令。除非另有说明，这些命令都可以在一个现代的 Microsoft® Windows® "命令行"窗口环境被执行。

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

以 # 开头的例子代表必须以 FreeBSD 超级用户身份执行的命令。您可以用 root 身份登录来输入这些命令，或者以普通账号登录然后用 su(1) 来获得超级用户权限。

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

以 % 开头的例子代表命令应该被普通账号执行。 除非另有说明， 在设置环境变量和使用的其他 shell 命令均为 C-shell 语法。

```
% top
```

致谢

您所看到的这本书是全球几百人努力的结果。
无论他们只是纠正一些错误或提交完整的章节，所有的贡献都是非常有用的。

一些公司通过提供资金让作者专注于文档开发、 提供出版资金等等方式来支持文档开发。 其中， BSDi (后并入http://www.windriver.com[Wind River Systems]) 资助 FreeBSD 文档计划成员来专职改善这本书直到 2000 年三月第一个印刷版 (ISBN 1-57176-241-8) 的出版。 Wind River Systems 同时资助其他作者来对输出结构做很多改进和给文章添加一些附加章节。 这项工作结束于 2001 年 11 月印刷第二版 (ISBN 1-57176-303-1)。在 2003-2004 两年中，http://www.freebsdmall.com[FreeBSD Mall]，
向为改进这本手册以使其第三版印刷版本能够出版的志愿者支付了报酬。

Part I: 起步

手册的以下章节主要是针对刚开始使用 FreeBSD 的用户及管理员:

- FreeBSD 入门。
- 安装过程向导。
- 教您 UNIX® 基本知识和基本原理。
- 展示如何在 FreeBSD 上安装大量的第三方应用程序。
- 介绍使用 X，UNIX® 窗口系统， 以及为一些能够提高工作效率的桌面环境配置细节。

我们尝试用最少的页数来保持前言的索引，以至于可以用最少翻页次数将该手册从头至尾读过。

Chapter 1. 介绍

1.1. 概述

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

阅读完这章，您将了解：

- 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 开放源代码开发模式的基础。
- 当然还有："FreeBSD" 这个名称的由来。

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。 您也可以阅读 FreeBSD 的历史， 或者最新的发行版本。 如果您有意捐助(代码， 硬件，基金)，请看为 FreeBSD 提供帮助这篇文章。

1.2.1. FreeBSD 能做些什么?

FreeBSD 有许多非凡的特性。其中一些是：

- _抢占式多任务_与动态优先级调整确保在应用程序和用户之间平滑公正的分享计算机资源，
  即使工作在最大的负载之下。
- 多用户设备 使得许多用户能够同时使用同一 FreeBSD 系统做各种事情。 比如，
  像打印机和磁带驱动器这样的系统外设， 可以完全地在系统或者网络上的所有用户之间共享，
  可以对用户或者用户组进行个别的资源限制， 以保护临界系统资源不被滥用。
- 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。 这意味着您的 FreeBSD 主机可以很容易地和其他系统互联，
  也可以作为企业的服务器，提供重要的功能， 比如 NFS(远程文件访问)以及 email 服务，
  或将您的组织接入 Internet 并提供 WWW，FTP，路由和防火墙(安全)服务。
- _内存保护_确保应用程序(或者用户)不会相互干扰。 一个应用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)，
  并且从开始就是如此设计的。
- 业界标准的 X Window 系统 (X11R7)为便宜的常见 VGA
  显示卡和监视器提供了一个图形化的用户界面(GUI)， 并且完全开放代码。
- 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 应用程序可以从 FreeBSD ports 和 packages 套件中找到。
  您可以顺利地从这里找到， 何须搜索网络?
- 可以在 Internet 上找到成千上万其它 easy-to-port 的应用程序。 FreeBSD 和大多数流行的商业 UNIX® 代码级兼容， 因此大多数应用程序不需要或者只要很少的改动就可以编译。
- 页式请求_虚拟内存_和"集成的 VM/buffer
  缓存"设计有效地满足了应用程序巨大的内存需求并依然保持其他用户的交互式响应。
- SMP 提供对多处理器的支持。
- 内建了完整的 C、 C++、 Fortran 开发工具。 许多附加的用于高级研究和开发的程序语言，
  也可以在通过 ports 和 packages 套件获得。
- 完整的系统_源代码_意味着您对您环境的最大程度的控制。 当您拥有了一个真正的开放系统时，
  为什么还要受困于私有的解决方案， 任商业公司摆布呢?

- 丰富的_在线文档_。
- 不仅如此!

FreeBSD 基于加州大学伯克利分校计算机系统研究组 (CSRG) 发布的 4.4BSD-Lite， 继承了 BSD 系统开发的优良传统。 除了 CSRG 优秀的工作之外， FreeBSD 项目花费了非常多的时间来优化调整系统，使其在真实负载情况下拥有最好的性能和可靠性。 在现今， 许多商业巨人正为给 PC 操作系统增加新功能、提升和改善其可靠性， 以便在其上展开激烈竞争的同时， FreeBSD 现在 已经能够提供所有这一切了!

FreeBSD 可以提供的应用事实上仅局限于您的想象力。
从软件开发到工厂自动化，从存货控制到遥远的人造卫星天线方位控制， 如果商业的 UNIX® 产品可以做到， 那么就非常有可能您也可以用 FreeBSD 来做！ FreeBSD 也极大地受益于全世界的研究中心和大学开发的数以千计的高质量的应用程序，这些程序通常只需要很少的花费甚至免费。 可用的商业应用程序， 每天也都在大量地增加。

因为 FreeBSD 自身的源代码是完全公开的，
所以对于特定的应用程序或项目，可以对系统进行最大限度的定制。
这对于大多数主流的商业生产商的操作系统来说几乎是不可能的。 以下是当前人们应用 FreeBSD 的某些程序的例子：

- Internet 服务： FreeBSD 内建的强大的 TCP/IP 网络使它得以成为各种 Internet 服务的理想平台，比如：
  - FTP 服务器
  - World Wide Web 服务器(标准的或者安全的 [SSL])
  - IPv4 and IPv6 路由
  - 防火墙和 NAT("IP 伪装") 网关
  - 电子邮件服务器
  - USENET 新闻组和电子布告栏系统
  - 还有许多…

    使用 FreeBSD， 您可以容易地从便宜的 386 类 PC 起步，并随着您的企业成长，一路升级到带有 RAID 存储的四路 Xeon 服务器。
- 教育： 您是一名计算机科学或者相关工程领域的学生吗?
  学习操作系统，计算机体系结构和网络没有比在 FreeBSD 可提供的体验下动手实践更好的办法了。许多可自由使用的 CAD、数学和图形设计包也使它对于那些主要兴趣是在计算机上完成 _其他_工作的人非常有帮助。
- 研究： 有完整的系统源代码，FreeBSD
  对于操作系统研究以及其他计算机科学分支都是一个极好的平台。 FreeBSD 可自由获得的本性，同样可以使处在不同地方的开发团队在开放的论坛上讨论问题、 交流想法与合作开发成为可能，且不必担心特别的版权协定或者限制。
- _网络：_需要一个新的路由器? 一台域名服务器 (DNS)? 一个隔离您的内部网络的防火墙? FreeBSD 可以容易的把丢弃在角落不用的 386 或者 486 PC 变成一台完善的带包过滤能力的高级路由器。
- X Window 工作站： FreeBSD 是廉价 X 终端的一种绝佳解决方案， 您可以选择使用免费的 X11 服务器。 与 X 终端不同，如果需要的话 FreeBSD 能够在本地直接运行程序，因而减少了中央服务器的负担。 FreeBSD 甚至能够在 "无盘" 环境下启动，这使得终端更为便宜和易于管理。
- 软件开发： 基本的 FreeBSD 系统带有包括著名的 GNU C/C++ 编译器和调试工具在内的一整套开发工具。

FreeBSD 可以通过 CD-ROM、DVD， 以及匿名 FTP 以源代码和二进制方式获得。请查看获取 FreeBSDf 了解获取 FreeBSD 的更多细节。

1.2.2. 谁在使用 FreeBSD?

FreeBSD 被世界上最大的 IT 公司用作设备和产品的平台， 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

等等许多。

1.3. 关于 FreeBSD 项目

下面的章节提供了项目的一些背景信息， 包括简要的历史、项目目标、以及项目开发模式。

1.3.1. FreeBSD 的简要历史

FreeBSD 项目起源于 1993 年早期， 部分作为 "Unofficial 386BSD Patchkit" 的副产物，patchkit 的最后 3 个协调维护人是：Nate Williams，Rod Grimes 和我。

我们最初的目标是做出一份 386BSD 的测试版以修正一些 Patchkit 机制无法解决的错误(bug)。
很多人可能还记得早期的项目名称叫做 "386BSD 0.5" 或者 "386BSD Interim" 就是这个原因。

386BSD 是 Bill Jolitz 的操作系统， 到那时已被严重地忽视了一年之久。 由于 Patchkit
在过去的每一天里都在急剧膨胀， 使得对其进行消化吸收变得越来越困难，
因此我们一致同意应该做些事情并决定通过提供这个临时的 "cleanup" 版本来帮助 Bill。 然而，Bill
却在事先没有指出这个项目应该如何开展下去的情况下，
突然决定退出这个项目，最终这个计划只好被迫停止。

没过多久， 我们认为即便没有 Bill 的支持， 项目仍有保留的价值， 因此，我们采用了 David Greenman 的意见，给其命名为 "FreeBSD"。在和当时的几个用户商量后， 我们提出了最初的目标，
而这件事明朗化后， 这个项目就走上了正轨，甚至可能成为现实。 为了拓展 FreeBSD
的发行渠道，我抱着试试看的心态， 联系了光盘商 Walnut Creek CDROM，
以便那些上网不方便的用户得到 FreeBSD。 Walnut Creek CDROM 不仅支持发行 FreeBSD 光盘版的想法，
还为这个计划提供了所需的计算机和高速网络接入。 在那时， 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 (以及在整个互联网范围内发行的) 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")磁带， 也有许多组件是 386BSD 和自由软件基金会提供的。 对于第一次发行，这算是相当成功了。 在 1994 年 5 月，我们发布了更加成功的 FreeBSD 1.1 版。

在这段时间， 发生了一些意外的情况。 Novell 和 U.C. Berkeley 就 Berkeley Net/2
磁带知识产权的马拉松式的官司达成了和解。 和解中的一部分是 U.C. Berkeley 作出的让步， 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， 而后者在不久前刚刚从 AT&T 收购了这些产权； 作为回报， Berkeley 得到了来自 Novell 的 "许诺"， 在 4.4BSD-Lite 版本正式发布时， 可以声明为不受限的 (unencumbered)， 现有的 Net/2 用户则强烈建议转移到这个版本。 这包括了 FreeBSD， 而我们的项目则被允许在 1994 年 6 月底之前继续发行基于 Net/2 的产品。 根据和解协议， 在最后期限之前我们发布了一个最终版本， 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 重新编写自己的过程。 "Lite" 版本中， Berkeley 的 CSRG 删除了用于让系统能够引导的一大部分代码 (由于各种各样的法律需求)， 而当时 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， 我们的项目才完成了这项过渡， 并通过网络以及 CD-ROM (在 12 月底) 上发布了 FreeBSD 2.0。 尽管系统中还有很多比较粗糙的地方， 这个版本还是取得了巨大的成功， 并在 1995 年 6 月发布了更强大和易于安装的 FreeBSD 2.0.5 版本。

我们于 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 分支的另一个版本应运而生，它就是 FreeBSD 2.1.7.1，在 1997 年 2 月发布并停止了 2.1-STABLE 的主流开发。现在，它处于维护状态， 仅仅提供安全性的增强和其他严重的错误修补的维护(RELENG_2_1_0)。

FreeBSD 2.2 版作为 RELENG_2_2 分支，于 1996 年 11 月从开发主线 ("-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，顺着 2.2 分支的更进一步的版本在开发。 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 分支结束的开始。

1999 年 1 月 20 日又出现了新的分支，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补升级至 3.5.1，加入了对 Kerberos 安全性方面的修补。 这是 3.X 分支最后一个发行版本。

随后在 2000 年 3 月 13 日出现了一个新的分支， 也就是 4.X-STABLE。 这之后发布了许多的发行版本: 4.0-RELEASE 于 2000 年 3月发布， 而最后的 4.11-RELEASE 则是在 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。 这是将近三年的开发的巅峰之作， 同时也标志了 FreeBSD 在先进的多处理器和应用程序线程支持的巨大成就， 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 月正式创建， 紧随其后的是 5.3-RELEASE， 它是 5-STABLE 分支的标志性发行版。 这个分支的最后一个版本， 5.5-RELEASE 是在 2006 年 5 月发布的。 RELENG_5 分支不会有后续的发行版了。

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， 则是在 2005 年 11 月发布的。 这个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。 这是最后一个支持 Alpha 硬件架构的版本。

RELENG_7 分支于 2007 年 10 月创建。 第一个这个分支的发行版是 7.0-RELEASE， 这个版本是 2008 年 2 月发布的。 最新的 11.2-RELEASE 是在 June 28, 2018 发布的。 RELENG_7 将不会有其它后续的发布版本。

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 还将会有其它后续的发布版本。

目前， 中长期的开发项目继续在 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 项目目标

FreeBSD 项目的目标是无附加条件地提供能够用于任何目的的软件。 我们中的许多人对代码 (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资金上的补偿， 但我们并没打算坚决地要求得到这类资助。 我们认为我们的首要 "使命" 是为任何人提供代码， 不管他们打算用这些代码做什么， 因为这样代码将能够被更广泛地使用， 从而最大限度地发挥其价值。 我认为这是自由软件最基本的，

同时也是为我们所倡导的一个目标。

我们源代码树中， 以 GNU 公共许可证 (GPL) 或者 GNU 函数库公共许可证 (LGPL) 发布的那些代码带有少许的附加限制， 还好只是强制性的要求开放代码而不是别的。 由于使用 GPL 的软件在商业用途上会增加若干复杂性， 因此，如果可以选择的话， 我们更偏好使用限制相对更宽松的 BSD 版权来发布软件。

1.3.3. FreeBSD 开发模式

FreeBSD 的开发是一个非常开放且有有伸缩性的过程， 就像从我们的贡献者列表里看到的，它是完全由来自全世界的数以百计的贡献者发展起来的。 FreeBSD 的开发基础结构允许数以百计的开发者通过互联网协同工作。 我们也经常关注着那些对我们的计划感兴趣的新开发者和新的创意， 那些有兴趣更进一步参与项目的人只需要在 FreeBSD 技术讨论邮件列表 联系我们。 FreeBSD 公告邮件列表 对那些希望了解我们工作所涉及到哪些领域的人也是有用的。

无论是独立地工作或者封闭式的团队工作， 了解 FreeBSD 计划和它的开发过程都是有益的：

SVN 和 CVS 代码库

在过去的几年中 FreeBSD 的中央源代码树是由 CVS (并行版本控制系统)来维护的，CVS 是一个与 FreeBSD 捆绑的可自由获得的源代码控制工具。自 2008 年六月起， 这个项目开始转为使用SVN (Subversion)。 这次转换被认为是非常必要的，因为 CVS 的对于快速扩展源代码树和历史记录的限制越趋明显。现在主源码库使用 SVN，客户端的工具像 CVSup 和 csup 这些依赖于旧的 CVS 基础结构依然可以使用 - 因为对于 SVN 源码库的修改会被导回进 CVS。 目前只有中央原代码树是由 SVN 控制的。文档，万维网和 Ports 库还仍旧使用着 CVS。 The primary repository resides on a machine in Santa Clara CA, USA 主 CVS 代码库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 树也同样能非常容易的你的机器上。 请参阅 同步你的源码树 获得更多的相关信息。

committer 列表

committer 是那些对 CVS 树有_写_权限的人， 他们被授权修改 FreeBSD 的源代码 (术语 "committer" 来自于 cvs(1) 的 commit 命令，这个命令用来把新的修改提交给 CVS 代码库)。提交修正的最好方法是使用 send-pr(1) 命令。如果您发现在系统中出现了一些问题的话， 您也可以通过邮件将它们发送至 FreeBSD committer 的邮件列表。

FreeBSD 核心团队

如果把 FreeBSD 项目看作一家公司，那么 _FreeBSD 核心团队_就相当于董事会。 核心团队的主要任务是提出总体上的发展计划，然后确定一个正确的方向。 邀请那些富有献身精神和可靠的开发者加入到 committer 队伍中来也是核心团队的工作之一， 这些新的成员将作为新核心团队成员和其他人一起继续前进。 当前的核心团队是 2010 年 7 月从 committer 中选举产生的。选举每两年一次。

一些核心团队的成员还负责特定的责任范围， 也就是说他们必须尽力确保某个子系统能工作正常。 FreeBSD 开发者的完整列表和他们的责任范围，请参见贡献者列表

核心团队的大部分成员加入 FreeBSD 开发的时候都是志愿的， 并没有从项目中获得任何财政上的资助， 所以"承诺"不应该被理解为"支持保证"。 前面所述"董事会"的类推并不十分准确， 或许更好的说法是，他们是一群愿意放弃他们的生活， 投身于 FreeBSD 项目而非选择其个人更好的生活的人！

外围贡献者

事实上，最大的开发团队正是为我们提供反馈和错误修补的用户自己。 FreeBSD 的非集中式的开发者保持联系的主要方式就是预订 FreeBSD 技术讨论邮件列表，很多事情在那里讨论。查看Internet上的资源了解众多 FreeBSD 邮件列表的更多信息。

FreeBSD 贡献者列表 很长并在不断增长， 为什么不加入它来为 FreeBSD 做贡献呢?

提供代码不是为这个计划做贡献的唯一方式； 有一个更完整的需要做的事情的列表，可以参见 FreeBSD 项目网站。

总的来说，我们的开发模式好像是一组没有拘束的同心圆。 这种集中式的开发模式，主要是考虑到 FreeBSD _用户_的方便， 同时让他们能很容易地维护同一份软件， 而不会把潜在的贡献者排除在外! 我们的目标是提供一个包含有大量具有一致性 应用程序的稳定的操作系统， 以利于用户的安装和使用，- 这种模式在完成目标的过程中工作得非常有效。

我们对于那些想要加入，成为 FreeBSD 开发者的期待是:
具有如同当前其他人一样的投入，来确保持续的成功!

1.3.4. 最新的 FreeBSD 发行版本

FreeBSD 是一个免费使用且带有完整源代码的基于 4.4BSD-Lite 的系统， 它广泛运行于 Intel i386™、i486™、Pentium®、 Pentium® Pro、 Celeron®、 Pentium® II、 Pentium® III、 Pentium® 4(或者兼容系统)、 Xeon™、 和 Sun UltraSPARC® 的计算机系统上。 它主要以 加州大学伯克利分校 的 CSRG 研究小组的软件为基础，并加入了 NetBSD、OpenBSD、386BSD 以及来自 自由软件基金会 的一些东西。

自从 1994 年末我们的 FreeBSD 2.0 发行以来， FreeBSD
的性能，可定制性，稳定性都有了令人注目的提高。
最大的变化是通过整合虚拟内存/文件系统中的高速缓存改进的虚拟内存系统，
它不仅提升了性能，而且减少了 FreeBSD 对内存的需要， 使得 5 MB 内存成为可接受的最小配置。
其他的改进包括完整的 NIS 客户端和服务器端的支持， 事务式 TCP 协议支持，按需拨号的 PPP，集成的 DHCP 支持，改进的 SCSI 子系统， ISDN 的支持，ATM，FDDI，快速 Gigabit 以太网(1000 Mbit)支持，提升了最新的 Adaptec 控制器的支持和修补了很多的错误。

除了最基本的系统软件，FreeBSD 还提供了一个拥有成千上万广受欢迎的程序组成的软件的 Ports Collection。 到本书付印时，已有超过 36000 个 ports (ports 包括从 http(WWW)
服务器到游戏、程序设计语言、编辑器以及您能想到的几乎所有的东西)。 完整的 Ports Collection 大约需要 3 GB 的存储空间。所有的只提供对原始代码的 "修正"。这使得我们能够容易地更新软件，而且减少了老旧的 1.0 Ports Collection 对硬盘空间的浪费。 要编译一个
port，您只要切换到您想要安装的程序的目录， 输入 make install，然后让系统去做剩下的事情。
您要编译的每一个程序完整的原始代码可以从 CD-ROM 或本地 FTP
获得，所以您只需要编译您想要软件的足够的磁盘空间。 几乎大多数的软件都提供了事先编译好的 "package" 以方便安装，对于那些不希望从源代码编译他们自己的 ports 的人只要使用一个简单的命令 (pkg_add)就可以安装。 有关 package 和 ports 的更多信息可以在安装应用程序. Packages 和 Ports中找到。

您可以在最近的 FreeBSD 主机的 /usr/shared/doc 目录下找到许多有用的文件来帮助您安装及使用 FreeBSD。 您也可以用一个 HTML 浏览器来查阅本地安装的手册， 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

您也可以查看在 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，简单好用的安装程序，叫做 sysinstall 。这是 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。

学习完本章之后，您将会知道：

- 如何制作 FreeBSD 安装磁盘
- FreeBSD如何参照及分割您的硬盘
- 如何启动 sysinstall.
- 在执行 sysinstall 时您将要回答的问题、 问题代表什么意义，以及该如何回答它们。

在阅读本章之前，您应该：

- 阅读您要安装的 FreeBSD 版本所附的硬件支持列表以确定您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") 体系结构的电脑。如果有其它体系结构的安装说明，我们将一并列出。虽然本文档经常保持更新，但有可能与您安装版本上所带的说明文档有些许出入。在这里建议您使用本说明文章作为一般性的安装指导参考手册。

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， 给出了这些信息的一些总结。随您安装 FreeBSD 的方式不同， 可能需要使用软驱或为 FreeBSD 支持的 CDROM 驱动器， 有时候也可能需要的是一块网卡。 这将在 准备引导介质 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。您需要至少 150 MB 的空闲硬盘空间， 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 和腾出更多的硬盘空间，要比使用更快的处理器更有用。

2.2.1.2. FreeBSD/amd64

有两类处理器同时能够支持运行 FreeBSD/amd64。 第一种是 AMD64 处理器， 包括 AMD Athlon™64、AMD Athlon™64-FX、 AMD Opteron™ 以及更高级别的处理器。

能够使用 FreeBSD/amd64 的另一种处理器是包含了采用 Intel® EM64T 架构支持的处理器。这类处理器包括 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， 则 必须 使用 BIOS 配置， 禁用 IO APIC。如果您没有找到这样的选项， 可能就只能转而禁用 ACPI 了。 Pro-150 芯片组存在一个 bug，目前我们还没有找到绕过这一问题的方法。

2.2.1.3. FreeBSD/sparc64

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

FreeBSD/sparc64 需要独占一块磁盘。 目前还没有办法与其它操作系统共享一块磁盘。

2.2.2. 支持的硬件

支持的硬件列表， 会作为 FreeBSD 发行版本的 FreeBSD 兼容硬件说明提供。 这个文档通常可以在 CDROM 或 FTP 安装文件的顶级目录找到， 它的名字是 HARDWARE.TXT， 此外， 在 sysinstall 的 documentation 菜单也可以找到。它针对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. 列出您电脑的硬件清单

在安装 FreeBSD 之前，您应该试着将您电脑中的硬件清单列出来。 FreeBSD 安装程序会将这些硬件(磁盘、网卡、光驱等等) 以及型号及制造厂商列出来。FreeBSD 也会尝试为这些设备找出最适当的 IRQ 及 IO 端口的设定。但是因为 PC 的硬件种类实在太过复杂，这个步骤不一定总是能成功。这时， 您就可能需要手动更改有问题的设备的设定值。

如果您已经安装了其它的操作系统，如 Windows® 或 Linux，那么您可以先由这些系统所提供的工具来查看您的设备设定值是怎么分配的。如果您真的没办法确定某些接口卡用什么设定值，那么您可以检查看看，说不定它的设定已经标示在卡上。常用的 IRQ 号码为 3、5 以及 7； IO 端口的值通常以 16 进制位表示，例如 Ox330。

我们建议您在安装 FreeBSD 之前把这些信息打印或记录下来，做成表格 的样子也许会比较有帮助，例如：

表 1. 硬件设备清单

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 接口主设备
CDROM	N/A	N/A	第一个 IDE 接口从设备
第二块硬盘	N/A	N/A	20 GB，IBM 制造, 第二个 IDE 接口主设备
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

在清楚地了解了您计算机的配置之后， 需要检查它是否符合您希望安装的 FreeBSD 版本的硬件需求。

2.3.2. 备份您的数据

如果您的电脑上面存有重要的数据资料， 那么在安装 FreeBSD 前请确定您已经将这些资料备份了，并且先测试这些备份文档是否有问题。FreeBSD 安装程序在要写入任何资料到您的硬盘前都会先提醒您确认，一旦您确定要写入，那么以后就没有反悔的机会。

2.3.3. 决定要将 FreeBSD 安装到哪里

如果您想让 FreeBSD 使用整个硬盘，那么请直接跳到下一节。

但是，如果您想让 FreeBSD 跟您已有的系统并存，
那么您必须对您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 硬盘可以被细分为许多块。 这些块被称为 partitions (分区)。 由于 FreeBSD
内部也有分区的概念，如此命名很容易导致混淆， 因此我们在 FreeBSD 中，将其称为磁盘 slice，或简称为 slices。 例如， FreeBSD 提供的用于操作 PC 磁盘分区的工具 fdisk 就将其称为 slice 而不是 partition。
由于设计的原因， 每个硬盘仅支持四个分区； 这些分区叫做 主分区(Primary partion)。
为了突破这个限制以便能使用更多的分区，就有了新的分区类型，叫做 扩展分区(Extended partition)。
一个硬盘可以拥有一个扩展分区。在扩展分区里可以建立许多个所谓的 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 分区的分区号为 165。

一般而言，每种操作系统都会有自己独特的方式来区别分区。 例如 DOS 及其之后的 Windows®，
会分配给每个主分区及逻辑分区一个 驱动器字符， 从 C: 开始。

FreeBSD 必须安装在主分区。FreeBSD 可以在这个分区上面存放系统数据或是您建立的任何文件。
然而，如果您有多个硬盘，您也可以在这些硬盘上(全部或部分)建立 FreeBSD 分区。在您安装 FreeBSD 的时候，必须要有一个分区可以给 FreeBSD 使用。 这个分区可以是尚未规划的分区，
或是已经存在且存有数据但您不再需要的分区。

如果您已经用完了您硬盘上的所有分区， 那么您必须使用其它操作系统所提供的工具 (如 DOS 或 Windows® 下的 fdisk) 来腾出一个分区给 FreeBSD 使用。

如果您的某个分区有多余的空间，您可以使用它。 但是使用前您需要先整理一下这些分区。

FreeBSD 最小安装需要约 100 MB 的空间，但是这仅是 非常 基本的安装，
几乎没有剩下多少空间可以建立您自己的文件。一个较理想的最小安装是 250 MB，不含图形界面；或是 350 MB 以上，包含图形界面。 如果您还需要安装其它的第三方厂商的套件， 那么将需要更多的硬盘空间。

您可以使用类似 PartitionMagic® 这样的商业版本工具， 或类似 GParted
这样的自由软件工具来调整分区尺寸， 从而为 FreeBSD 腾出空间。 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。

目前已经有报告显示改变 Microsoft® Vista 分区尺寸时会出现问题。 在进行此类操作时， 建议您准备一张 Vista 安装 CDROM。如同其他的磁盘维护操作一样， 强烈建议您事先进行备份。

不当的使用这些工具可能会删掉您硬盘上的数据资料！
在使用这些工具前确定您有最近的、没问题的备份数据。

例 1. 使用已存在的分区

假设您只有一个 4GB 的硬盘，而且已经装了 Windows® 然后您将这个硬盘分成两个分区 C: 跟 D:，每个分区大小为 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。

这意味着您的盘上有两个分区，一个驱动器符号是一个分区 (如 c:、d:)。 您可以把所有存放在 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. 缩减已现在的分区

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

为了安装 FreeBSD，您必须从下面两种方式中选择一种：

1. 备份 Windows® 的数据资料，然后重新安装 Windows®， 并给 Windows® 分配 2 GB 的空间。
2. 使用上面提及的 PartitionMagic® 来整理或切割您的分区。

2.3.4. 收集您的网络配置相关资料

如果您想通过网络(FTP 或是 NFS)安装 FreeBSD， 那么您就必须知道您的网络配置信息。在安装 FreeBSD 的过程中将会提示您输入这些资料，以顺利完成安装过程。

2.3.4.1. 使用以太网或电缆/DSL Modem

如果您通过局域网或是要通过网卡使用电缆/DSL 上网， 那么您必须准备下面的信息：

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

如果您不知道这些信息， 您可以询问系统管理员或是您的网络服务提供者。 他们可能会说这些信息会由 DHCP 自动分配；如果这样的话，请记住这一点就可以了。

2.3.4.2. 使用 Modem 连接

如果您由 ISP 提供的拨号服务上网，您仍然可以通过它安装 FreeBSD，只是会需要很长的时间。

您必须知道：

1. 拨号到 ISP 的电话号码。
2. 您的 modem 是连接到哪个 COM 端口。
3. 您拨号到 ISP 所用的账号和密码。

2.3.5. 检查 FreeBSD 发行勘误

虽然我们尽力确保每个 FreeBSD 发行版本的稳定性， 但偶尔也会有一些错误进入发行版。极少数情况下，这些问题甚至可能会影响安装。 当发现和修正问题之后，它们会列在 FreeBSD 网站中的 FreeBSD 发行勘误中。 在您安装之前，应该首先看一看这份勘误表，以了解可能存在的问题。

关于所有释出版本的信息，包括勘误表，可以在 FreeBSD 网站 的 发行版信息 一节中找到。

2.3.6. 准备安装介质

FreeBSD 可以通过下面任何一种安装介质进行安装：

安装介质

- CDROM 或 DVD
- USB 记忆棒
- 在同一计算机上的 DOS 分区
- SCSI 或 QIC 磁带
- 软盘

网络

- 通过防火墙的一个 FTP 站点，或使用 HTTP 代理。
- NFS 服务器
- 一个指定的并行或串行接口

如果您购买了 FreeBSD 的 CD 或 DVD，那么您可以直接进入下一节 (准备引导介质)。

如果您还没有 FreeBSD 的安装文件， 则应按照 准备您自己的安装介质 来准备。 读完那节之后，

您就可以回到这节并从 准备引导介质 继续了。

2.3.7. 准备引导介质

FreeBSD 的安装过程开始于将您的电脑开机进入 FreeBSD 安装环境 -
－并非在其它的操作系统上运行一个程序。 计算机通常使用安装在硬盘上的操作系统进行引导，
也可以配置成使用一张"bootable(可引导)"的软盘进行启动。 大多数现代计算机也都可以从光驱或 USB 盘来引导系统。

如果您有 FreeBSD 的安装光盘或 DVD(或者是您购买的，
或者是您自己准备的。)并且您的计算机可以从光驱进行启动 (通常在 BIOS 中会有 "Boot Order" 或类似的选项可以设置)，那么您就可以跳过此小节。 因为 FreeBSD 光盘及 DVD 光盘都是可以引导的， 用它们开机您不用做什么特别的准备。

要创建引导系统所需的记忆棒， 需按下面的操作进行：

1. 获取记忆棒映像文件

   记忆棒映像文件可以从 arch 对应的 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 需要替换为您使用的平台和版本。 例如， FreeBSD/i386 12.0-RELEASE 的记忆棒映像位于 ftp://ftp.FreeBSD.org/pub/FreeBSD/releases/i386/ISO-IMAGES/12.0/FreeBSD-12.0-RELEASE-i386-memstick.img。

   记忆棒的映像文件扩展名是 .img。 在 ISO-IMAGES/ 目录中提供了多种不同的映像，
   您需要根据使用的 FreeBSD 版本， 有时也包括硬件来选择合适的映像。

   在继续安装之前， 务必 备份 您目前保存在 USB 记忆棒上的数据，
   接下来的操作将会 擦除 这些数据。

2. 准备记忆棒

   下面的例子中， 目标记忆棒对应的设备名是 /dev/da0。
   请小心地确认这是希望覆盖的设备， 否则可能会损坏您的现有数据。

   设置 kern.geom.debugflags sysctl 为允许写入目标设备的主引导记录。

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

3. 将映像文件写入记忆棒

   .img 文件 不是 直接复制到记忆棒中的那种普通文件。
   这个映像是一份包含启动盘全部内容的映像。 这意味着简单地从一个地方复制到另一个地方是 不能赋予其引导系统的能力的。 您必须使用 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(软盘)请依照下列步骤：

1. 获取开机软盘映像文件

   请注意， 从 FreeBSD 8.0 开始， 我们不再提供软盘映像了。
   请参阅前面关于如何使用 USB 记忆棒， 或 CDROM 和 DVD 来安装 FreeBSD

的介绍。

开机软盘映像文件可以在您的安装介质的 floppies/ 目录下找到， 另外您也可以从下述网站的 floppies 目录下载： ftp://ftp.FreeBSD.org/pub/FreeBSD/releases/架构名/版本-RELEASE/floppies/。 将 <架构名> 和 <版本> 替换为您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， 应该是 ftp://ftp.FreeBSD.org/pub/FreeBSD/releases/i386/11.2-RELEASE/floppies/。

软盘映像文件的扩展名是 .flp。 在 floppies/ 目录中包括了许多不同的映像文件， 随您安装的 FreeBSD 版本， 某些时候也随硬件的不同， 您需要使用的映像文件可能会有所不同。 您通常会需要四张软盘， 即 boot.flp、 kern1.flp、 kern2.flp， 以及 kern3.flp。 请查阅同一目录下的 README.TXT 文件以了解关于这些映像文件的最新信息。

您的 FTP 程序必须使用 二进制模式 来下载这些映 像文件。有些浏览器只会用 text (或_ASCII _) 模式来传输数据， 用这些浏览器下载的映像文件做成的软盘将无法正常开机。

2. 准备软盘

   您必须为您下载的每一个映像文件准备一张软盘。 并且请避免使用到坏掉的软盘。 最简单的方式就是您先将这些软盘格式化， 不要相信所谓的已格式化的软盘。在 Windows® 下的格式化程序不会告诉您出现多少坏块， 它只是简单的标记它们为 "bad" 并且忽略它们。 根据建议您应该使用全新的软盘来存放安装程序。

   如果您在安装 FreeBSD 的过程中造成当机、 冻结或是其它怪异现象，第一个要怀疑的就是引导软盘。 请用其它的软盘制作映像文件再试试看。

3. 将映像文件写入软盘中

   .flp 文件 并非 一般的文件，您不能直接将它们复制到软盘上。 事实上它是一张包含完整磁盘内容的映像文件。这表示您 不能 简单的使用 DOS 的 copy 命令将文件写到软盘上， 而必须使用特别的工具程序将映像文件直接写到软盘中。

   如果您使用 MS-DOS® 或 Windows® 操作系统来制作引导盘， 那么您可以使用我们提供的 fdimage 程序来将映像文件写到软盘中。

   如果您使用的是光盘，假设光盘的驱动器符号为 E:，那么请执行下面的命令：

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

   重复上述命令以完成每个 .flp 文件的写入， 每换一个映像文件都必须更换软盘； 制作好的软盘请注明是使用哪个映像文件做的。 如果您的映像文件存放在不同的地方，请自行修改上面的指令指向您存放 .flp 文件的地方。要是您没有 FreeBSD 光盘， 您可以到 FreeBSD 的 FTP 站点 tools目录 中下载。

   如果您在 UNIX® 系统上制作软盘(例如其它 FreeBSD 机器)， 您可以使用 dd(1) 命令来将映像文件写到软盘中。 如果您用 FreeBSD,可以执行下面的命令：

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

   在 FreeBSD 中，/dev/fd0 指的是第一个软驱(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的名称， 这时您就要查阅该系统的说明文件。

您现在可以安装 FreeBSD 了

2.4. 开始安装

默认情况下, 安装过程并不会改变任何您硬盘中的数据， 除非您看到下面的讯息：

> 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!

在看到这最后的警告讯息前您都可以随时离开，
安装程序界面不会变更您的硬盘。如果您发现有任何设定错误，
这时您可以直接将电源关掉而不会造成任何伤害。

2.4.1. 开机启动

2.4.1.1. 引导 i386™ 系统

1. 从电脑尚未开机开始说起
2. 将电脑电源打开。刚开始的时候它应该会显示进入系统设置菜单或 BIOS 要按哪个键，常见的是 `F2`、 `F10`、 `Del` 或 `Alt` + `S`。不论是要按哪个键，请按它进入 BIOS 设置画面。
   有时您的计算机可能会显示一个图形画面，典型的做法是按 `Esc` 将关掉这个图形画面，
   以使您能够看到必要的设置信息。
3. 找到设置开机顺序的选项，它的标记为 "Boot Order"
   通常会列出一些设备让您选择，例如：Floppy、 CDROM、First Hard Disk 等等。

   如果您要用光盘安装，请选择 CDROM。 如果使用 USB 盘， 或者软盘来引导系统，
   也应类似地确认选择了正确的引导设备。 如有疑问， 请参考您的主板说明手册。

   储存设定并离开，系统应该会重新启动。
4. 如果您根据 准备引导介质 制作了 "可引导" 的 USB 记忆棒， 在开机前将其插到计算机上。

   如果您是从光盘安装， 那么开机后请立即将 FreeBSD 光盘放入光驱中。

   > 对于 FreeBSD 7.3 和更早的版本， 可以使用软盘引导， 这些软盘可以根据
   > 准备引导介质 来制作。 其中， boot.flp 是启动盘。
   > 引导系统时应使用这张软盘。

   如果您开机后发现计算机引导了先前已经装好的其他操作系统， 请检查： ..
   是不是软盘或光盘太晚放入而错失开机引导时间。 如果是， 请将它们放入后重新开机。 .. BIOS
   设定不对，请重新检查 BIOS 的设定。 .. 您的 BIOS 不支持从这些安装介质引导。
5. FreeBSD 即将启动。如果您是从光盘引导， 您会见到类似下面的画面：

   ```
   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]
\
```

如果您从软盘启动， 则应看到类似下面的画面：

```
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...
```

请根据提示将 boot.flp 软盘取出， 插入 kern1.flp 这张盘， 然后按 `Enter`。
您只需从第一张软盘启动， 然后再需要时根据提示插入其他软盘就可以了。

6. 不论是从光盘、 USB 记忆棒或软盘引导， 接下来都会进入 FreeBSD 引导加载器菜单：

图 1. FreeBSD Boot Loader Menu

您可以等待十秒， 或按 Enter。

2.4.1.2. 引导 sparc64

多数 sparc64 系统均配置为从硬盘自动引导。 如果希望安装 FreeBSD，就需要从网络或 CDROM 启动了，这需要首先进入 PROM (OpenFirmware)。

要完成这项工作，首先需要重启系统，并等待出现引导消息。
具体的信息取决于您使用的型号，不过它应该会是类似下面这样：

```
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.
```

如果您的系统此时开始了从硬盘引导的过程，则需要按下 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 ~#) 以便进入 PROM 提示符。 它应该是类似下面这样：

```
ok     ①
ok {0} ②
```

① 这是在只有一颗 CPU 的系统上的提示。

② 这是用于 SMP 系统的选项， 这里的数字， 是系统中可用的 CPU 数量。

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

2.4.2. 查看设备探测的结果

前面屏幕显示的最后几百行字会存在缓冲区中以便您查阅。

要浏览缓冲区，您可以按下 Scroll Lock 键， 这会开启画面的卷动功能。然后您就可以使用方向键或 PageUp 、 PageDown 键来上下翻阅。 再按一次 Scroll Lock 键将停止画面卷动。

在您浏览的时候会看到类似 典型的设备探测结果的画面。 真正的结果依照您的电脑装置而有所不同。

典型的设备探测结果

```
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 pci
0
usb0: <VIA 83572 USB controller> on uhci0
usb0: USB revision 1.0
uhub0: VIA UHCI root hub, class 9/0, rev 1.00/1.00, addr1
uhub0: 2 ports with 2 removable, self powered
pci0: <unknown card> (vendor=0x1106, dev=0x3040) at 7.3
dc0: <ADMtek AN985 10/100BaseTX> port 0xe800-0xe8ff mem 0xdb000000-0xeb0003ff ir
q 11 at device 8.0 on pci0
dc0: Ethernet address: 00:04:5a:74:6b:b5
miibus0: <MII bus> on dc0
ukphy0: <Generic IEEE 802.3u media interface> on miibus0
ukphy0: 10baseT, 10baseT-FDX, 100baseTX, 100baseTX-FDX, auto
ed0: <NE2000 PCI Ethernet (RealTek 8029)> port 0xec00-0xec1f irq 9 at device 10.
0 on pci0
ed0 address 52:54:05:de:73:1b, type NE2000 (16 bit)
isa0: too many dependant configs (8)
isa0: unexpected small tag 14
```

```
orm0: <Option ROM> at iomem 0xc0000-0xc7fff on isa0
fdc0: <NEC 72065B or clone> at port 0x3f0-0x3f5,0x3f7 irq 6 drq2 on isa0
fdc0: FIFO enabled, 8 bytes threshold
fd0: <1440-KB 3.5” drive> on fdc0 drive 0
atkbdc0: <Keyboard controller (i8042)> at port 0x60,0x64 on isa0
atkbd0: <AT Keyboard> flags 0x1 irq1 on atkbdc0
kbd0 at atkbd0
psm0: <PS/2 Mouse> irq 12 on atkbdc0
psm0: model Generic PS/@ mouse, device ID 0
vga0: <Generic ISA VGA> at port 0x3c0-0x3df iomem 0xa0000-0xbffff on isa0
sc0: <System console> at flags 0x100 on isa0
sc0: VGA <16 virtual consoles, flags=0x300>
sio0 at port 0x3f8-0x3ff irq 4 flags 0x10 on isa0
sio0: type 16550A
sio1 at port 0x2f8-0x2ff irq 3 on isa0
sio1: type 16550A
ppc0: <Parallel port> at port 0x378-0x37f irq 7 on isa0
pppc0: SMC-like chipset (ECP/EPP/PS2/NIBBLE) in COMPATIBLE mode
ppc0: FIFO with 16/16/15 bytes threshold
plip0: <PLIP network interface> on ppbus0
ad0: 8063MB <IBM-DHEA-38451> [16383/16/63] at ata0-master UDMA33
acd0: CD-RW <LITE-ON LTR-1210B> at ata1-slave PIO4
Mounting root from ufs:/dev/md0c
/stand/sysinstall running as init on vty0
```

请仔细检查探测结果以确定 FreeBSD 找到所有您期望出现的设备。 如果系统没有找到设备，
则不会将其列出。 定制内核 能够让您为系统添加默认的 GENERIC 内核所不支持的设备， 如声卡等。

在 FreeBSD 6.2 和更高版本中， 在探测完系统设备之后， 将显示 选择国家及地区菜单。
请使用光标键来选择国家或地区。 接着按 Enter， 系统将自动设置地区。 您也可以很容易地退出 sysinstall 程序并从头来过。

图 2. 选择国家及地区菜单

如果您在国家及地区菜单中选择了 United States (美国)，则系统会使用标准的美国键盘映射；
如果选择了不同的国家，则会显示下面的菜单。使用光标键选择正确的键盘映射，然后按 Enter 来确认。

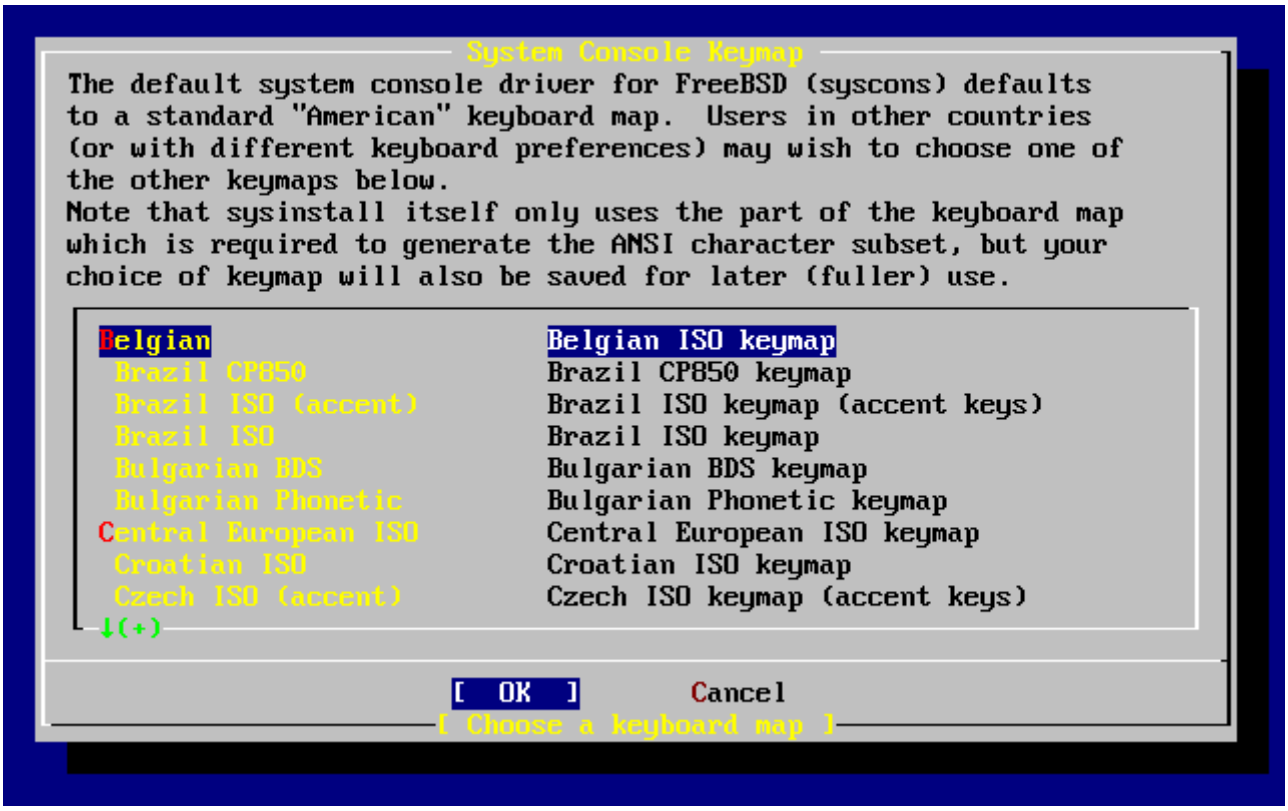


图 3. 选择键盘菜单

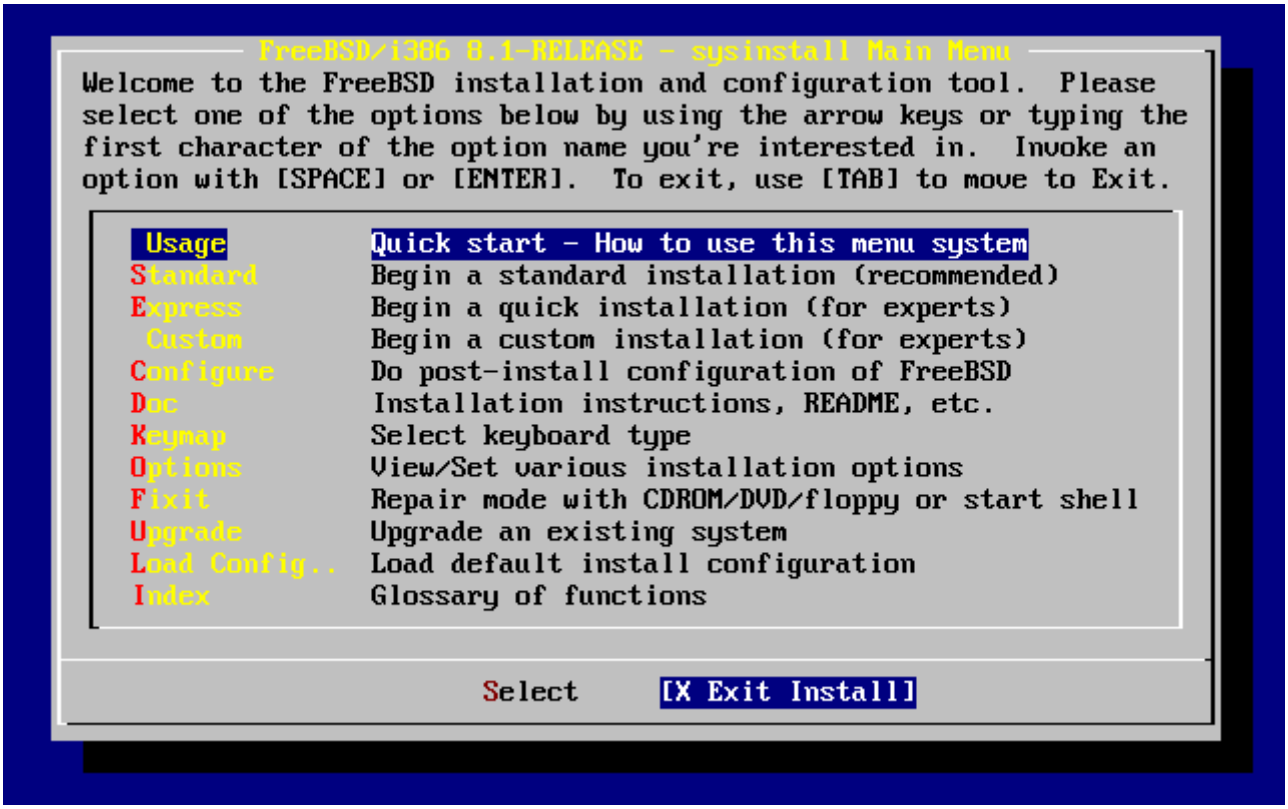


图 4. 选择离开 Sysinstall

在主界面使用方向键选择 Exit Install 您会看到 如下的信息：

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

          [ Yes ]  No
```

如果此处选择了 [yes] 但 CDROM 还留在光驱里， 则会再次进入安装程序。

如果您是从软盘启动， 则在重启系统之前， 需要将 boot.flp 软盘取出。

2.5. 介绍 Sysinstall

sysinstall 是 FreeBSD 项目所提供的安装程序。它以 console(控制台)为主，
分为多个菜单及画面让您配置及控制安装过程。

sysinstall 菜单画面由方向键、 Enter、Tab、Space， 以及其它按键所控制。在主画面的 Usage
菜单有这些按键的说明。

要查看这些说明，请将光标移到 Usage 项目，然后 [Select] 按键被选择， 选取 Sysinstall 主菜单的 Usage 项目，然后按下 Enter 键。

安装画面的使用说明会显示出来，阅读完毕请按 Enter 键回到主画面。

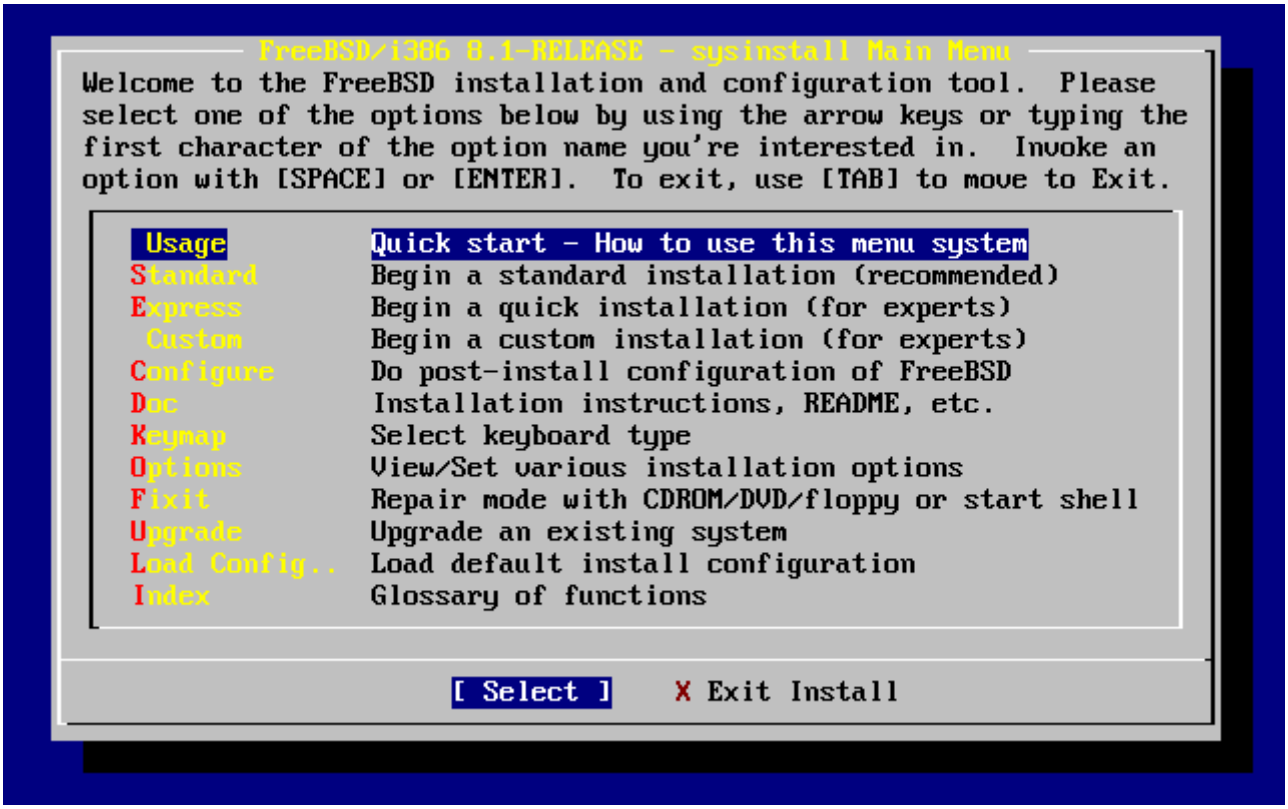


图 5. 选取 Sysinstall 主菜单的 Usage 项目

2.5.1. 选择 Documentation(说明文件) 菜单

用方向键从主菜单选择 Doc 条目然后按 Enter 键。

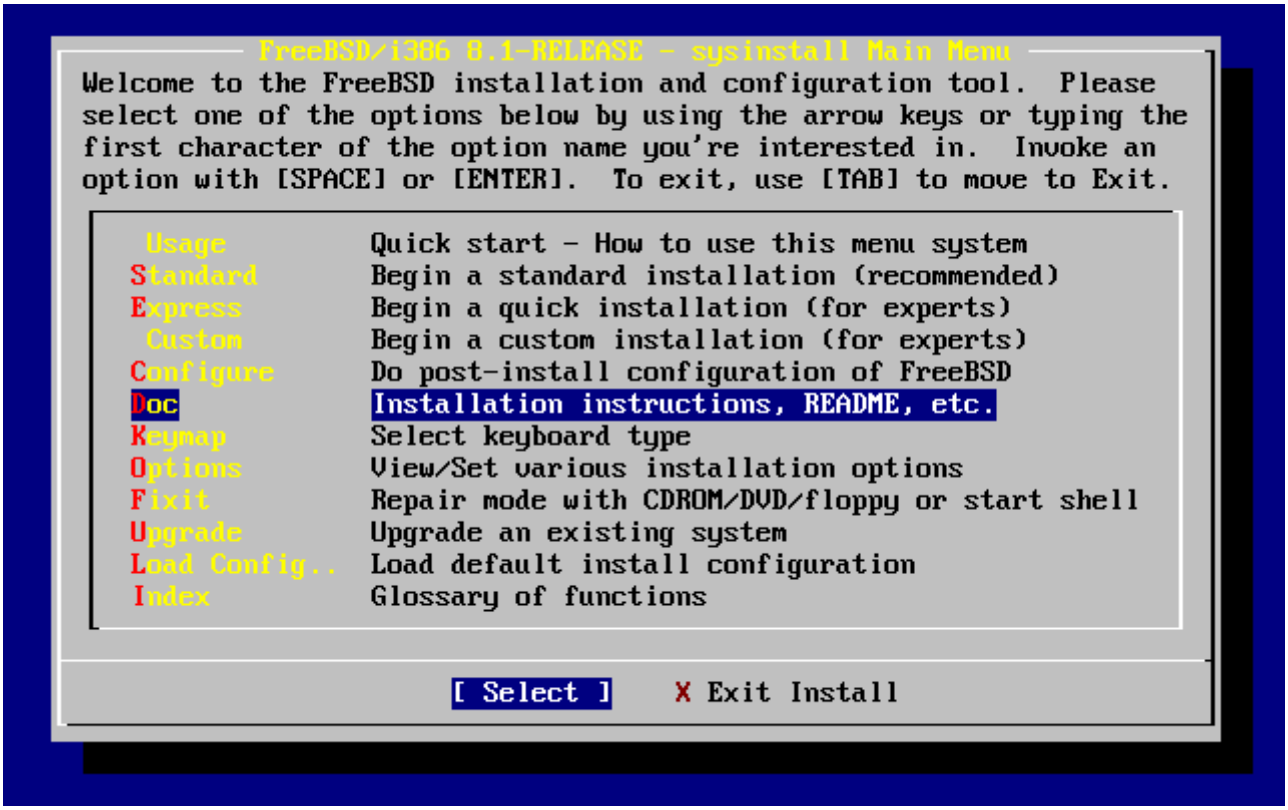


图 6. 选择说明文件菜单

这将会进入说明文件菜单。

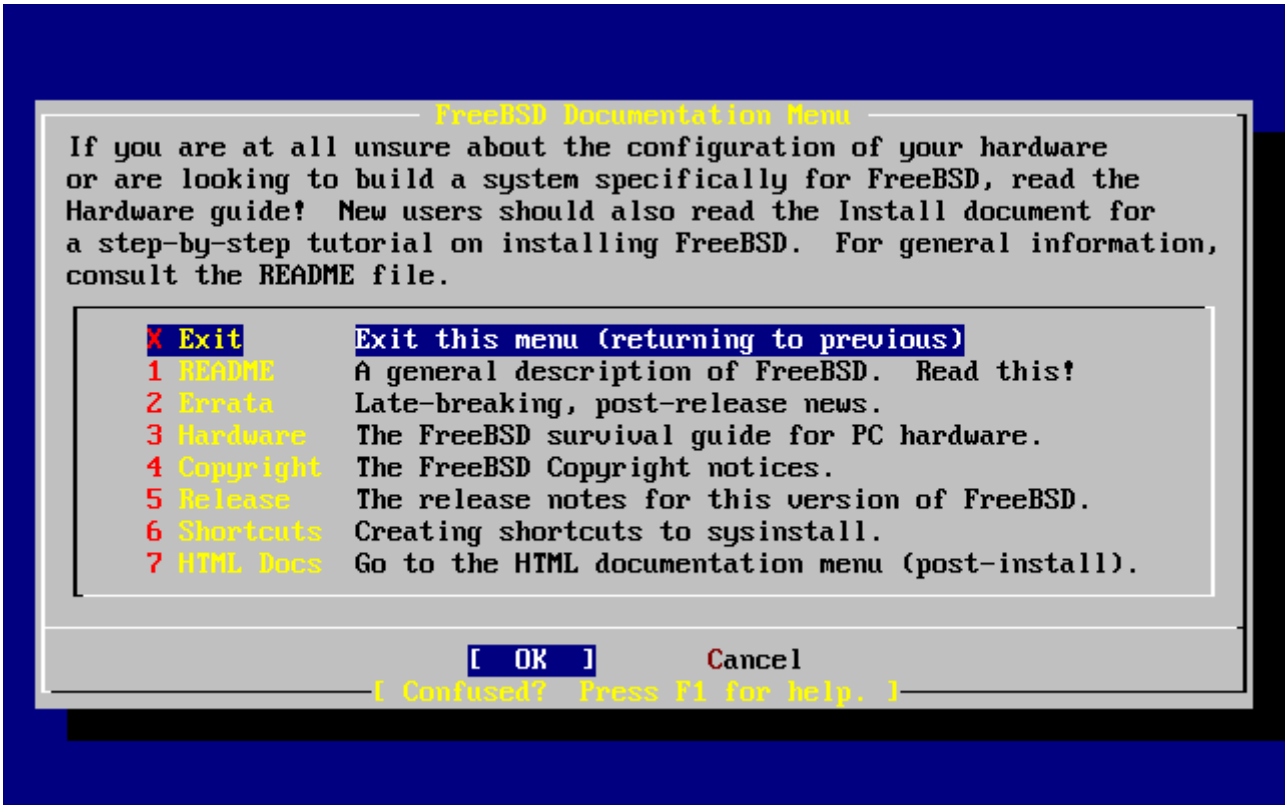


图 7. Sysinstall 说明文件菜单

阅读这些说明文件很重要。

要阅读一篇文章，请用方向键选取要阅读的文章然后按 Enter 键。阅读中再按一下 Enter 就会回到说明文件画面。

若要回到主菜单，用方向键选择 Exit 然后按下 Enter 键。

2.5.2. 选择键盘对应(Keymap)菜单

如果要改变键盘按键的对应方式， 请在主菜单选取 Keymap 然后按 Enter 键。一般情况下不改变此项，除非您使用了非标准键盘或非美国键盘。

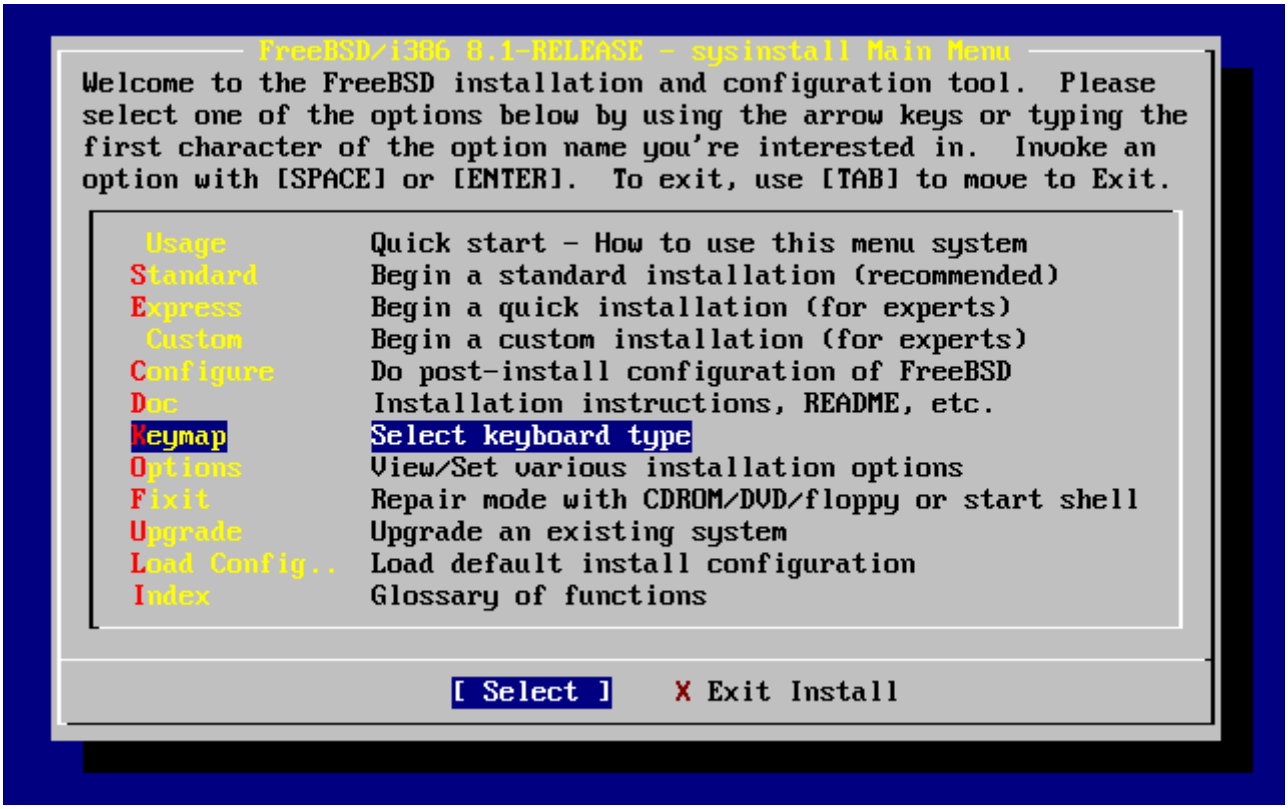


图 8. Sysinstall 主菜单

您可以使用上下键移动到您想使用的键盘对应方式， 然后按下 Space 键以选取它；再按 Space 键可以取消选取。当您完成后， 请选择 [OK] 然后按 Enter 键。

这一屏幕只显示出部分列表。选择 [Cancel] 按 Tab 键将使用默认的键盘对应， 并返回到主菜单

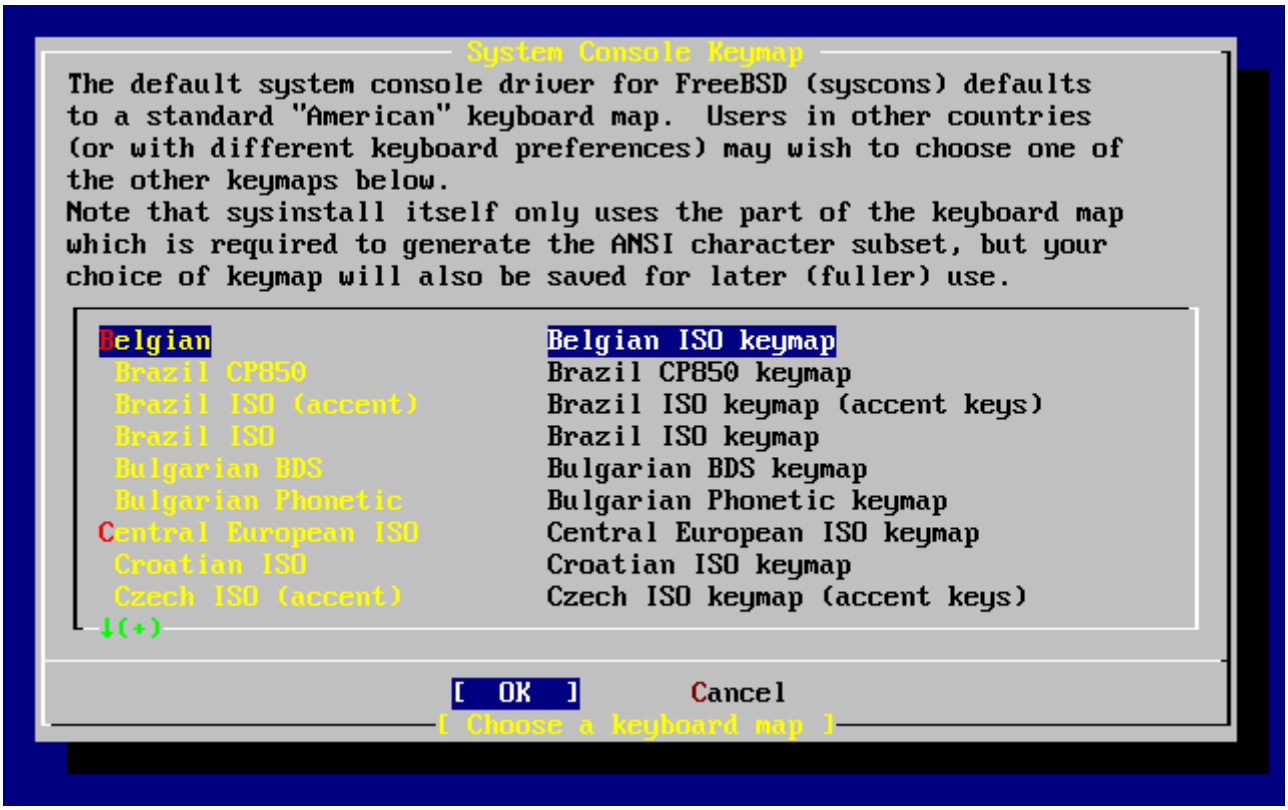


图 9. Sysinstall 键盘对应菜单

2.5.3. 安装选项设置画面

选择 Options 然后按 Enter 键。

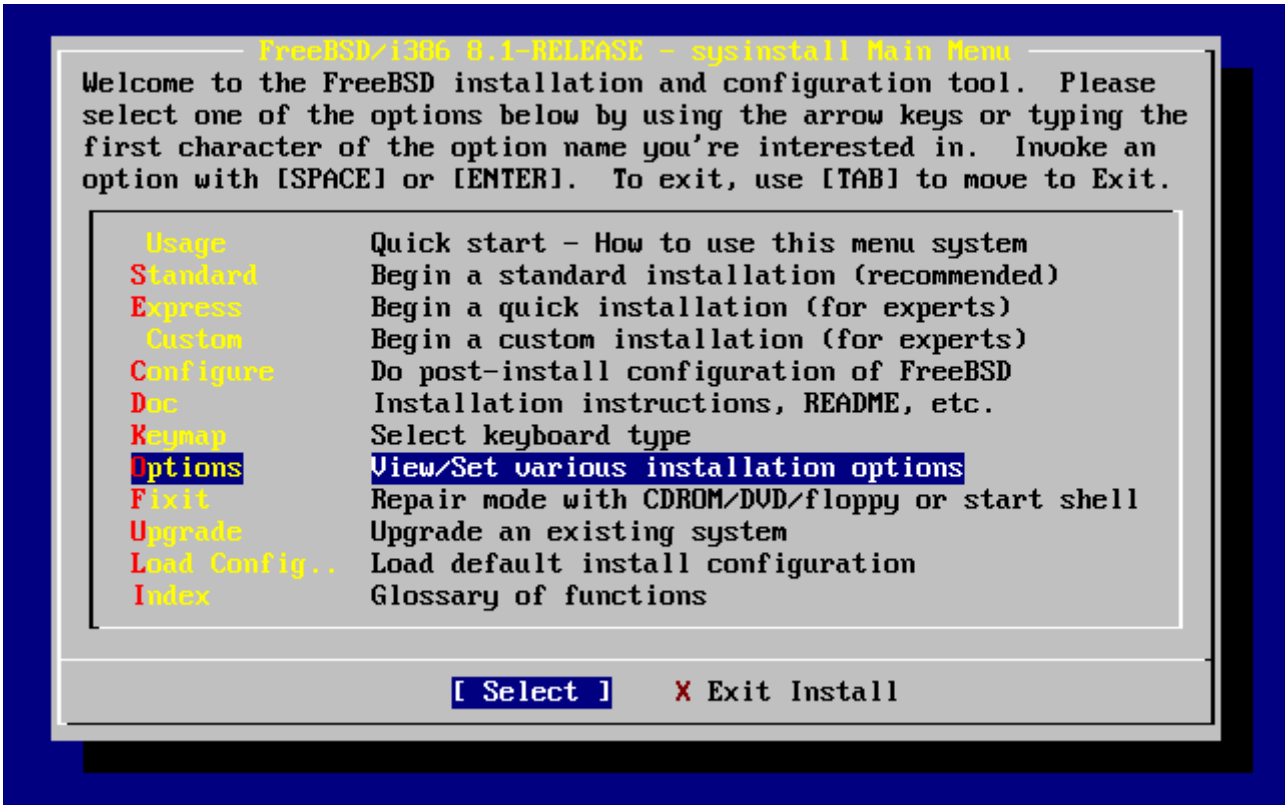


图 10. Sysinstall 主菜单

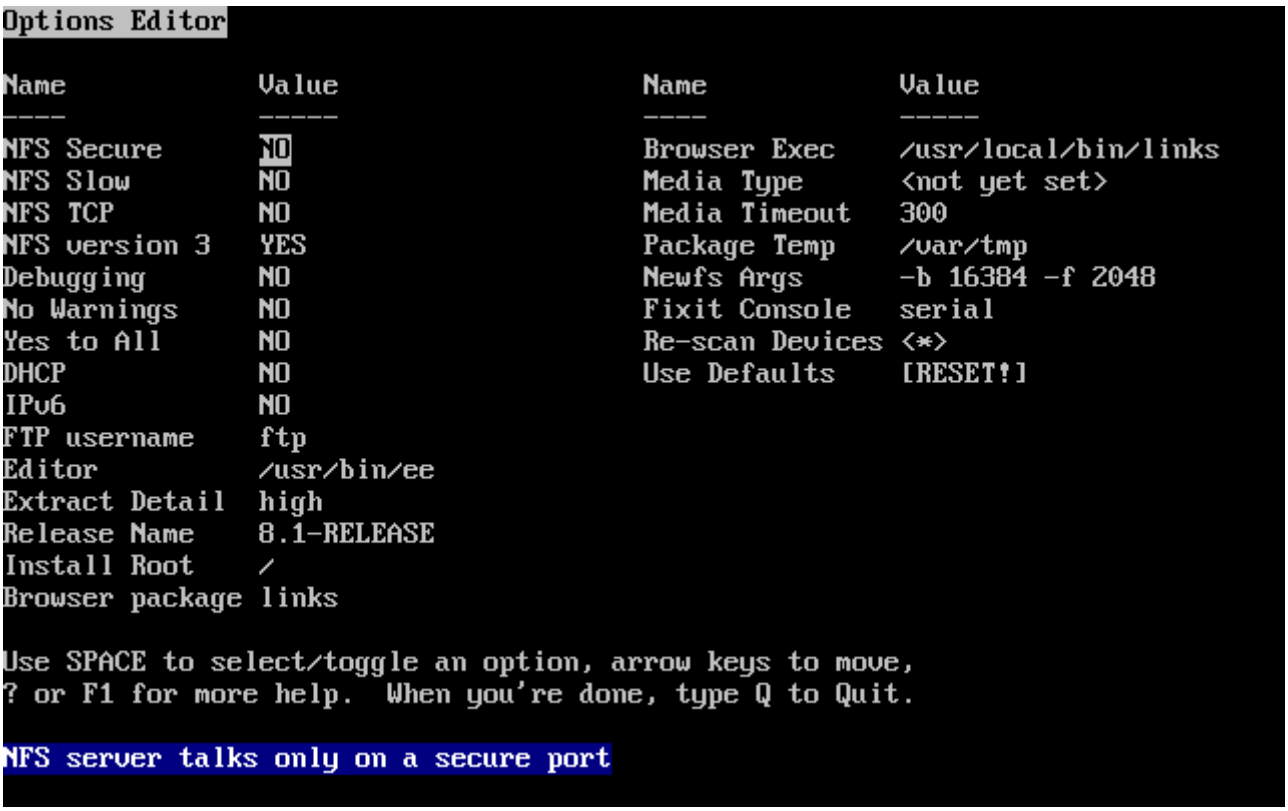


图 11. Sysinstall 选项设置

预设值通常可以适用于大部分的使用者，您并不需要改变它们。 版本名称要根据安装的版本进行变化。

目前选择项目的描述会在屏幕下方以蓝底白字显示。 注意其中有一个项目是 Use Defaults(使用默认值) 您可以由此项将所有的设定还原为预设值。

可以按下 F1 来阅读各选项的说明。

按 Q 键可以回到主画面。

2.5.4. 开始进行标准安装

Standard(标准) 安装适用于那些 UNIX® 或 FreeBSD 的初级使用者。用方向键选择 Standard 然后按 Enter 键可开始进入标准安装。

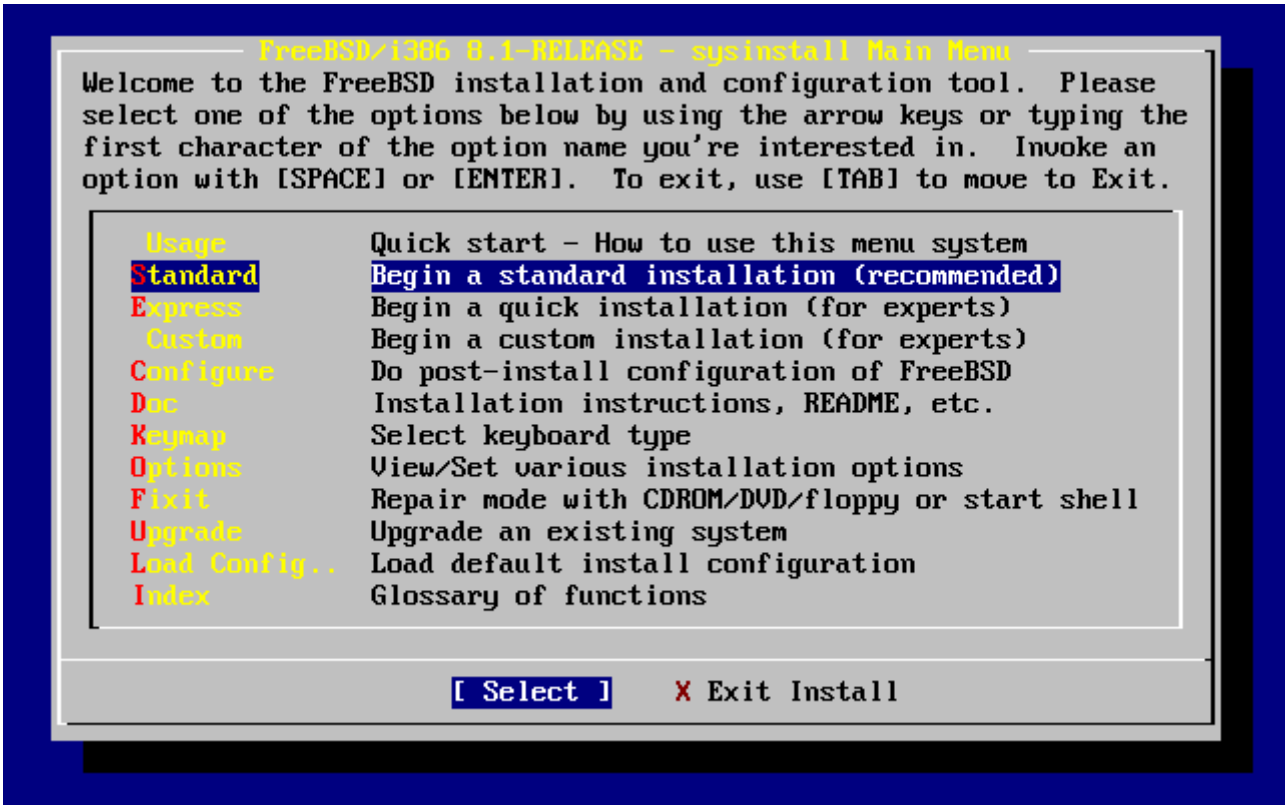


图 12. 开始进行标准安装

2.6. 分配磁盘空间

您的第一个工作就是要分配 FreeBSD 用的硬盘空间以便 sysinstall 先做好一些准备。
为了完成这个工作，您必须先对 FreeBSD 如何找到磁盘信息做一个了解。

2.6.1. BIOS 磁盘编号

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

在 pc 架构，当您跑像 MS-DOS® 或 Microsoft® Windows® 这种跟 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。这让使用者不一定非要有所谓的 "primary master" 硬盘开机。 许多人发现最简单而便宜备份系统的方式就是再去买一块一模一样的硬盘，然后定期将数据从第一块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，或者有坏轨道， 他们可以调整 BIOS 中的开机顺序而直接用第二块硬盘开机。就像交换硬盘的数据线，但是无需打开机箱。

比较昂贵，配有 SCSI 控制卡的系统通常可以延伸 BIOS 的功能来让 SCSI 设备 (可达七个) 达到类似改变顺序的功能。

习惯于使用这种方式的使用者可能会感到惊讶， 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 逻辑磁盘对应" 是怎么回事。这会让人感觉很疑惑，明明就是一样的硬盘而且资料也完全从另一块复制过来的， 结果却没办法像以前那样用。

当使用 FreeBSD 以前，请将 BIOS 中的硬盘开机顺序调回正常的顺序， 并且以后不要再改变。如果一定要交换硬盘顺序， 那请用硬件的方式， 打开机箱并调整调线。

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， 他发现这旧的 SCSI硬盘发生了许多小问题。之后， 他就跟 Bill 说起这件事。

又过了几天，Bill 决定是该解决问题的时候了， 所以他从后面房间的硬盘 "收藏" 中找出了一个一模一样的硬盘，并且经过表面测试后显示这块硬盘没有问题。 因此，Bill 将它的 ID 调成 4，然后安装到 Fred 的机器， 并且将资料从磁盘 0 复制到磁盘 4。现在新硬盘装好了， 而且看起来好像一切正常；所以，Bill 认为现在应该可以开始用它了。 Bill 于是到 SCSI BIOS 中设定 SCSI ID 4 为开机盘，用磁盘 4 重新开机后，一切跑得很顺利。

继续用了几天后，Bill 跟 Fred 决定要来玩点新的： 该将 FreeBSD 升级了。Bill 将 ID 0 的硬盘移除 (因为有问题) 并且又从收藏区中拿了一块一样的硬盘来。然后他用 Fred 神奇的网络 FTP 磁盘将新版的 FreeBSD 安装在这块硬盘上； 安装过程没什么问题发生。

Fred 用了这新版本几天后，觉得它很适合用在工程部门… 是时候将以前放在旧系统的工作资料复制过来了。 因此， Fred 将 ID4 的 SCSI 硬盘 (里面有放着旧系统中复制过来的最新资料) mount 起来，结果竟然发现在 ID4 的硬盘上， 他以前的所有资料都不见了！

资料跑到哪里去了呢?

当初 Bill 将 ID0 硬盘的资料复制到 ID4 的时候， ID4 即成为一个 "新的副本"。 而当他调 SCSI BIOS 设定 ID4 为开机盘，想让系统从 ID4 开机， 这其实只是他自己笨，因为大部分的系统可以直接调 BIOS 而改变开机顺序， 但是 FreeBSD 却会把开机顺序还原成正常的模式，因此，Fred 的 FreeBSD 还是从原来那块 ID0 的硬盘开机的。所有的资料都还在那块硬盘上， 而不是在想象之中的 ID4 硬盘。

幸运的是， 在我们发现这件事的时候那些资料都还在， 我们将这些资料从最早的那块 ID0 硬盘取出来并交还给 Fred， 而 Bill 也由此了解到计算机计数是从 0 开始的。

虽然我们这里的例子使用 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 创建分区

如果不再做改变，数据将会写进硬盘。如果您犯了一个错误想重新开始， 请选择 sysinstall 安装程序的退出按钮(exit)。或按 U 键来 Undo 操作。 如果您的操作没有结果， 您总可以重新启动您的计算机来达到您的目的。

当您在 sysinstall 主菜单选择使用标准安装后，您会看到下面的信息：

```
                        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 键， 然后您就会看到一个列表列出所有在探测设备的时候找到的硬盘。
选择要分区的硬盘 范例显示的是有找到两个 IDE 硬盘的情形，这两个硬盘分别为 ad0 和 ad2。

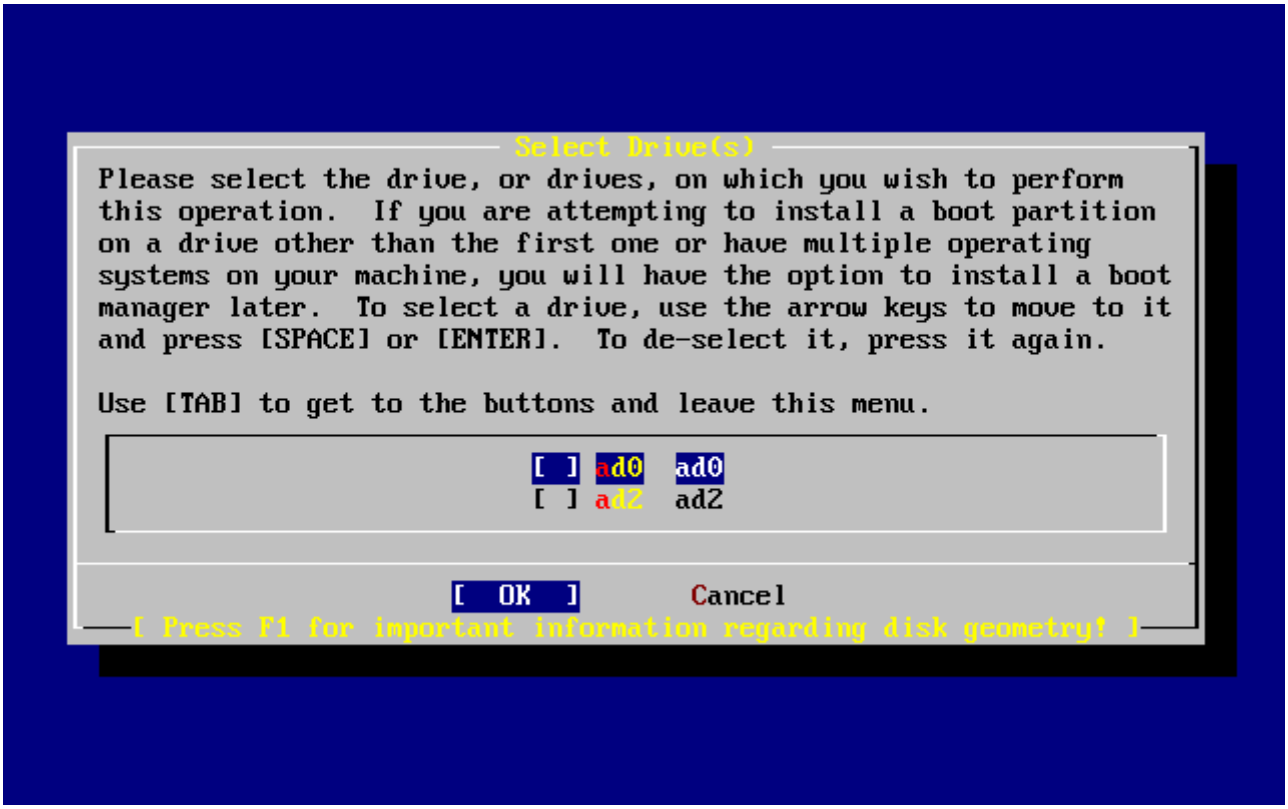


图 13. 选择要分区的硬盘

您可能正在奇怪，为什么 ad1 没有列出来？ 为什么遗失了呢?

试想，如果您有两个 IDE 硬盘，一个是在第一个 Primary master， 一个是 Secondary master，这样会发生什么事呢？ 如果 FreeBSD 依照找到的顺序来为他们命名，如 ad0 和 ad1 那么就不会有什么问题。

但是，现在问题来了。如果您现在想在 primary slave 加装第三个硬盘， 那么这个硬盘的名称就会是 ad1，之前的 ad1 就会变成 ad2。 这会造成什么问题呢？因为设备的名称 （如 ad1s1a）是用来寻找文件系统的， 因此您可能会发现，突然，您有些文件系统从此无法正确地显示出来，必须修改 FreeBSD 配置文件（译注：/etc/fstab）才可以正确显示。

为了解决这些问题，在配置内核的时候可以叫 FreeBSD 直接用 IDE 设备所在的位置来命名，而不是依据找到的顺序。使用这种方式的话， 在 secondary master 的 IDE 设备就永远是ad2，即使您的系统中没有 ad0 或 ad1 也不受影响。

此为 FreeBSD 内核的默认值，这也是为什么上面的画面只显示 ad0 和 ad2 的原因。 画面上这台机器的两颗硬盘是装在 primary 及 secondary 的 master 上面； 并没有任何一个硬盘安装在 slave 插槽上。

您应该选择您想安装 FreeBSD 的硬盘，然后按下 [OK]。之后 FDisk 就会开始，您会看到类似 典型的尚未编辑前的 Fdisk 分区表的画面。

FDisk 的显示画面分为三个部分。

第一部分是画面上最上面两行，显示的是目前所选择的硬盘的信息。 包含它的 FreeBSD 名称、硬盘分布以及硬盘的总容量。

第二部分显示的是目前选择的硬盘上有哪些分区， 每个分区的开始及结束位置、所占容量、FreeBSD 名称、 它们的描述以及类别（sub-type）。此范例显示有两个未使用的小分区， 还有一个大的 FAT 分区，（很可能是 MS-DOS® 或 Windows® 的 C: ）， 以及一个扩展分区（在 MS-DOS® 或 Windows® 里面还可以包含逻辑分区）。

第三个部分显示 FDisk 中可用的命令。

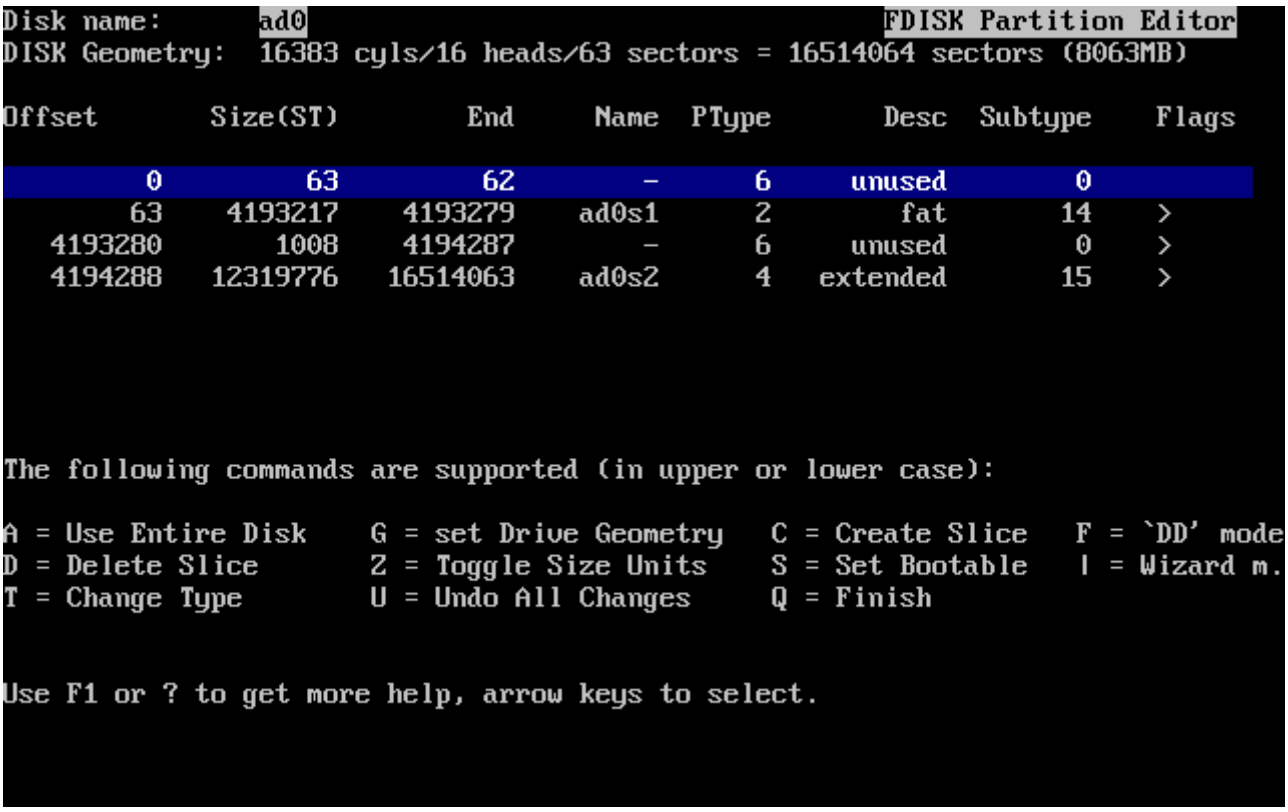


图 14. 典型的尚未编辑前的 Fdisk 分区表

接下来要做的事跟您要怎么给您的硬盘分区有关。

如果您要让 FreeBSD 使用整个硬盘（稍后您确认要 sysinstall 继续安装后会删除所有这个硬盘上的资料），那么您就可以按 A 键（Use Entire Disk ）目前已有的分区都会被删除，取而代之的是一个小的，标示为 unused 的分区，以及一个大的 FreeBSD 分区。之后，请用方向键将光标移到这个 FreeBSD 分区，然后按 S 以将此分区标记为启动分区。您会看到类似 Fdisk 分区使用整个硬盘 的画面。注意，在 Flags 栏中的 A 记号表示此分区是 激活 的，因而启动将从此分区进行。

要删除现有的分区以便为 FreeBSD 腾出空间，您可以将光标移动到要删除的分区后按 D 键。然后就可按 C 键，并在弹出的对话框中输入将要创建的分区的大小。输入合适的大小后按 Enter 键。一般而言，这个对话框中的初始值是可以分配给该分区的最大值。它可能是最大的邻接分区或未分配的整个硬盘大小。

如果您已经建立好给 FreeBSD 的分区（使用像 PartitionMagic®类似的工具），那么您可以按下 C 键来建立一个新的分区。同样的，会有对话框询问您要建立的分区的大小。

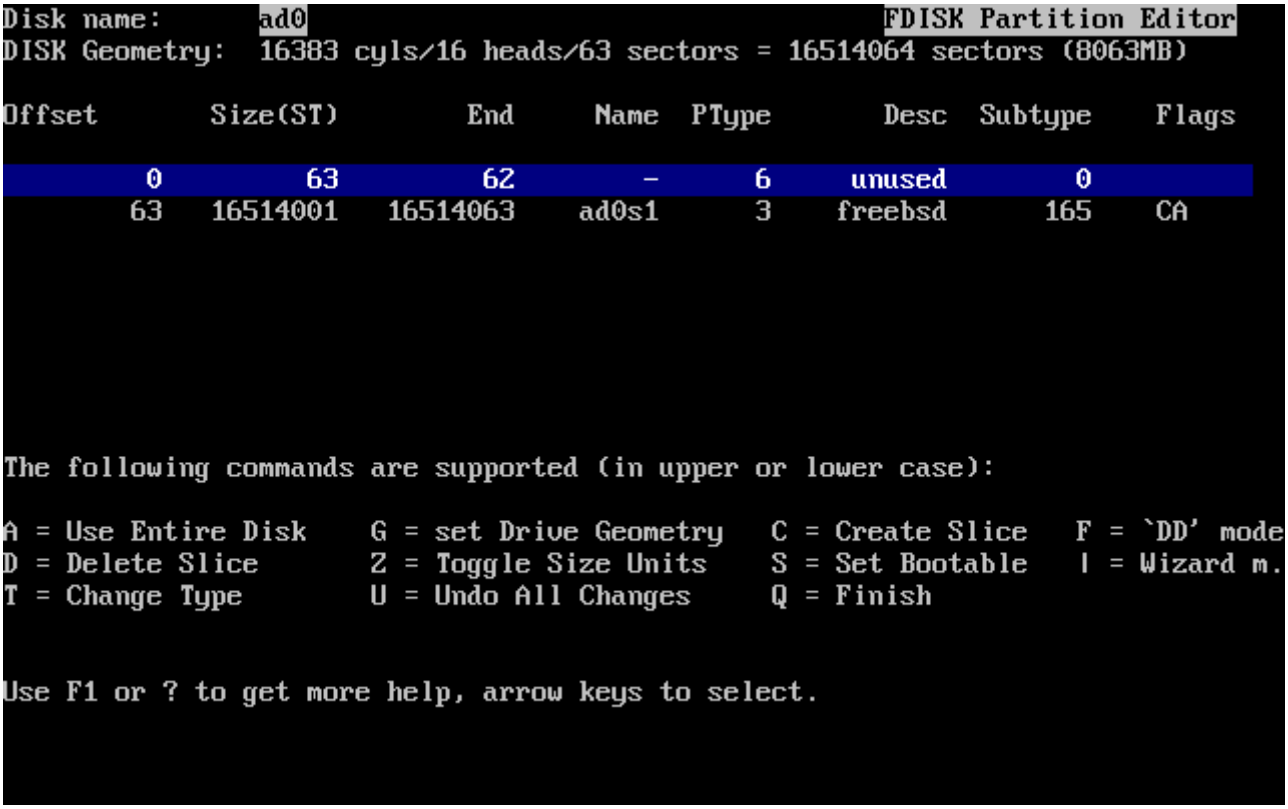


图 15. Fdisk 分区使用整个硬盘

完成后，按 Q 键。您的变更会存在 sysinstall 中， 但是还不会真正写入您的硬盘。

2.6.3. 安装多重引导

在这步骤您可以选择要不要安装一个多重引导管理器。 一般而言，如果碰到下列的情形，
您应该选择要安装多重引导管理程序。

- 您有一个以上的硬盘，并且 FreeBSD 并不是安装在第一个硬盘上。
- 除了 FreeBSD，您还有其它的操作系统安装在同一块硬盘上，
  所以您需要在开机的时候选择要进入哪一个系统。

如果您在这台机器上只安装一个 FreeBSD 操作系统， 并且安装在第一个硬盘， 那么选择 Standard
安装就可以了。如果您已经使用了一个第三方的多重引导程序， 那么请选择 None。

选择好配置后请按 Enter。

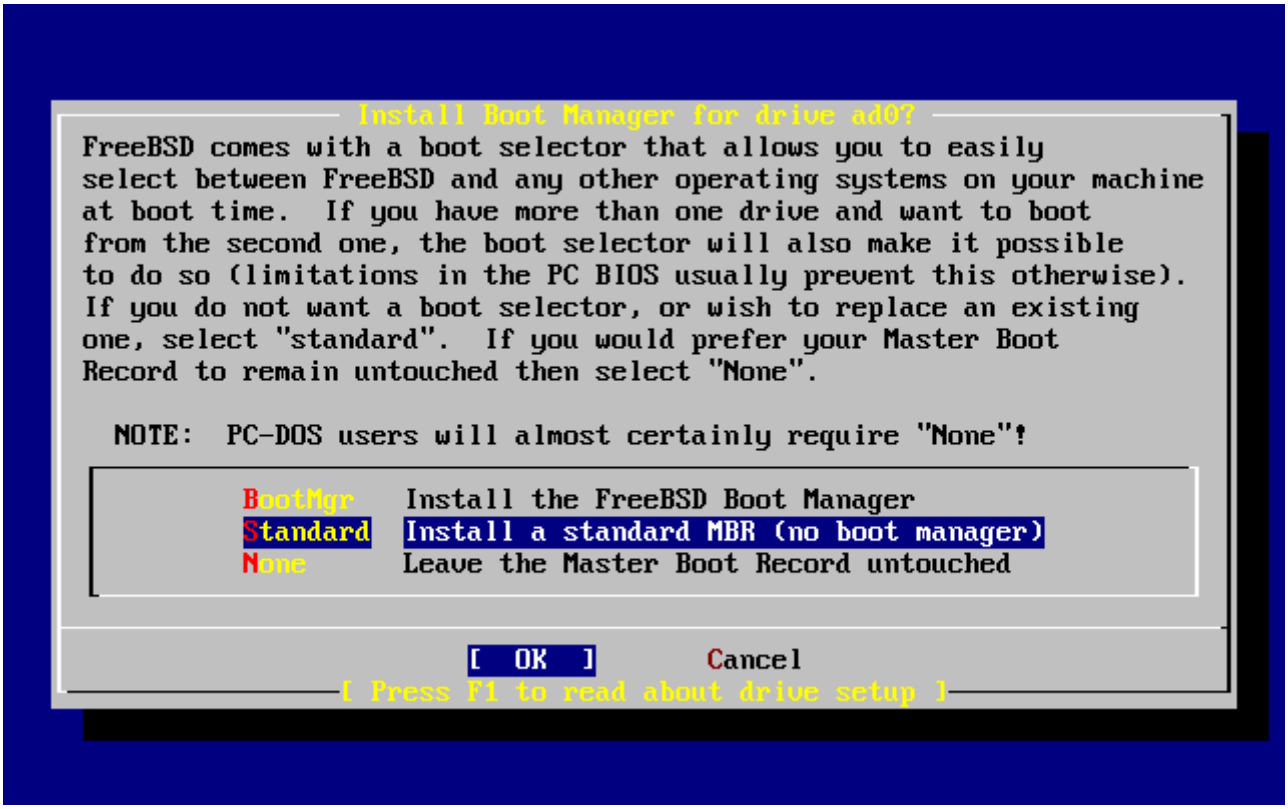


图 16. Sysinstall 多重引导管理程序

按下 F1 键所显示的在线说明中有讨论一些操作系统共存可能发生的问题。

2.6.4. 在其它硬盘上创建分区

如果您的系统上有一个以上的硬盘， 在选择完多重引导管理程序后会再回到选择硬盘的画面。 如果您要将 FreeBSD 安装在多个硬盘上，那么您可以在这里选择其它的硬盘， 然后重复使用 FDisk 来建立分区。

如果您想让 FreeBSD 来管理其它的硬盘， 那么两个硬盘都必须安装 FreeBSD 的多重引导管理程序。

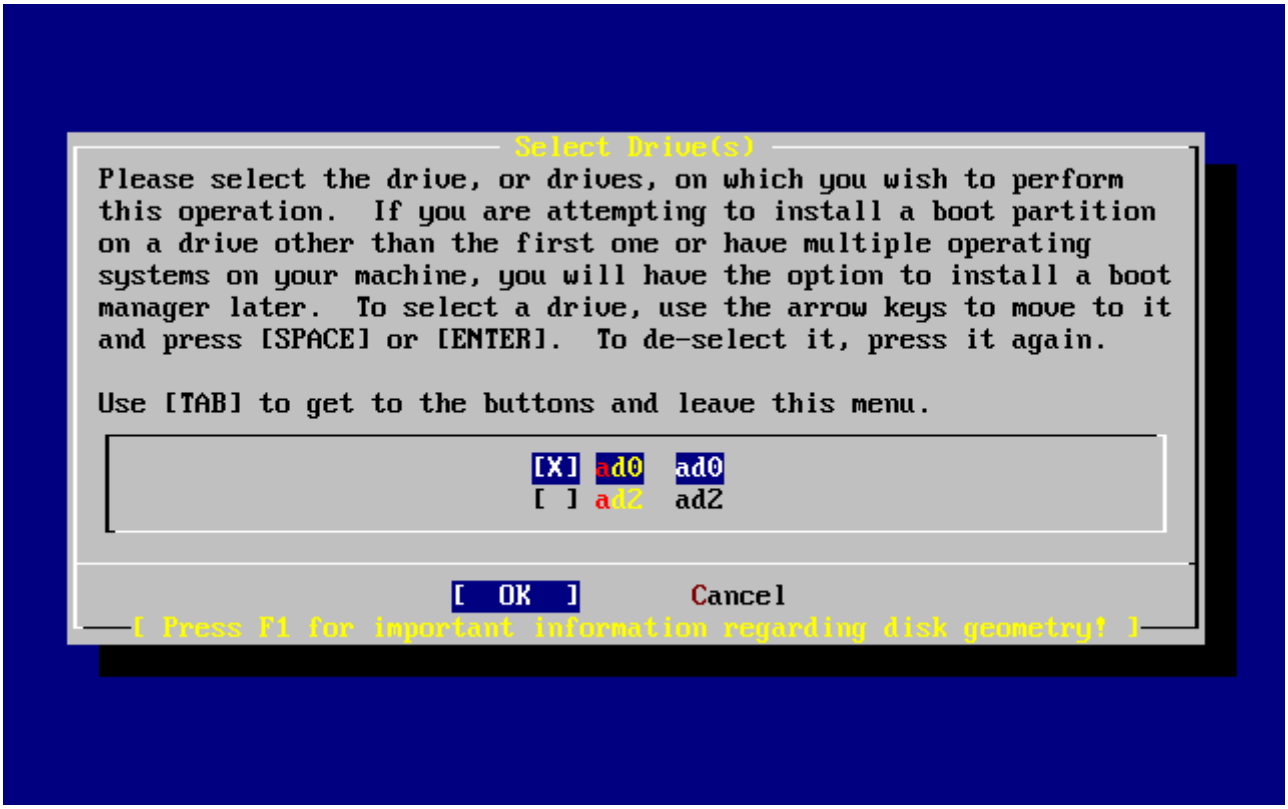


图 17. 离开选择硬盘画面

Tab 键可以在您最后选择的硬盘、 [OK] 以及 [Cancel] 之间进行切换。

用 Tab 键将光标移动到 [OK] 然后按 Enter 键继续安装过程。

2.6.5. 使用 bsdlabel 创建分区

您现在必须在刚刚建立好的 slice 中规划一些 label。 请注意，每个 label 的代号是 a 到 h，另外，习惯上 b、 c 和 d 是有特殊用途的，不应该随意变动。

某些应用程序可以利用一些特殊的分区而达到较好的效果，
尤其是分区分散在不同的硬盘的时候。但是，现在您是第一次安装FreeBSD，
所以不需要去烦恼如何分割您的硬盘。最重要的是，
装好FreeBSD然后学习如何使用它。当您对FreeBSD有相当程度的熟悉后，
您可以随时重新安装FreeBSD，然后改变您分区的方式。

下面的范例中有四个分区 - 一个是磁盘交换分区，另外三个是文件系统。

表 2. 为第一个硬盘分区

分区	文件系统	大小	描述
a	/	1 GB	这是一个根文件系统（root filesystem）。 任何其它的文件系统都会挂在根目录（译注：用根目录比较亲切） 下面。 1 GB 对于此目录来说是合理的大小， 因为您往后并不会在这里存放太多的数据； 在安装 FreeBSD 后会用掉约 128 MB 的根目录空间。 剩下的空间是用来存放临时文件用的，同时， 您也应该预留一些空间， 因为以后的FreeBSD版本可能会需要较多的 /（根目录）空间。
b	N/A	2-3 x RAM	b 分区为系统磁盘交换分区（swap space）。选择正确的交换空间大小可是一门学问唷。 一般来说，交换空间的大小应该是您系统上内存（RAM） 大小的2到3倍。 交换空间至少要有 64 MB。因此， 如果您的电脑上的 RAM 比 32 MB 小， 请将交换空间大小设为 64 MB。 如果您有一个以上的硬盘， 您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间，这样做能够提高 swap 的性能。 如果是这种情形， 先算出您总共需要的交换空间大小 （如128 MB），然后除以您拥有的硬盘数目（如2块）， 算出的结果就是每个硬盘上要配置的交换空间的大小。 在这个例子中， 每个硬盘的交换空间为 64 MB。

分区	文件系统	大小	描述
e	/var	512 MB 至 4096 MB	/var 目录会存放不同长度的文件、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。当这些文件放在另外的文件系统（译注：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）	您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。

上面例子中的数值仅限于有经验的用户使用。 通常我们鼓励用户使用 FreeBSD 分区编辑器中一个叫做 Auto Defaults的自动分区布局功能。

如果您要将FreeBSD安装在一个以上的硬盘， 那么您必须在您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. 为其它磁盘分区

分区	文件系统	大小	描述
b	N/A	见描述	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，习惯上还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规划为 a分区而不是 e分区。然而， 习惯上 a分区是保留给根目录 (/) 用的。您不一定要遵守这个习惯，但是 sysinstall 会，所以照着它做会使您的安装比较清爽、干净。您可以将这些文件系统挂在任何地方，本范例建议将它们挂在 /diskn 目录，n 依据每个硬盘而有所不同，但是，您喜欢的话也可将它们挂在别的地方。

分区的配置完成后，您可以用sysinstall. 来建立它们了。您会看到下面的信息：

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 键开始FreeBSD分区表编辑器，称做 Disklabel。

Sysinstall Disklabel 编辑器 显示您第一次执行 Disklabel的画面。 画面分为三个区域。

前几行显示的是您正在编辑的硬盘以及您正在建立的 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 还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。

画面中间区域显示已建立的区区，每个分区的文件系统名称、所占的大小以及一些关于建立这些文件系统的参数选项。

下方的第三区显示在 Disklabel 中可用的按键。

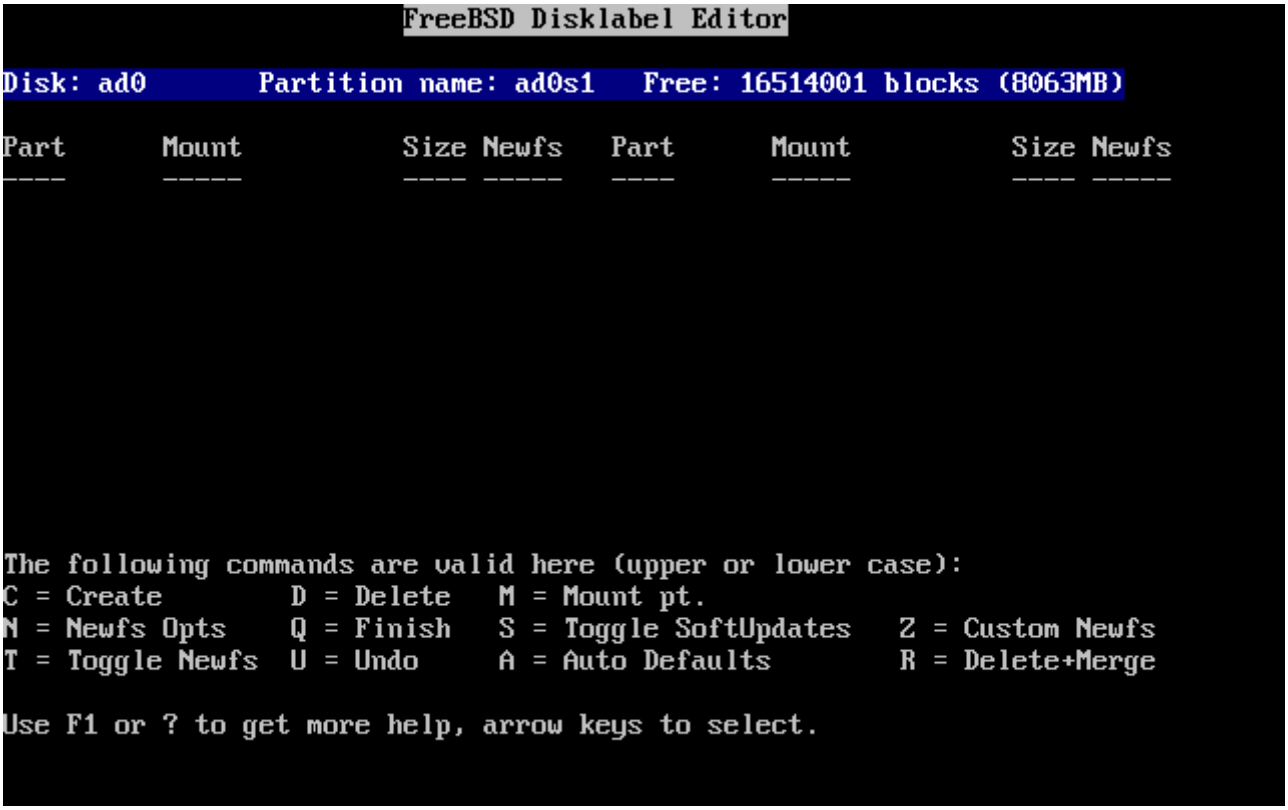


图 18. Sysinstall Disklabel 编辑器

Disklabel 您可以自动配置分区以及给它们预设的大小。这些默认的分区是由内部的分区尺寸算法根据磁盘的大小计算出的。 您可以按 A 键使用此功能。您会看到类似 Sysinstall Disklabel 编辑器-使用自动配置的画面。根据您硬盘的大小，自动分配所配置的大小不一定合适。但是没有关系， 您并不一定要使用预设的大小。

默认情况下会给/tmp 目录一个独立分区，而不是附属在 / 之下。这样可以避免将一些临时文件放到根目录中（译注： 可能会用完根目录空间）。

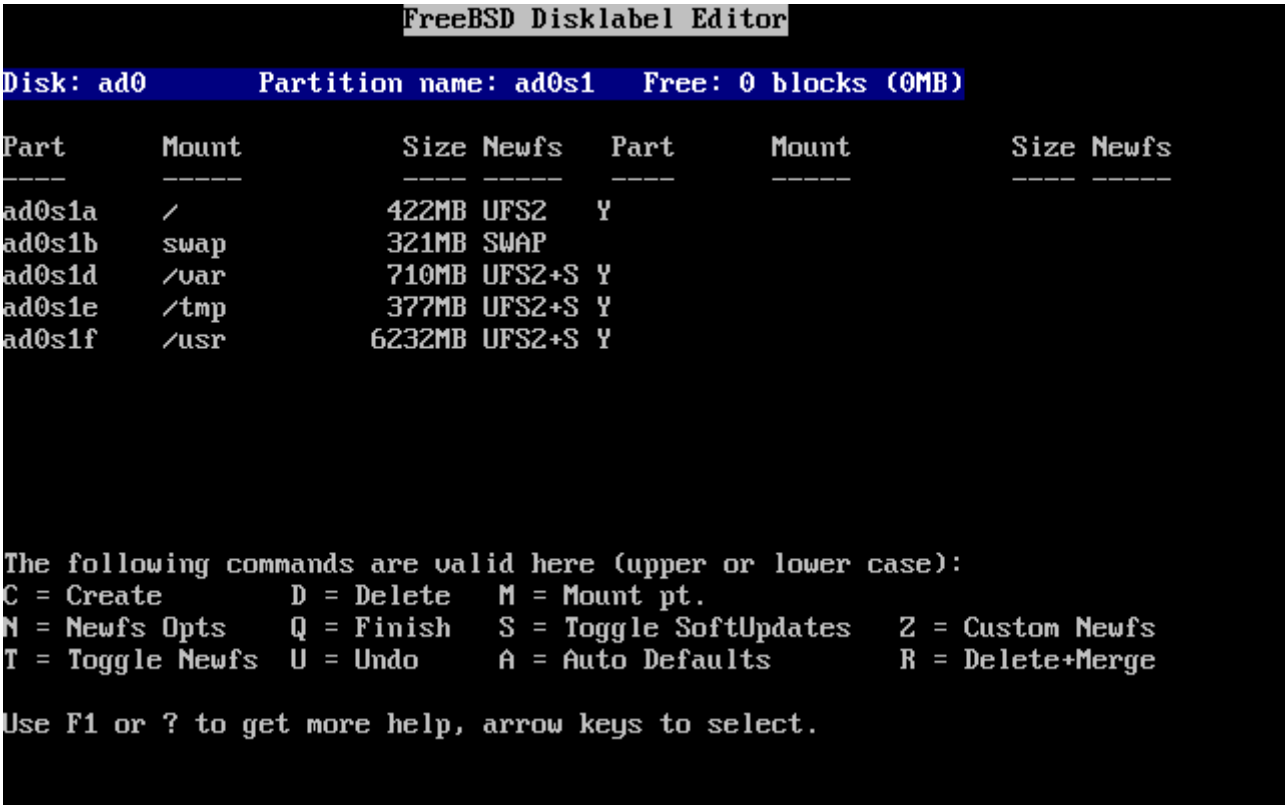


图 19. Sysinstall Disklabel 编辑器-使用自动配置

如果您不想使用默认的分区布局， 则需要用方向键移动光标并选中第一个分区， 然后按 D 来删除它。
重复这一过程直到删除了所有推荐的分区。

要建立第一个分区 (a，作为 / - 根文件系统)， 请确认您已经在屏幕顶部选中了正确的 slice， 然后按 C。
接下来将出现一个对话框， 要求您输入新分区的尺寸 (如 根目录使用空间 所示)。
您可以输入以块为单位的尺寸，或以 M 表示MB、 G 结尾表示GB， 或者 C 表示柱面数的方式来表达尺寸。

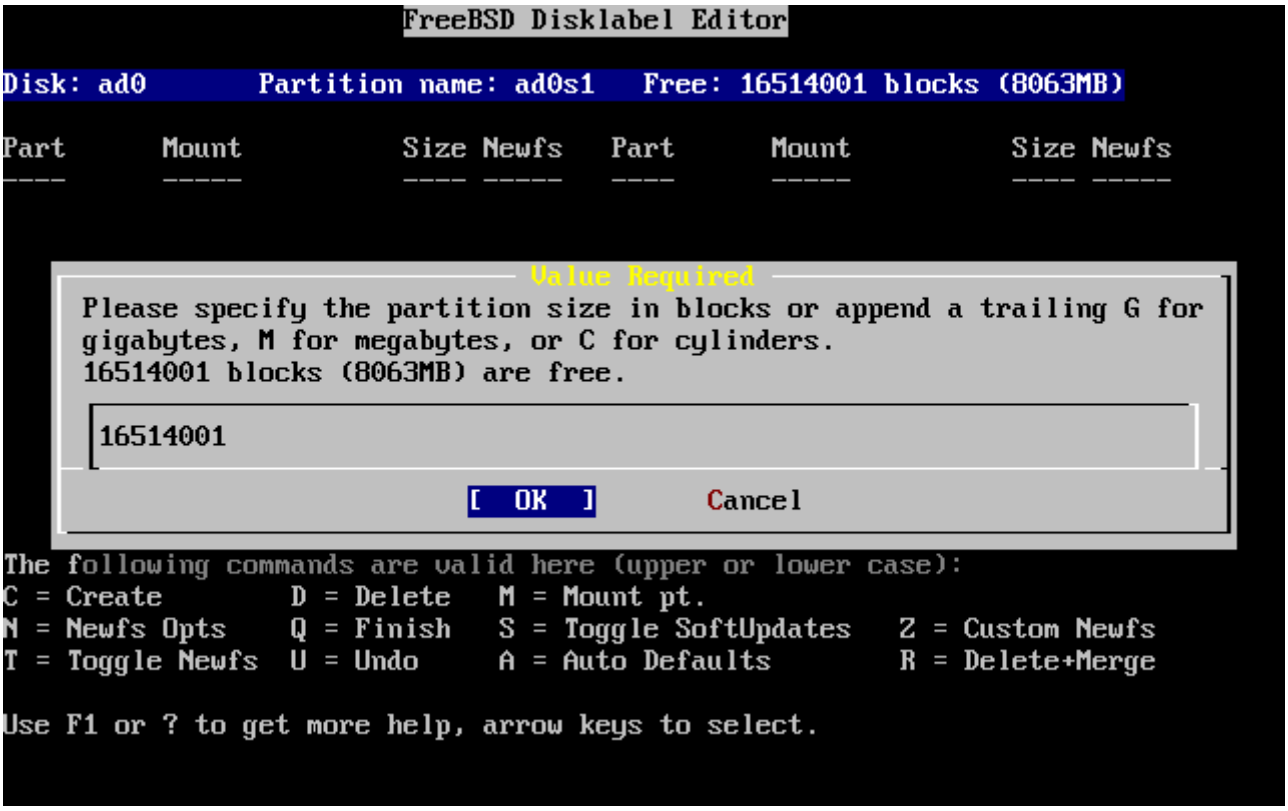


图 20. 根目录使用空间

如果使用此处显示的默认尺寸， 则会创建一个占满整个 slice 空余空间的

partition。如果希望使用前面例子中描述的 partition 尺寸， 则应按 Backspace 键删除这些数字， 并输入 512M， 如 编辑要分区大小 所示。 然后， 按下 [OK]。

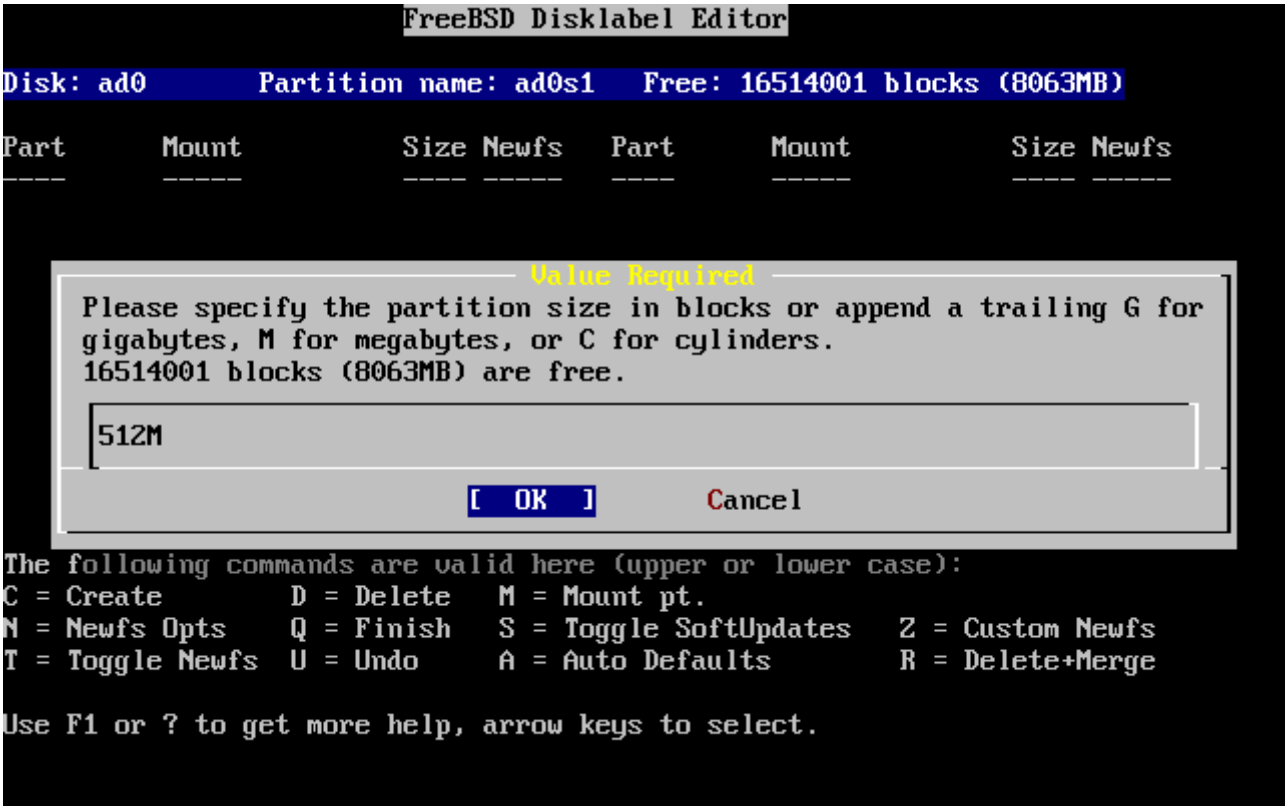


图 21. 编辑要分区大小

输入完大小后接着问您要建立的分区是文件系统还是交换空间，如 选择根分区类型所示。第一个分区是文件系统， 所以确认选择 FS后按 Enter 键。

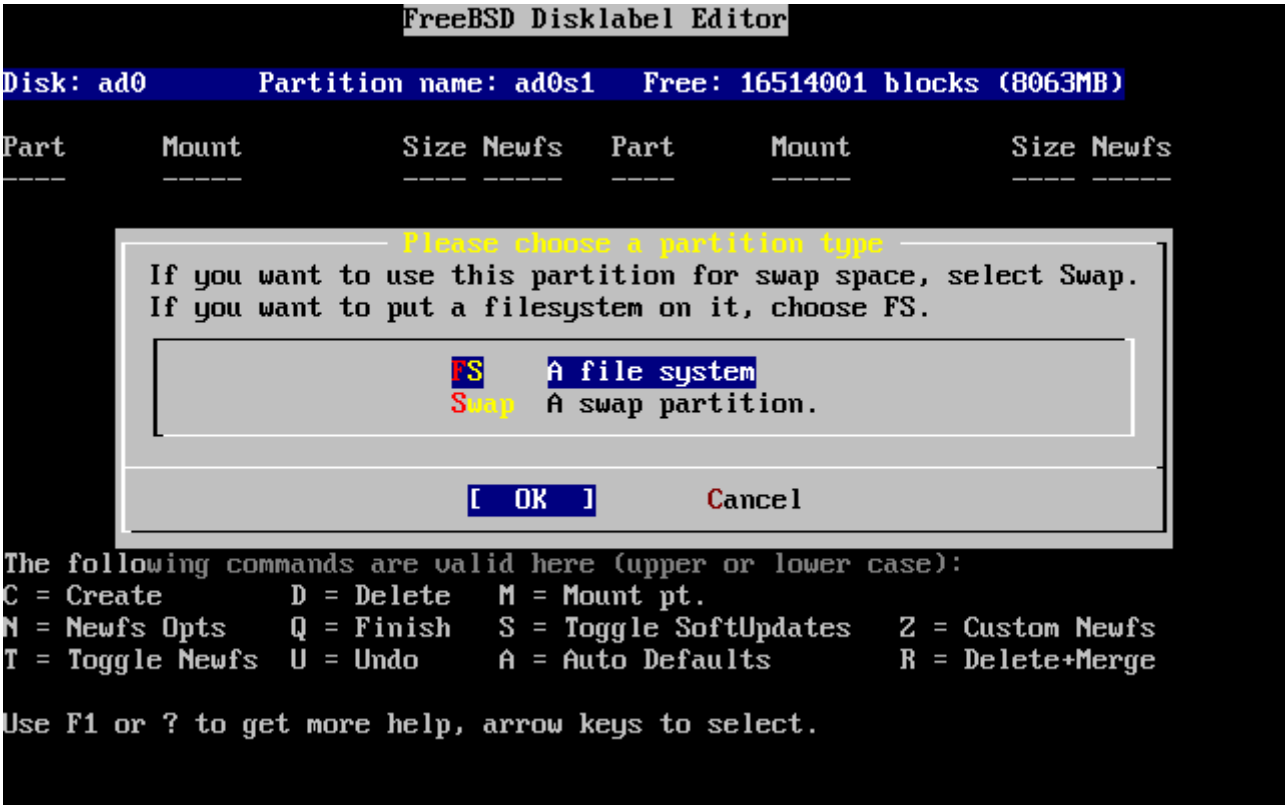


图 22. 选择根分区类型

最后，因为您要建立的是一个文件系统，所以必须告诉 Disklabel 这个文件系统要挂接在什么地方，如 选择根挂接点所示。根文件系统的挂接点 /, 所以请输入 /,然后按 Enter 键。

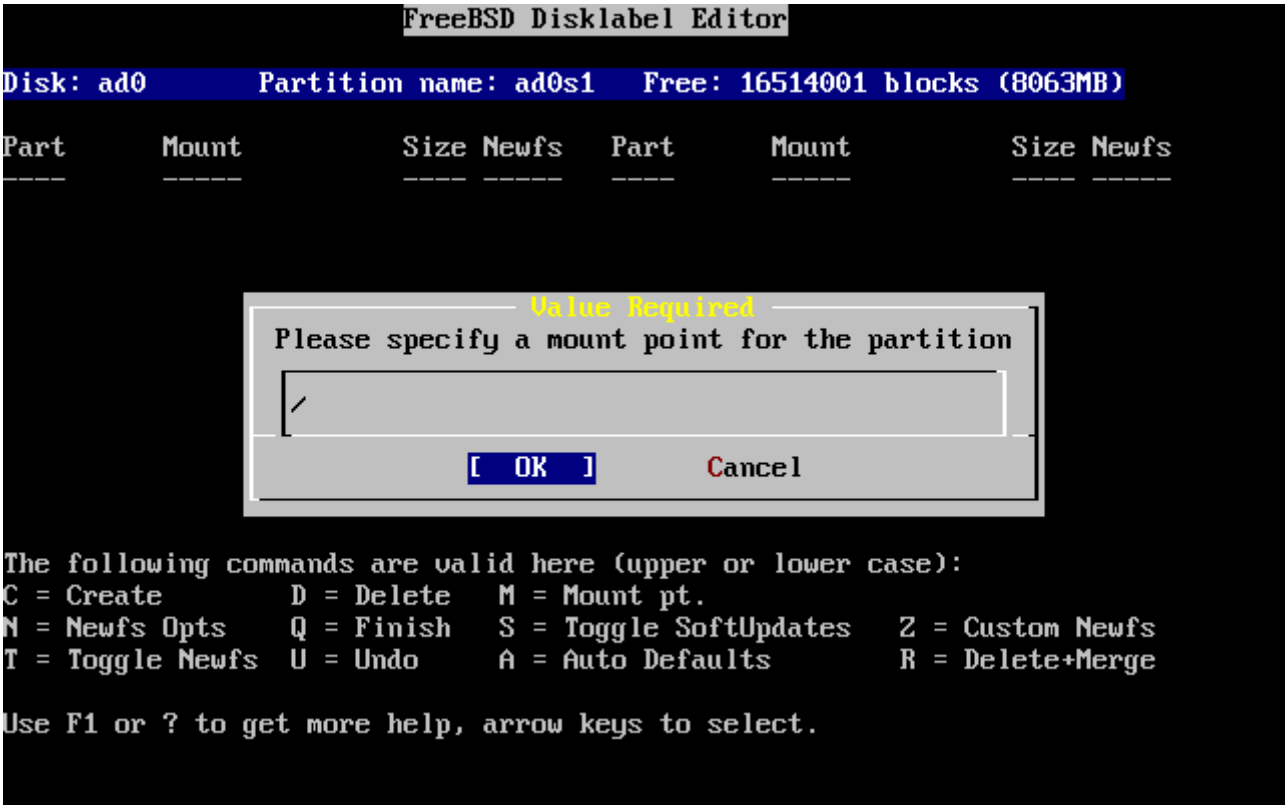


图 23. 选择根挂接点

刚刚制作好的分区会显示在画面上。 您应该重复上述的动作以建立其它的分区。当建立交换空间的时候，系统不会问您要将它挂接在哪里，因为交换空间是不用挂在系统上的。 当您在建立最后一个分区 /usr的时候， 您可以直接使用默认的大小，即所有此分区剩余的空间。

您最终的 FreeBSD DiskLabel 编辑器画面会类似 Sysinstall Disklabel 编辑器，实际数字按您的选择而有所不同。 按下 Q 键完成分区的建立。

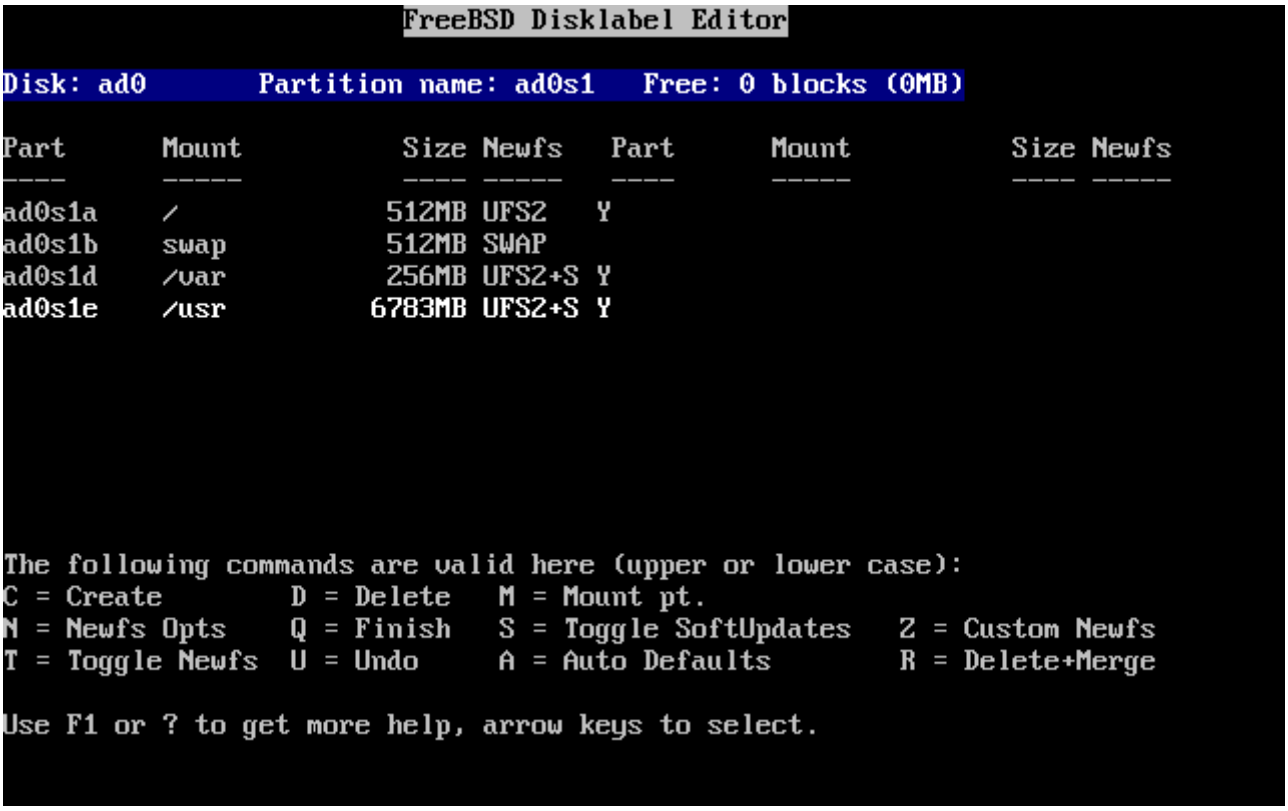


图 24. Sysinstall Disklabel 编辑器

2.7. 选择要安装的软件包

2.7.1. 选择要安装的软件包

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。内建的选项包括了运行所需要的最小系统，
到把所有软件包全都装上的常用配置。UNIX® 或 FreeBSD
新手通常直接选择一个设定好的软件包就可以了， 而有经验的使用者则可以考虑自己订制安装哪些软件包。

按下 F1 可以看到有关软件包的更多选项信息， 以及它们都包含了哪些软件，之后，可以按 Enter
回到软件包选择画面。

如果需要图形用户界面， 则配置 X 服务以及选择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， 您还需要选择包含源代码的那个选项。 要了解为什么应该编译和构建新的内核，
请参见 配置FreeBSD的内核。

显然， 包含所有组件的系统是最万能的。 如果磁盘空间足够， 用光标键选择 选择软件包 中的 All 并按
Enter。 如果担心磁盘空间不够的话， 则选择最合适的选项。 不要担心选择的是否是最合适的，
因为其他软件包可以在安装完毕后再加入进来。

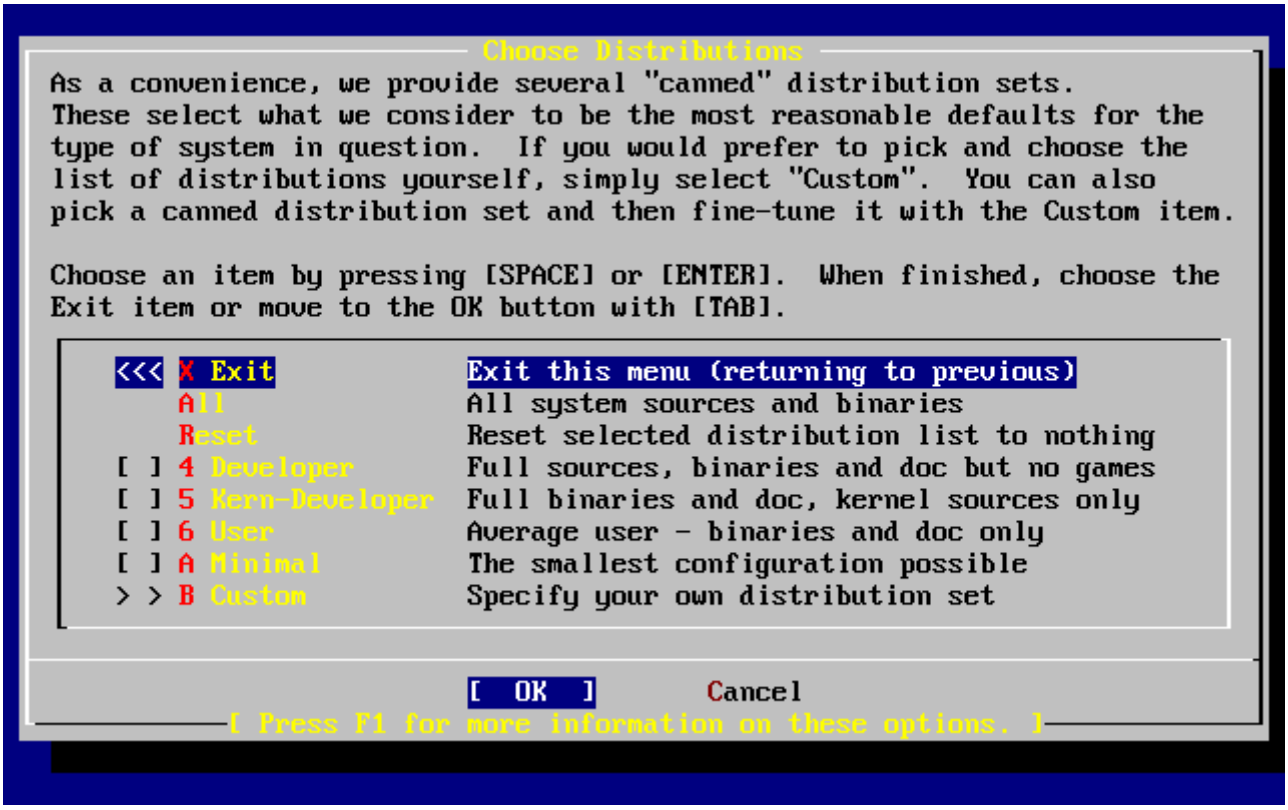


图 25. 选择软件包

2.7.2. 安装ports软件包

当选择完您想要安装的部分后，接着会询问您要不要安装FreeBSD Ports
软件包；Ports软件包可以让您简单方便地安装软件包。Ports本身并不包含编辑
软件所需要的程序源代码，而是一个包含自动下载、编辑以及安装的文档集合。 安装应用程序: Packages 和
Ports 一章讨论如何使用Ports.

安装程序并不会检查您是否有足够的硬盘空间, 在选择这一项之前请先确定您有足够的硬盘空间。 目前
FreeBSD 12.0 版本中， FreeBSD Ports Collection 大约占用 3 GB 大小的硬盘空间。
对于近期的版本您可能需要更多一些空间来安装他们。

```
                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
```

选择 [yes] 将会安装 Ports Collection， 而选择 [no] 则将跳过它。 选好后按 Enter 继续。 此后，选择安装的软件包的屏幕将再次出现。

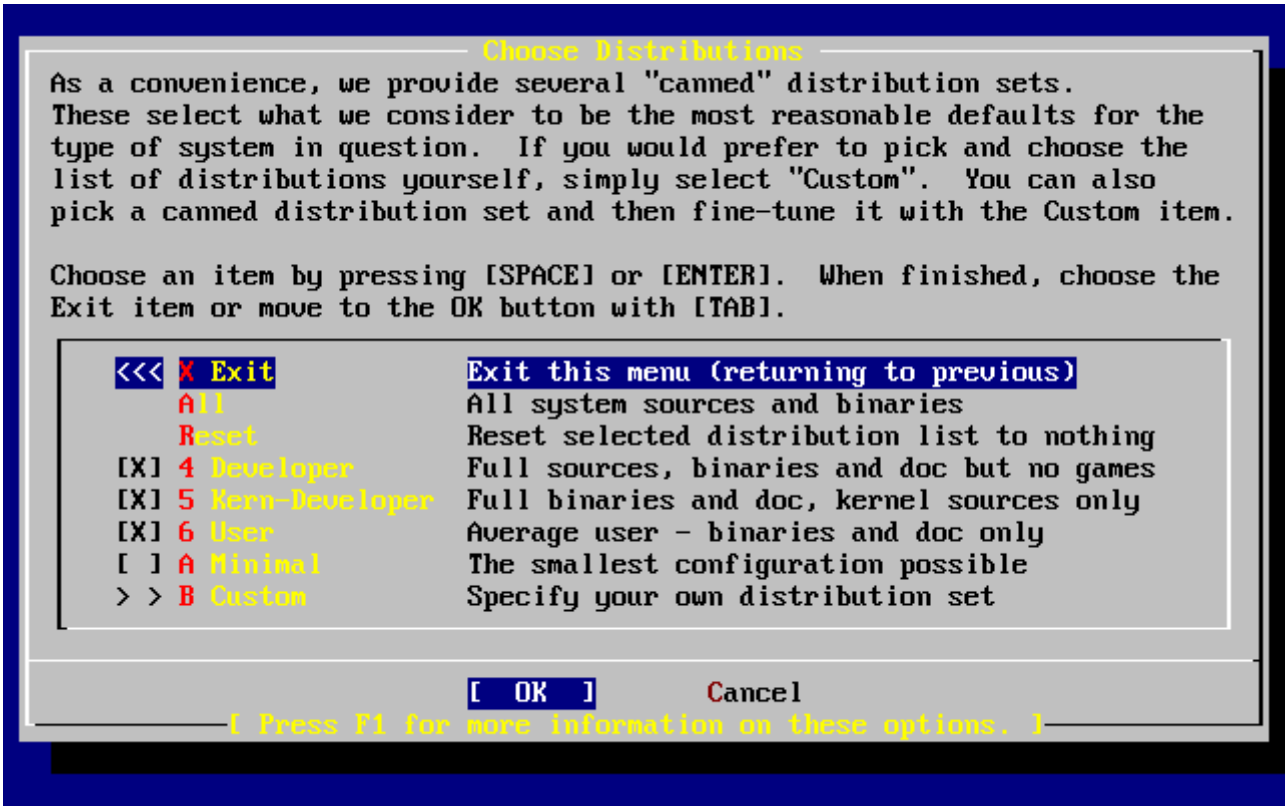


图 26. 确认您要安装的软件包

如果对您的选择感到满意，请选择Exit 退出，确保[OK] 被高亮显示，然后按Enter 继续。

2.8. 选择您要使用的安装介质

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。确认 [OK] 被选取，然后按 Enter 开始安装程序。

如果要使用其它的方式安装， 请选择适当的安装介质然后按照屏幕指示进行安装。

按 F1 可以显示安装介质的在线说明。按一下 Enter 可返回选择安装介质画面。

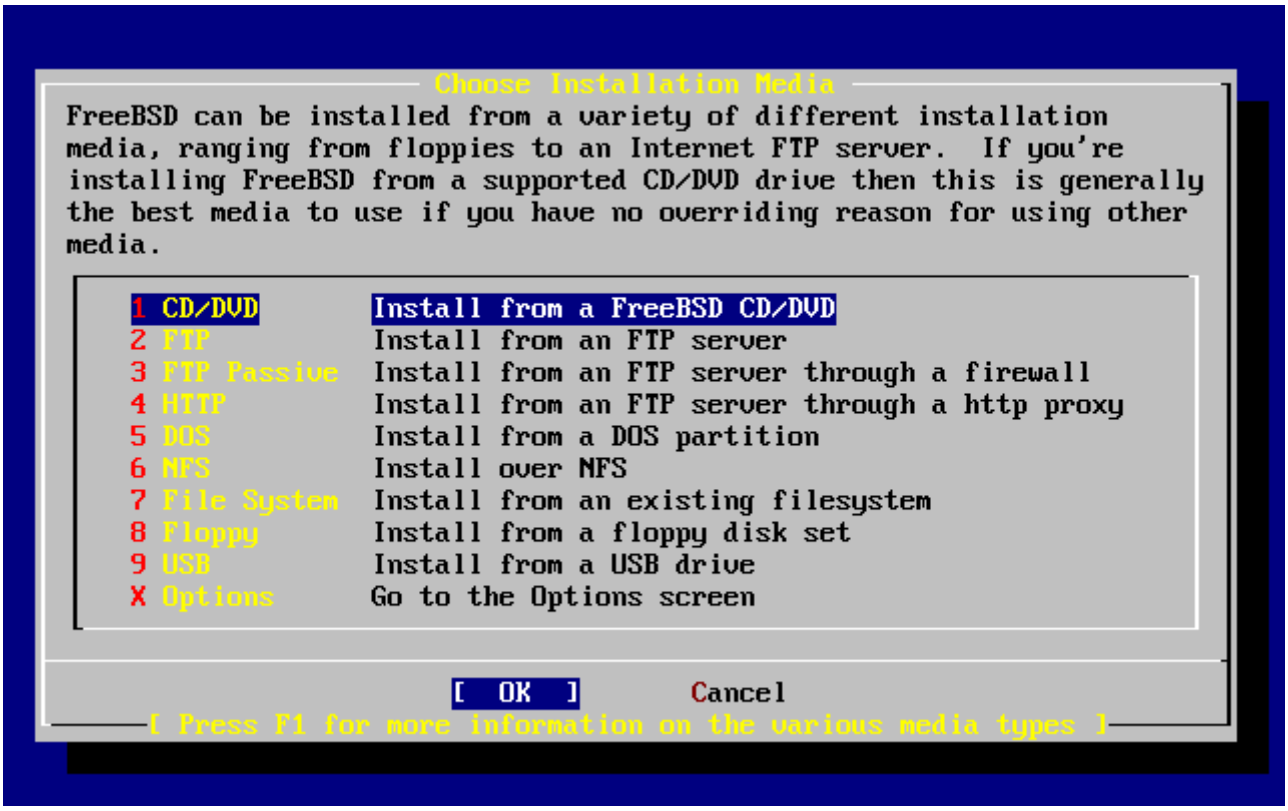


图 27. 选择安装介质

FTP安装模式

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

主动式FTP： 从FTP服务器安装

这个选项将会使所有的FTP传输使用 "Active"模式。
这将无法通过防火墙，但是可以使用在那些比较早期，
不支持被动模式的FTP站。如果您的连接在使用被动（默认值）
模式卡住了，请换主动模式看看！

被动模式FTP：通过防火墙从FTP服务器安装

此选项会让 sysinstall 使用 "Passive"模式来安装。这使得使用者可以穿过
不允许用非固定TCP PORTS连入的防火墙。

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

此选项会让 sysinstall 通过HTTP协议 （像浏览器一样）连到proxy服务器。
proxy服务器会解释送出的请求，然后通知FTP服务器。
因为通过HTTP协议，所以可以穿过防火墙。
要用这种方式，您必须指定proxy服务器的地址。

对于一个 FTP 代理服务器而言，
通常在使用者登入名称中加入您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，假设您要从
ftp.FreeBSD.org 安装，通过 FTP 代理服务器 foo.example.com， 使用1234端口。

在这种情况下，您可以到 options 菜单，将 FTP username 设为 ftp@ftp.FreeBSD.org，密码设为您的电子邮件地址。 安装介质部分，指定FTP (或是被动式 FTP，如果代理服务器支持的话) 以及URL为 ftp://foo.example.com:1234/pub/FreeBSD。

因为ftp.FreeBSD.org的 /pub/FreeBSD 目录会被抓取到 foo.example.com之下，您就可以从 这台 机器 (会从 ftp.FreeBSD.org 抓取文件) 安装。

2.9. 安装确认

到此为止，可以开始进行安装了， 这也是您避免更动到您的硬盘的最后机会。

```
                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
```

选择 [yes] 然后按下 Enter 确认安装

安装所需的时间会根据您所选择的软件、 安装介质以及您电脑的速度而有所不同。
在安装的过程中会有一些信息来显示目前的进度。

当您看到下面的信息表示已经安装完成了：

```
                      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。

选择 [no] 然后按 Enter 会取消安装，不会对您的系统造成更动。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东西也没有安装，按下 Enter
后会离开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， 就可以进行进一步的配置了。 引导新安装的 FreeBSD 系统之后， 使用 sysinstall 并选择 Configure。

2.10.1. 配置网卡

如果您之前配置用 PPP 通过 FTP 安装，那么这个画面将不会出现；
正像所说的那样，您可以稍后再做配置。

如果想更多的了解网卡或将FreeBSD配置为网关或路由器，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
```

如果要配置网卡，请选择 [yes] 然后按 Enter。 否则请选择 [no] 继续。

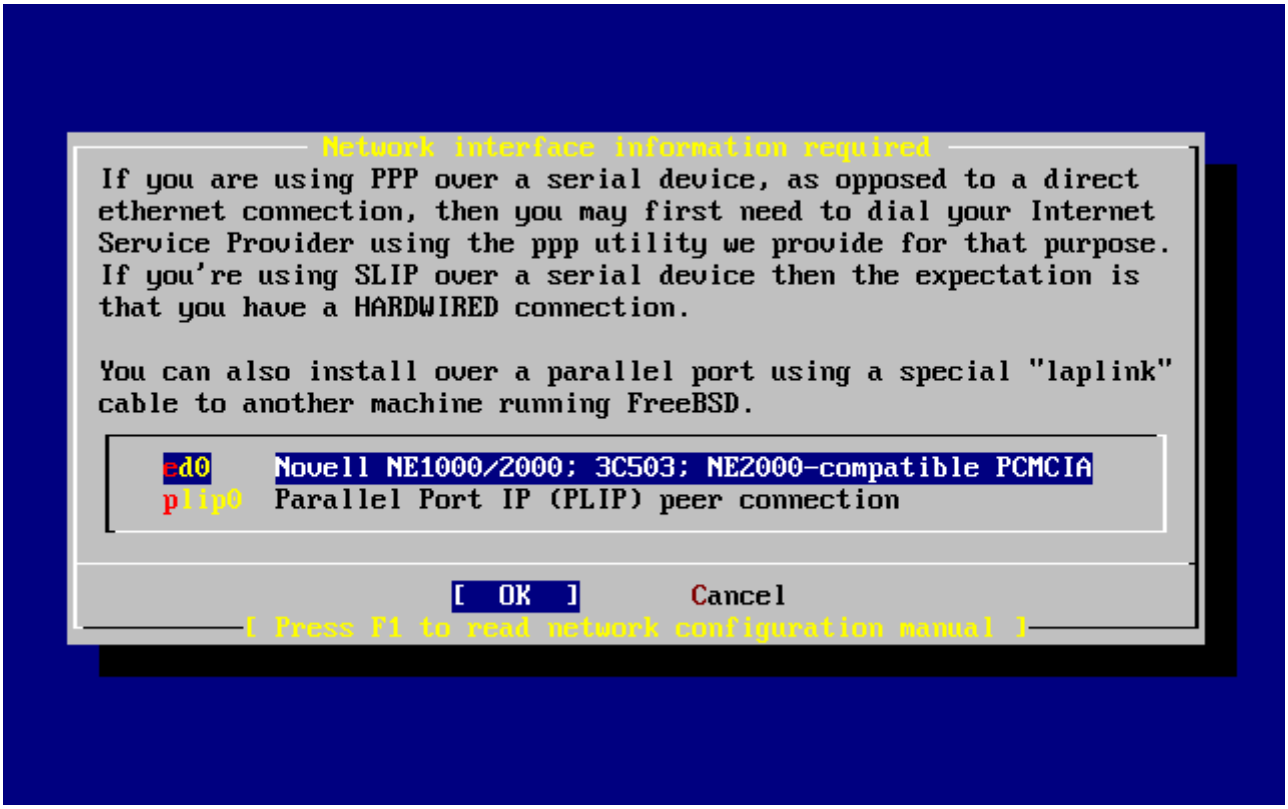


图 28. 选择网卡设备

用方向键选择您要配置的网卡接口，然后按 Enter。

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

Yes [ No ]
```

目录私人区域网络IP协议IPv4已经足够，所以选择 [no] 然后按 Enter。

如果想试试新的IP通信协议 IPv6 ， 使用 RA 服务，请选择 [yes] 然后按 Enter。 寻找 RA 服务器将会花费几秒的时间。

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

Yes [ No ]
```

如果您不需要 DHCP (Dynamic Host Configuration Protocol 动态主机配置协议) ，选择 [no] 然后按 Enter。

选择 [yes] 会执行dhclient， 如果成功，它会自动将网络配置信息填上。更多的信息请参考 网络自动配置 (DHCP) 。

下面的网络配置显示了怎样把以太网设备配置成区域网络网关的角色。

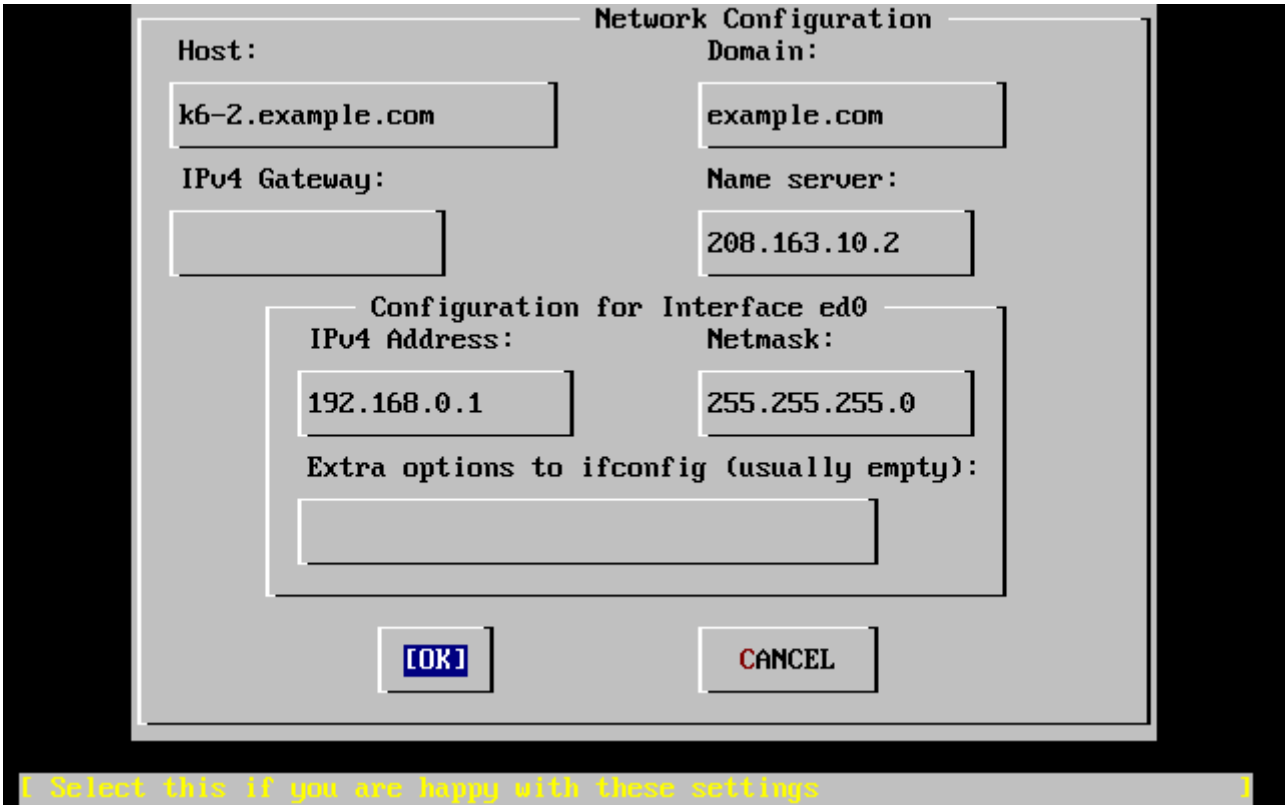


图 29. 配置 ed0接口

使用Tab键可以在各个栏目之间进行切换，请输入适当 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）
: 您机器所在的域名称，如本例的 example.com

IPv4 Gateway（IPv4网关）
: 输入将数据包传送到远端网络的机器IP地址。 只有当机器是网络上的一个节点时才要输入。
如果这台机器要作为您局域网的网关， 请将此处设为空白。IPv4网关，
也被称作默认网关或默认路由器。

域名服务器
: 本地网络中的域名服务器的IP地址。 本例中假设机器所在的网络中没有域名服务器，
所以填入的是ISP提供的域名服务器地址 （208.163.10.2。）

IPv4 地址
: 本机所使用的IP地址。本例为 192.168.0.1。

子网掩码
: 在这个局域网中所使用的地址块是 192.168.0.0 - 192.168.0.255， 对应的子网掩码是 255.255.255.0。

ifconfig 额外参数设定
: 任何ifconfig命令跟网卡接口有关的参数。 本范例中没有。

使用 Tab 键选择 [OK]然后按 Enter键。

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

```
                [ Yes ]  No
```

选择 [yes] 然后按 Enter 将会将机器的网卡转为启用状态。 机器下次启动的时候即可使用。

2.10.2. 配置网关

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

                [ Yes ]  No
```

如果这台机器要作为本地网络和其它机器之间传送数据包的网关，请选择 [yes] 然后按 Enter。
如果这台机器只是网络上的普通节点，请选择 [no] 并按 Enter 继续。

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 ]
```

如果选择 [no]， 许多网络服务，如 telnetd 将不会启用。 这样， 远端用户将无法 telnet 进入这台机器。本机上的用户还是可以 telnet到远端机器的。

这些服务可以在安装完成后修改/etc/inetd.conf 配置文件来启用它们。请参阅 总览 以获得更多的信息。

如果您想现在就配置这些网络服务，请选择 [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
```

选择 [yes] 继续。

```
                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

选择 [yes] 将允许您添加网络服务 (或将相应网络服务每行开头的 # 除掉即可。)

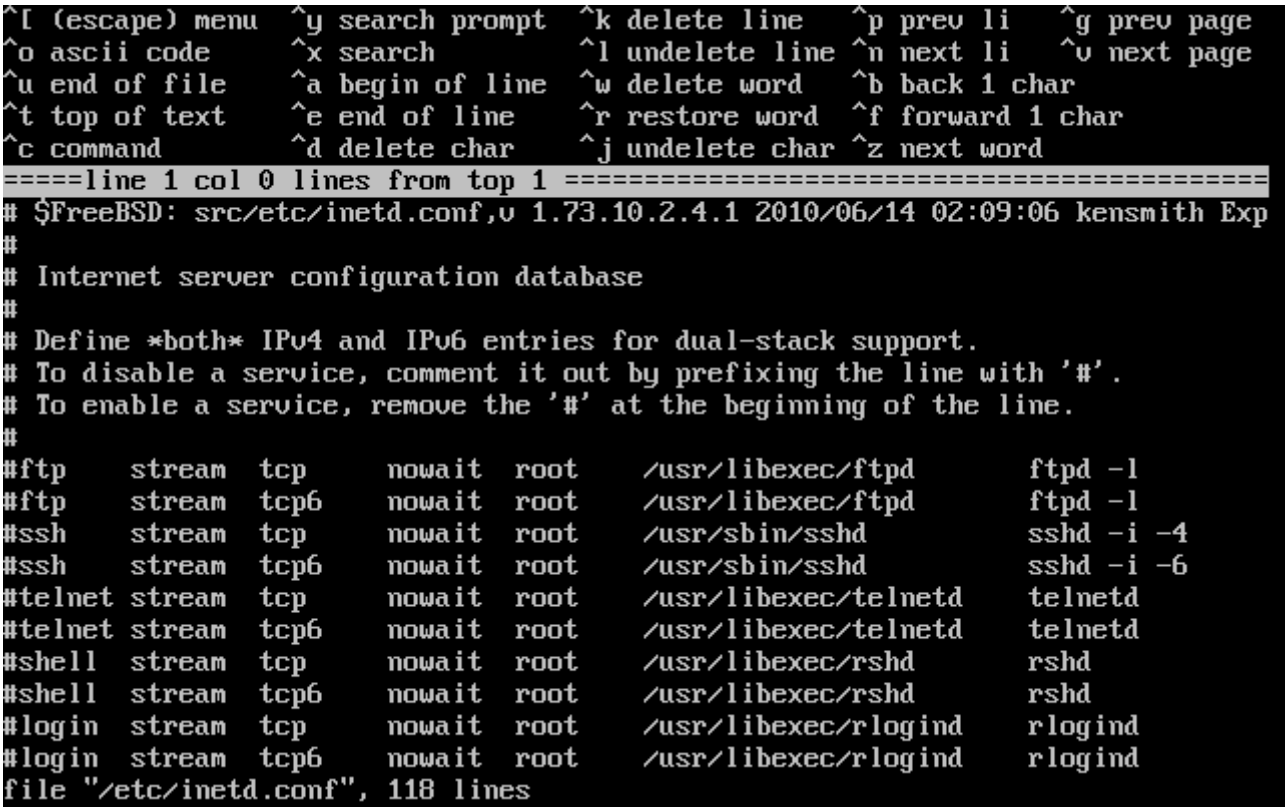


图 30. 编辑 inetd.conf配置文件

在加入您想启用的服务后，按下 Esc 键会出现一个 对话框可以让您离开以及保存修改。

2.10.4. 启用 SSH 登录

User Confirmation Requested
Would you like to enable SSH login?
Yes [No]

选择 [yes] 便会启用 sshd(8)， 也就是 OpenSSH 服务程序。 它能够让您以安全的方式从远程访问机器。 如欲了解关于 OpenSSH 的进一步详情， 请参见 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访问

选择默认的 [no] 并按下 Enter 键将仍然可以让在这台机器上有账号的用户访问 FTP。

2.10.5.2. 允许匿名 FTP访问

如果您选择允许匿名 FTP 存取， 那么网络中任何人都可以使用FTP来访问您的机器。
在启用匿名访问之前应该考虑网络的安全问题。 如果要知道更多有关网络安全的信息， 请参阅 安全。

要启用FTP匿名访问，用方向键选择 [yes] 并按 Enter 键。 系统会给出进一步的确认信息：

```
                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
```

这些信息会告诉您 FTP 服务还需要在 /etc/inetd.conf 中启用。 假如您希望允许匿名 FTP 连接， 请参见 配置网络服务。 选择 [yes] 并按 Enter 继续； 系统将给出下列信息：

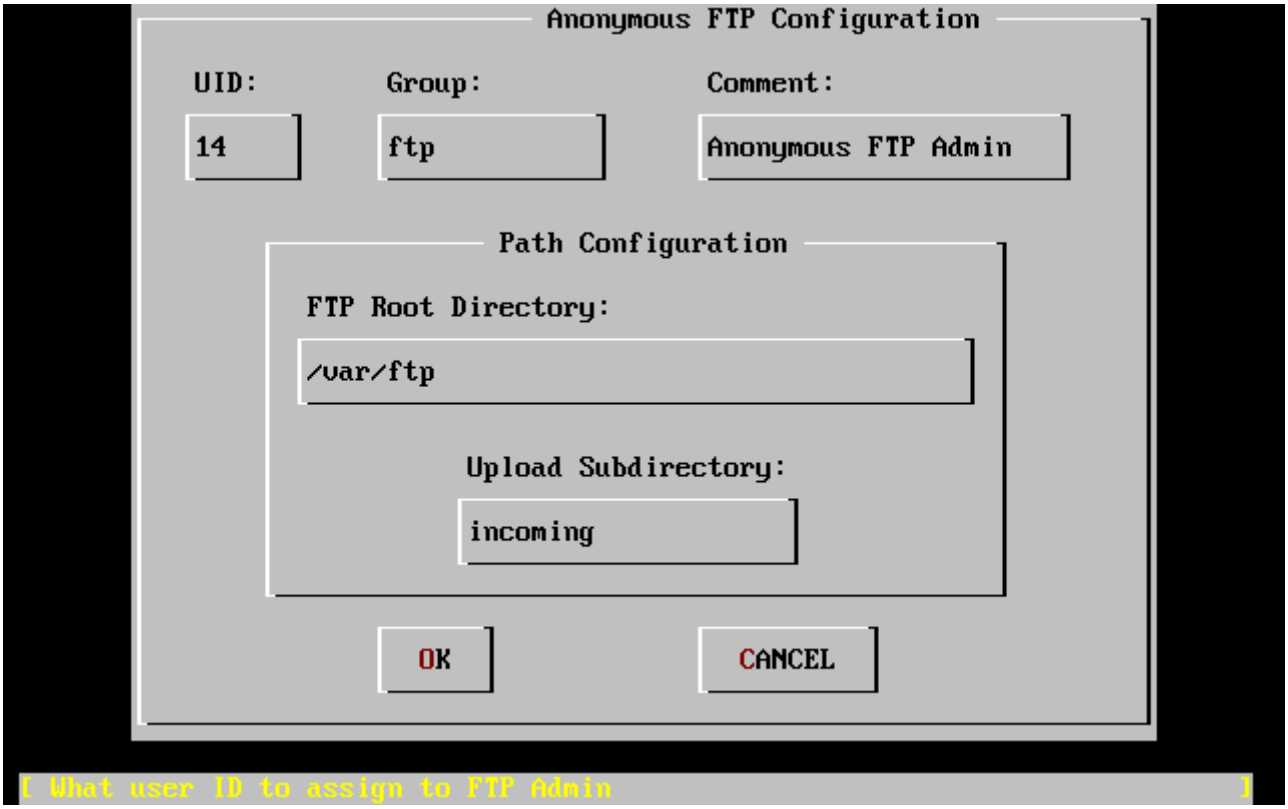


图 31. 默认的匿名 FTP 配置

使用 Tab 在不同的信息字段之间切换， 并填写必要的信息：

UID
: 用于分配给匿名 FTP 用户的用户 ID。 所有上传的文件的属主都将是这个 ID。

Group
: 匿名 FTP 用户所在的组。

Comment
: 用于在 /etc/passwd 中描述该用户的说明性信息。

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 用户上传的文件的存放位置。

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

当您对一切配置都满意后，请按 Enter 键继续。

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

                  [ Yes ]   No
```

如果您选择 [yes] 并按下 Enter 键， 系统会自动打开文本编辑器让您编辑FTP的欢迎信息。

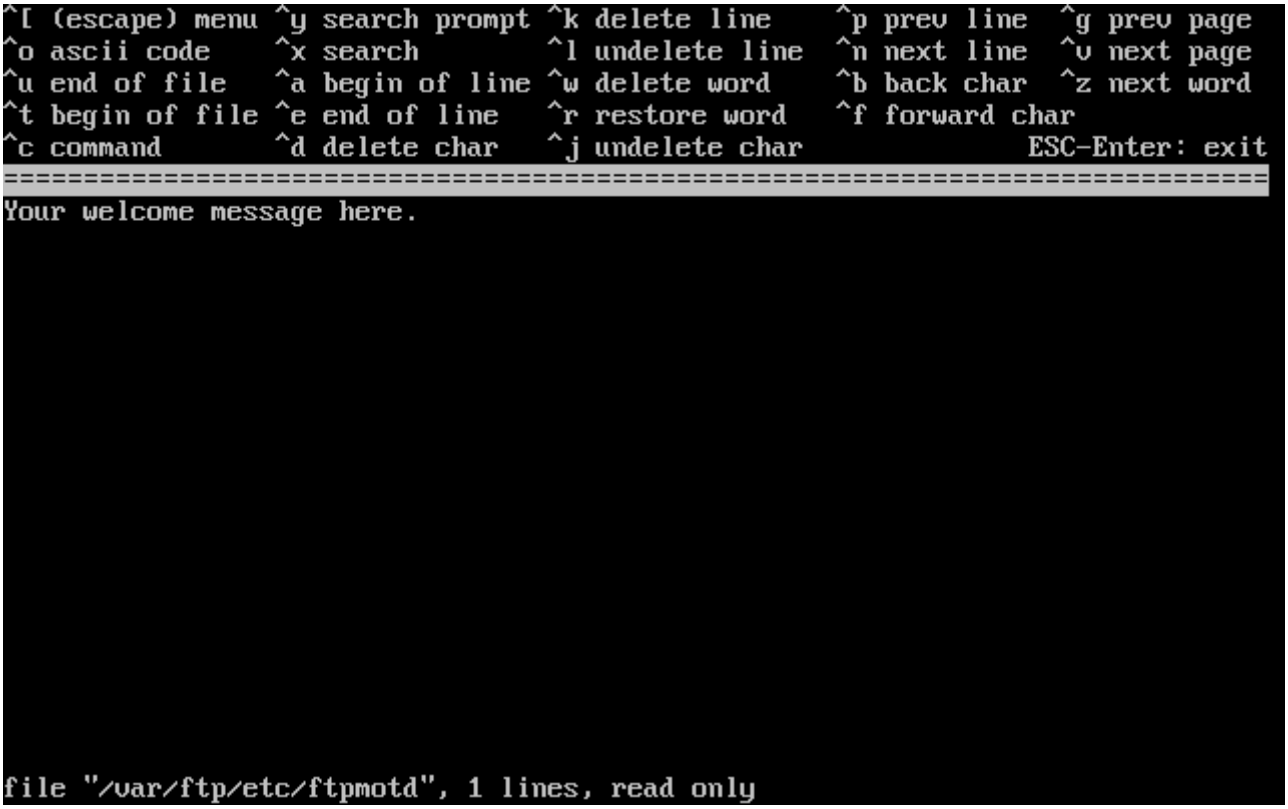


图 32. 编辑FTP欢迎信息

此文本编辑器叫做 ee。 按照指示修改信息文本或是稍后再用您喜爱的文本编辑器来修改。
请记住画面下方显示的文件位置。

按 Esc 将弹出一个默认为 a) leave editor的对话框。按 Enter 退出并继续。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) 可以让您可以在网络上共享您的文件。
一台机器可以配置成NFS服务器、客户端或两者并存。请参考 网络文件系统（NFS) 以获得更多的信息。

2.10.6.1. NFS 服务器

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

Yes [No]

如果您不想安装网络文件系统，请选择 [no] 然后按 Enter 键。

如果您选择 [yes] 将会出现一个对话框提醒您必须先建立一个 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 键继续。系统会启动文本编辑器让您编辑 exports 文件。

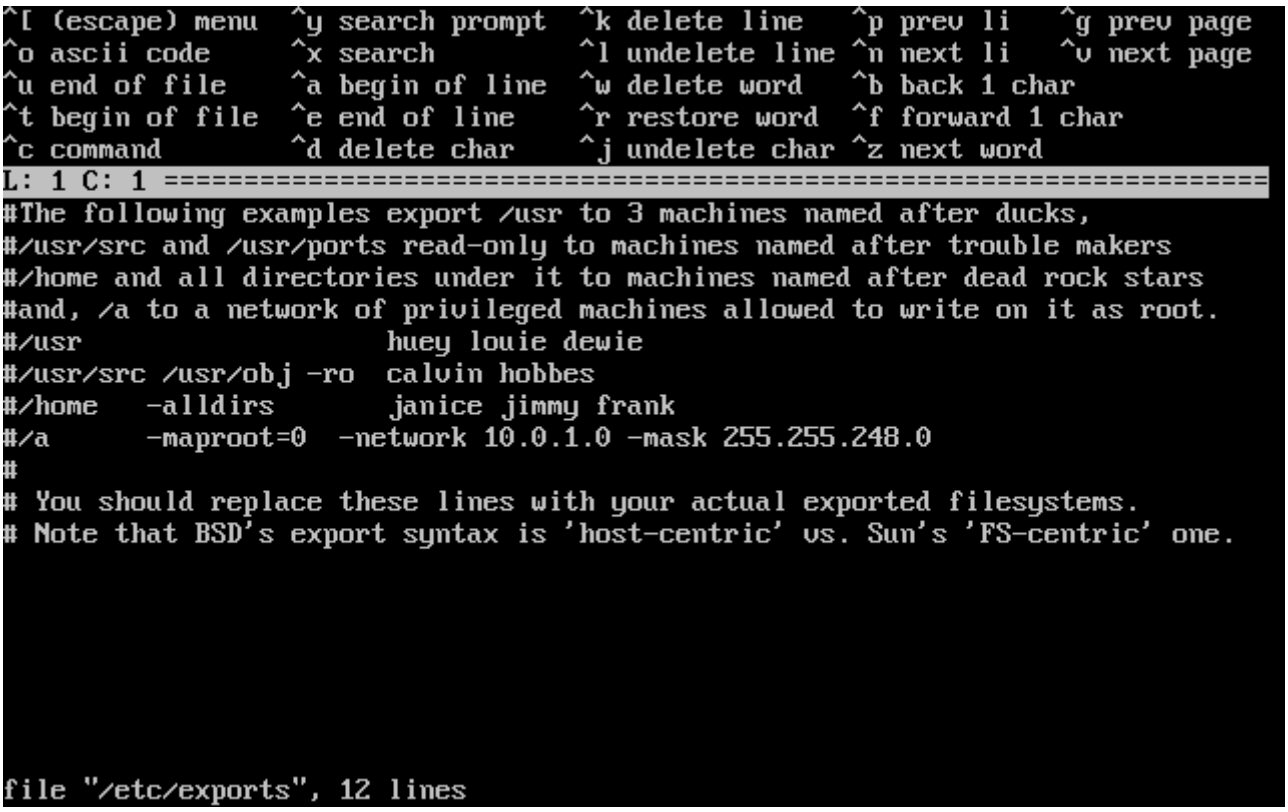


图 33. 编辑 exports文件

按照指示加入真实输出的文件目录或是稍后用您喜爱的编辑器自行编辑。
请记下画面下方显示的文件名称及位置。

按下 Esc 键会出现一具对话框，默认选项是 a) leave editor。按下 Enter 离开并继续。

2.10.6.2. NFS 客户端

NFS 客户端允许您的机器访问NFS服务器。

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

                    Yes  [ No ]
```

按照您的需要，选择 [yes] 或 [no] 然后按 Enter。

2.10.7. 配置系统终端

系统提供了几个选项可以让您配置终端的表现方式。

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

                    [ Yes ]  No
```

要查阅及配置这些选项，请选择 [yes] 并按 Enter。

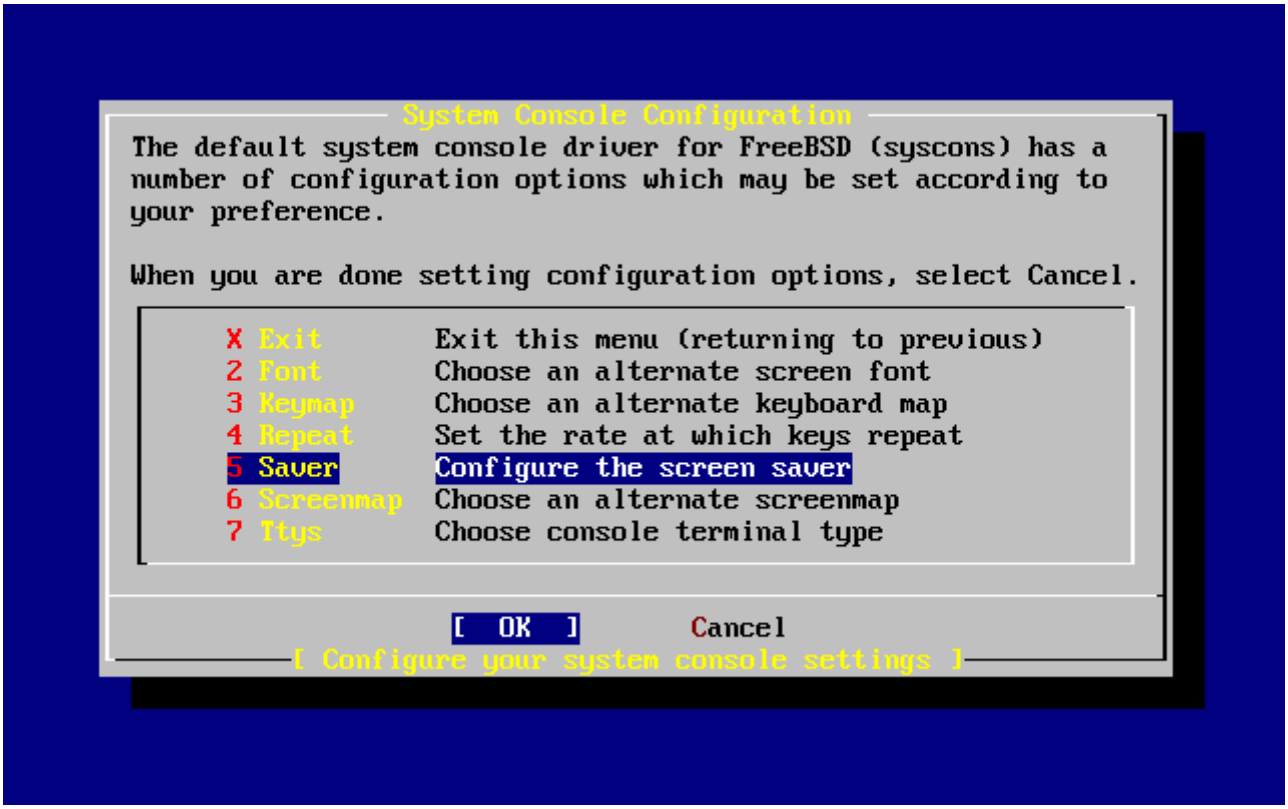


图 34. 系统终端配置选项

最常用的选项就是屏幕保护程序了。使用方向键将光标移动到 Saver 然后按 Enter。

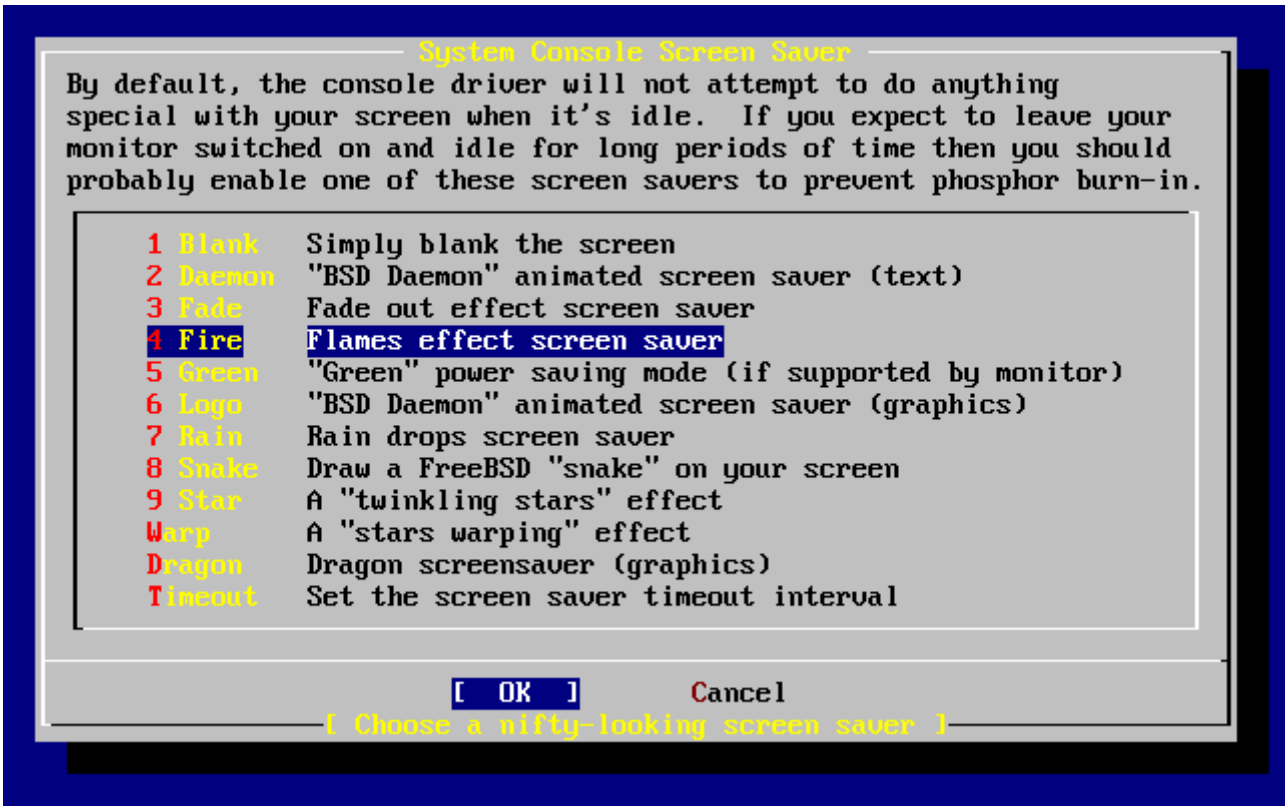


图 35. 屏幕保护程序选项

选择您想使用的屏幕保护程序，然后按 Enter。 之后回到系统终端配置画面。

默认开启屏幕保护程序的时间是300秒。如果要更改此时间，请再次选择 Saver 。然后选择 Timeout 并按 Enter 键。系统会弹出一个对话框如下：

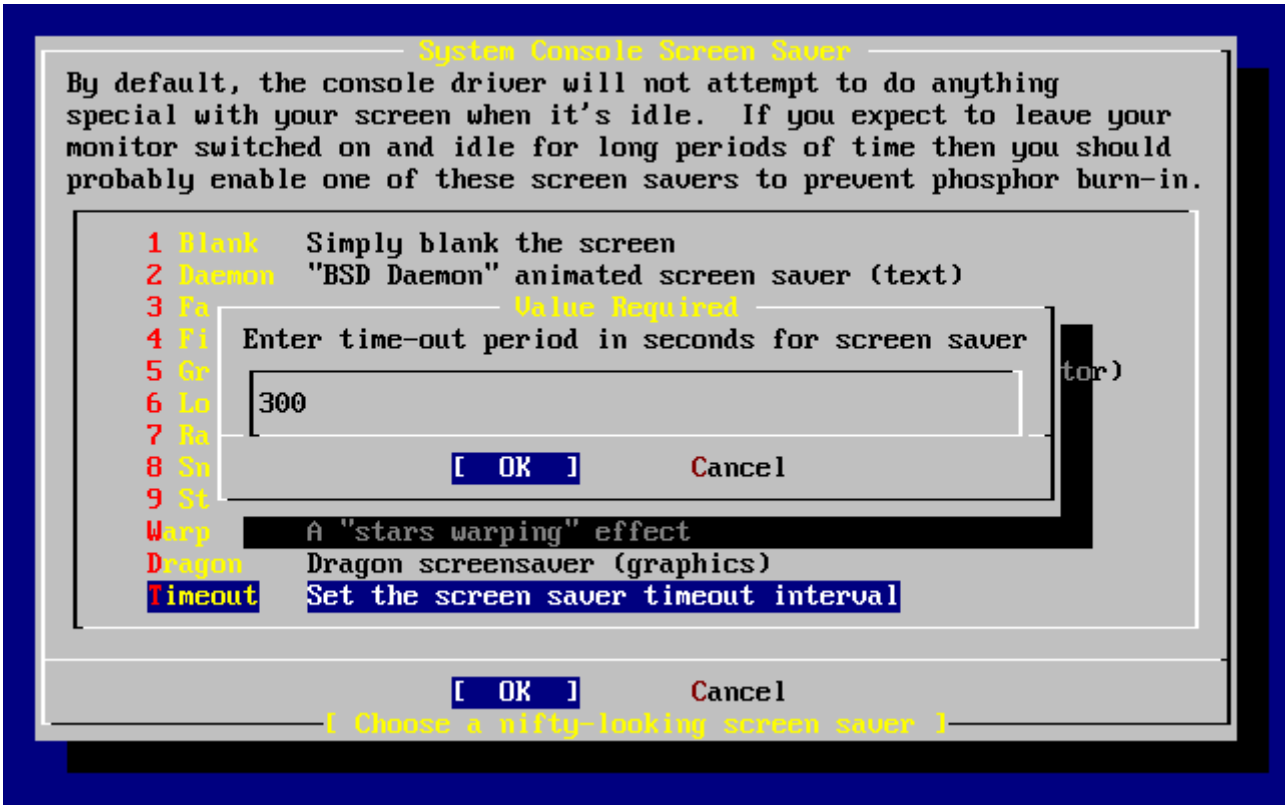


图 36. 屏幕保护时间设置

您可以直接改变这个值，然后选 [OK]并按 Enter 键回到系统终端配置画面。

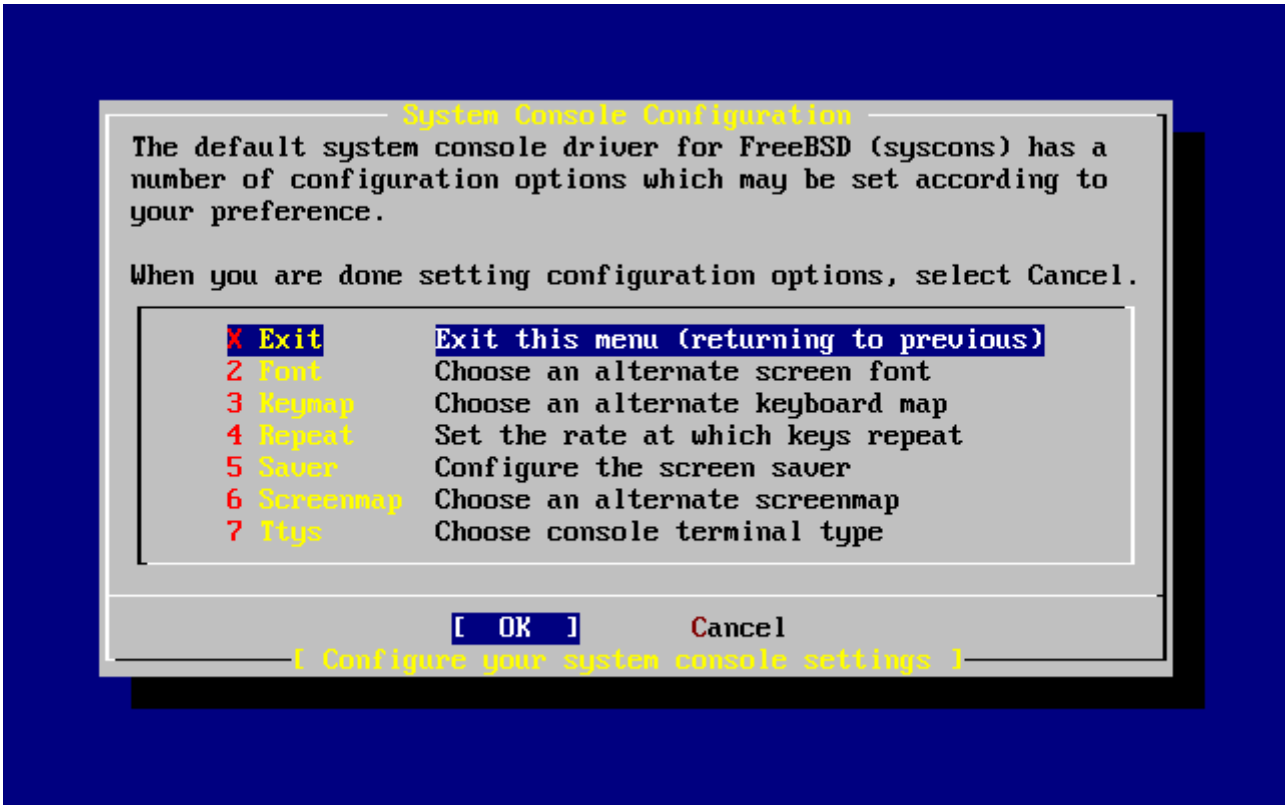


图 37. 退出系统终端配置

选择 Exit 然后按下 Enter 键会回到安装后的配置画面。

2.10.8. 配置时区

配置您机器的时区可以让系统自动校正任何区域时间的变更，

并且在执行一些跟时区相关的程序时不会出错。

例子中假设此台机器位于美国东部的时区。 请参考您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
```

选择 [yes] 并按下 `Enter` 键以配置时区。

```
                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 ]
```

这里按照您机器时间的配置，选择 [yes] 或 [no] 然后按 `Enter`。

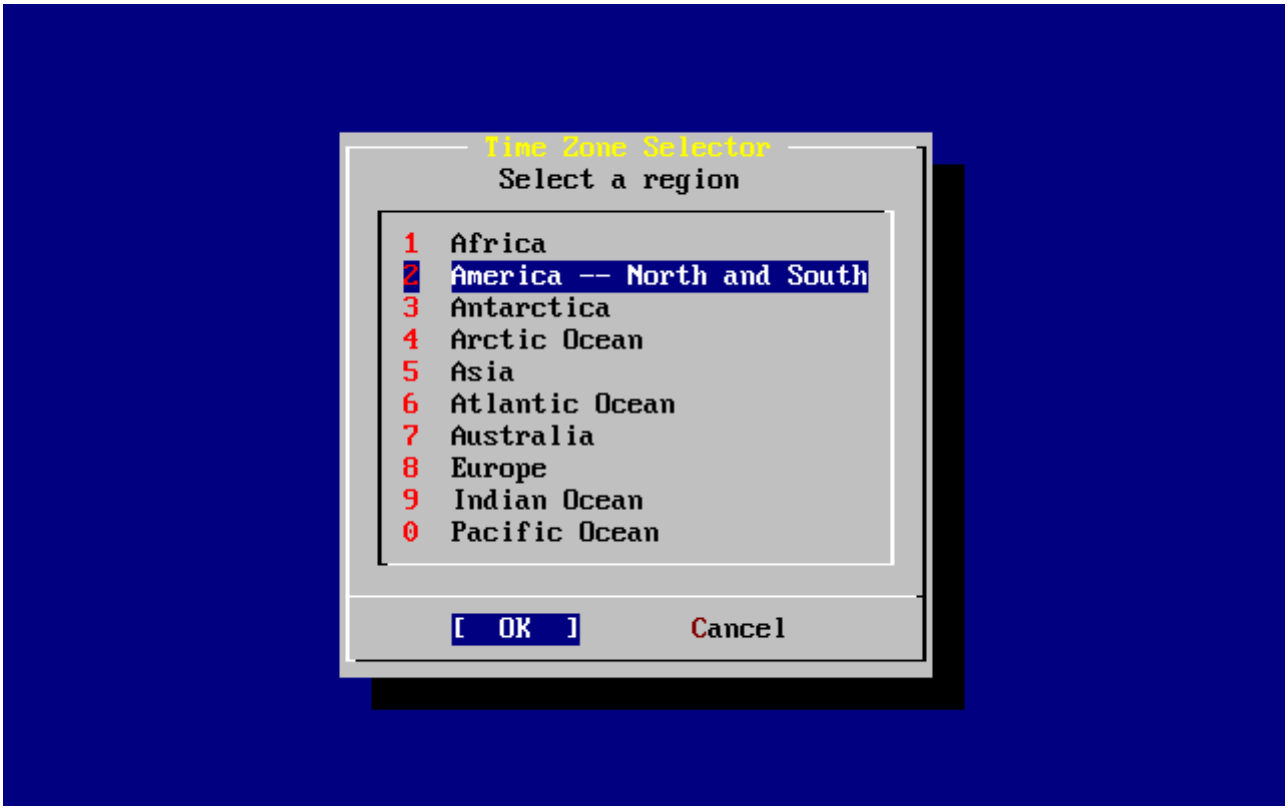


图 38. 选择您所处的地理区域

请选择适当的区域然后按 `Enter`。

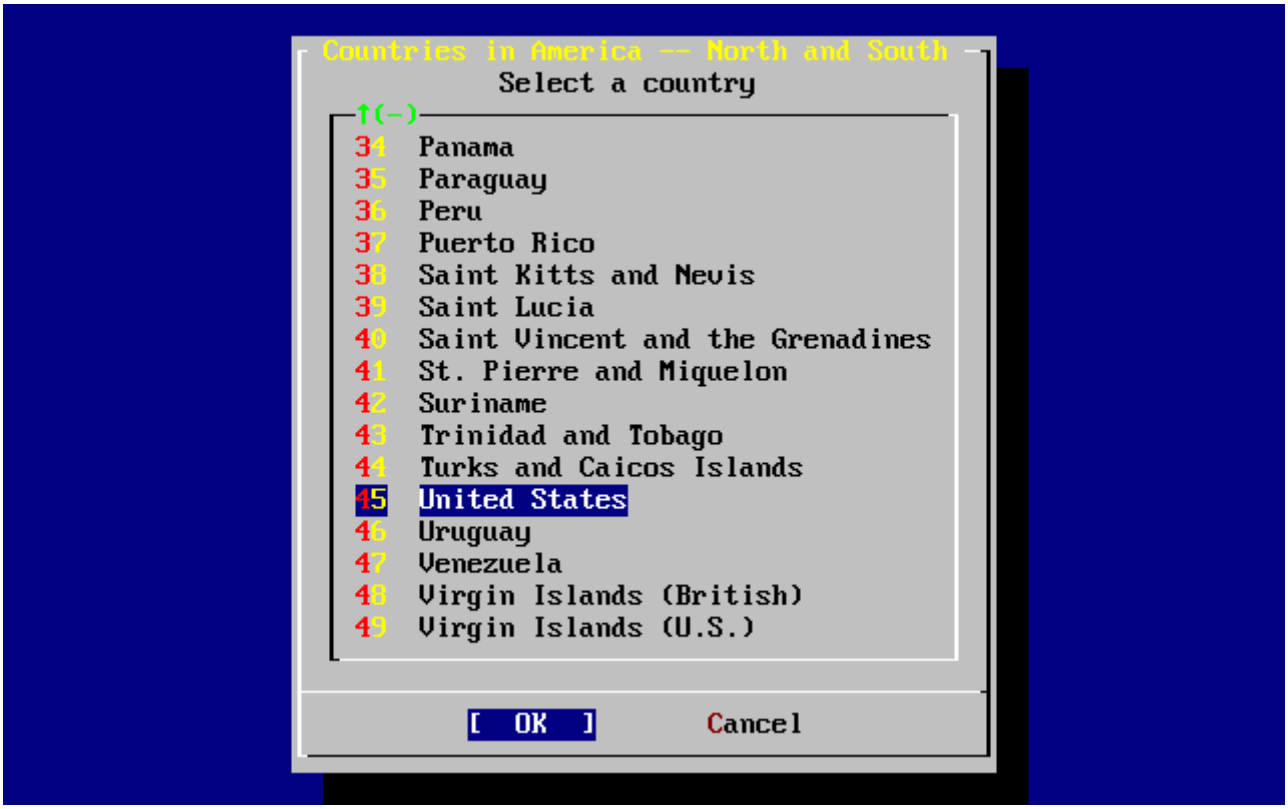


图 39. 选择您所在的国家

选择您所在的国家然后按 Enter。

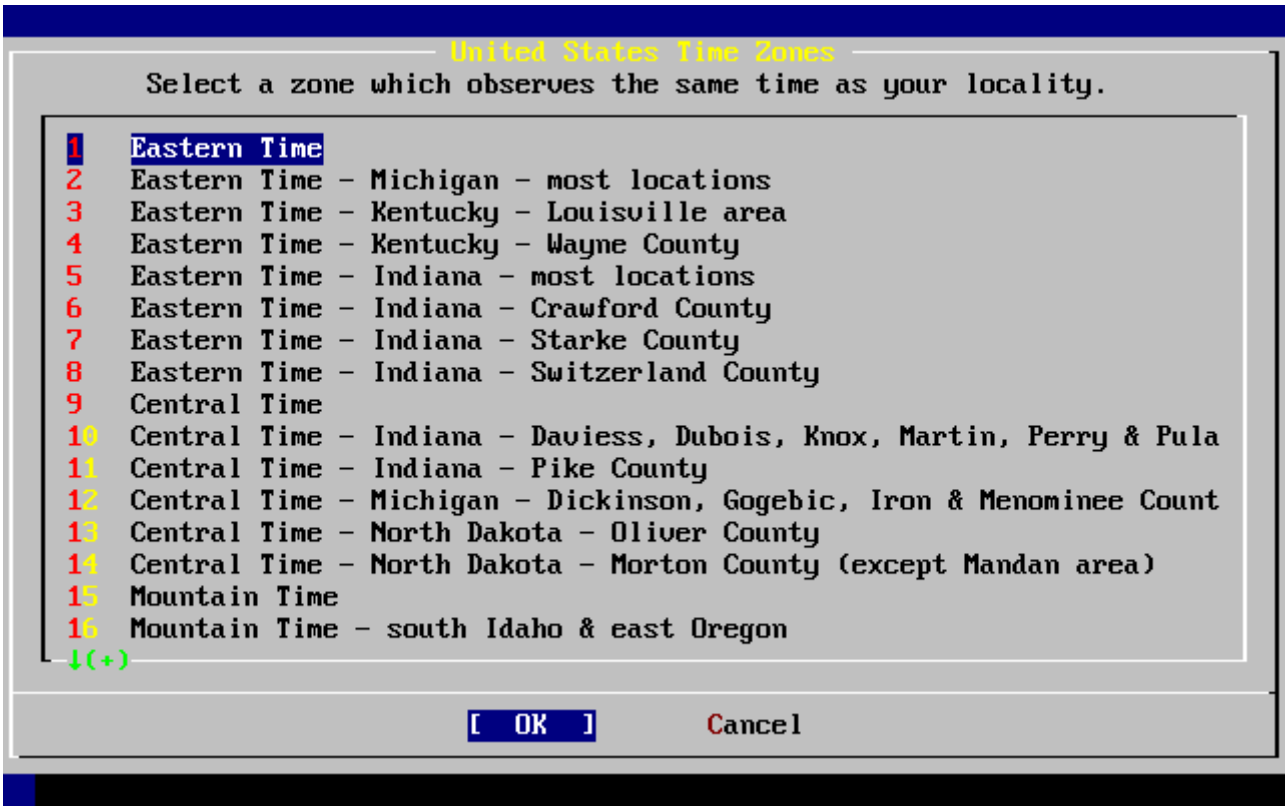


图 40. 选择您所在的时区

选择您所在的时区然后按 Enter。

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

```
        [ Yes ]  No
```

检查一下时区的缩写是否正确，如果没错，请按 Enter 返回系统安装后的配置画面。

2.10.9. Linux 兼容性

这节内容只适用于 FreeBSD 7.X 安装过程， 如果您安装的是 FreeBSD 8.X 或更高版本，系统不会给出这个提示。

```
          User Confirmation Requested
     Would you like to enable Linux binary compatibility?

          [ Yes ]  No
```

选择 [yes] 并按下 Enter 键， 将允许您在FreeBSD中执行Linux的软件。安装程序会安装一些为了跟 Linux 兼容的软件包。

如果您是通过FTP安装，那么您必须连到网络上。
有时候FTP站并不会包含所有的安装软件包（例如Linux兼容软件包）；
不过，稍后您还可以再安装这个项目。

2.10.10. 配置鼠标

此选项可以让您在终端上使用三键鼠标剪贴文字。 如果您用的鼠标是两个按钮，请参考手册 moused(8)；以取得有关模拟三键鼠标的信息。范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，请选择 [yes]， 如果是 USB 鼠标， 则应选择 [no] 并按 Enter。

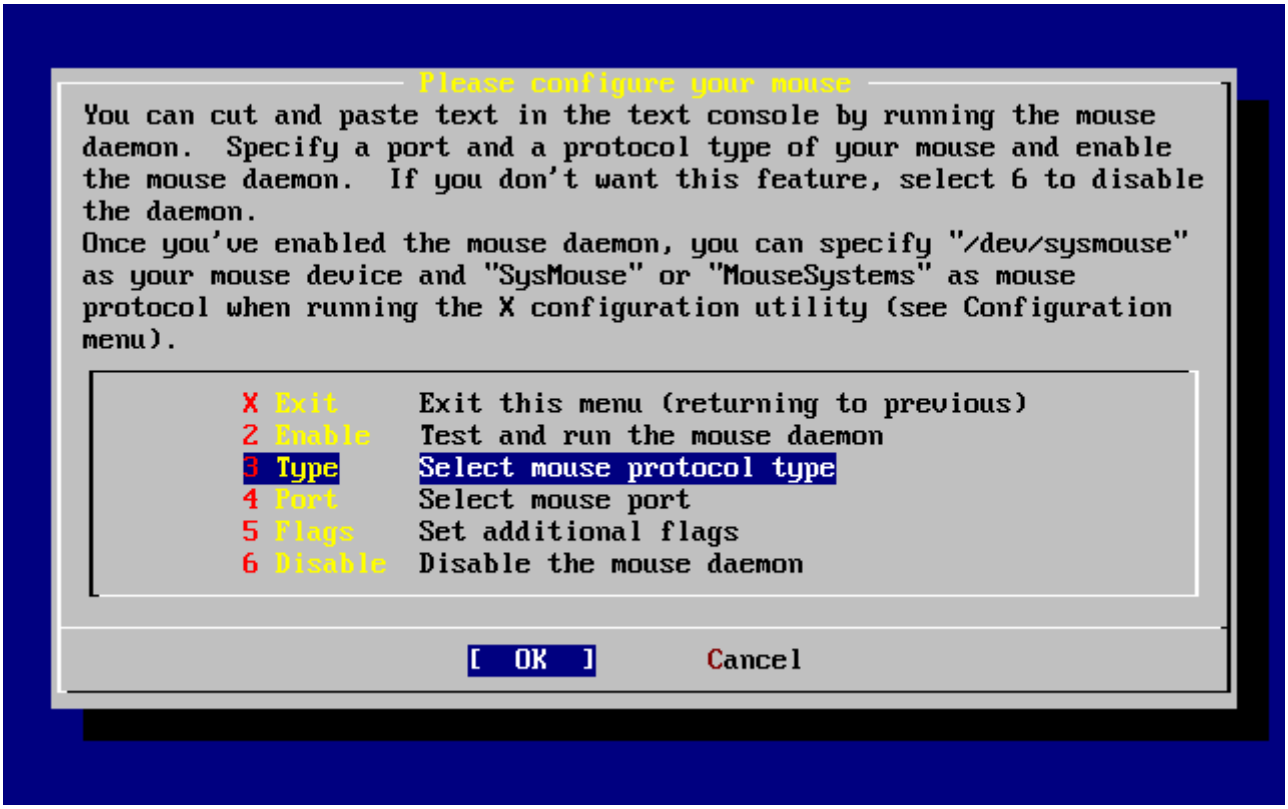


图 41. 选择鼠标类型

使用方向键选择 Type 然后按 Enter。

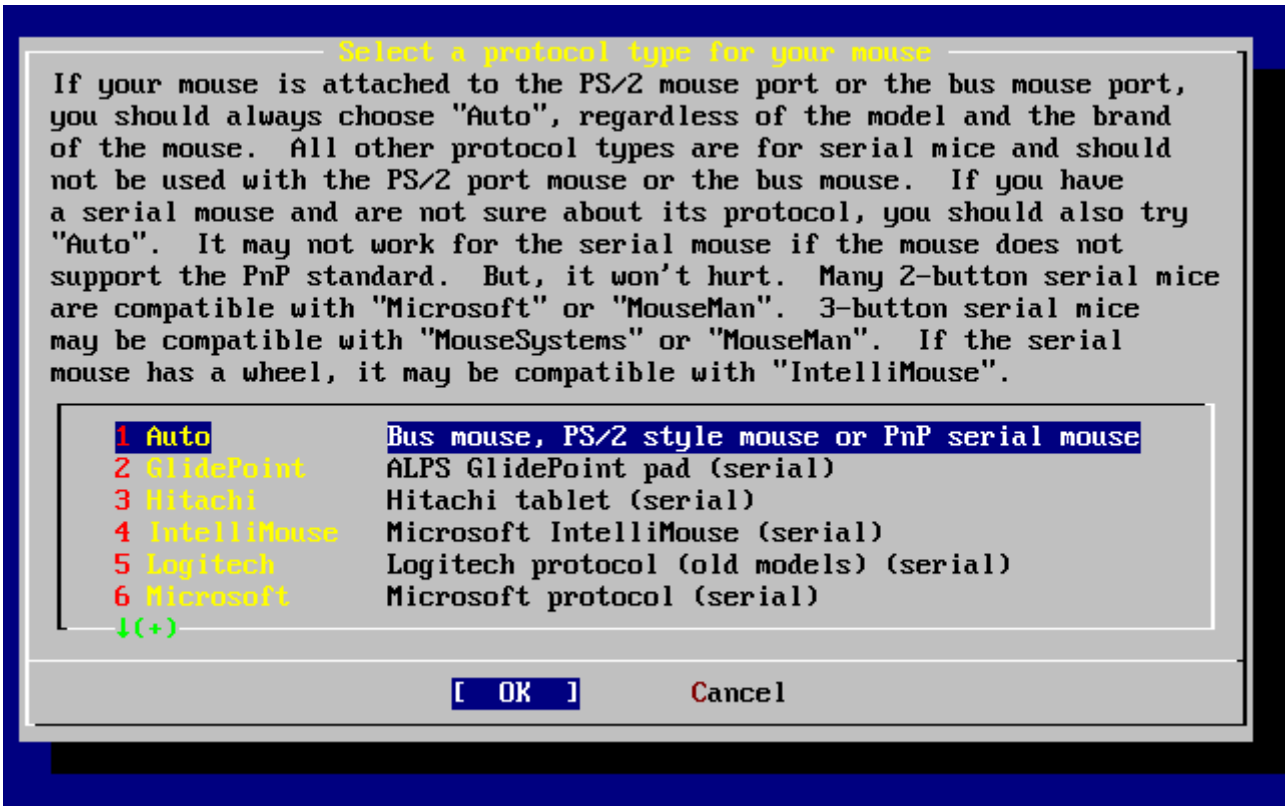


图 42. 设置鼠标协议

在这个例子中使用的类型是ps/2鼠标，所以可以使用默认的 Auto（自动）。
您可以用方向键选择合适的项目，确定选择了 [OK] 后按 Enter 键离开此画面。

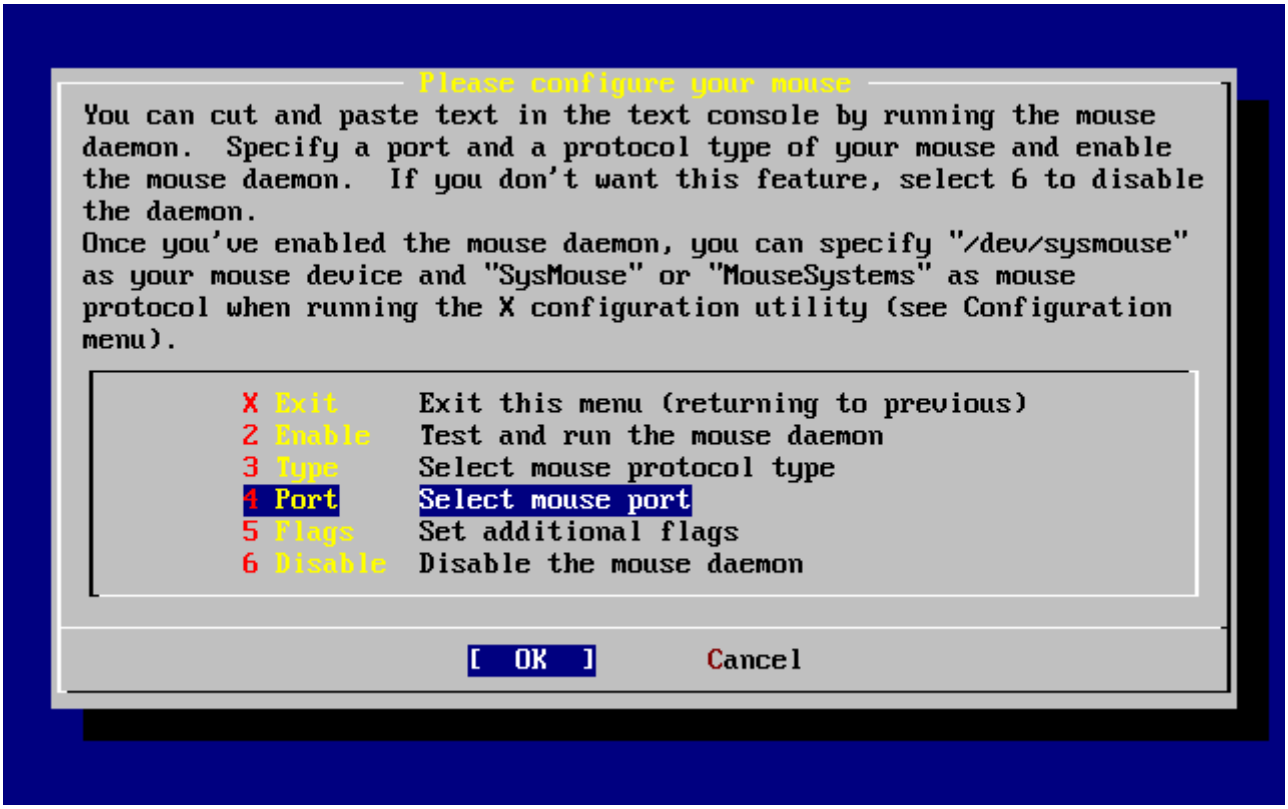


图 43. 配置鼠标端口

选择 Port 然后按 Enter。

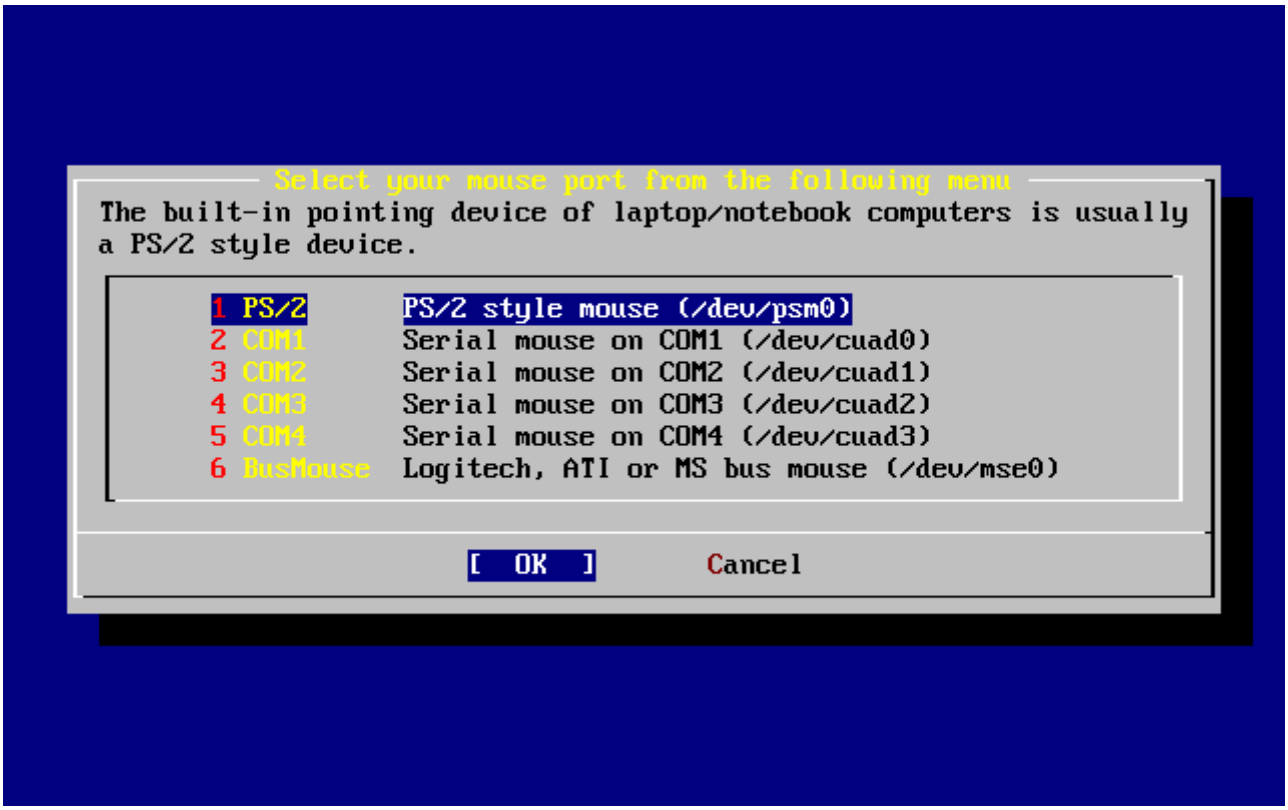


图 44. 配置鼠标端口

假设这台机器用的是ps/2鼠标，您可以采用默认的 PS/2 选项。请选择适当的项目然后按 Enter。

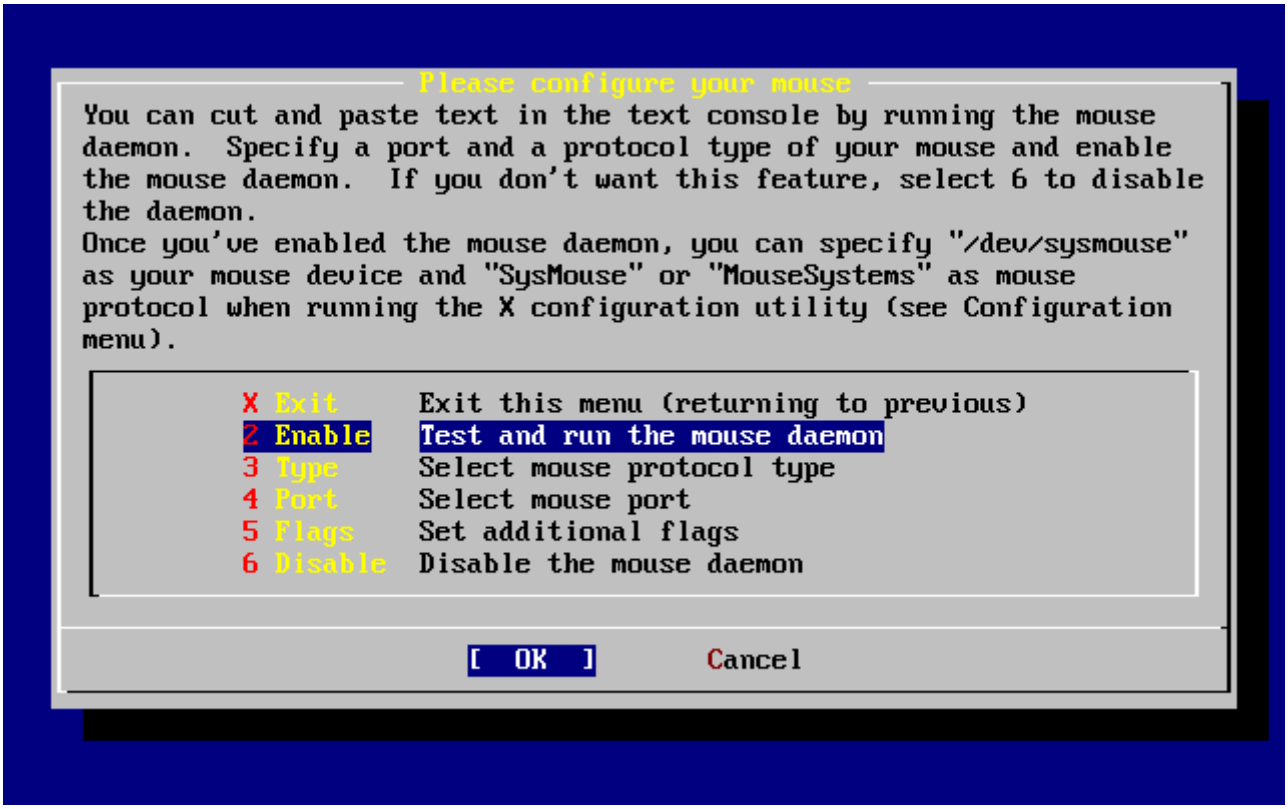


图 45. 启动鼠标服务进程

选择Enable然后按 Enter 来启动和测试鼠标。

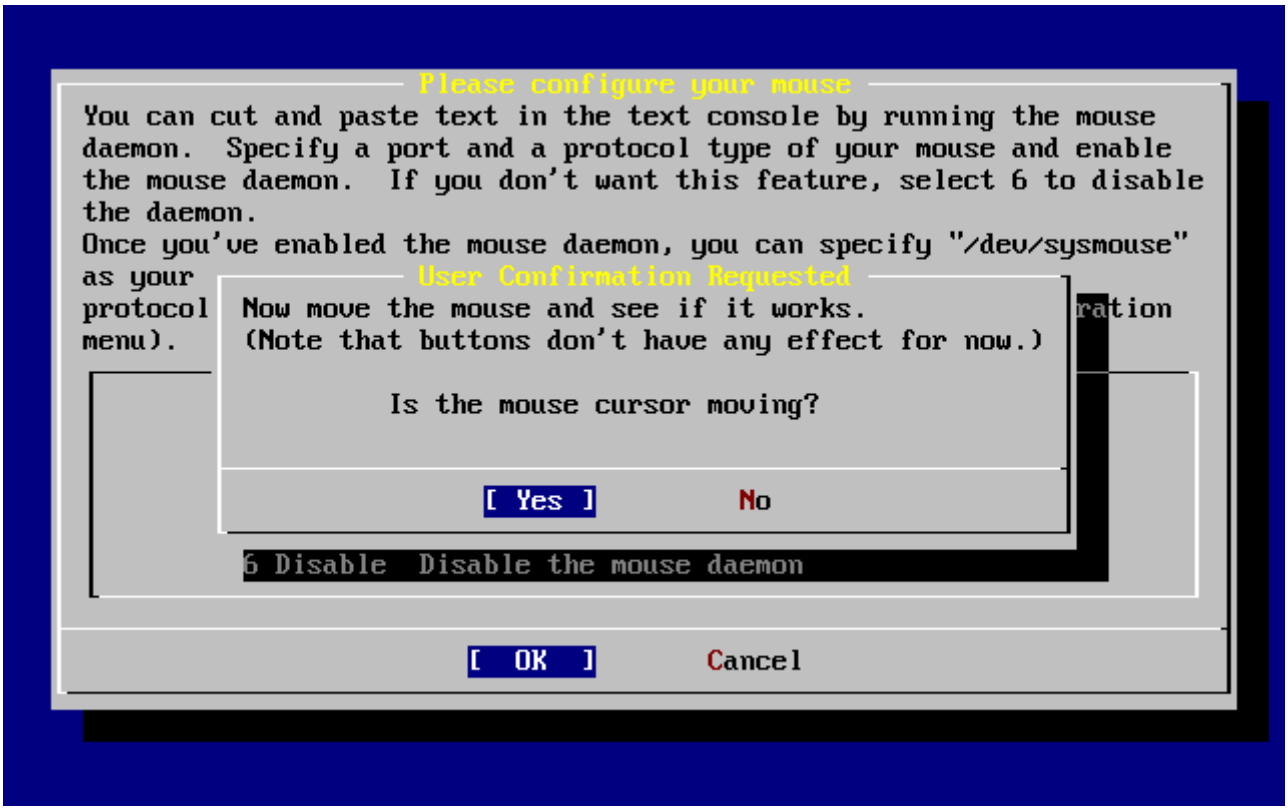


图 46. 测试鼠标功能

鼠标指针可以在屏幕上移动，指明鼠标服务已经正常启用。那么请选择 [yes] 按 Enter 键。否则鼠标没有配置成功 - 选择 [no] 并尝试不同的配置 选项。

选择 Exit 并按 Enter 退回到系统安装完成后的配置画面。

2.10.11. 安装预编译的软件包 (package)

Package 是事先编译好的二进制文件， 因此， 这是安装软件的一种便捷的方式。

在这里作为例子我们将给出安装一个 package 所需的过程。 如果需要， 还可以在这一阶段加入其他 package。 安装完成之后， sysinstall 依然可以用来安装其他 package。

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

                    [ Yes ]  No
```

选择 [yes] 并按 Enter 将进入 package 选择界面：

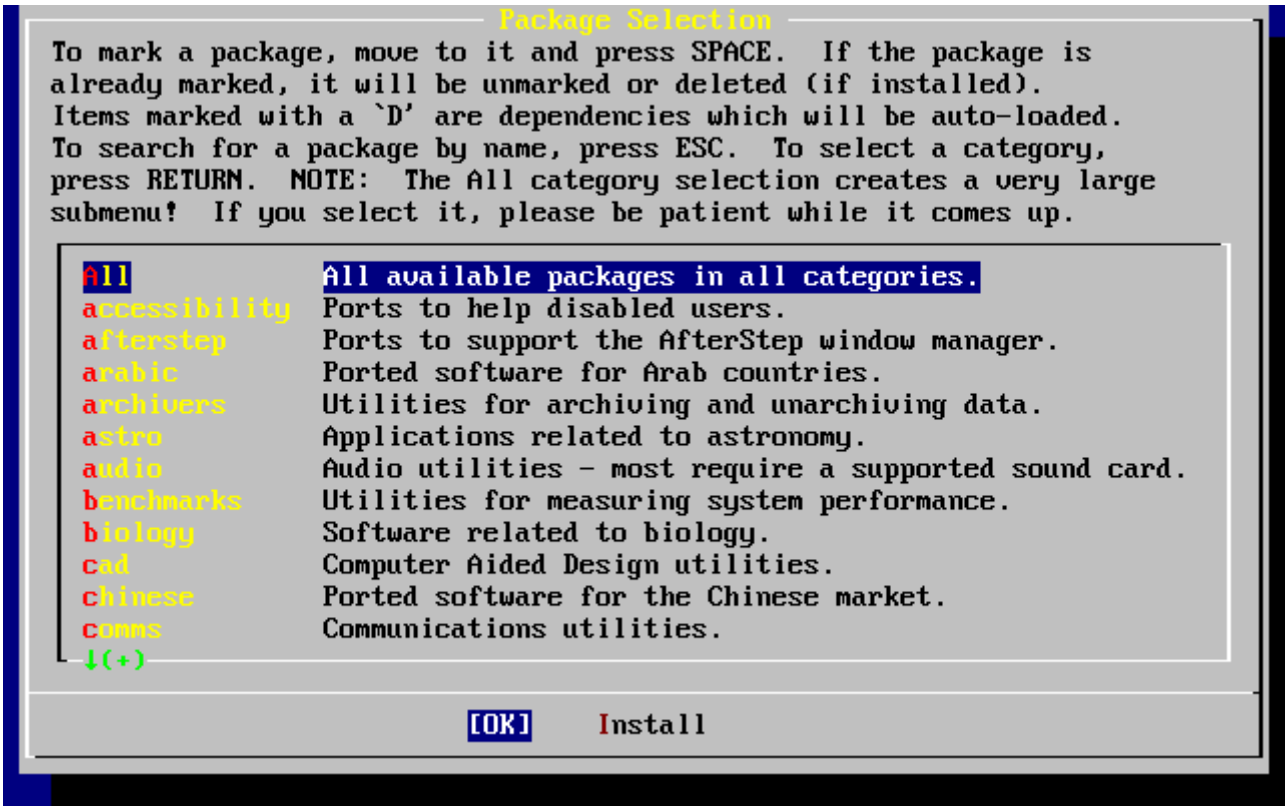


图 47. 选择 Package 类别

在任何时候， 只有当前安装介质上存在的 package 才可以安装。

如果选择了 All 或某个特定的分类， 则系统会列出全部可用的 package。 用光标键移动光棒选中需要的 package， 并按 Enter。

系统会显示可供选择的 package：

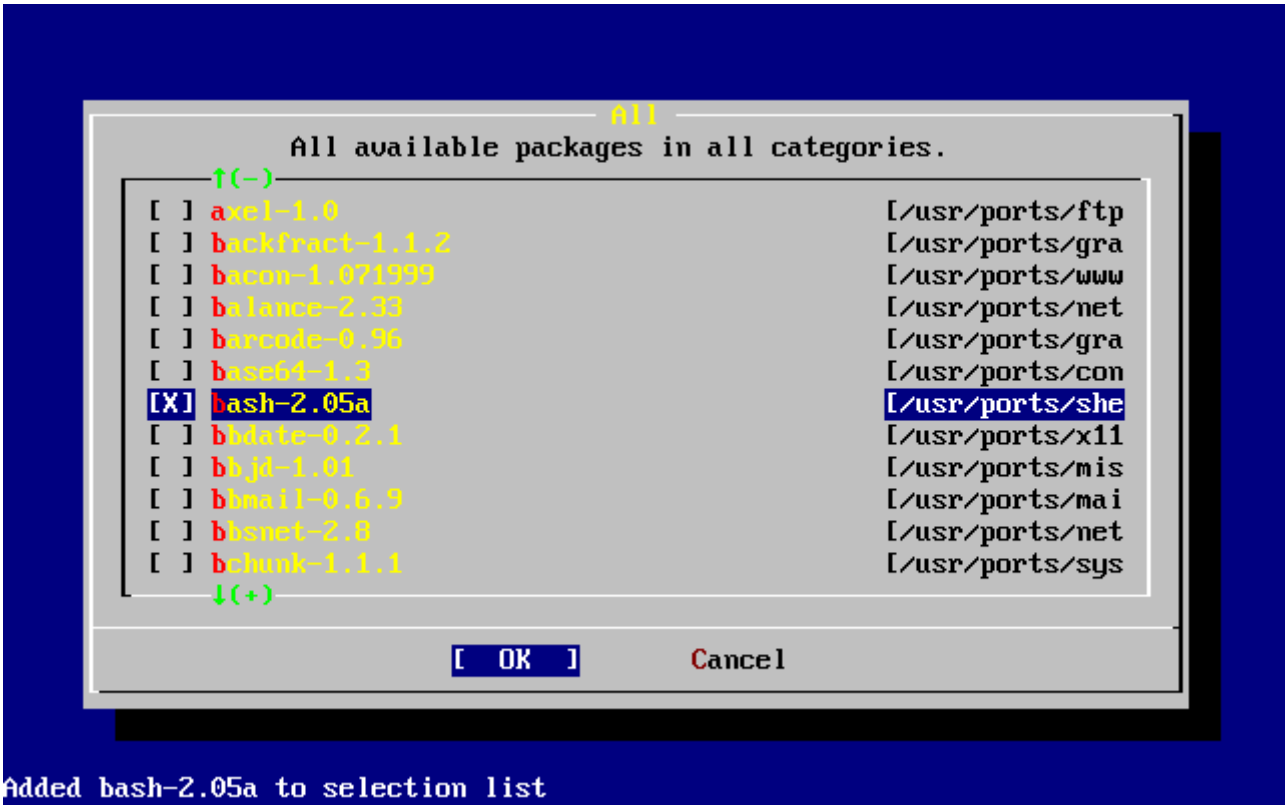


图 48. 选择 Package

如图所示， 我们选择了 bash shell。 您可以根据需要使用 Space 键来勾选选定的 package。 在屏幕左下角会给出 package 的简短说明。

反复按下 Tab 键， 可以在最后选中的 package、 [OK] 和 [Cancel] 之间来回切换。

当您把需要的 package 都标记为安装之后， 按一下 Tab 切换到 [OK]， 随后按下 Enter 就可以回到 package 选择菜单了。

左右方向键可以用于在 [OK] 和 [Cancel] 之间进行切换。 这种方法也可以用来选择 [OK]， 随后按下 Enter 也可以回到 package 选择菜单。

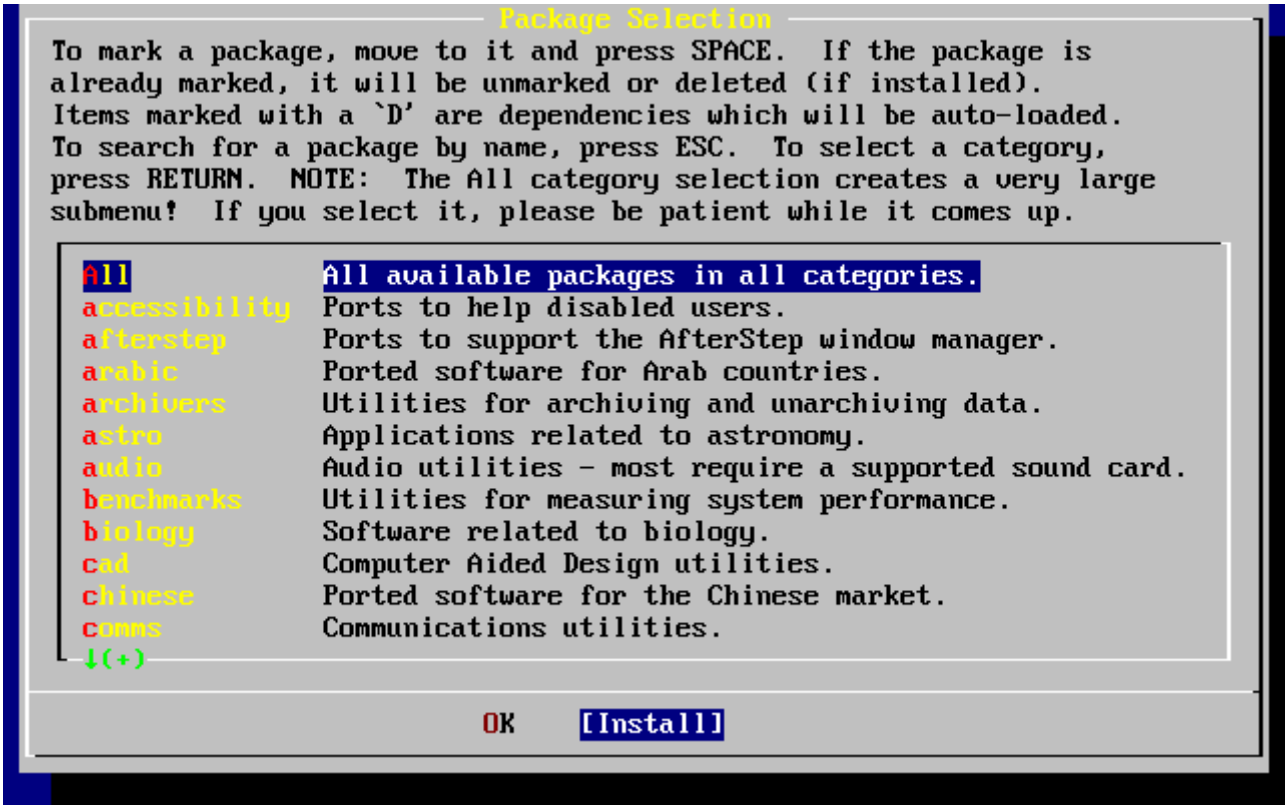


图 49. 安装预编译软件包

使用 Tab 和左右方向键选择 [Install] 并按 Enter。 接下来需要确认将要安装的预编译包：

图 50. 确认将要安装的预编译包

选择 [OK] 并按下 Enter 就可以开始预编译包的安装了。在这个过程中您会看到安装的相关信息，直到安装完成为止。请留意观察是否有错误信息出现。

在完成预编译包的安装之后， 就进入了最后的配置阶段。 如果您没有选择任何预编译包，并希望直接进入最后的配置阶段， 则可以选择 [Install] 来跳过。

2.10.12. 添加用户和组

在安装系统的过程中， 您应添加至少一个用户， 以避免直接以 root 用户的身份登录。
用以保存其用户数据的根分区通常很小， 因此用 root 身份运行程序可能将其迅速填满。
下面的提示信息介绍了这样做可能带来的更大隐患：

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

                    [ Yes ]  No
```

选择 [yes] 并按 Enter 即可开始创建用户的过程。

图 51. 选择用户

用箭头键来选择 User 然后按 Enter。

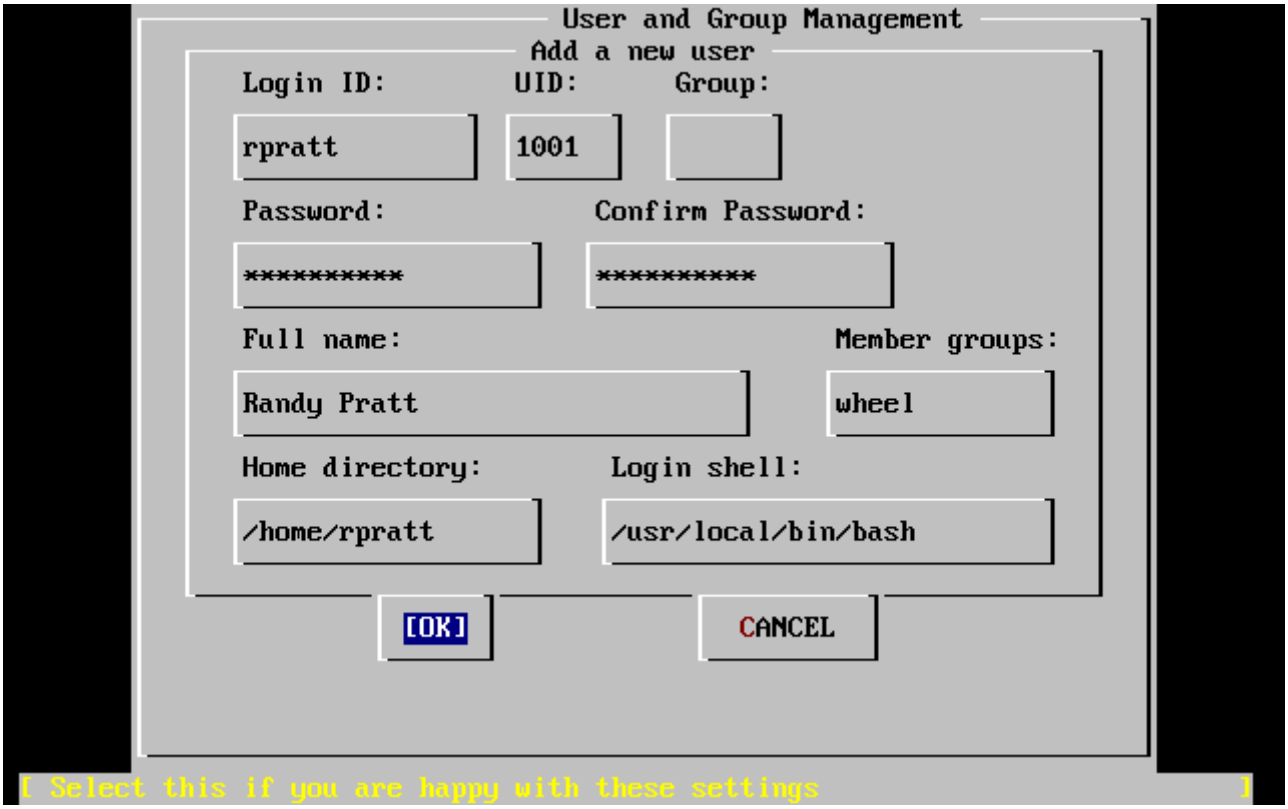


图 52. 添加用户信息

下面的描述信息会出现在屏幕的下方，可以使用 Tab 键来切换不同的项目，以便输入相关信息：

Login ID
: 新用户的登录名（强制性必须写）

UID
: 这个用户的ID编号（如果不写，系统自动添加）

Group
: 这个用户的登录组名（如果不写，系统自动添加）

Password
: 这个用户的密码（键入这个需要很仔细！）

Full name
: 用户的全名（解释、备注）

Member groups
: 这个用户所在的组

Home directory
: 用户的主目录（如果不写，系统自动添加）

Login shell
: 用户登录的shell（默认是/bin/sh）。

你可以将登录 shell 由 /bin/sh 改为 /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您不能登录的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。

用户也可以被添加到 wheel 组中成了一个超级用户，从而拥有 root 权限。

当您感觉满意时，键入 [OK] 键， 用户和组管理菜单将会重新出现。

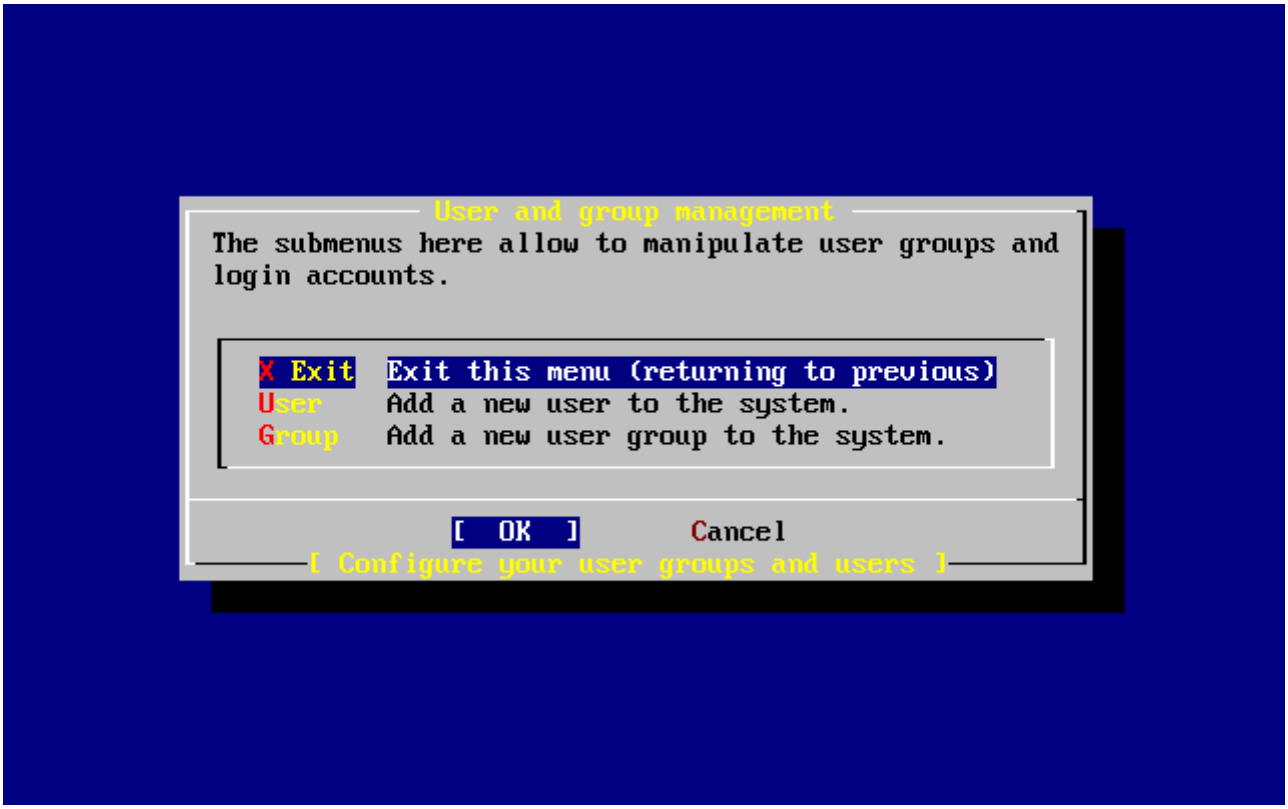


图 53. 退出用户和组管理

如果有其他的需要， 此时还可以添加其他的组。 此外， 还可以通过 sysinstall 在安装完成之后添加它们。

当您完成添加用户的时候，选择Exit 然后键入 Enter 继续下面的安装。

2.10.13. 设置 root 密码

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

                [ OK ]

          [ Press enter or space ]
```

键入 Enter 来设置 root 密码。

密码必须正确地输入两次。 毋庸讳言， 您需要选择一个不容易忘记的口令。
请注意您输入的口令不会回显， 也不会显示星号。

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

密码成功键入后，安装将继续。

2.10.14. 退出安装

如果您需要设置 其他网络设备， 或需要完成其他的配置工作， 可以在此时或者事后通过 sysinstall 来进行配置。

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

                     Yes  [ No ]
```

选择 [no] 然后键入 Enter 返回到主安装菜单。

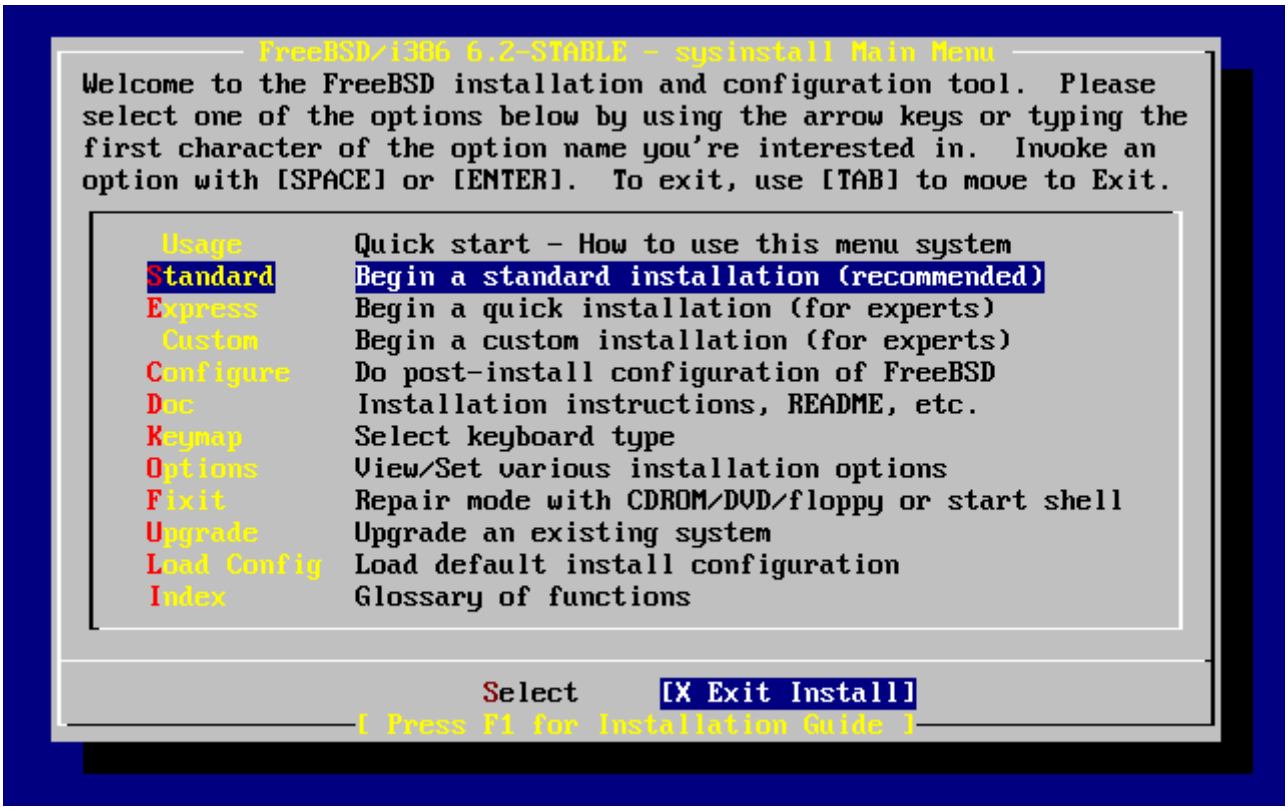


图 54. 退出安装

选择 [X Exit Install] 然后键入 Enter。您可能需要确认是否真的退出安装：

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

                     [ Yes ]  No
```

选择 [yes]。 如果您是从 CDROM 引导的系统， 则会出现下面的提示信息要求您取出光盘：

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

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

在系统开始重启之前， CDROM 驱动器是锁住的。 CDROM 解锁后就可以取出光盘了 (动作要快)。 按 [OK] 重启系统。

此后系统将重新启动， 因此请留意是否会出现一些错误信息。 进一步的细节， 请参见 FreeBSD 的启动过程。

2.10.15. 配置其他网络服务

如果之前缺少这一领域的经验， 那么配置网络服务对于新手而言， 很可能会是一件很有挑战的事情。 网络， 包括 Internet， 对于包括 FreeBSD 在内的所有现代操作系统而言都至关重要。 因此， 首先对 FreeBSD 提供的丰富的网络性能加以了解会很有帮助。 在安装过程中了解这些知识，
能够确保用户更好地理解他们可以用到的各种服务。

网络服务是一些可以接收来自网络上任何地方的人所提交的输入信息的程序。
人们一直都在努力确保这些程序不会做任何 "有害的" 事情。 不幸的是，
程序员们并不是十全十美的完人，因此，网络服务程序中的漏洞，
便有可能被攻击者利用来做一些坏事。因而，
只启用那些您知道自己需要的服务就很重要了。如果存在疑问，
那么就最好不要在您发现需要它之前启动任何网络服务。 您可以事后通过再次运行 sysinstall
或直接手工配置 /etc/rc.conf 来随时启用这些服务。

选择 Networking 选项将下显示一个类似下面的菜单：

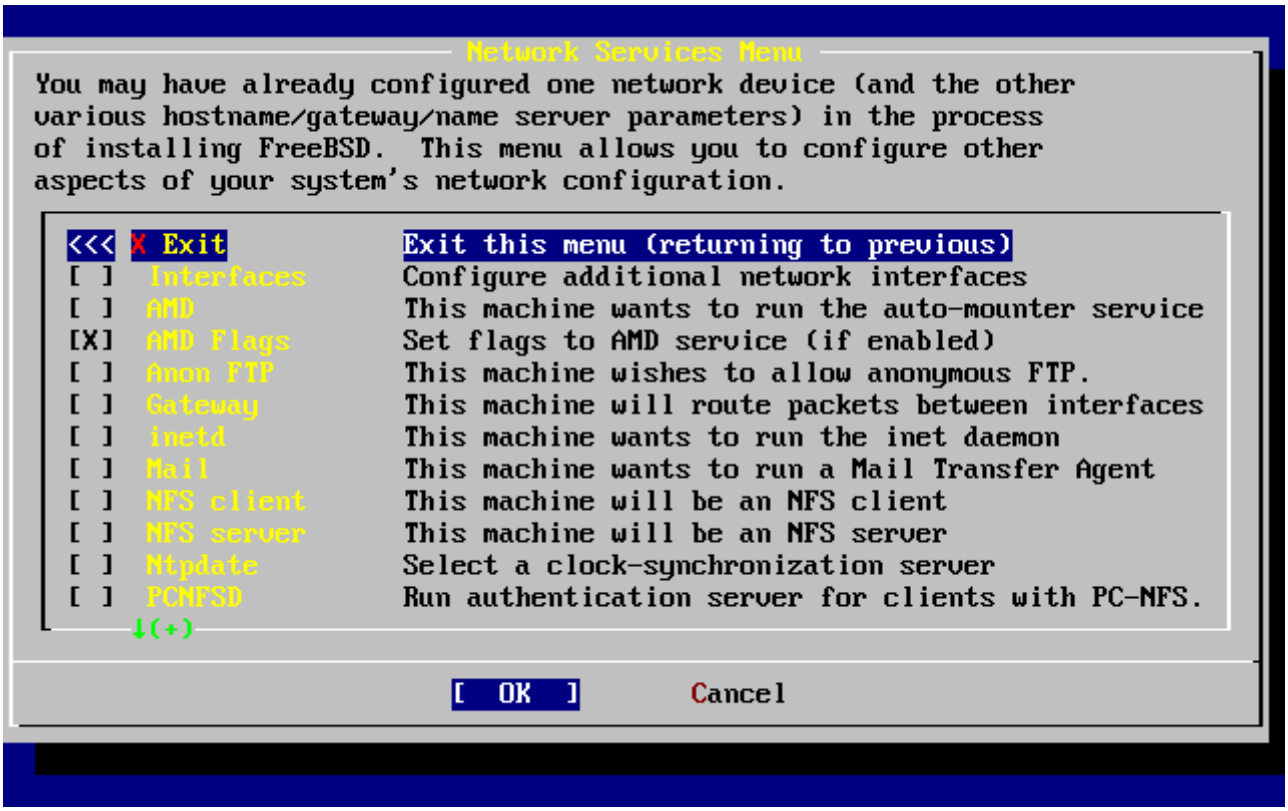


图 55. 网络配置之上层配置

第一个选项， Interfaces， 已经在前面的 配置网卡 中做过配置， 因此现在可以略过它。

选择 AMD 选项， 将添加对于 BSD 自动挂接程序的支持。 这个程序通常会和 NFS 协议 (详情参见下文) 配合使用，以便自动挂载远程文件系统。 启用它不需要在此时进行特殊的额外配置。

下一行是 AMD Flags 的参数选项。 选择它之后，会弹出一个让您选择 AMD 参数的子菜单。
菜单中包含一系列的选项：

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

-a 选项用来设置默认的挂接位置，这里使用的是 /.amd_mnt目录。-l 指定默认的 日志 文件； 但是，当使用 syslogd 时， 所有在日志中记录的活动， 都会发送到系统日志服务去。 /host
用来挂接远程主机上输出的文件系统，而 /net 目录则用来挂接从特定 IP 地址输出的文件系统。
/etc/amd.map 文件定义了用于 AMD 的默认输出选项。

Anon FTP 允许匿名 FTP 访问。 选中这个选项， 可以使这台机器成为一台匿名 FTP 服务器。
要注意启用这个选项的安全风险。 系统将使用另外的菜单来说明安全风险和进一步的配置。

Gateway 选项可以使将本机配置成为一台以前我们介绍过的网关。 如果您在安装过程中不小心选中了 Gateway， 也可以在这里用这个选项来取消。

Inetd 选项用来配置或完全禁用前面讨论过的 inetd(8) 服务程序。

Mail 用来配置系统默认的 MTA 或邮件传输代理。 选择这个选项将出现下面的菜单：

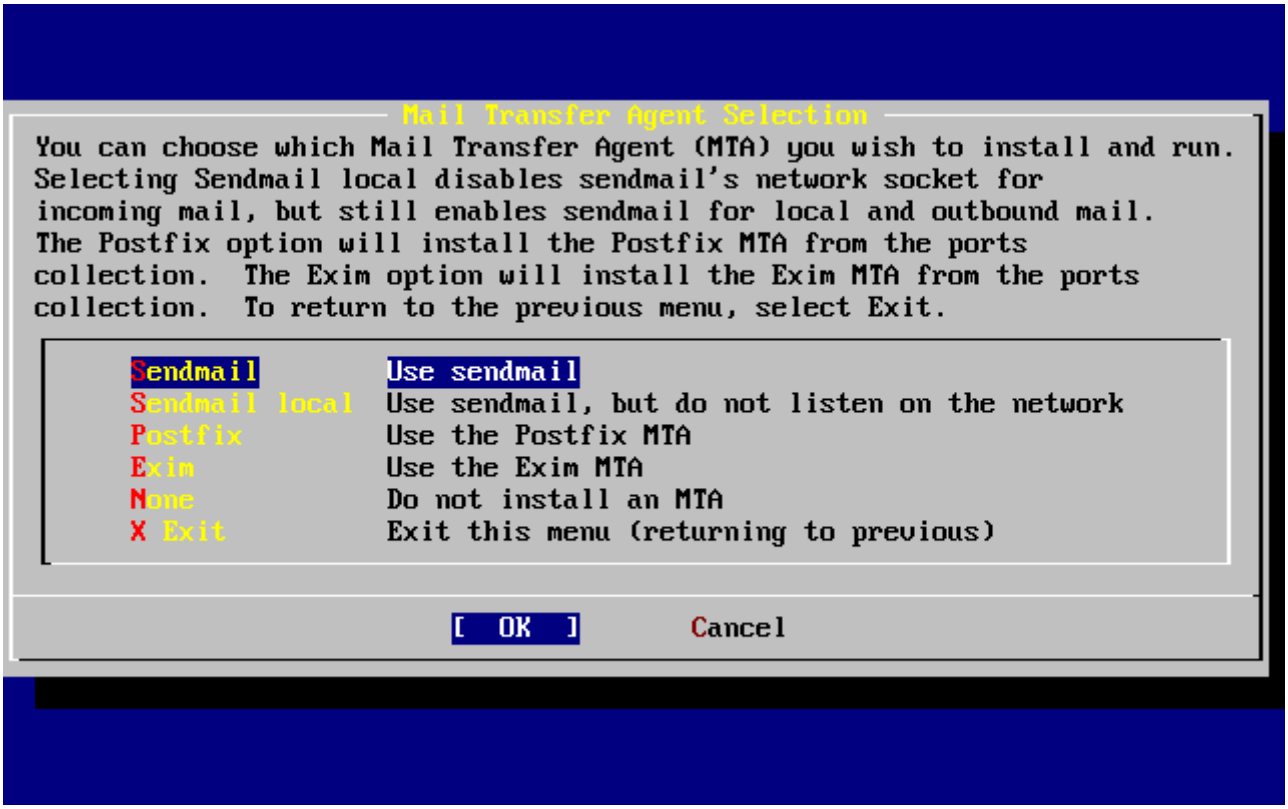


图 56. 选择默认的 MTA

这里给您提供了一个安装MTA 并将其配置为默认值的机会。MTA
是一种能够将邮件头递给本系统或互联网上的用户的邮件服务。

选择 Sendmail 将会安装十分流行的 sendmail 服务， 这也是 FreeBSD 的默认配置。Sendmail local 选项表示将 sendmail 设为默认的 MTA，但禁止其从 Internet 上接收邮件的能力。
此外还有一些其他选项，Postfix 和 Exim 与 Sendmail 的功能类似。 它们两者也可以投递邮件； 不过，有些用户会喜欢使用它们代替 sendmailMTA。

选择 MTA 或决定不挑选 MTA 之后， 网络配置菜单的下一项将是 NFS client。

NFS client 客户端可以使系统通过 NFS 与服务器进行通信。 NFS 服务器通过 NFS
协议可以使其它在网络上的机器来访问自己的文件系统。
如果这台机器要作为一台独立的服务器，这个选项可以保留不选。 如果启用它，
您在之后还需要进行更多的其他配置； 请参见 网络文件系统（NFS)
以了解关于配置客户机和服务器的进一步详情。

接下来的 NFS server 选项， 可以让您将本机系统配置为 NFS 服务器。 这会自动将启动 RPC
远程过程调用的信息写入配置文件。 RPC 是一种在多个主机和程序之间进行连接组织的机制。

下一项是 Ntpdate 选项， 它能够处理时间同步。 当选择它后， 会出现一个像下面所似的菜单：

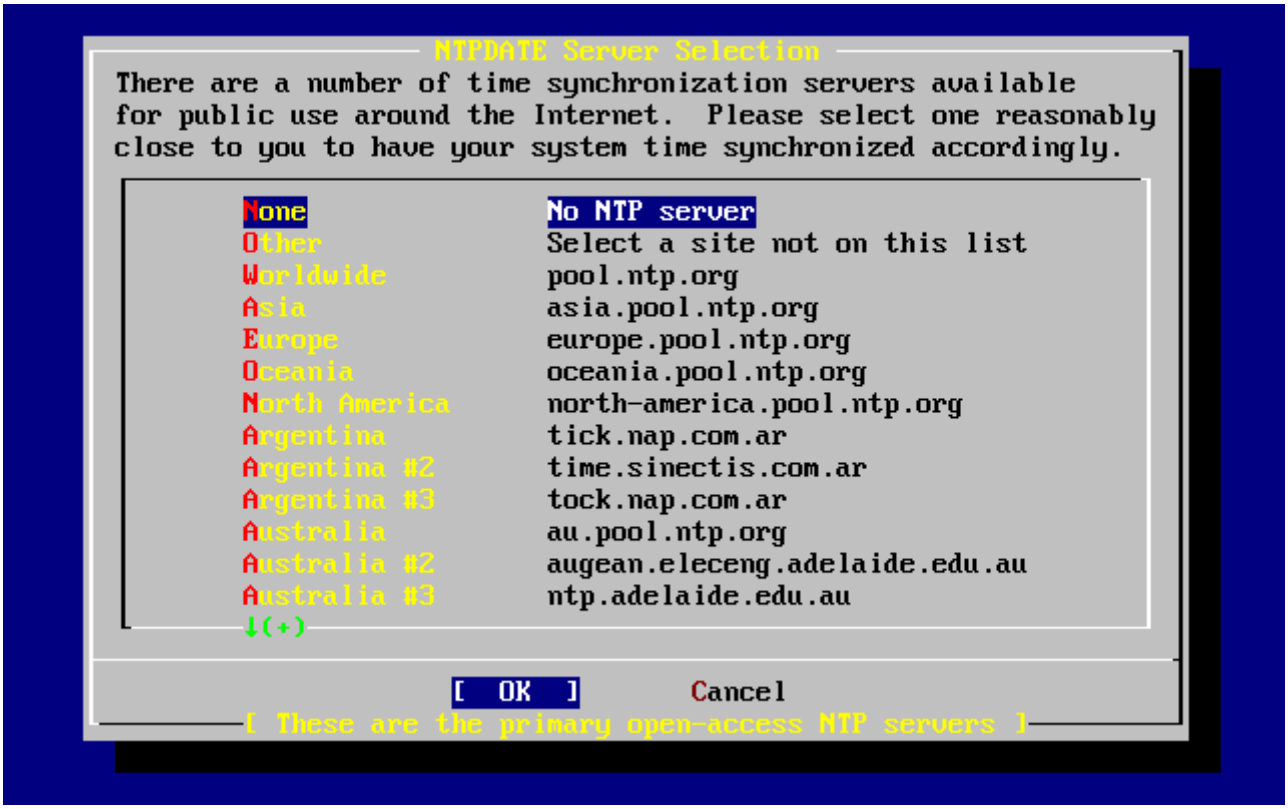


图 57. Ntpdate 配置

从这个菜单选择一个离您最近的服务器。 选择较近的服务器，有助于提高时间同步的精度，因为较远的服务器的连接延迟可能会比较大。

下一个选项是 PCNFSD。 这个选项将安装第三方软件包 net/pcnfsd。 它可以用来为无法自行提供 NFS 认证服务的操作系统， 如微软的 MS-DOS® 提供服务。

滚屏到下一页看一下其它选项：

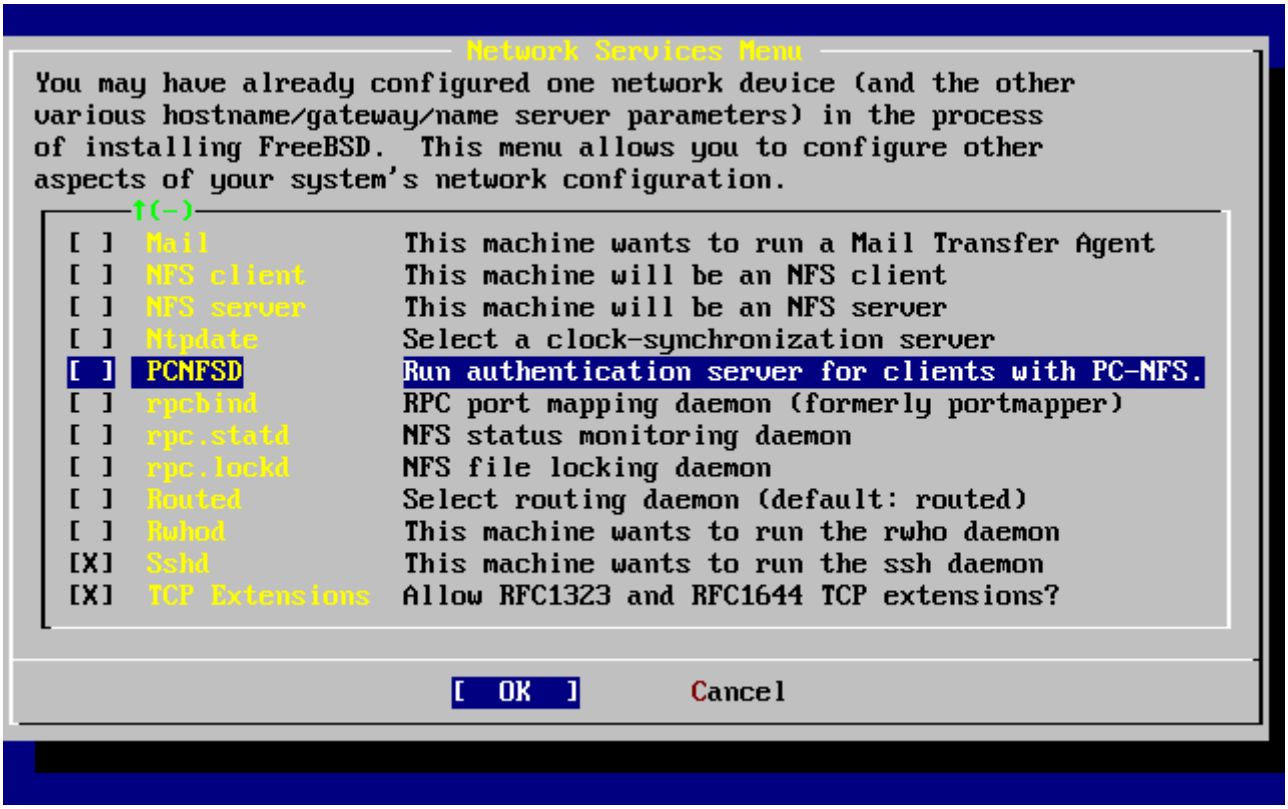


图 58. 网络配置之下层配置

rpcbind(8)， rpc.statd(8) 和 rpc.lockd(8) 这三个程序是用来提供远程过程调用 (RPC) 服务的。 rpcbind 程序管理 NFS 服务器和客户端的通信， 这是 NFS 正确工作的必要前提。rpc.statd 程序可以和其它主机上 rpc.statd 程序交互， 以提供状态监控。这些状态报告默认情况下会保存到 /var/db/statd.status 文件中。 最后的一项是 rpc.lockd 选项， 如果启用，则将提供文件上锁服务。通常将它和 rpc.statd 联用， 以监视哪些主机会请求对文件执行上锁操作， 以及这种操作的频繁程度。
尽管后两项功能对于调试非常有用， 但它们并不是 NFS 服务器和客户端正常运行所必需的。

下一个项目是Routed，这是一个路由程序。 routed(8) 程序管理网络路由表，发现多播路由，
并且支持在网络上与它物理相连的主机来复制它的路由表的请求。
它被广泛地应用在本地网络中并扮演着网关的角色。
当选择它后，一个子菜单会来询问您这个程序的默认位置。 默认的位置已经被定义过， 您可以选择 Enter 键， 也可以按下其它的键。 这时会出来另一个菜单来询问您传递给 routed程序的参数。 默认的是 -q 参数。

接下来是 Rwhod 选项， 选中它会启用 rwhod(8) 程序在系统初时化的时候。
rwhod程序通过网络周期性的广播系统 信息或以"客户"的身份来收集这些信息。 更多的信息可以查看 ruptime(1) 和 rwho(1) 手册页。

倒数第二个选项是sshd(8) 程序。它可以通过使用 OpenSSH 来提供安全的shell服务，
我们推荐通过使用它来使用 telnet 和 FTP 服务。 sshd
服务通过使用加密技术来创建从一台机器到另一台机器的安全连接。

最后有一个 TCP 扩展选项。 这可以用来扩展在 RFC 1323 和 RFC 1644 里定义的 TCP 功能。当许多主机以高速连接本机时，可能会引起某些连接被丢弃。 我们不推荐使用这个选项，
但是当使用独立的主机时可以从它上面得到一些好处。

现在您已经配置完成了网络服务， 您可以滚动屏幕到顶部选择 X Exit 项， 退出进入下一个配置部分，
或简单地选择两次 X Exit 之后选择 [X Exit Install] 来退出 sysinstall。

2.10.16. FreeBSD 的启动过程

2.10.16.1. FreeBSD/i386 的启动过程

如果启动正常，您将看到在屏幕上有很多信息滚动， 最后您会看到登录命令行。您可以通过键入 Scroll-Lock 和使用 PgUp 与 PgDn 来查看信息，再键入 Scroll-Lock 回到命令行。

记录信息可能不会显示（缓冲区的限制）。您可以通过键入 dmesg 来查看。

使用您在安装过程中设置的用户名/密码来登录。（例子中使用 rpratt）。除非必须的时候请不要用 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: CDROM <DELTA OTC-H101/ST3 F/W by OIPD> at ata0-slave using PIO4
Mounting root from ufs:/dev/ad0s1a
swapon: adding /dev/ad0s1b as swap device
Automatic boot in progress...
/dev/ad0s1a: FILESYSTEM CLEAN; SKIPPING CHECKS
/dev/ad0s1a: clean, 48752 free (552 frags, 6025 blocks, 0.9% fragmentation)
/dev/ad0s1f: FILESYSTEM CLEAN; SKIPPING CHECKS
/dev/ad0s1f: clean, 128997 free (21 frags, 16122 blocks, 0.0% fragmentation)
/dev/ad0s1g: FILESYSTEM CLEAN; SKIPPING CHECKS
/dev/ad0s1g: clean, 3036299 free (43175 frags, 374073 blocks, 1.3% fragmentation)
/dev/ad0s1e: filesystem CLEAN; SKIPPING CHECKS
/dev/ad0s1e: clean, 128193 free (17 frags, 16022 blocks, 0.0% fragmentation)
Doing initial network setup: hostname.
ed0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        inet 192.168.0.1 netmask 0xffffff00 broadcast 192.168.0.255
        inet6 fe80::5054::5ff::fede:731b%ed0 prefixlen 64 tentative scopeid 0x1
        ether 52:54:05:de:73:1b
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
        inet6 fe80::1%lo0 prefixlen 64 scopeid 0x8
        inet6 ::1 prefixlen 128
        inet 127.0.0.1 netmask 0xff000000
```

```
Additional routing options: IP gateway=YES TCP keepalive=YES
routing daemons:.
additional daemons: syslogd.
Doing additional network setup:.
Starting final network daemons: creating ssh RSA host key
Generating public/private rsa1 key pair.
Your identification has been saved in /etc/ssh/ssh_host_key.
Your public key has been saved in /etc/ssh/ssh_host_key.pub.
The key fingerprint is:
cd:76:89:16:69:0e:d0:6e:f8:66:d0:07:26:3c:7e:2d root@k6-2.example.com
 creating ssh DSA host key
Generating public/private dsa key pair.
Your identification has been saved in /etc/ssh/ssh_host_dsa_key.
Your public key has been saved in /etc/ssh/ssh_host_dsa_key.pub.
The key fingerprint is:
f9:a1:a9:47:c4:ad:f9:8d:52:b8:b8:ff:8c:ad:2d:e6 root@k6-2.example.com.
setting ELF ldconfig path: /usr/lib /usr/lib/compat /usr/X11R6/lib
/usr/local/lib
a.out ldconfig path: /usr/lib/aout /usr/lib/compat/aout /usr/X11R6/lib/aout
starting standard daemons: inetd cron sshd usbd sendmail.
Initial rc.i386 initialization:.
rc.i386 configuring syscons: blank_time screensaver moused.
Additional ABI support: linux.
Local package initialization:.
Additional TCP options:.

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

login: rpratt
Password:
```

生成 RSA 和 DSA密钥在比较慢的机器上可能要花很长时间。这只是一个
新安装后的首次启动，以后的启动会变得更快一点。

如果已经完成 X 服务器的配置， 且指定了默认的桌面窗口管理器， 就可以在命令行键入 startx
来启动它了。

2.10.17. FreeBSD 关机

正确的关闭操作系统是很重要的。不要仅仅关闭电源。 首先，您需要成为一个超级用户，通过键入 su
命令来实现。然后输入 root 密码。这需要用户是 wheel 组的一名成员。然后， 以root键入 shutdown -h
now命令。

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

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

您也能够使用 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，像如何报告发生的问题， 如何双重启动 FreeBSD 和 MS-DOS® 或 Windows®。

2.11.1. 当您遇到错误时，应该怎么做?

由于 PC 结构的限制， 硬件检测不可能 100% 地可靠， 但是有些问题是您可以自己解决的。

首先检查一下您使用的 FreeBSD 版本的 硬件兼容说明 文档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， 则需要联编 定制的内核。
这能够支持默认的 GENERIC 内核所不支持的设备。
在引导盘上的内核假定绝大多数的硬件，均为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， 则可能需要编辑内核配置， 并重新编译内核， 以便告诉 FreeBSD
到哪里去查找设备。

除此之外，也可能遇到这种情况吗，即探测某种并不存在的设备时， 会干扰到其他设备的检测并使其失败。
这种情况吗下应禁止驱动程序检测可能导致冲突的设备。

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给用户下载新的 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 。因为更新的过程 可能
会发生问题，进而损害 BIOS 芯片。

2.11.2. 使用 MS-DOS® 和 Windows® 文件系统

目前， FreeBSD 尚不支持通过 Double Space™ 程序压缩的文件系统。 因此，如果希望 FreeBSD
访问数据， 则应首先解压缩这些文件系统。 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)
命令能够把这样的文件系统挂接到现有的目录结构中， 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 中的某一行被调用或者被
mount(8) 工具并配合适当的参数来调用。

/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 的细节， 请参阅 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。 您的情况可能与引不同，查看命令 dmesg 和
mount 的输出。 它们应该可以让您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标记磁盘 slices， 特别需要指出的是，MS-DOS® 的扩展分区通常会比 MS-DOS® 主分区被标记为更高的数值。 可以使用 fdisk(8) 工具来帮助测定哪些 slices 属于 FreeBSD 哪些是属于其他的操作系统。

NTFS 分区也可以通过类似 mount_ntfs(8) 命令挂接在FreeBSD上。

2.11.3. 排除故障时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发生了死机、 安装过程中行为异常， 或没有检测到软驱。

FreeBSD 在启动过程中广泛使用了 i386、 amd64 及 ia64 平台提供的 ACPI 服务来检测系统配置。不幸的是， 在 ACPI 驱动和主板 BIOS 中存在一些 bug。 如果遇到这种情况， 可以在系统引导时禁用 ACPI， 其方法是在第三阶段引导加载器时使用 hint hint.acpi.0.disabled：

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

这一设置会在系统重启之后失效，因此，如果需要的话，您应在 /boot/loader.conf 文件中增加 hint.acpi.0.disabled="1"。 关于引导加载器的进一步详情， 请参见 概述。

2.11.3.2. 在硬盘安装 FreeBSD 之后的首次启动时， 内核加载并检测了硬件， 但给出下列消息并停止运行：

系统在处理引导盘非系统中的第一块盘时有一个由来已久的问题。 BIOS 采用的编号方式有时和 FreeBSD 不一致， 而设法将其变为一样则很难正确地实现。

因而， 在发生这种情况时，FreeBSD 可能会需要一些帮助才能找到磁盘。有两种常见的情况，在这些情况下您都需要手工告诉 FreeBSD 根文件系统模块的位置。 这是通过告诉引导加载器 BIOS 磁盘编号、磁盘类型以及 FreeBSD 中的该种磁盘的编号来实现的。

第一种情况是有两块 IDE 硬盘， 分别配置为对应 IDE 总线上的主 (master) 设备， 并希望 FreeBSD 从第二块硬盘上启动。 BIOS 将两块硬盘识别为磁盘 0 和磁盘 1， 而 FreeBSD 则将其分别叫做 ad0 和 ad2。

FreeBSD 位于 BIOS 磁盘 1， 其类型是 ad 而 FreeBSD 磁盘编号则是 2， 因此， 您应输入：

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

注意， 如果您的主总线上有从设备， 则这一配置是不必要的 (因为这样配置是错的)。

第二种情况是从 SCSI 磁盘启动，但系统中安装了一个或多个 IDE 硬盘。这时，FreeBSD 磁盘编号会比 BIOS 磁盘编号小。如果您有两块 IDE 硬盘， 以及一块 SCSI 硬盘，则 SCSI 硬盘将会是 BIOS 磁盘 2，类型为 da 而 FreeBSD 磁盘编号是 0， 因此， 您应输入：

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

来告诉 FreeBSD 您希望从 BIOS 磁盘 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，则应以 1: 代替。

一旦您确定了应选用的正确配置， 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 将使用这个文件的内容， 作为对 boot: 提示的默认回应。

2.11.3.3. 在硬盘安装 FreeBSD 之后的首次启动时， Boot Manager 只是给出了 F? 的菜单提示，但并不继续引导过程。

在您安装 FreeBSD 进行到分区编辑器时所设置的磁盘尺寸信息不对。请回到分区编辑器并指定正确的磁盘尺寸。 这种情况必须重新安装 FreeBSD。

如果您无法确定在您机器上的正确尺寸信息，可以用一个小技巧： 在磁盘开始的地方安装一个小的 DOS 分区， 并在其后安装 FreeBSD。 安装程序能够看到这个 DOS 分区， 并利用它推测磁盘的尺寸信息， 这通常会有所帮助。

下面的技巧不再推荐使用， 在这里仅供参考：

如果您正准备建立只运行 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)， 选择 FreeBSD 独占整个硬盘每一个扇区的非标准选项。 这会扫除关于磁盘尺寸的一切烦恼， 但会限制您以后运行 FreeBSD 以外的其他操作系统的能力。

2.11.3.4. 系统找到了 ed(4) 网卡， 但总是报设备超时 (device timeout) 错误。

您的网卡可能使用了与 /boot/device.hints 文件中指定的 IRQ 不同的中断请求号。 ed(4) 驱动默认情况下并不支持 "软" 配置 (在 DOS 中使用 EZSETUP 配置的值)， 但如果您在网卡的 hints 中指定 -1， 便会使用软配置。

您应使用网卡的跳线进行硬配置 (根据需要修改内核设置) 或通过 hint hint.ed.0.irq="-1" 将 IRQ 指定为 -1。 这会告诉内核使用软配置。

另一个可能是您的网卡使用 IRQ 9， 这会与 IRQ 2 共用同一中断请求线， 同时也是导致问题的一个常见原因 (特别是 VGA 卡使用 IRQ 2 的时候！)。 您应尽量避免使用 IRQ 2 或 9。

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

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

2.12. 高级安装指南

这节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键盘的系统上安装FreeBSD

这种类型的安装叫做 "headless install（无头安装）"， 因您正要安装FreeBSD的机器不是没带显示器，就是没有显卡。 您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外一台机器来充当主显示设备和键盘。 要这样做，只要执行下面的步骤： 创建安装 USB 记忆棒，请看 准备引导介质一节说明； 此外， 也可下载 ISO 映像文件， 具体请参阅 创建一张安装光盘。

要将安装介质改为使用串口控制台， 需要按下面这些步骤来操作 (如果使用 CDROM 则可跳过第一步)：

1. 令安装 USB 记忆棒引导并进入串口控制台

   如果使用刚刚制作的 USB 记忆棒引导系统， 则 FreeBSD 会进入正常的安装模式。 我们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 盘。

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

   您需要根据实际情况修改挂点的名称。

   现在挂好了记忆棒， 您需要对其进行配置令其进入串口控制台。 为此， 需要在 USB 记忆棒中的 loader.conf 文件中加入下面的这行配置：

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

这样就完成了对 USB 记忆棒的配置， 您应使用 umount(8) 命令将其卸下：

```
# umount /mnt
```

现在就可以拔下 USB 记忆棒并进入这一过程的第三步了。

2. 令安装 CD 引导并进入串口控制台

   如果您直接使用 ISO 映像 (see 创建一张安装光盘) 制作的 CD 引导， 则 FreeBSD 会引导进入正常的安装模式。 我们希望引导到串口控制台来完成安装。 为了做到这一点， 您需要展开、 修改并重新生成 ISO 文件， 然后再刻录光盘。

   在保存例如 FreeBSD-8.1-RELEASE-i386-disc1.iso ISO 的 FreeBSD 系统上用 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
   ```

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

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

   最后， 从修改好的目录树中创建新的 ISO 映像。 这里我们使用通过 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
   ```

   这样就完成了对 ISO 映像的配置， 您可以使用您熟悉的工具将其刻录到 CD-R 上了。

3. 连接 Null-modem 线

   现在需要一根 null-modem 线 来连接两台机器。 只要连接两台机器的串口。 这里不能使用普通的串口线， 而必须使用 null-modem 线， 因为它需要一些内部交叉的连线。

4. 开始启动安装

   现在可以开始安装了。 将 USB 记忆棒插到您准备进行 headless 安装的机器上， 然后开机。 如果您使用的是 CDROM， 则在开机之后立即将光盘放进光驱。

5. 连接您的无头机器

   现在您已经通过cu(1)连接到了那台机器。

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

   在 FreeBSD 7.X 上应使用下面的命令：

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

这样就可以了！ 您现在可以通过 cu 会话来控制那台 headless 的机器了。 接着系统会提示选择终端类型。 选择 FreeBSD 彩色控制台并继续安装！

2.13. 准备您自己的安装介质

为了避免重复 "FreeBSD disc" 在这里指 FreeBSD CDROM or DVD 那即意味着您要购买或自己制做。

有好几个原因需要您创建自己的FreeBSD安装介质。 这可能是物理介质，如磁带，使用 sysinstall 程序找到的安装文件， FTP 站点或 MS-DOS®分区。

例如：

- 您有许多机器连接到本地网络，使用一个FreeBSD光盘。 您要使用FreeBSD来创建一个本地FTP站点，然后使用这个FTP站点来代替连接到Internet。
- 您有一张 FreeBSD 光盘， FreeBSD 不支持您的 CD/DVD 驱动器， 但 MS-DOS®/Windows® 支持。 您要复制安装文件到一个DOS分区， 然后使用这些文件进行安装。
- 您要安装的计算机没有 CD/DVD驱动器和网卡，但您可以连接一个 "Laplink-style" 串口或并口线缆到那台计算机。
- 您要通过一个磁带机来安装FreeBSD.

2.13.1. 创建一张安装光盘

FreeBSD 的每个发行版本都为每一支持的平台提供至少两张 CDROM 映像 ("ISO images")。如果您有刻录机， 这些映像文件可以被("burned") 成FreeBSD的安装光盘。 如果没有刻录机，而上网带宽却很便宜，它也是一种很好的安装方式。

1. 下载正确的 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。选择合适的 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	这个 CD 映像可以让您从光驱启动并进入安装过程，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，您需要通过网络 (例如从 FTP 服务器) 来完成安装。
FreeBSD-版本-RELEASE-架构-dvd1.iso.gz	这个 DVD 映像包括用于安装 FreeBSD 操作系统基本组件、预编译包和文档所需的全部文件。它也支持引导进入基于 "livefs" 的修复模式。

文件名	包含内容
FreeBSD-版本-RELEASE-架构-memstick.img	这个映像可以写进 USB 记忆棒， 用于引导系统并完成安装。 它也支持引导进入基于 "livefs" 的修复模式。 这个版本的映像中包含了文档所需要的全部文件 ，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	这个 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	这个 CD 映像包含了能填满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	另一个包含了能填满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	这个 CD 映像包含了用以支持引导进入基于 "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习惯。 它们的 ISO 文件名不使用 FreeBSD- 前缀。

您 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 带宽很廉价， 则应使用 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软件 (参见 安装应用程序: Packages 和 Ports)。

如果打算安装 FreeBSD 并安装常用的软件包， 则应使用 dvd1。

其它的映像盘也很有用， 但不是必须的， 尤其是在您有高速的网络连接时。

2. 刻录 CDs

   您必须把这些映像文件刻录成光盘。 如果您在其它的FreeBSD系统上完成此项工作，请看 创建和使用光学介质(CD) 得到更多的信息，（特别是 burncd 和 cdrecord）

   如果您在其它的系统平台上执行，您需要相应的刻录软件。
   映像文件使用的是标准的ISO格式，必须被您的刻录软件所支持。

如果有兴趣制作一张定制的 FreeBSD 版本， 请参考 Release Engineering Article。

2.13.2. 为 FreeBSD 安装盘建立局域网 FTP 站点

FreeBSD 光盘的布局和 FTP 站点相同。 这样， 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. 在要作为FTP站点的那台FreeBSD机器上， 确定FreeBSD磁盘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 服务器的账号。 您可以利用 vipw(8) 命令编辑 /etc/passwd 文件， 加入下面这一行叙述：

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

3. 确定在 /etc/inetd.conf 配置文件中开启了FTP服务。

任何本地网络中的机器在安装 FreeBSD 选择安装介质时就可以选择透过 FTP 站点，然后选取 "Other" 后输入 ftp:// 本地FTP服务器 即可以透过本地的FTP站点来安装FreeBSD。

如果用作 FTP 客户端的引导介质 (通常是软盘) 与本地局域网的 FTP 站点上的版本不一致， sysinstall 会不允许您完成安装。 如果您使用的版本差距不很大，并且希望绕过这一判断， 则应进入 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服务公开给外面的网际网络（非本地用户），
您的电脑必须承担被侵入或其它的风险。 我们强烈建议您要有完善的安全机制才这样做。

2.13.3. 创建安装软盘

如果您从软盘安装（我们_不_推荐那样做），
或者是由于不支持硬件或者更简单的理由是因为您坚持要使用软盘安装。 您必须准备几张软盘。

至少这些软盘必须是 1.44 MB 的，用来容纳所有在 base (基本系统) 目录下的文件。如果您在 DOS 操作系统下准备就 必须 使用 MS-DOS® 的 FORMAT 命令来格式化软盘。 如果您使用的是 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，并选择 "Format")。

不要 指望厂家的预先格式化！ 最好还是亲自进行格式化。
过去用户报告的很多问题都是由于不正确地使用格式化设备所造成的， 所以我们需要在这里着重提一下。

如果您在另外一台FreeBSD的机器上做了启动盘的话， 进行格式化是一个不错的主意。
虽然您不需要把每张盘都做成DOS文件系统。您也可以使用 bsdlabel 和 newfs
命令来创建一个UFS文件系统，具体操作按下面的顺序进行：

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

然后您就可以像其它的文件系统一样挂上和写入这些磁盘。

格式化这些磁盘后，您必须把文件复制到磁盘中。 这些发行文件被分割成刚好可存进五张 1.44 MB 软盘。检查您所有的磁盘， 找出所有可能适合的文件。 直到您找到所有需要的配置并且将它们以这种方式安置。第一个配置都应该有一个子目录在磁盘上， 例如： a:\base\base.aa、 a:\base\base.ab， 等等。

base.inf 文件， 也应放在 base 的第一张盘上， 因为安装程序需要读取这个文件，以了解在获得发布包时需要下载多少文件。

一旦您进入选择安装介质的屏幕， 选择 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， 您需要将发布文件复制到该分区根目录下的 freebsd 目录中。 例如: c:\freebsd。 您必须复制一部分 CDROM 或 FTP 上的目录结构， 因此， 如果您从光盘进行复制， 建议使用 DOS 的 xcopy 命令。 下面是准备进行 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: 盘是您的空闲空间， E: 盘是您挂接的 CDROM。

如果您没有光盘驱动器，您可以从以下网站下载发行包。ftp.FreeBSD.org.
每一个发行包都在一个目录中，例如 base 发行包可以在 12.0/base/ 目录中找到。

对很多发行包来说，如果您希望从 MS-DOS®分区安装的话 （您有足够的空间），安装 c:\freebsd - 下的每个文件－这个 BIN 发行包只是最低限度的要求。

2.13.5. 创建一个安装磁带

从磁带安装也许是最简单的方式， 比在线使用 FTP 安装或使用 CDROM 还快。安装的程序假设是简单地被压缩在磁带上。
在您得到所有配置文件后，简单地解开它们，用下面的命令:

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

在您安装的时候，您要确定留有足够的空间给临时目录（允许您选择） 来容纳磁带安装时 全部 的内容。由于不是随机访问 磁带的，所以这种安装方法需要很多临时空间。

开始安装时，在从软盘启动 之前， 磁带机必须已经放在驱动设备中。否则， 安装过程中可能会找不到它。

2.13.6. 通过网络安装

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 线缆))。

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 支持绝大多数常见 PC 以太网卡； 系统能够支持的网卡 (以及所需的配置) 可以在 FreeBSD 发行版附带的硬件兼容说明中找到。
如果您使用的是系统支持的 PCMCIA 以太网卡， 在为笔记本加电 之前 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热插 PCMCIA 卡。

此外， 您还需要知道自己的 IP 地址、 网络类型对应的子网掩码， 以及机器名。 如果您正通过 PPP 连接安装而没有固定的静态 IP， 不用怕， 这个 IP 地址会由您的 ISP 自动分配。
您的系统管理员会告诉您进行网络配置所需的信息。 如果您需要通过名字而不是 IP 地址来访问其他主机， 则还需要配置一个域名服务器， 可能还需要一个网关地址 (在使用 PPP 时， 这个地址是服务提供商的 IP 地址)。 如果您希望通过 HTTP 代理服务器来完成 FTP 安装， 还需要知道代理服务器的地址。
如果您不知道这些信息， 则应在进行这种安装 之前 向系统管理员或 ISP 询问。

如果您使用一个 MODEM，那您就只有 PPP 这一种选择了。在您安装的过程中，
要确定您能很容易地获得完整且快速的关于您服务提供商的信息。

如果您使用 PAP 或 CHAP 方式连接到您的 ISP，
（换句话说，如果您不使用脚本在Windows®中连接到您的ISP） ， 那么您需要在 ppp 提示符下输入 dial 命令。否则，当 PPP 连接者只提供一种最简单的终端模拟器，您必须知道如何使用针对 MODEM 的 "AT commands"拨号到您的 ISP。 想知道更深入的信息可以参考 使用手册中的用户级PPP那节 以及 FAQ 。

如果您有一些问题，可以使用 set log local … 命令将日志显示在屏幕上。

您也可以通过并口电缆连接到另外一台 FreeBSD 机器上进行安装，您可以考虑使用 "laplink"
并口电缆进行安装。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NFS安装之前

NFS 安装方式是非常方便的。只需要简单地将 FreeBSD
文件复制到一台服务器上，然后在安装时选择NFS介质。

如果这个服务器要 "特权端口" 才能支持 （如SUN的工作站），您需要在安装前在 Options 菜单中设置 NFS Secure。

如果你使用了一块低质量的以太网卡比较糟糕， 速度很慢，则应考虑 NFS Slow的选项。

为了达到NFS安装的目的，这个服务器必须支持 subdir 加载。 例如，如果您的 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，而不仅仅是 /usr 或 /usr/archive/stuff。

在 FreeBSD的 /etc/exports 配置文件中， 是由 -alldirs 选项来控制的。其它 NFS
服务器也许有不同的方式。如果您从服务器得到 permission denied 这个信息，
可能是因为您没有正确的启用它。

Chapter 3. 安装 FreeBSD（适用于 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，而在之前则是指 sysinstall。 本章介绍 bsdinstall 的使用， 有关 sysinstall 的使用参见 安装 FreeBSD。

学习完本章之后， 您将知道：

- 如何创建 FreeBSD 安装介质。
- FreeBSD 如何划分目标硬盘。
- 如何启动 bsdinstall。
- 运行 bsdinstall 时需要回答的问题， 问题的具体含义， 以及应该如何回答。

阅读本章之前， 您应该：

- 查看将要安装的 FreeBSD 版本所附的硬件支持列表， 以确定您的硬件能够被支持。

一般来说， 此安装说明是针对 i386™（“PC 兼容机”） 架构的计算机；同时也会尽可能地对其他架构下的安装予以说明。 虽然本文档经常更新，但仍可能与所安装版本上附带的说明文档有些许出入，因此建议您仅将其作为常规的安装指导。

3.2. 硬件需求

3.2.1. 最低配置

安装 FreeBSD 所需的最低配置， 随版本及硬件架构而有所不同。

以下几节对这些信息进行了总结。 根据所选的安装方式， 可能需要使用 FreeBSD 支持的 CDROM 或网络适配器， 详见 准备安装介质。

3.2.1.1. FreeBSD/i386

FreeBSD/i386 需要 486 或更快的处理器， 最小 64 MB 的内存， 以及至少 1.1 GB 的硬盘空间。

通常情况下对于老旧的计算机而言， 安装更大的内存和腾出更多的硬盘空间，会比使用更快的处理器对性能的提升更加明显。

3.2.1.2. FreeBSD/amd64

FreeBSD/amd64 支持两种处理器。 第一种是 AMD64 处理器， 包括 AMD Athlon™64、 AMD Athlon™64-FX、 AMD Opteron™ 以及更高级别的处理器。

能够使用 FreeBSD/amd64 的另一种处理器是采用了 Intel® EM64 架构的处理器。 这类处理器包括 Intel® Core™ 2 Duo、 Quad 和 Extreme 家族， 还包括 Intel® Xeon™ 3000、 5000 和 7000 系列， 以及 Intel® Core™ i3、 i5 和 i7。

对于使用了 nVidia nForce3 Pro-150 的机器， 必须 在 BIOS 设置中禁用 IO APIC，如果没有这样的选项就只能转而禁用 ACPI。 因为 Pro-150 芯片组存在bug，而目前还没有能够规避此问题的方法。

3.2.1.3. FreeBSD/powerpc Apple® Macintosh®

支持所有内建 USB 的 New World Apple® Macintosh® 系统， 同时也为配置多 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 蓝白机上的 FireWire® 也不被支持。

3.2.1.4. FreeBSD/sparc64

有关 FreeBSD/sparc64 的系统支持， 详见 FreeBSD/sparc64 项目。

FreeBSD/sparc64 需要独占一块磁盘。 目前还不支持与其他操作系统共享同一块磁盘。

3.2.2. 支持的硬件

FreeBSD 发行版所支持的硬件架构及设备会列在硬件兼容说明文件中， 此文件通常名为 HARDWARE.TXT，位于发行版介质的根目录下。 这些内容也可以在 FreeBSD 网站的 发行版信息 页面上找到。

3.3. 安装前的准备工作

3.3.1. 备份您的数据

在将 FreeBSD 安装至目标机器前， 应首先备份其上的重要数据并对备份进行测试。 FreeBSD 安装程序对硬盘做任何改动前都会进行询问， 而一旦操作开始就无法撤销。

3.3.2. 决定将 FreeBSD 安装在何处

如果整个硬盘上仅安装 FreeBSD 一个操作系统， 那么请直接跳过此节； 但如果需要让 FreeBSD 与其他操作系统并存， 那么首先应当了解 FreeBSD 的硬盘布局结构。

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）。

有两种硬盘分区方式。 传统的 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创建为 extended partition (扩展分区)， 并在其中建立 logical partitions (逻辑分区)。 正如您看到的那样， 这种方法十分笨拙。

新式的 GUID Partition Table (GUID 分区表) (GPT) 提供了更为简便的磁盘分区方法。 与传统的 MBR 分区相比， GPT 功能更为强大。 常见的 GPT 实现可以在一块磁盘上支持多达 128 个分区，从而无需再采用类似逻辑分区这样迭床架屋的结构。

一些旧式的操作系统， 如 Windows® XP 并不兼容 GPT 分区格式。 如果需要让 FreeBSD 与这样的操作系统共用一块硬盘， 就必须使用 MBR 分区了。

FreeBSD 的标准引导加载器需要使用一个主分区或 GPT 分区。 (有关 FreeBSD 引导过程的详情， 请参阅 FreeBSD 引导过程。) 如果所有的主分区或 GPT 分区都已在使用中， 则必须为 FreeBSD 腾出一个来使用。

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。 比较实用的情况下， 如果不使用图形界面， 最小安装应分配至少 3 GB 的空间， 而使用图形界面， 则应分配至少 5 GB 的空间。 此外， 第三方应用程序可能还需要更多的空间。

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 分区编辑器。 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现有的数据。 在修改磁盘分区之前，应先做一次完整的备份并校验其完整性。

调整 Microsoft® Vista 分区大小时可能会遇到一些问题。 如果要这样做， 请提前准备好 Vista 安装光盘。

例 3. 使用现有的分区

> 假设一台安装了 Windows® 的计算机上有一块 40 GB 的硬盘， 分成了两个 20 GB 的分区。 Windows® 将它们分别叫做 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: 移动到 C:， 就可将第二个分区腾出来供 FreeBSD 使用了。

例 4. 缩小现有的分区

> 假设一台安装了 Windows® 的计算机上有一块 40 GB 的硬盘，
> 一个大的分区使用了整块磁盘的全部空间。 Windows® 将这个 40 GB 分区叫做 C:。 目前占用了 15 GB 空间。 现希望将 Windows® 分区减少到 20 GB， 并将余下的 20 GB 分给 FreeBSD 使用。
>
> 可以在以下两种方法中任选一种：
>
> 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 这样的分区调整工具来缩小 Windows® 分区， 并腾出空间给 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的一种。
如果希望同时运行多种不同的操作系统， 可以使用在 虚拟化 中介绍的方法。

3.3.3. 收集网络配置信息

某些 FreeBSD 安装方式需要通过网络连接下载相关文件。 若要连接至以太网 (或电视电缆/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， 则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. 本地网络域名
5. DNS 服务器的 IP 地址

3.3.4. 检查 FreeBSD 发行勘误

尽管 FreeBSD 项目会确保每个发行版尽可能地稳定， 但 bug 总是在所难免。 极少数情况下， 这些 bug 甚至会影响安装。 一旦这些问题被发现并修正后， 就会列在 FreeBSD 网站的 FreeBSD 发行勘误 中。
在安装之前， 应首先检查这些勘误， 以确保安装可以顺利进行。

有关所有发行版的信息及勘误， 可以在 FreeBSD 网站 的 发行版信息 一节中找到。

3.3.5. 准备安装介质

FreeBSD 的安装介质包括 CD、 DVD 及 USB 记忆棒。 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执行安装程序这种方式进行安装。

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， 还有一种 bootonly 安装介质。
这种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进行下载。 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。

如果您已经有 FreeBSD 的安装 CD、 DVD 或 USB 记忆棒， 则可以跳过此节。

FreeBSD 的安装 CD 或 DVD 映像均为可引导的 ISO 文件。 只需要 CD 或 DVD 其中的一种即可完成安装操作。 任选一种在当前操作系统中刻录成可引导光盘即可。

若要创建可引导的记忆棒， 请执行以下操作：

1. 获取记忆棒映像

   FreeBSD 9.0-RELEASE 和更高版本的记忆棒映像文件可以在 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 则是指要安装的版本号。 举例来说， FreeBSD/i386 9.0-RELEASE 的记忆棒映像位于 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， 映像文件的下载位置略有不同。 关于 FreeBSD 8.X 和更早版本的安装操作请参阅 安装 FreeBSD。

   记忆棒映像的扩展名为 .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， 有时也包括安装对象的硬件状况进行选择。

   执行以下步骤前， 应 备份 USB 记忆棒上的数据， 因为之后的操作将 擦除 这些数据。

2. 将映像文件写入记忆棒

   Procedure: 在 FreeBSD 中操作

   在下面的例子中， 目标记忆棒对应的设备节点是 /dev/da0。
   操作前请仔细确认目标设备是否正确， 以免损坏现有的数据。

   a. 使用 dd(1) 写入映像

      扩展名为 .img 的映像文件 不是 一种普通的文件。 它是对记忆棒上完整内容所做的 映像， 因此 不能 只是像普通文件一样简单的复制， 而应使用 dd(1) 将其直接写入目标设备：

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

   Procedure: 在 Windows® 中操作

   操作前请确认是否为目标设备选择了正确的驱动器号，
   否则可能会覆盖并损坏您的现有数据。

   a. 获取 Image Writer for Windows®

      Image Writer for Windows® 是一种能将映像正确写入到记忆棒中的免费应用程序。 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

      双击图标 Win32DiskImager 运行程序后， 确定 Device

下面显示的驱动器号所对应的是记忆棒。 点击文件夹图标以选择需要写入的映像文件，然后点击 [Save] 接受选择。 在确认所有操作无误且没有其他窗口访问记忆棒后， 点击 [Write] 将映像文件写入记忆棒。

系统不再支持从软盘进行安装了。

您现在可以开始安装 FreeBSD 了。

3.4. 开始安装

默认情况下， 在您看到下面这条信息之前， 安装程序不会对硬盘数据做任何修改：

```
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， 抑或您担心进行了某些错误的配置， 也可以直接关闭电源。

3.4.1. 开机启动

3.4.1.1. 引导 i386™ 及 amd64 系统

1. 若要使用 准备安装介质 所述的 USB 记忆棒引导， 则应在开机前将其插入计算机。

   若要使用 CDROM 引导， 则应在开机后立刻将其放入计算机。

2. 根据所使用的安装介质， 选择从 CDROM 或 USB启动。 在 BIOS 设置中，可以选择特定的引导设备。 大多数系统还可以在启动时选择引导设备， 通常需要按 F10、 F11、 F12 或 Escape 键。
3. 如果您的计算机正常启动并加载了现有的操作系统， 那么请检查：
   a. USB 记忆棒插入过晚或 CDROM 放入过晚， 请将其拔下或取出，然后重新启动计算机并再次尝试。
   b. BIOS 设置错误， 请重新设置。
   c. BIOS 不支持从当前介质启动； 可以使用 Plop Boot Manager， 它能够让老式计算机支持 CD 或 USB 启动。
4. FreeBSD 将开始启动。 如果使用的是CDROM， 则会看到类似这样的显示（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 引导加载器会显示：

图 59. FreeBSD 引导加载器菜单

您可以等待十秒或按 Enter 键。

3.4.1.2. 引导 Macintosh® PowerPC®

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

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

对于不带键盘的 Xserves 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 系统均设置成了硬盘自启动。 若要安装 FreeBSD， 则应从网络或 CDROM 启动，
这就需要首先进入 PROM（OpenFirmware）。

重启系统后等待引导信息出现， 虽然其具体内容取决于机器型号， 但应该会类似：

```
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.
```

如果此时系统已经开始从硬盘启动， 那么请按下 L1 + A 或 Stop + A 或在串口控制台发送 BREAK（在 tip(1) 或 cu(1) 中是 ~#）以进入 PROM 提示符， 它应该如下所示：

```
ok      ①
ok {0} ②
```

① 这是在单 CPU 系统上的提示符。

② 这是在 SMP 系统上的提示符， 其中的数字表示可用的 CPU 个数。

现在， 放入 CDROM 并在 PROM 提示符后输入 boot cdrom。

3.4.2. 查看设备探测结果

为了便于查阅， 屏幕上所显示的最后几百行字符会始终保存在缓冲区里。

若要浏览缓冲区， 可以按下 Scroll Lock 键来开启屏幕的滚动功能； 开启后即可使用方向键、 PageUp 键或 PageDown 键进行翻阅； 再次按下 Scroll Lock 键将关闭滚动功能。

浏览时将看到内核进行了设备探测， 其结果类似 典型的设备探测结果 中的文本，
但具体内容会因计算机中所包含的设备而有所不同。

典型的设备探测结果

```
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,CLFLUSH,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,0xe0000000-0xefffffff 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
usbus0: <Intel 82801I (ICH9) USB controller> on uhci0
uhci1: <Intel 82801I (ICH9) USB controller> port 0xcf40-0xcf5f irq 21 at device 26.1 on pci0
usbus1: <Intel 82801I (ICH9) USB controller> on uhci1
uhci2: <Intel 82801I (ICH9) USB controller> port 0xcf20-0xcf3f irq 19 at device 26.2 on pci0
usbus2: <Intel 82801I (ICH9) USB controller> on uhci2
ehci0: <Intel 82801I (ICH9) USB 2.0 controller> mem 0xff9ff800-0xff9ffbff irq 19 at device
26.7 on pci0
usbus3: EHCI version 1.0
```

```
usbus3: <Intel 82801I (ICH9) USB 2.0 controller> on ehci0
hdac0: <Intel 82801I High Definition Audio Controller> mem 0xff9f8000-0xff9fbfff irq 22 at
device 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
usbus0: 12Mbps Full Speed USB v1.0
usbus1: 12Mbps Full Speed USB v1.0
usbus2: 12Mbps Full Speed USB v1.0
usbus3: 480Mbps High Speed USB v2.0
ugen0.1: <Intel> at usbus0
uhub0: <Intel UHCI root HUB, class 9/0, rev 1.00/1.00, addr 1> on usbus0
ugen1.1: <Intel> at usbus1
uhub1: <Intel UHCI root HUB, class 9/0, rev 1.00/1.00, addr 1> on usbus1
ugen2.1: <Intel> at usbus2
uhub2: <Intel UHCI root HUB, class 9/0, rev 1.00/1.00, addr 1> on usbus2
ugen3.1: <Intel> at usbus3
uhub3: <Intel EHCI root HUB, class 9/0, rev 2.00/1.00, addr 1> on usbus3
uhub0: 2 ports with 2 removable, self powered
uhub1: 2 ports with 2 removable, self powered
uhub2: 2 ports with 2 removable, self powered
uhub3: 6 ports with 6 removable, self powered
ugen2.2: <vendor 0x0b97> at usbus2
uhub8: <vendor 0x0b97 product 0x7761, class 9/0, rev 1.10/1.10, addr 2> on usbus2
ugen1.2: <Microsoft> at usbus1
ada0 at ahcich0 bus 0 scbus1 target 0 lun 0
ada0: <Hitachi HTS543225L9SA00 FBEOC43C> ATA-8 SATA 1.x device
ada0: 150.000MB/s transfers (SATA 1.x, UDMA6, PIO 8192bytes)
ada0: Command Queueing enabled
ada0: 238475MB (488397168 512 byte sectors: 16H 63S/T 16383C)
```

```
ada0: Previously was known as ad4
ums0: <Microsoft Microsoft 3-Button Mouse with IntelliEyeTM, class 0/0, rev 1.10/3.00, addr
2> on usbus1
SMP: AP CPU #1 Launched!
cd0 at ahcich1 bus 0 scbus2 target 0 lun 0
cd0: <TEAC DV-W28S-RT 7.0C> Removable CD-ROM SCSI-0 device
cd0: 150.000MB/s transfers (SATA 1.x, ums0: 3 buttons and [XYZ] coordinates ID=0
UDMA2, ATAPI 12bytes, PIO 8192bytes)
cd0: cd present [1 x 2048 byte records]
ugen0.2: <Microsoft> at usbus0
ukbd0: <Microsoft Natural Ergonomic Keyboard 4000, class 0/0, rev 2.00/1.73, addr 2> on
usbus0
kbd2 at ukbd0
uhid0: <Microsoft Natural Ergonomic Keyboard 4000, class 0/0, rev 2.00/1.73, addr 2> on
usbus0
Trying to mount root from cd9660:/dev/iso9660/FREEBSD_INSTALL [ro]...
```

请仔细检查设备探测结果， 以确定 FreeBSD 找到了所有您希望使用的设备。
没有找到的设备并不会在这里列出， 因为默认的 GENERIC 内核中不包含它们； 可以通过 内核模块 对这些设备提供支持。

设备探测完成后， 您将看到 选择安装介质的使用方式， 表明安装介质共有三种用途： 安装 FreeBSD 、作为"`Live CD`"或引导至 FreeBSD 的命令行界面。 请使用方向键选择一项后按 Enter 键确认。

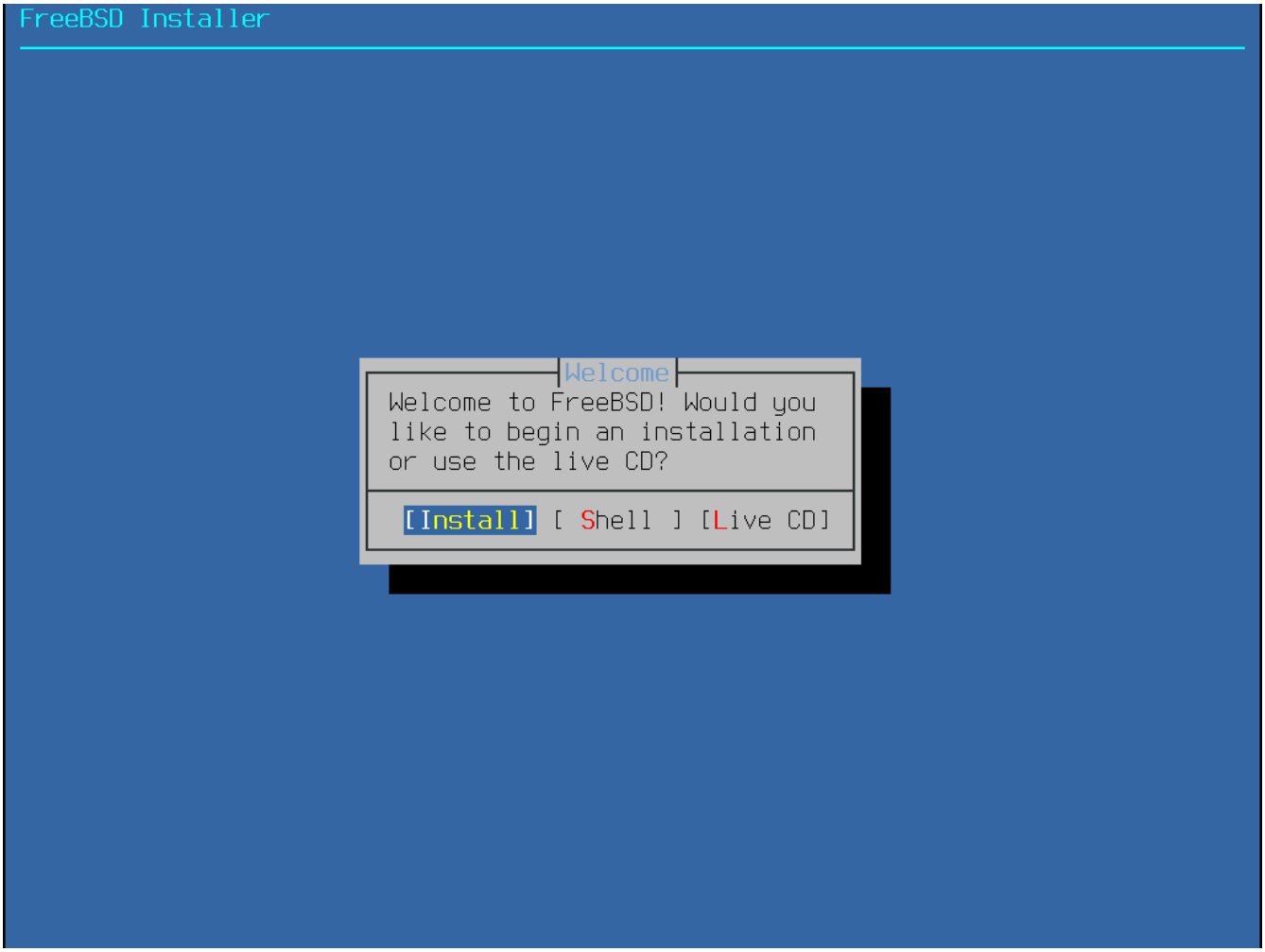


图 60. 选择安装介质的使用方式

在这里， 请选择 [Install] 以运行安装程序。

3.5. 介绍 bsdinstall

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

Kris Moore <kmoore@FreeBSD.org> 为 PC-BSD 编写的 pc-sysinstall 也可以用于 安装 FreeBSD。 虽然有时会同 bsdinstall 混淆， 但实际两者并不相关。

bsdinstall 菜单系统的主要控制键包括方向键、 Enter 键、 Tab 键、 Space 键等。

3.5.1. 选择键盘映射

根据当前正在使用的系统控制台， bsdinstall 可能会首先提示选择一种非默认的键盘布局。

图 61. 键盘映射选择

选择了 [YES] 后， 将显示下面的键盘选择画面； 否则将不显示此画面而直接使用默认键盘映射。

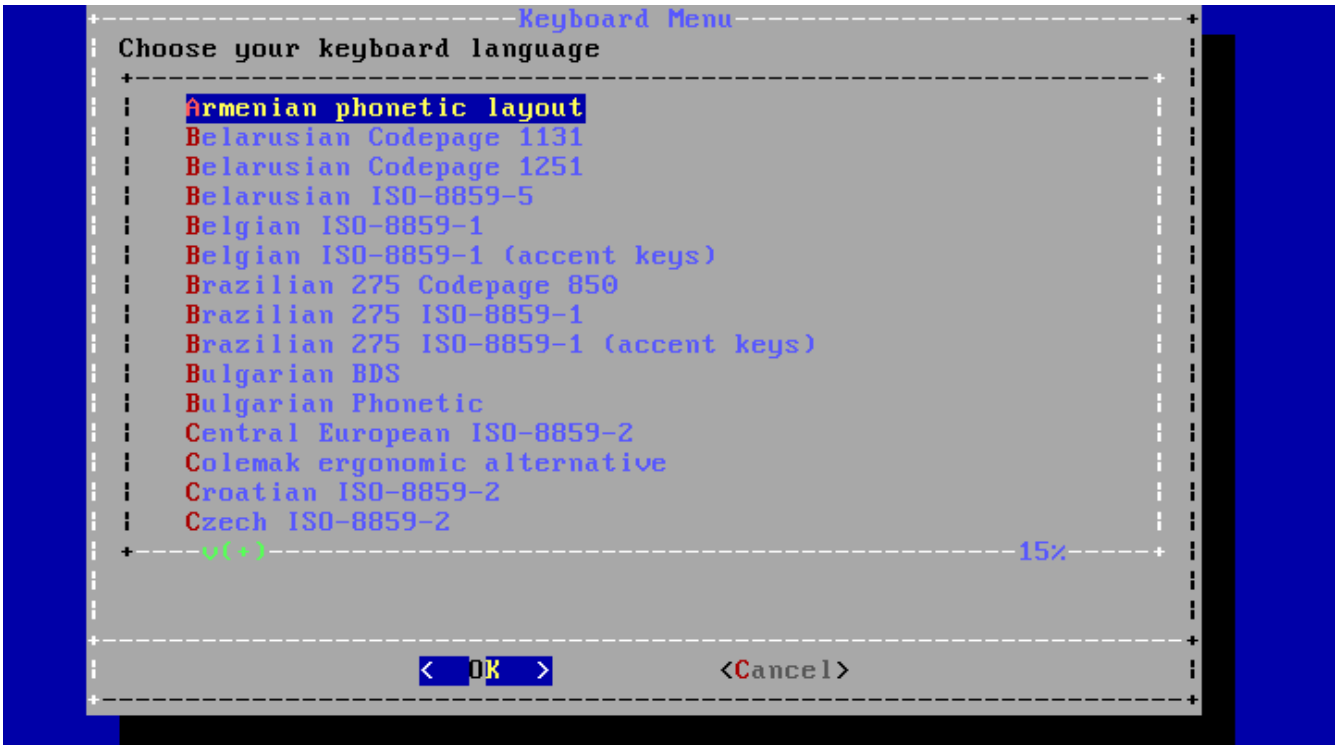


图 62. 键盘选择菜单

使用上/下方向键选择最适合当前系统的键盘映射后， 按 Enter 键确认。

按 Esc 键以使用默认的键盘映射。 如果不清楚该选择哪一项， 推荐 United States of America ISO-8859-1。

3.5.2. 设置主机名

下面， bsdinstall 将提示为新安装的系统设置主机名。

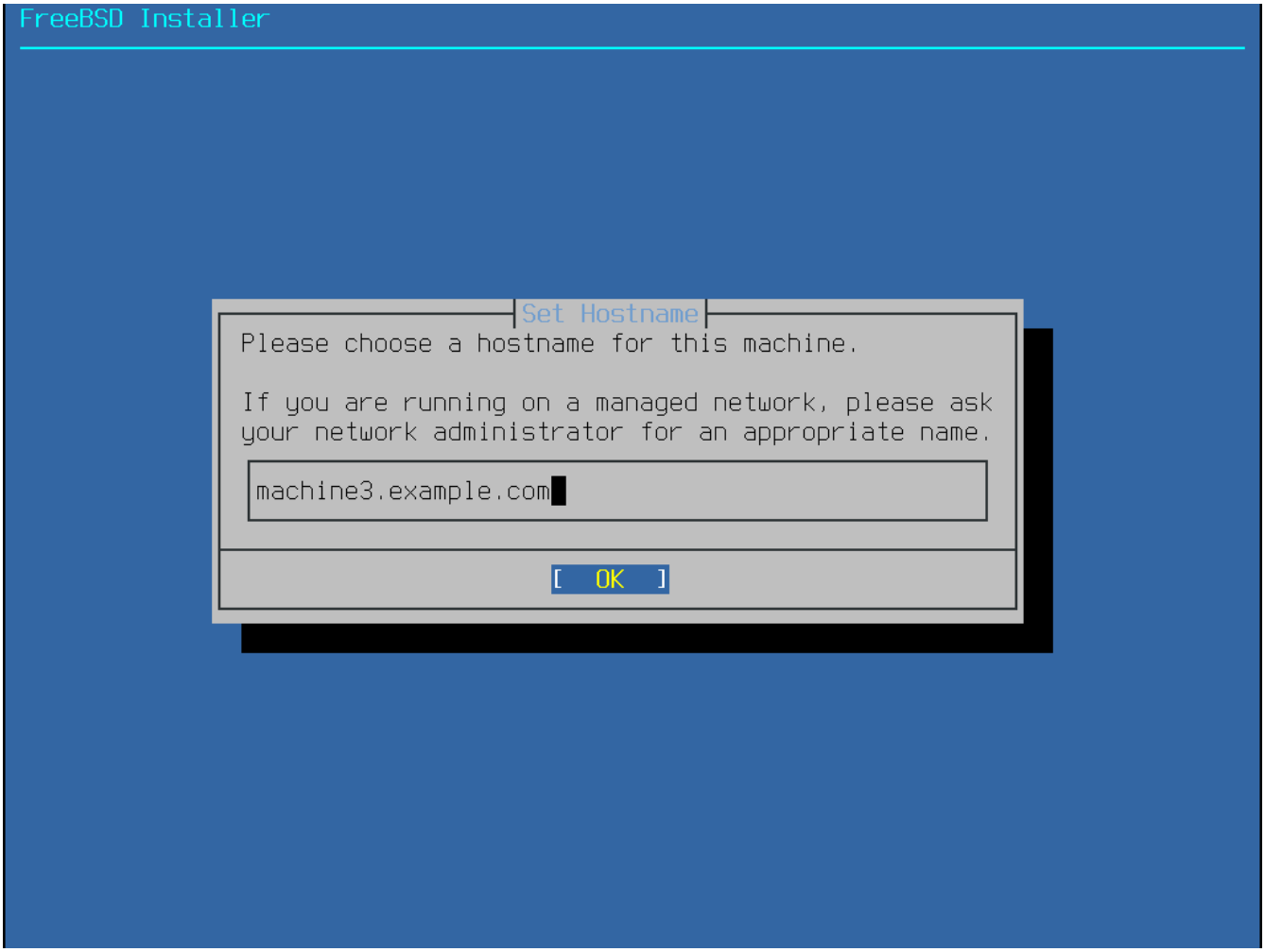


图 63. 设置主机名

应该输入完整的主机名， 例如 machine3.example.com。

3.5.3. 选择要安装的组件

下面， bsdinstall 将提示选择要安装的组件。

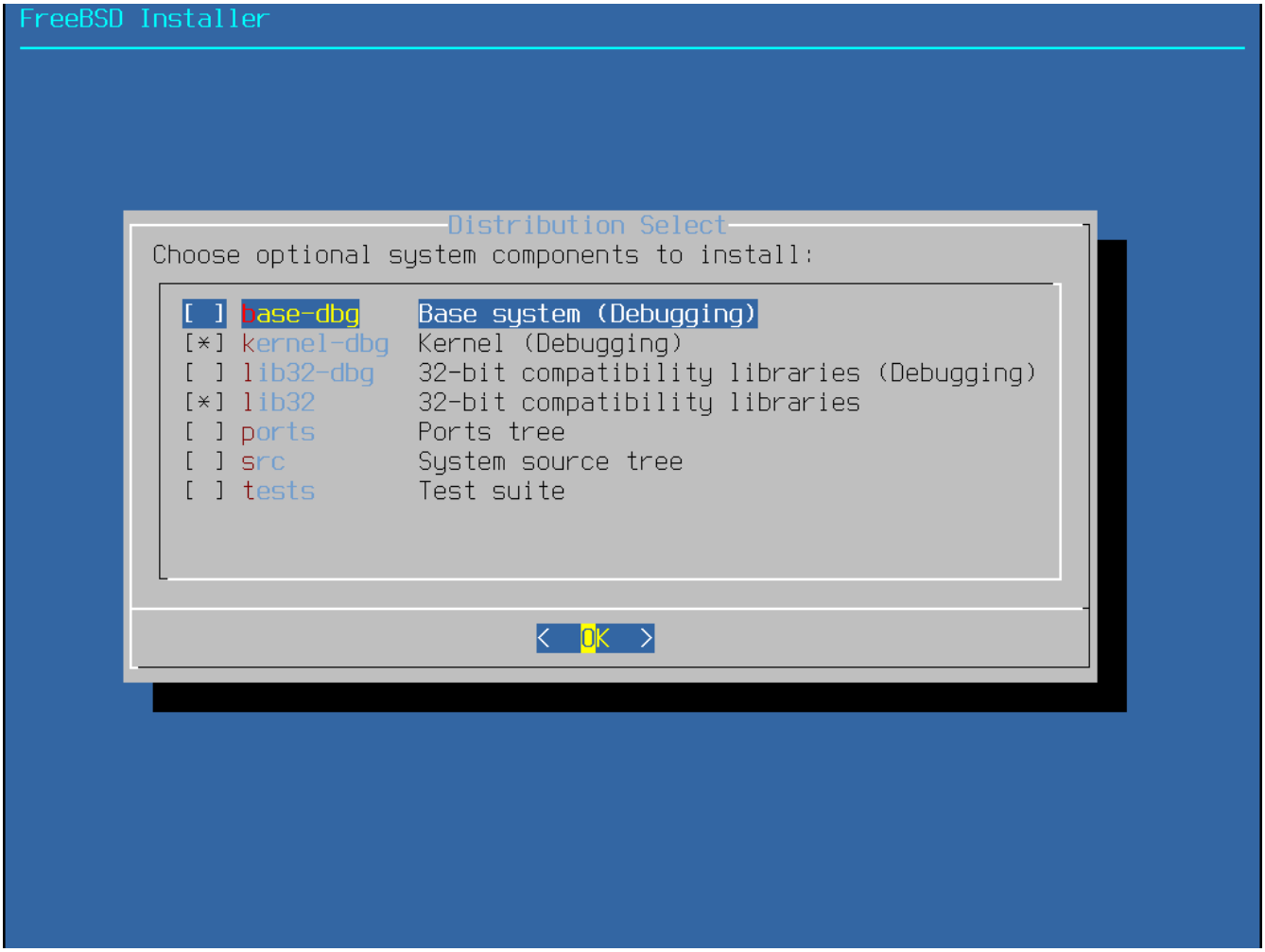


图 64. 选择要安装的组件

安装哪些组件很大程度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（统称"基系统"）。

根据安装类型的不同， 某些组件可能不会显示。

可选组件

- doc - 附加文档， 主要是与项目历史相关的内容。 稍后还可以安装 FreeBSD 文档计划所提供的文档。
- games - 一些传统的 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 集提供了一种简单而方便的途径来安装软件。 在 ports 集中， 并不包含编译软件所需的源代码， 取而代之的是一组能够自动下载、 编译并安装第三方软件包的文件。 安装应用程序. Packages 和 Ports 会讲述如何使用 ports 集。

  选择此项时， 必须保证有足够的硬盘空间， 注意安装程序并不会对此进行检查。 FreeBSD 9.0 的 ports 集约需 3 GB 的磁盘空间； 您也可以为稍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， 它们主要用于联编特定软件（例如设备驱动或内核模块）或者 FreeBSD 本身的开发。

  完整的源代码树需要 1 GB 的磁盘空间， 而重新编译整个 FreeBSD 系统则额外还需要 5 GB 的空间。

3.6. 通过网络安装

bootonly 安装介质中并不会包含所有的安装文件。 如果使用这种介质进行安装，
那么需要的文件就必须通过网络下载。

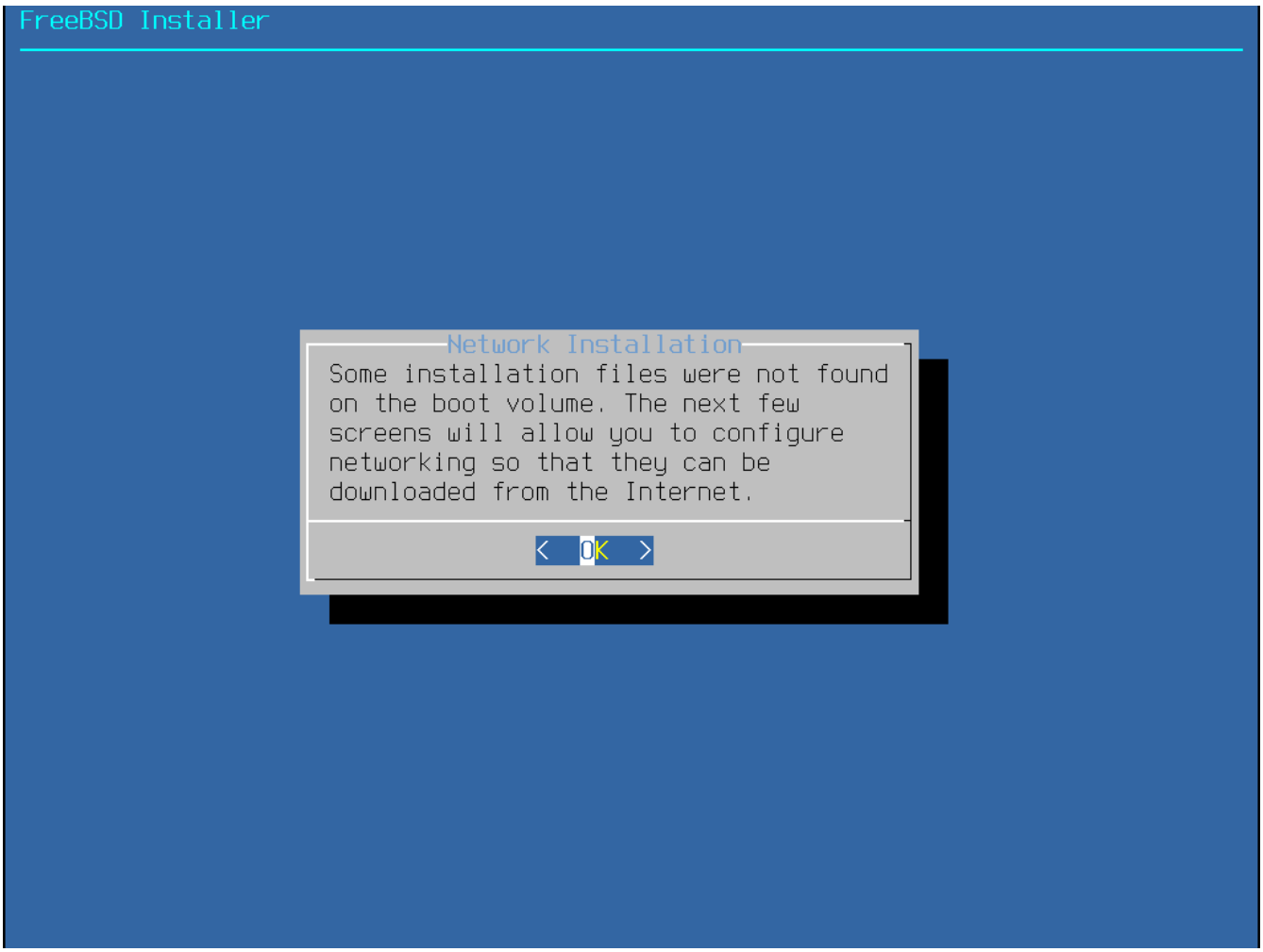


图 65. 通过网络安装

根据 配置网络接口 配置了网络连接后， 即可开始选择像站点。 镜像站点上缓存有 FreeBSD 的安装文件，
选择一个更近的镜像站点有助于更快的获取这些文件， 从而减少安装时间。

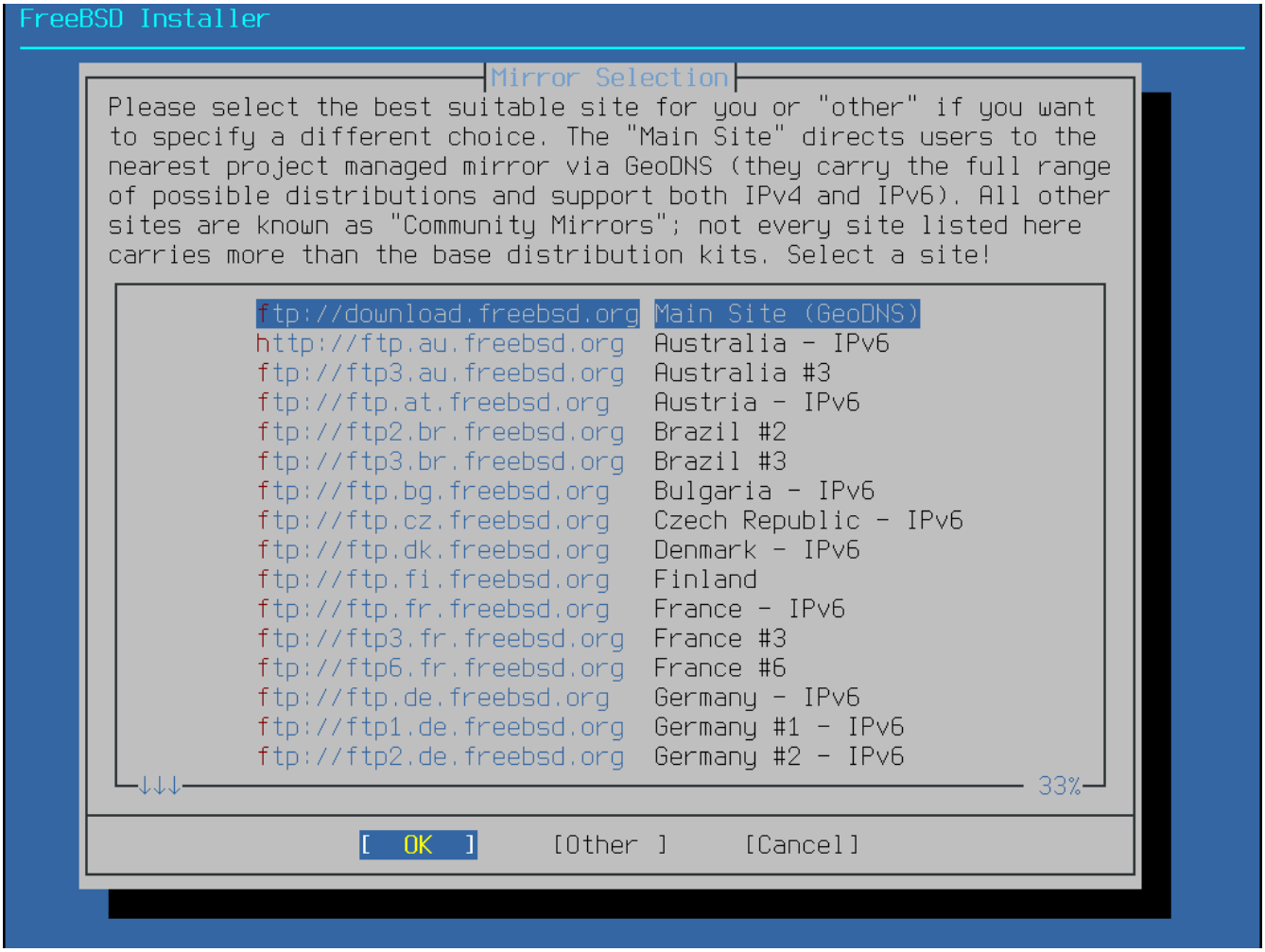


图 66. 选择一个镜像站点

连接至所选镜像站点并查询到所需文件后， 安装将继续进行。

3.7. 分配磁盘空间

FreeBSD 提供了三种方式来分配磁盘空间： Guided（向导式） 分区能够自动设置磁盘分区； 而 Manual（手动式） 分区则允许高级用户创建自定义分区； 还可以进入 shell 中直接使用类似 gpart(8)、fdisk(8) 与 bsdlabel(8) 这样的命令行程序。

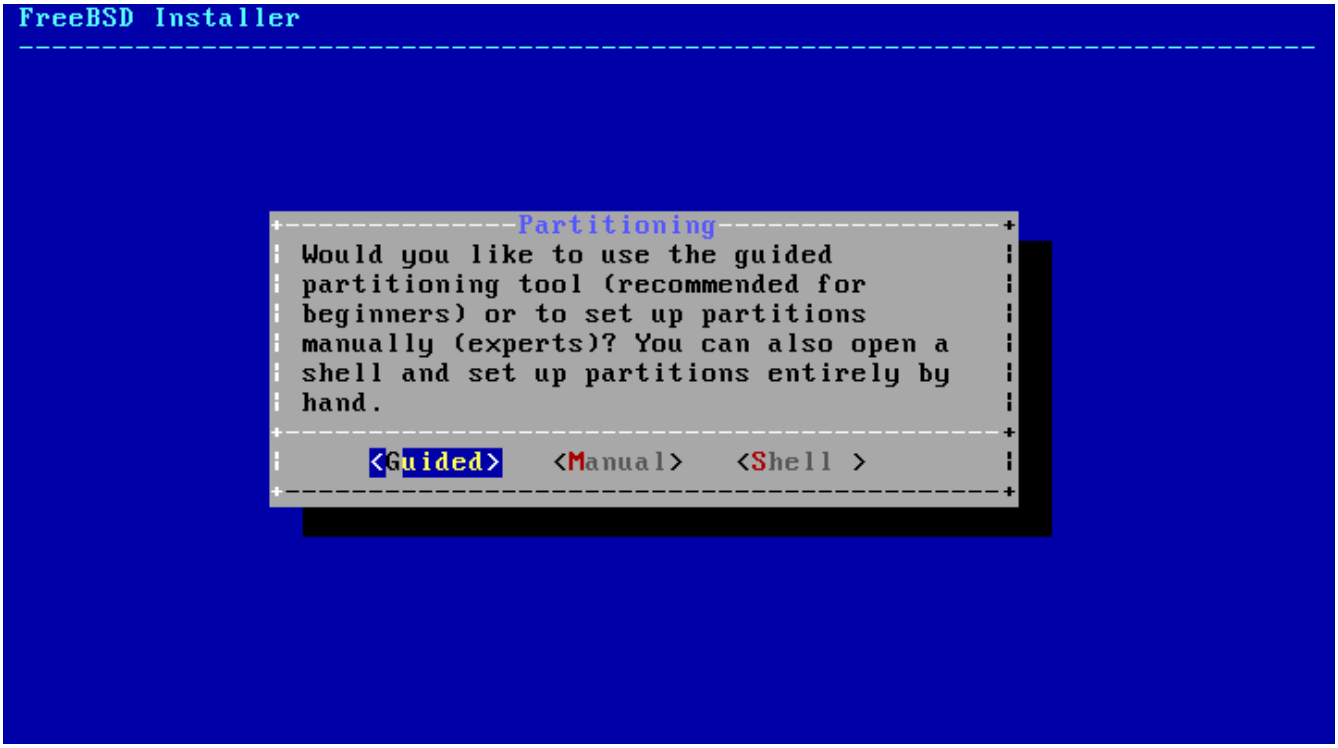


图 67. 选择分配磁盘空间的方式

3.7.1. 向导式分区

如果机器上配有多块磁盘， 则需要为 FreeBSD 的安装指定目标磁盘。

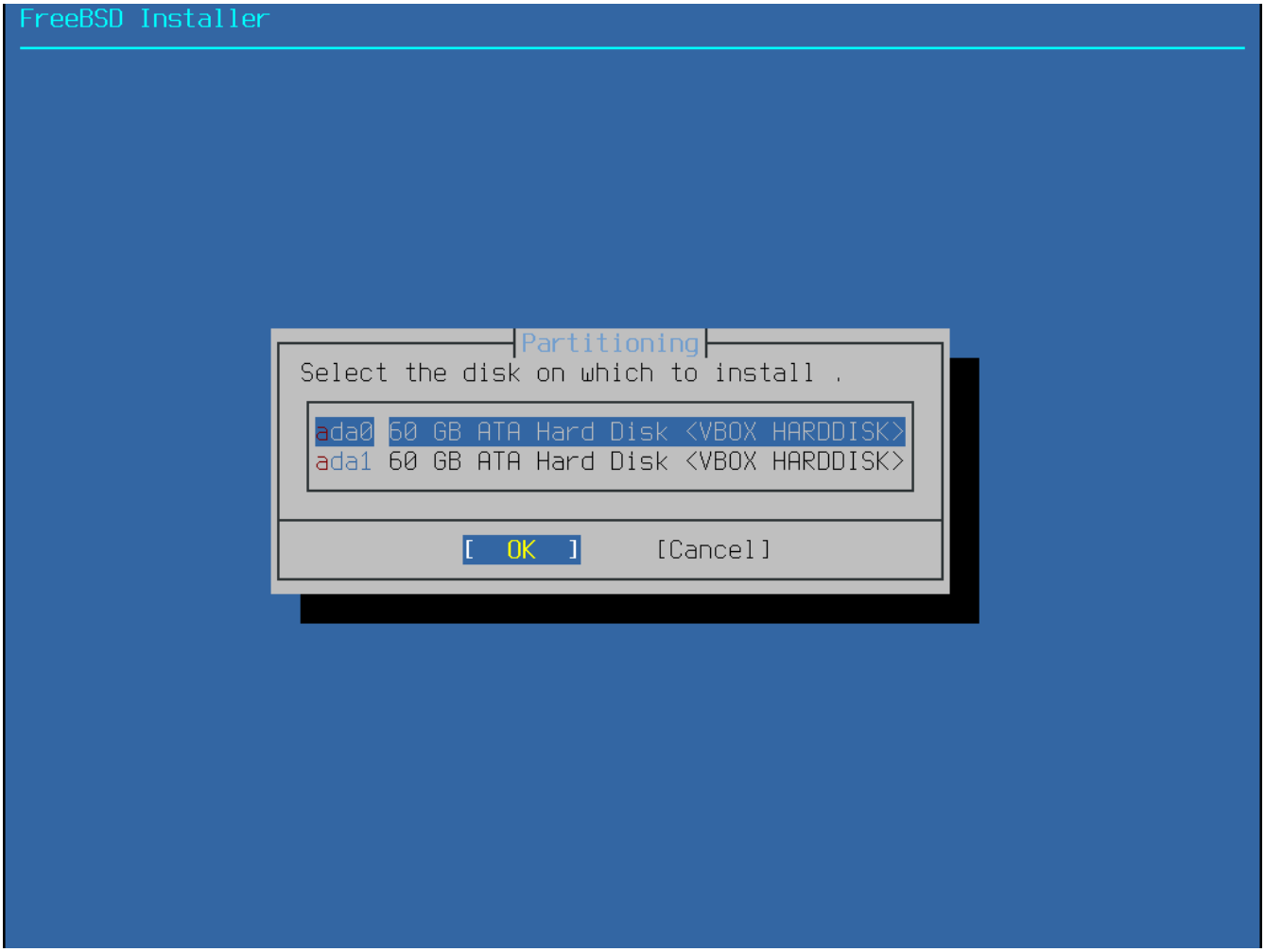


图 68. 从多块磁盘中进行选择

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]，
则创建分区布局时会直接使用整个磁盘； 若选择的是 [Partition]，
则创建分区时仅会使用磁盘上的空闲空间。

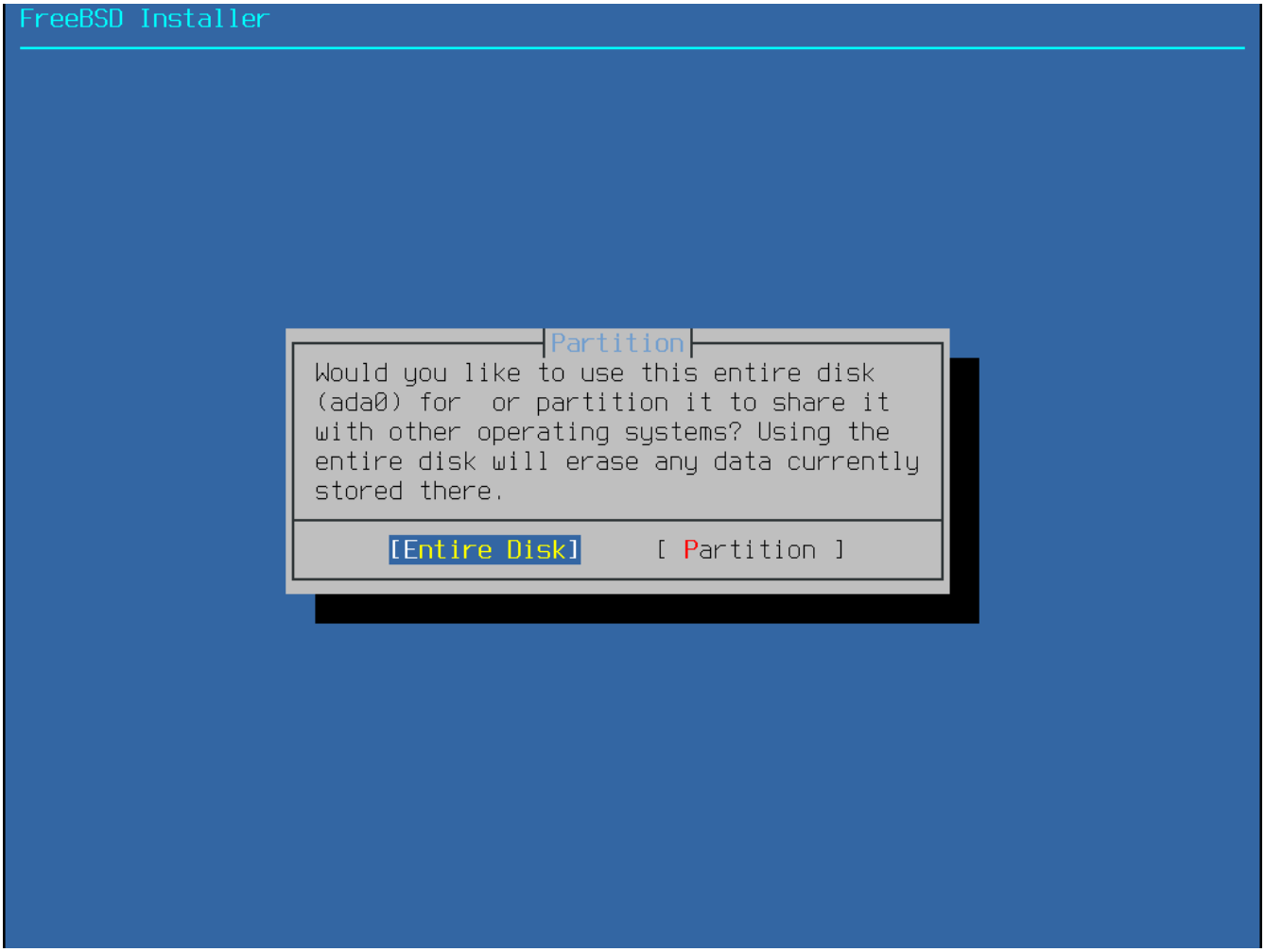


图 69. 选择如何创建分区布局

请仔细检查分区布局的创建结果。 如果发现有错误之处， 可以选择 [Revert] 来还原之前的分区； 此外，
也可以选择 [Auto] 重新让 FreeBSD 自动创建分区。 也可以手动创建、 修改或删除分区。
正确创建了分区之后， 请选择 [Finish] 以继续安装。

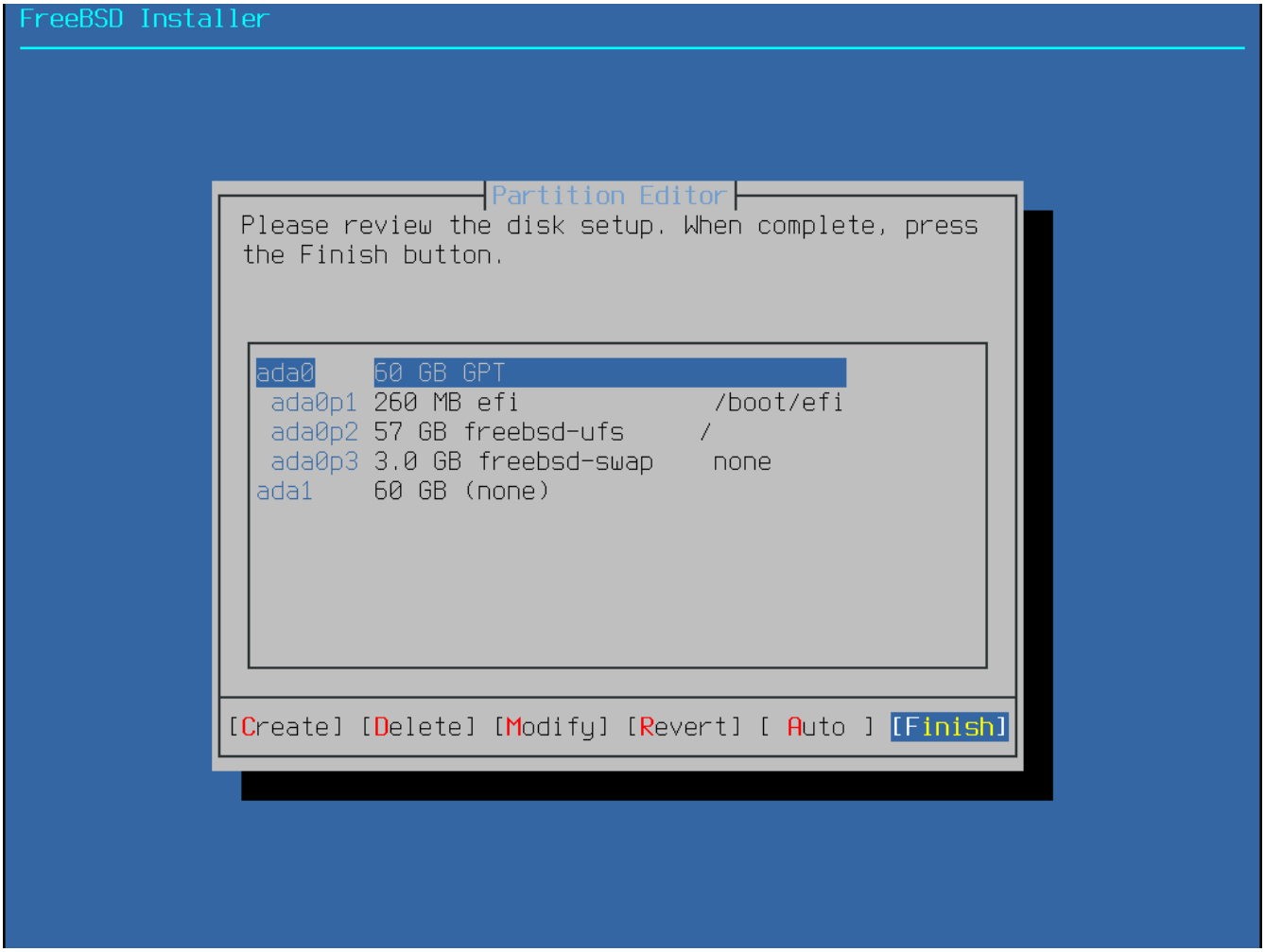


图 70. 检查已创建分区

3.7.2. 手动式分区

手动式分区将直接使用分区编辑器进行操作。

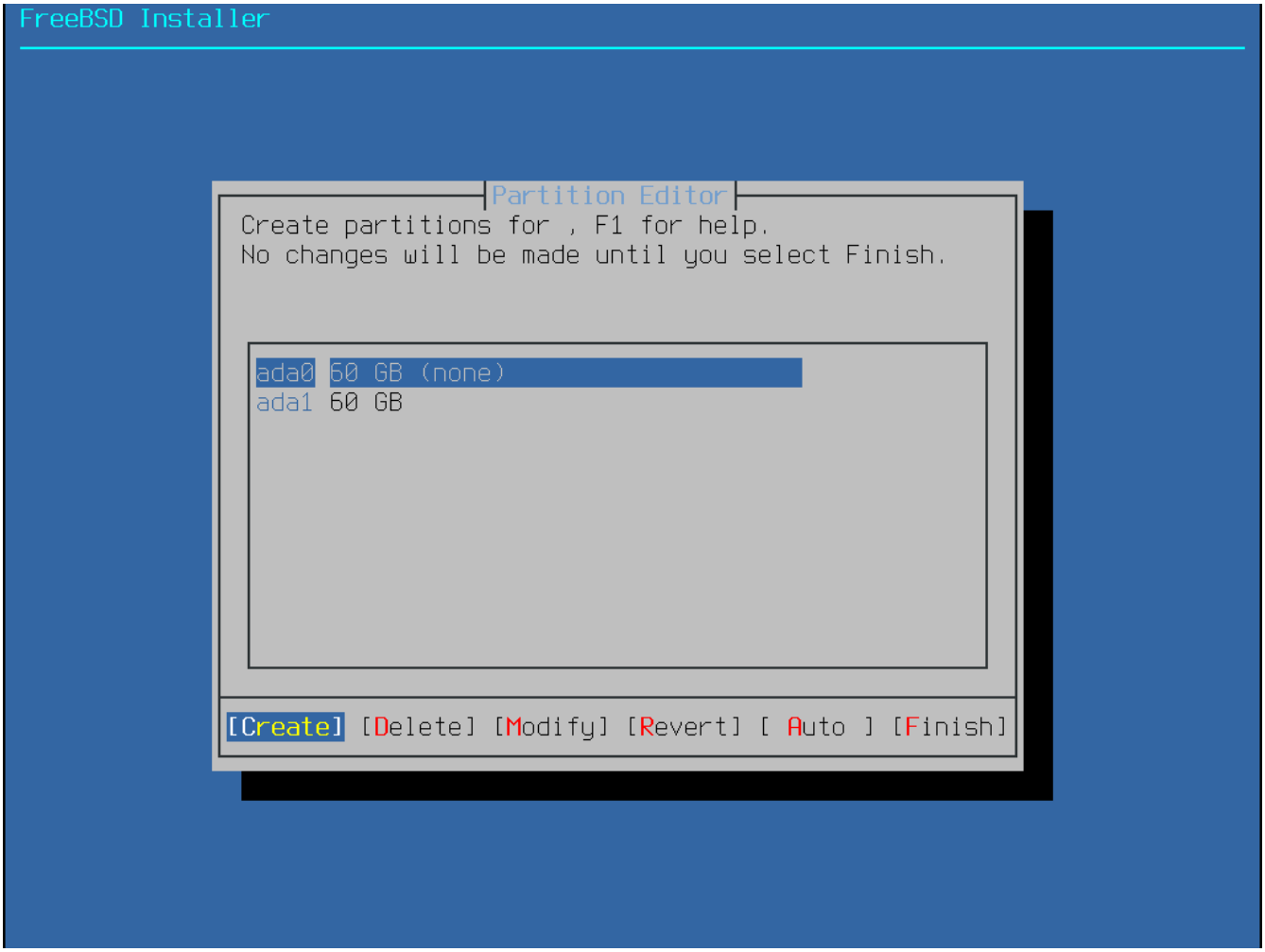


图 71. 手动创建分区

高亮目标驱动器（本例中为 ada0）并选择 [Create] 以显示 partitioning scheme（分区方案） 菜单。

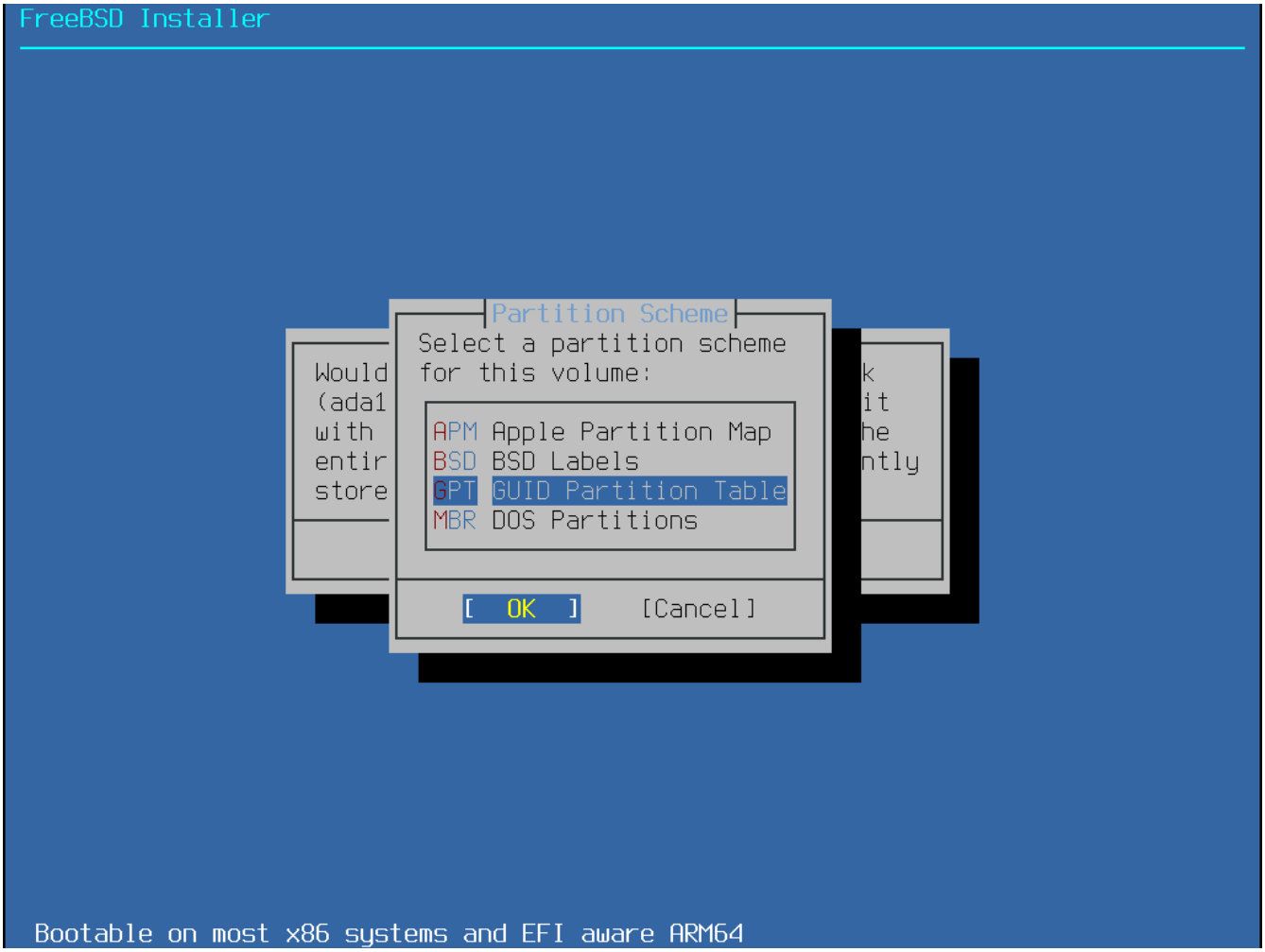


图 72. 手动创建分区

对于 PC 兼容机来说， GPT 分区通常是最合适的选择， 而某些不兼容 GPT 的老式操作系统则可能需要使用 MBR 分区。 除此之外的分区方案仅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	不带 MBR 的 BSD Label， 有时也称作危险的专用模式， “dangerously dedicated mode” 。 请参阅 bsdlabel(8)。
GPT	GUID 分区表。
MBR	Master Boot Record， 主引导记录。
PC98	MBR 变体， 用于 NEC PC-98 计算机。
VTOC8	Volume Table Of Contents， 用于 Sun SPARC64 和 UltraSPARC 计算机。

确定了分区方案并创建完成后， 可再次选择 [Create] 以创建新的分区。

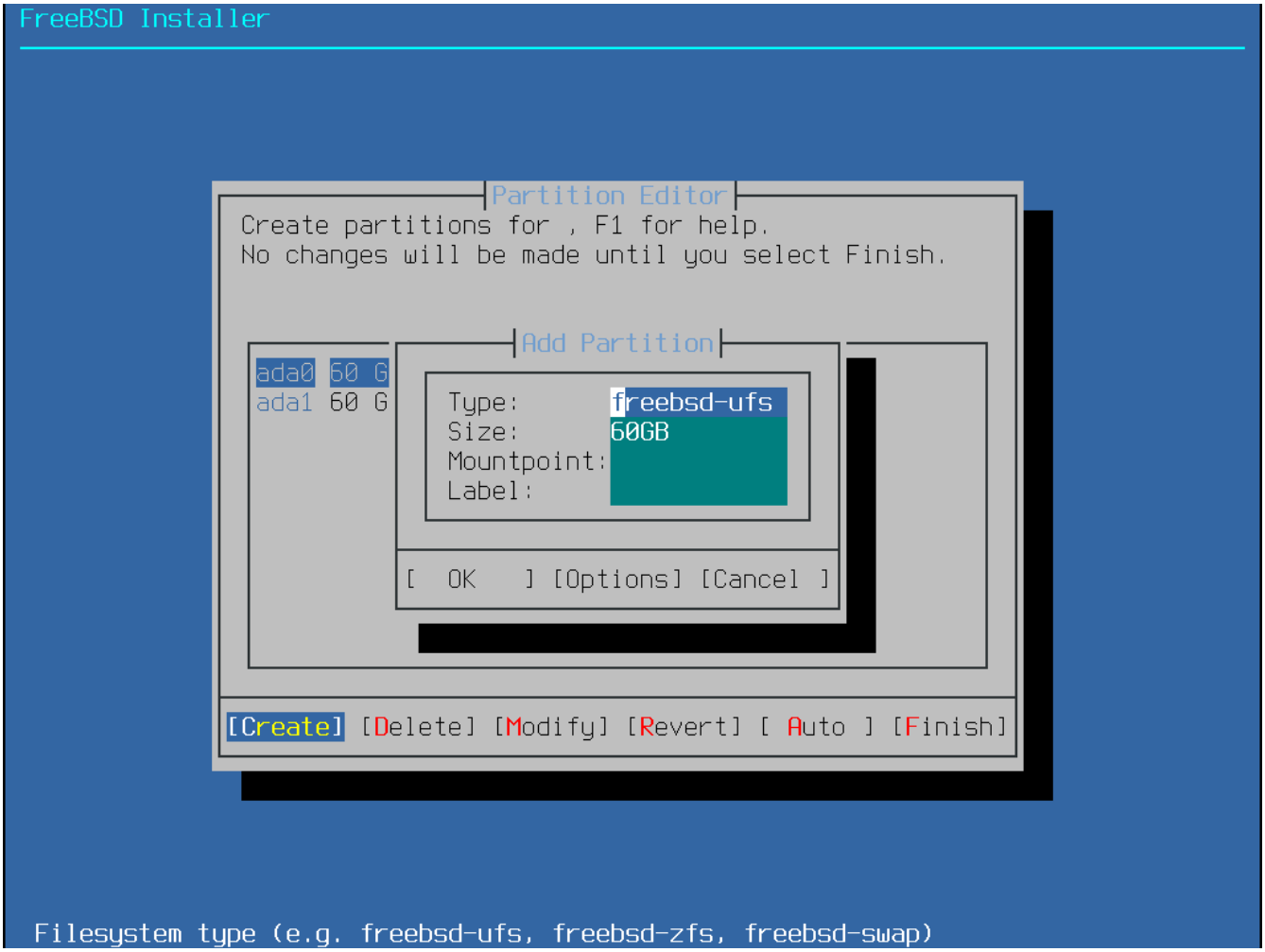


图 73. 手动创建分区

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， 它必须处于首位。
- 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。 有些用户会喜欢传统的分区格局， 为 /、 /var、 /tmp， 以及 /usr 文件系统分别创建分区。 请参阅 创建传统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。

正确地对齐磁盘扇区能够获取最佳性能。 无论磁盘的每个扇区为 512 字节还是 4K 字节， 将分区大小设置为 4K 字节的倍数都能够确保对齐。 实际操作中， 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， 目前由于引导代码所限， 此分区不能大于 512K。

若分区包含文件系统，则需要在 Mountpoint 项中为其输入挂载点； 若仅创建了一个 UFS 分区， 则应在此项中输入 /。

最后需要输入的是 Label（标签） 项， 用于命名所创建的分区。
如果将驱动器连接至不同的控制器或端口， 其名称或编号会发生改变， 但对应的标签并不会变化。 在类似 /etc/fstab 这样的文件中， 通过标签引用分区比通过驱动器名加分区编号引用更加灵活，
因为这样引用使系统对硬件的改变更加宽容。 GPT 的标签会在磁盘连接后出现在 /dev/gpt/ 中；
而其他分区方案中的标签也有不同的功能， 它们会出现在 /dev/ 中的不同目录里。

为避免冲突， 请给每个文件系统指定独一无二的标签。 与计算机的名称、用途或位置相关的字符均可添加至标签。 例如， 实验室计算机的 UFS 根目录可以命名为 “labroot” 或 “rootfs-lab”。

例 5. 创建传统的分割式文件系统分区

在传统的分区布局中， 目录 /、 /var、 /tmp 及 /user 都是位于自己分区上的独立文件系统； 在 GPT 分区方案中也可以创建这样的分区布局。 本例中所使用的是一块 20G 的硬盘， 如果使用更大的硬盘，建议创建更大的交换或 /var 分区。 标签的前缀 ex 是指 “example”，具体操作时您可以使用任何独一无二的字符。

分区类型	大小	挂载点	标签
freebsd-boot	512K		
freebsd-ufs	2G	/	exrootfs
freebsd-swap	4G		exswap
freebsd-ufs	2G	/var	exvarfs
freebsd-ufs	1G	/tmp	extmpfs
freebsd-ufs	接受默认值（剩余空间）	/usr	exusrfs

创建了自定义分区后， 请选择 [Finish] 以继续安装。

3.8. 安装确认

下面， 安装程序将真正对硬盘进行写操作， 这也是取消安装的最后机会。

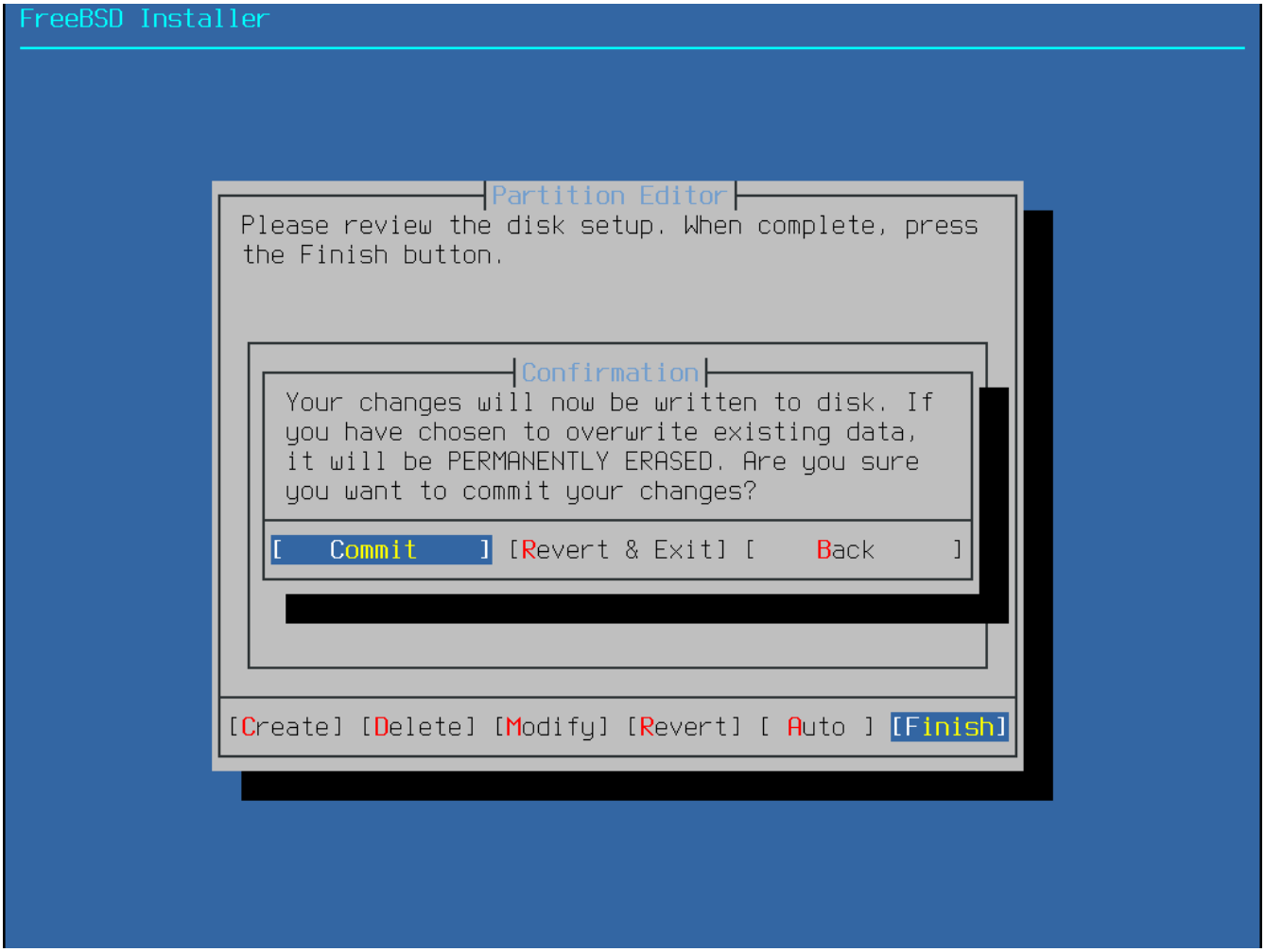


图 74. 最后确认

选择 [Commit] 并按 Enter 键确认安装； 选择 [Back] 以返回分区编辑器进行修改； 选择 [Revert & Exit]以退出安装而不修改任何硬盘数据。

根据所选组件、 安装介质和机器速度的不同， 需要的时间会有所变化。
安装时会有一系列信息显示目前的进度。

首先， 安装程序会将分区布局写入磁盘， 并执行 newfs 初始化分区。

如果是通过网络安装， bsdinstall 将根据之前所选的组件下载对应的文件。

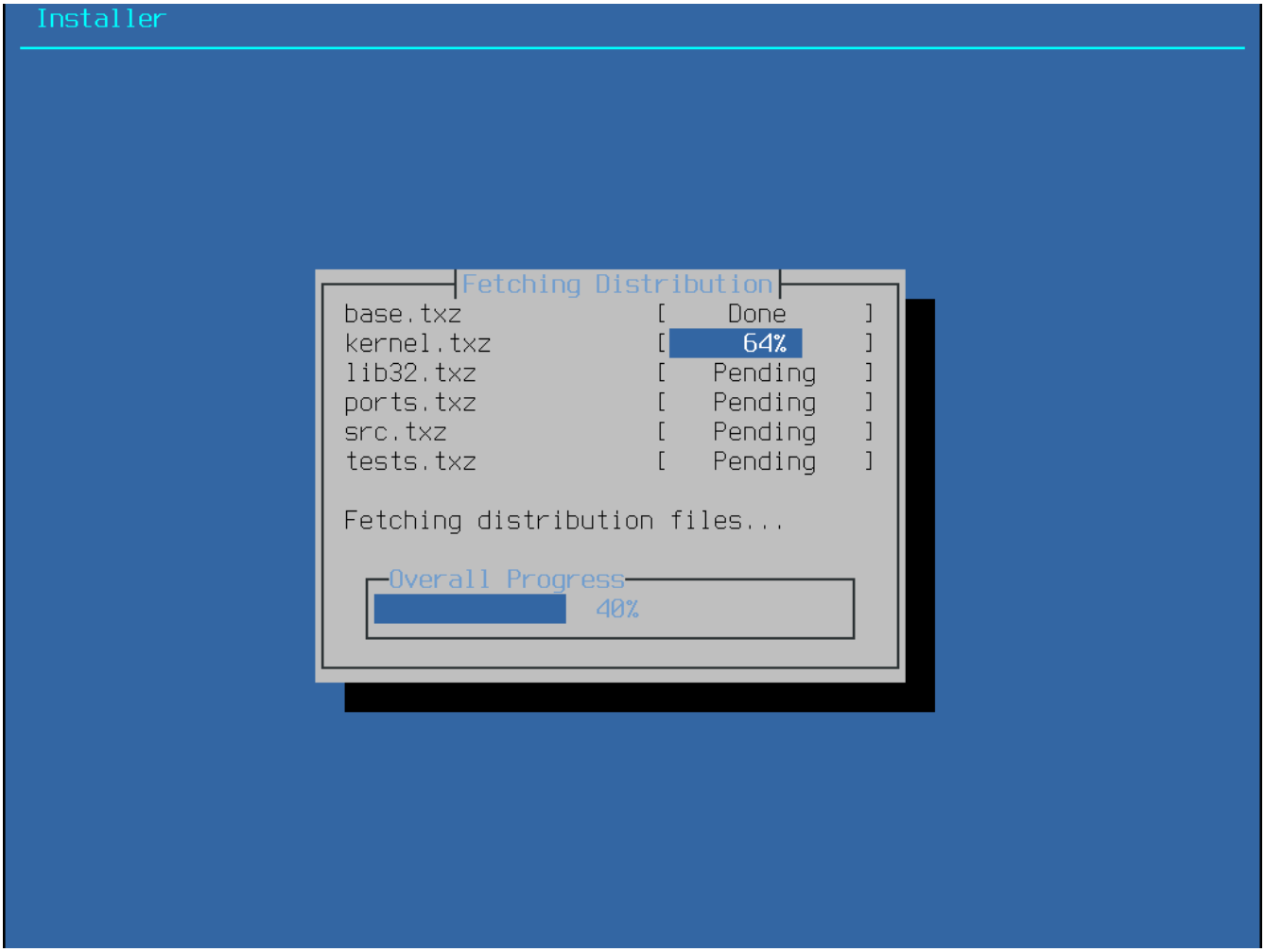


图 75. 获取组件对应的文件

接下来， 会验证这些文件的完整性， 以防止其在下载时损坏或从安装介质中误读。

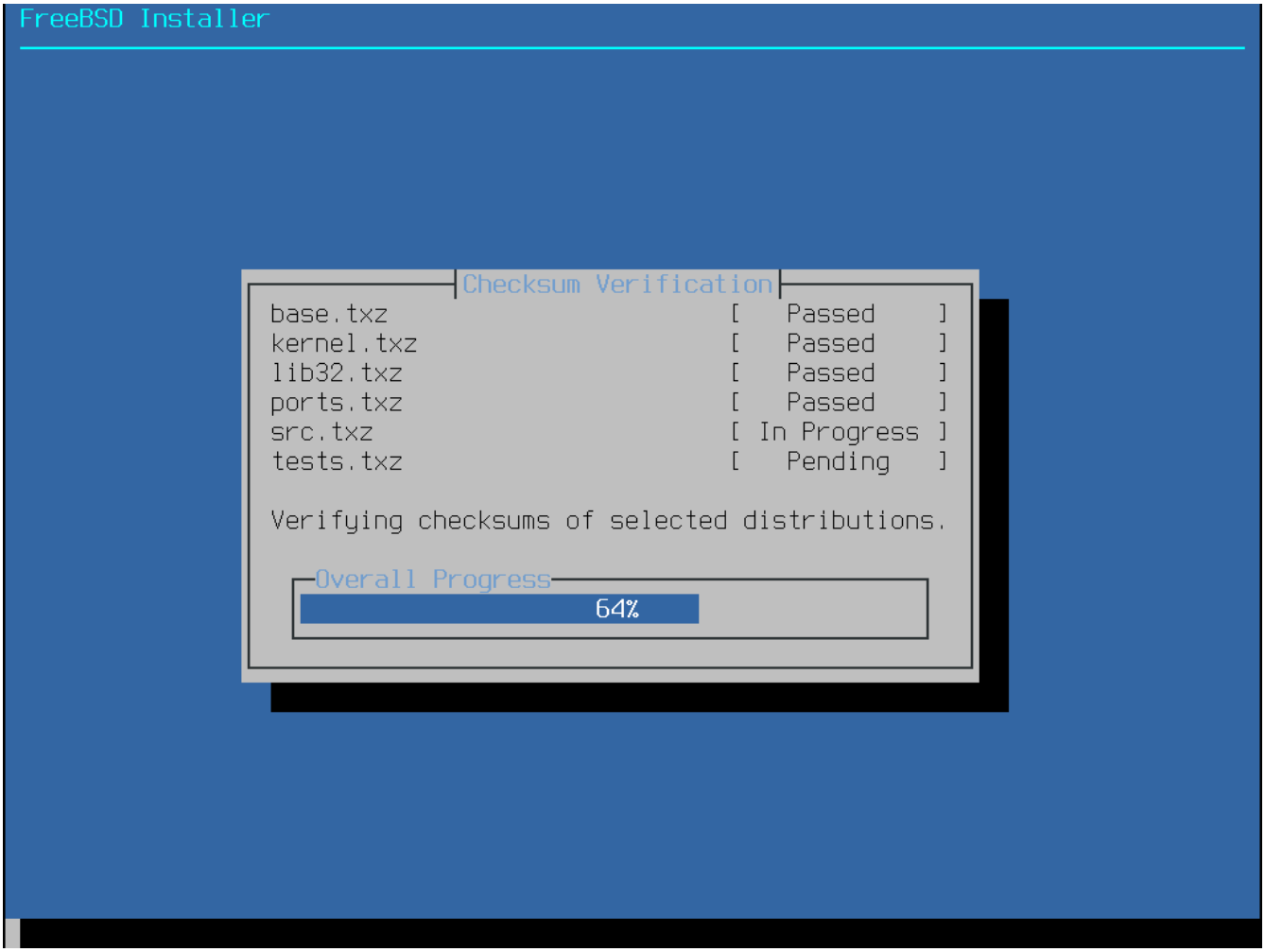


图 76. 验证组件对应的文件

最后， 验证过的组件文件会被提取至磁盘。

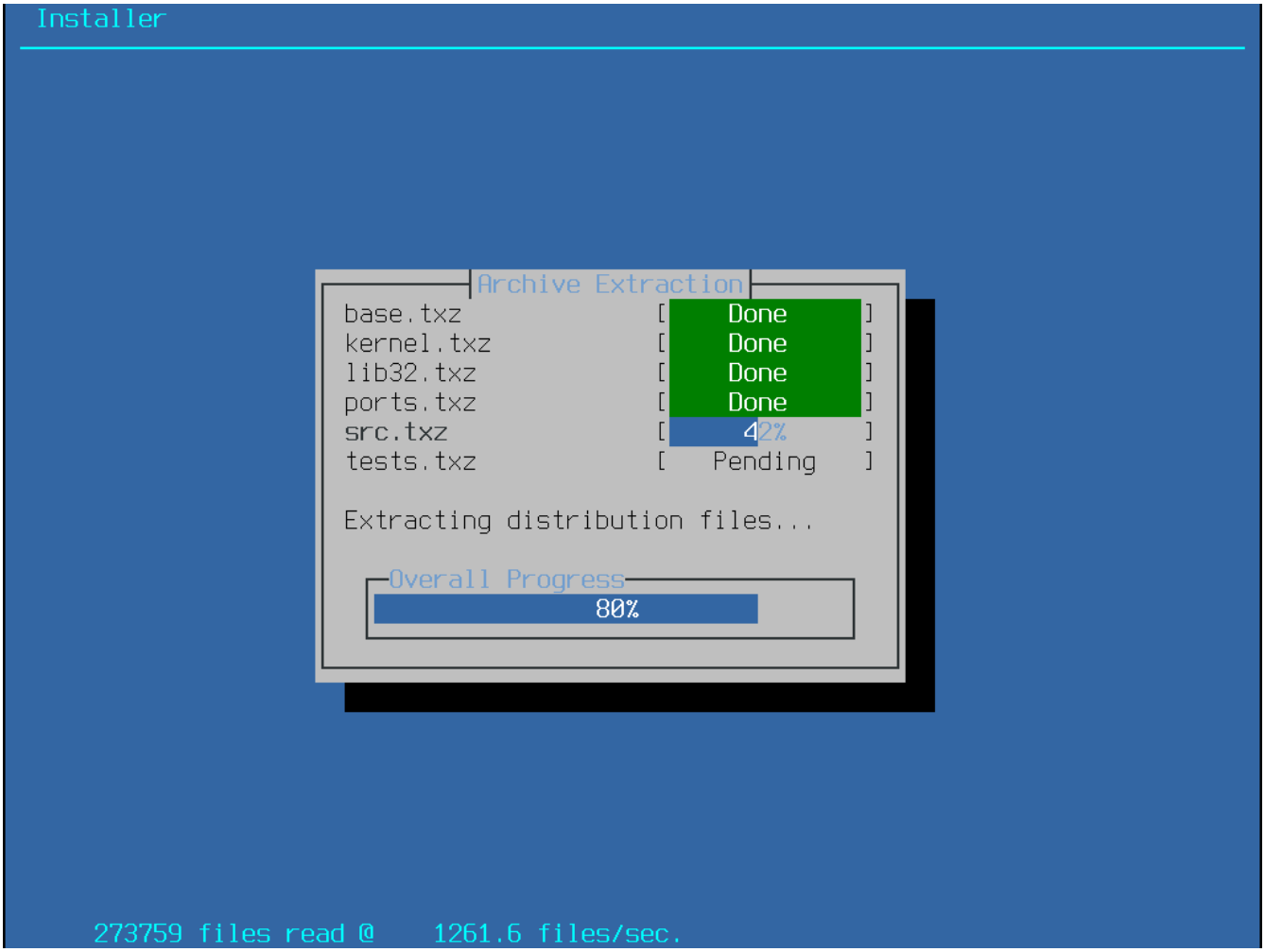


图 77. 提取组件对应的文件

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

3.9. 安装后的配置

成功安装 FreeBSD 后， 还需要依次进行一些配置。 在重启进入新系统前，
这些配置始终可以通过最终的配置菜单进行修改。

3.9.1. 设置 root 密码

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

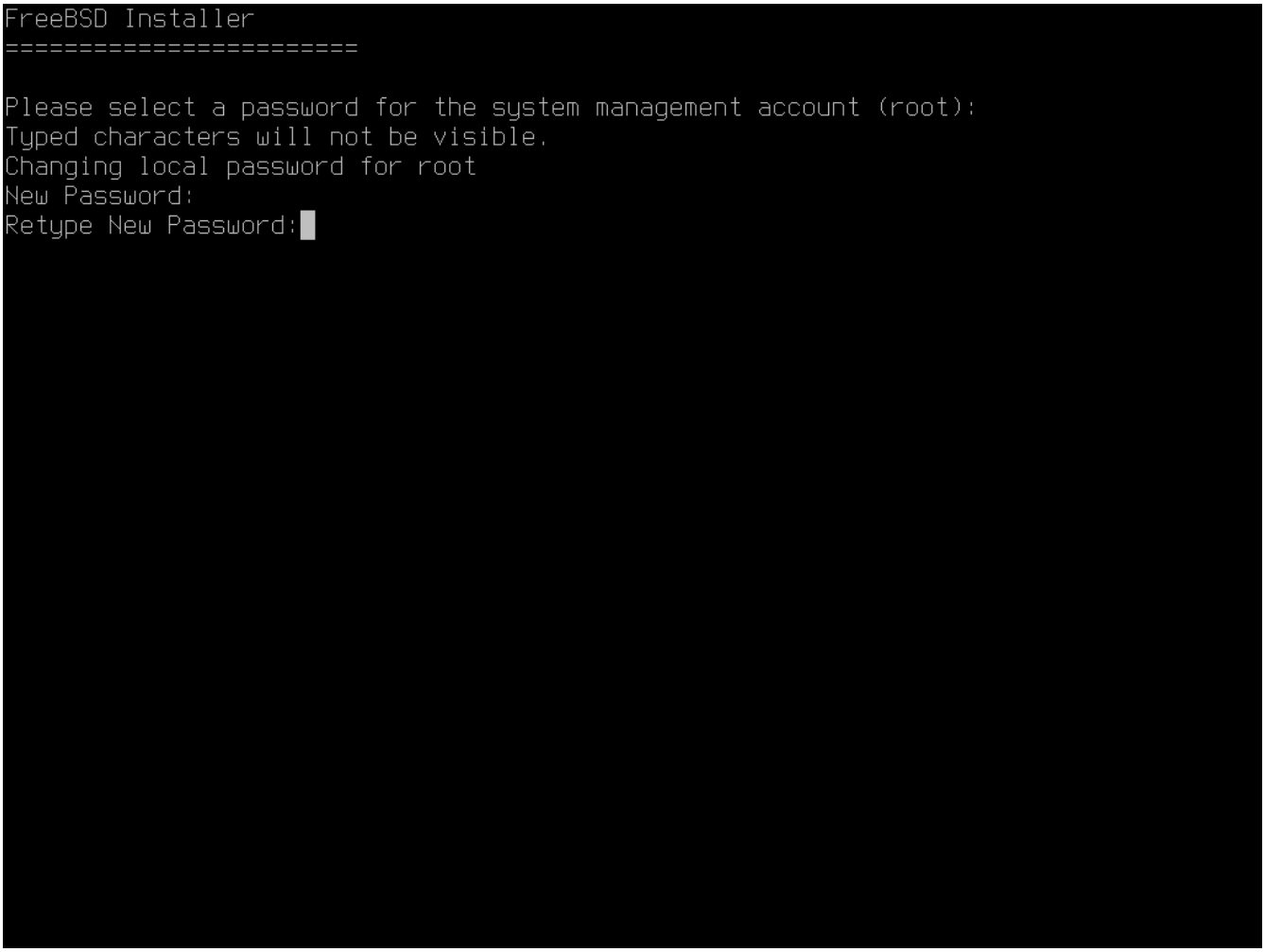


图 78. 设置 root 密码

成功设置密码后， 安装将继续进行。

3.9.2. 配置网络接口

如果已经在 bootonly 安装时配置过网络接口， 则可略过此步。

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侦测到的， 请选择一个进行配置。

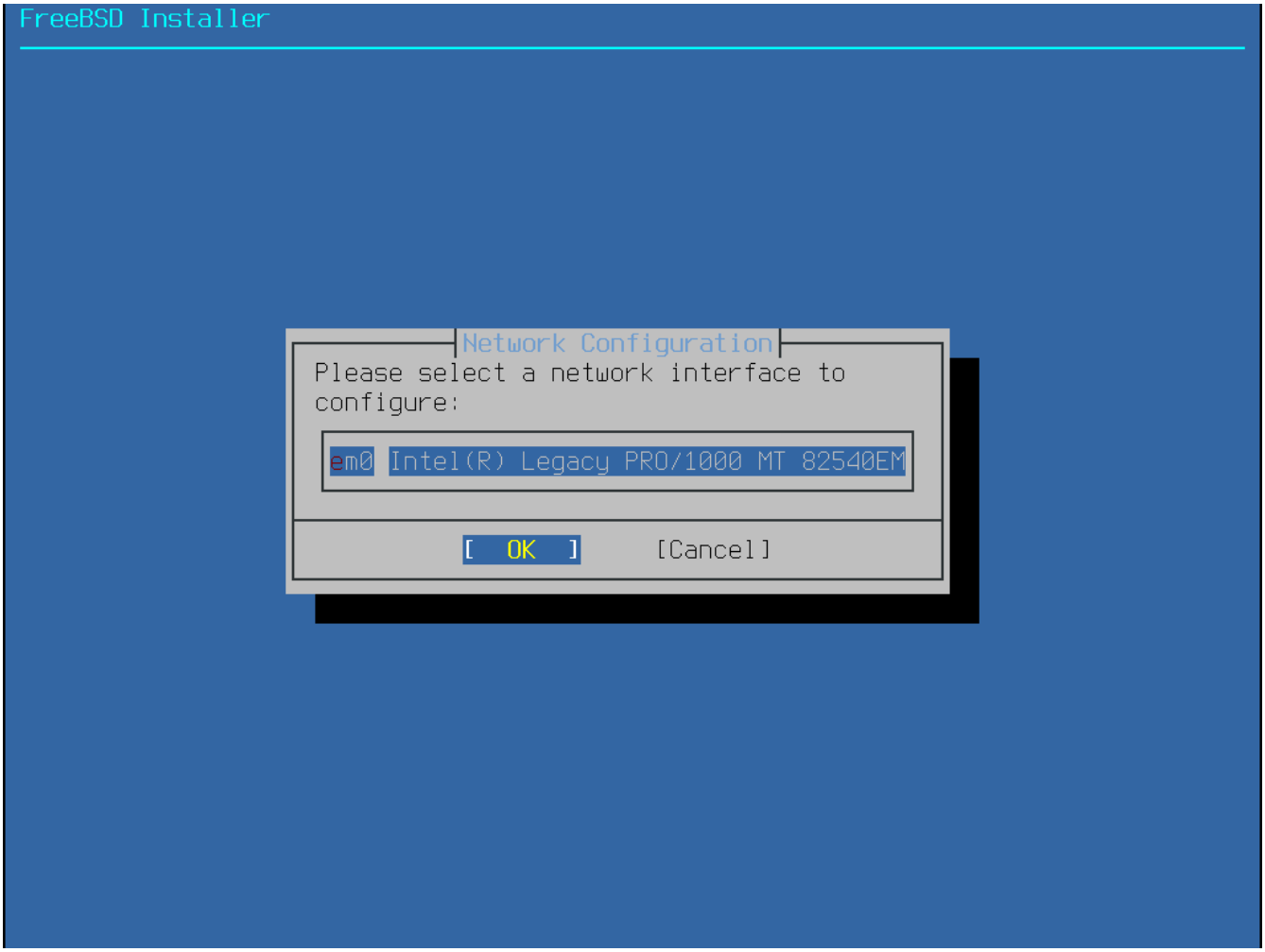


图 79. 选择一个网络接口

3.9.2.1. 配置无线网络接口

如果选择了无线网络接口， 则必须输入相关的无线网络验证及安全参数， 以允许其连接至特定的网络。

无线网络是通过 Service Set Identifier（服务集标识符， 简写为 SSID）来表示的，
它是唯一表示无线网络的短字符串。

大多数无线网络都会以加密方式传输数据， 藉此保护信息不被未经授权者查看。 强烈建议采用 WPA2 加密。 旧式的加密类型， 如 WEP， 几乎没有任何安全性可言。

若要连接至一个无线网络， 首先需要扫描无线接入点。

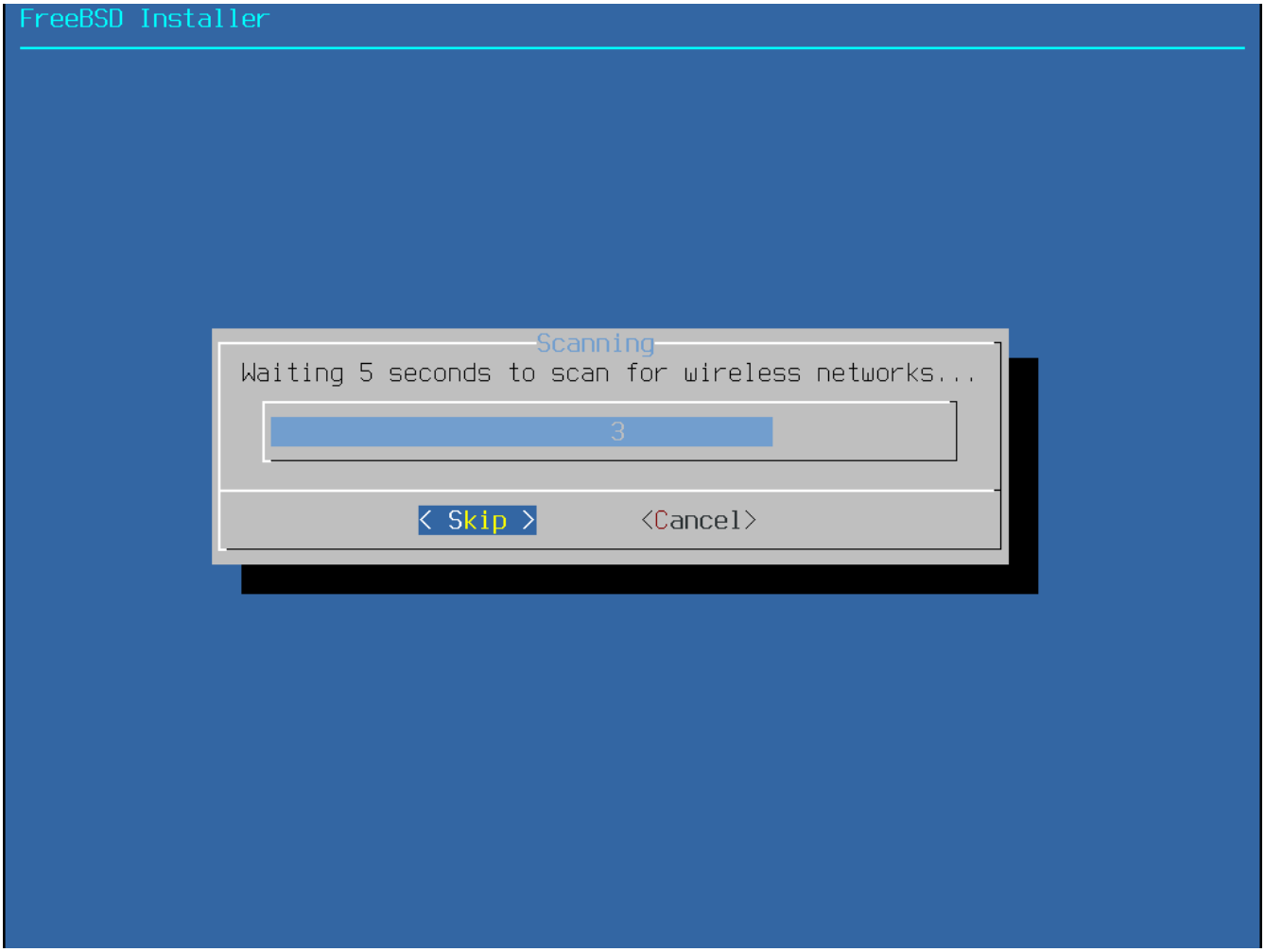


图 80. 扫描无线接入点

扫描完成后， 会列出所有发现的 SSID 以及它们支持的加密类型说明。 如果需要连接的 SSID 没有列出，请选择 [Rescan] 再次扫描。 如果还没有出现， 请检查天线， 或将计算机移至更靠近接入点的地方。 在做过这些改善措施之后， 再重新扫描。

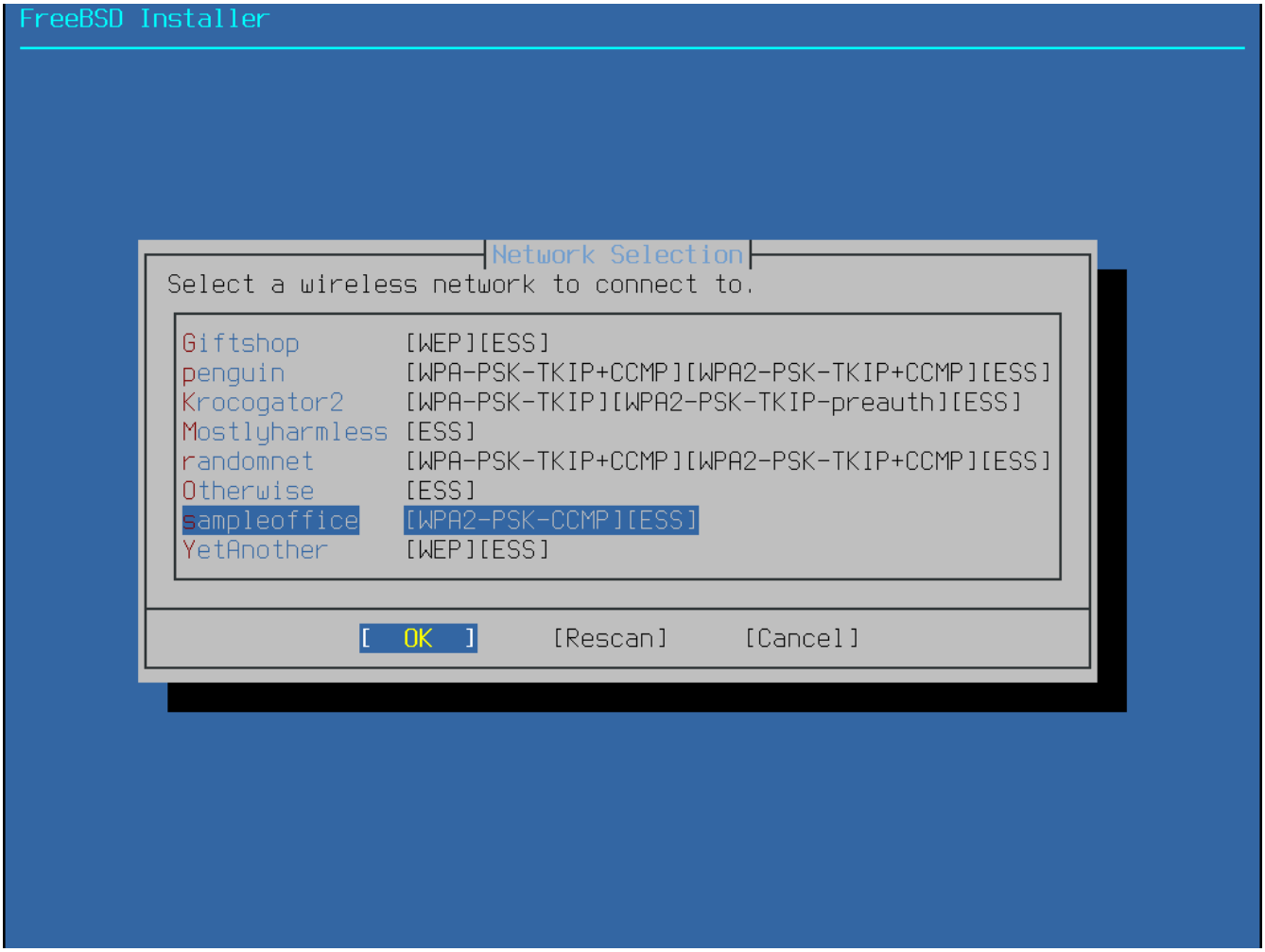


图 81. 选择一个无线网络

选择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（也叫预共享密钥，(简称 PSK）。 为安全起见， 在输入框中键入的字符将显示为星号。

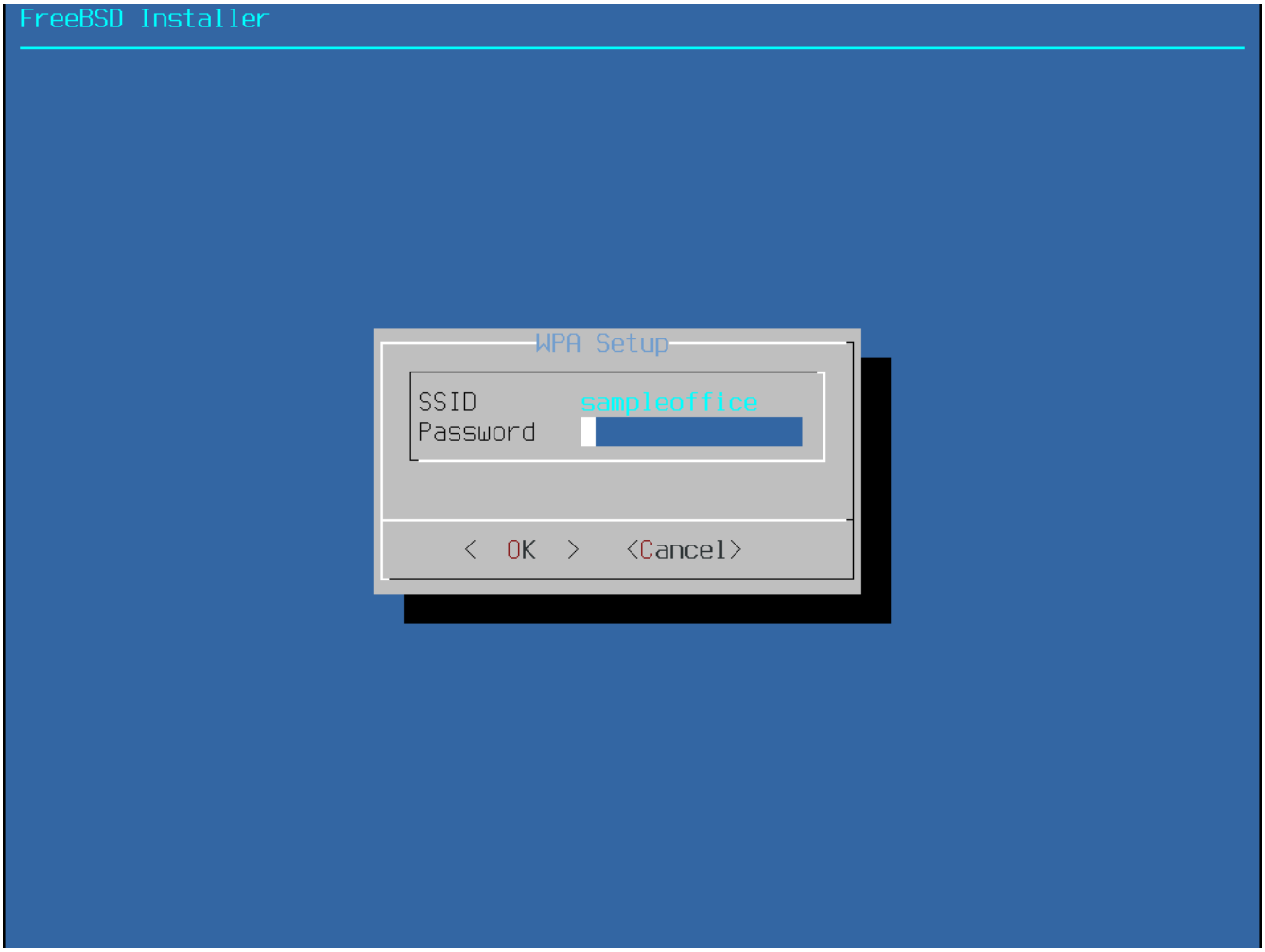


图 82. WPA2 设置

在选择了无线网络并输入了连接所需的信息后， 网络配置将继续进行。

3.9.2.2. 配置 IPv4 网络

选择是否使用 IPv4 网络。 这是最常见的网络连接类型。

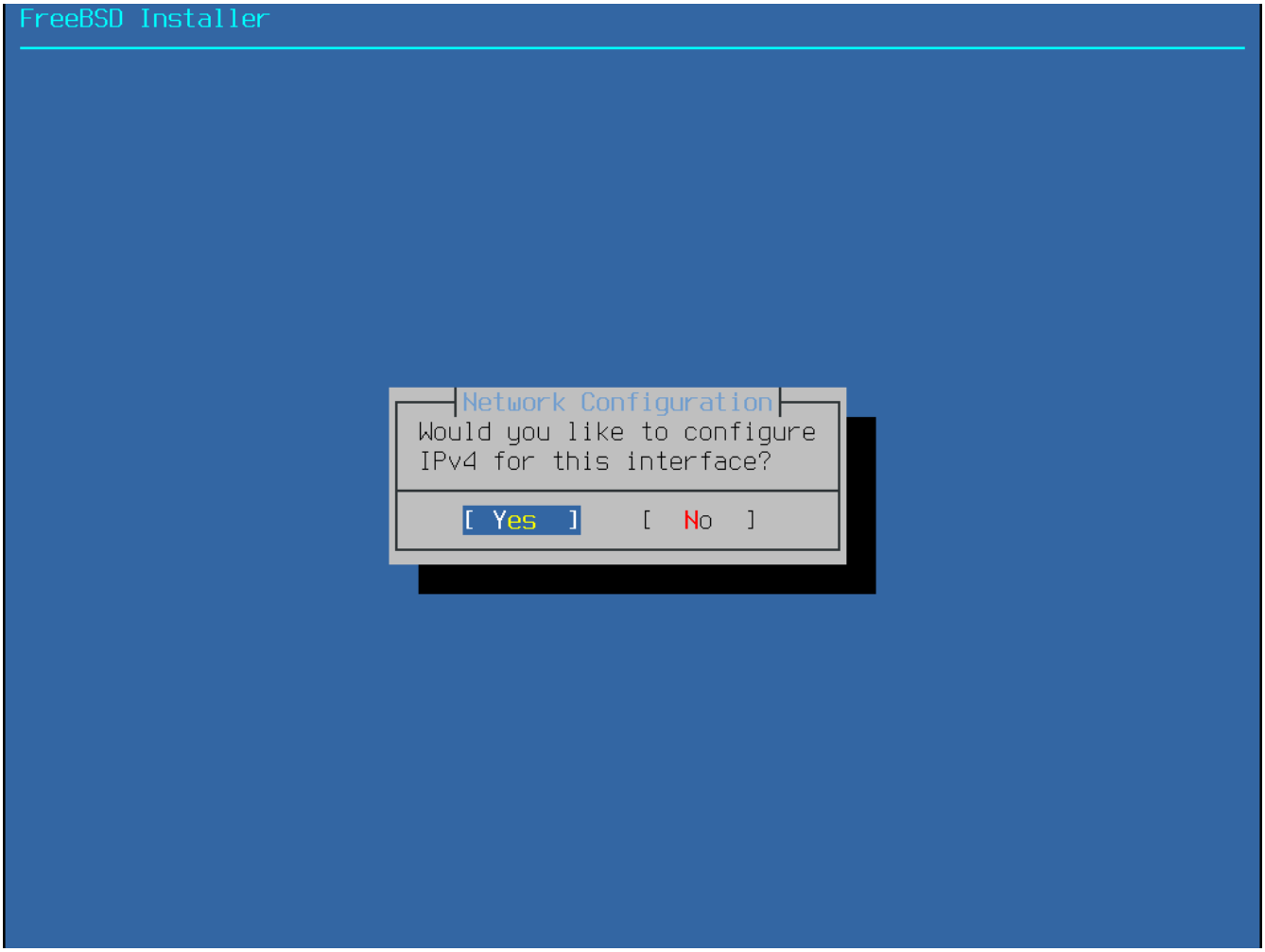


图 83. 选择 IPv4 网络

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， 这是首选的方式。 而 Static （静态） 方式则需要手工输入网络的配置信息。

不要随意输入网络的配置信息， 因为这样的话网络就无法正常工作。
请向网络管理员或服务提供商那里取得 收集网络配置信息 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， 请选择 [Yes] 以自动配置网络接口。

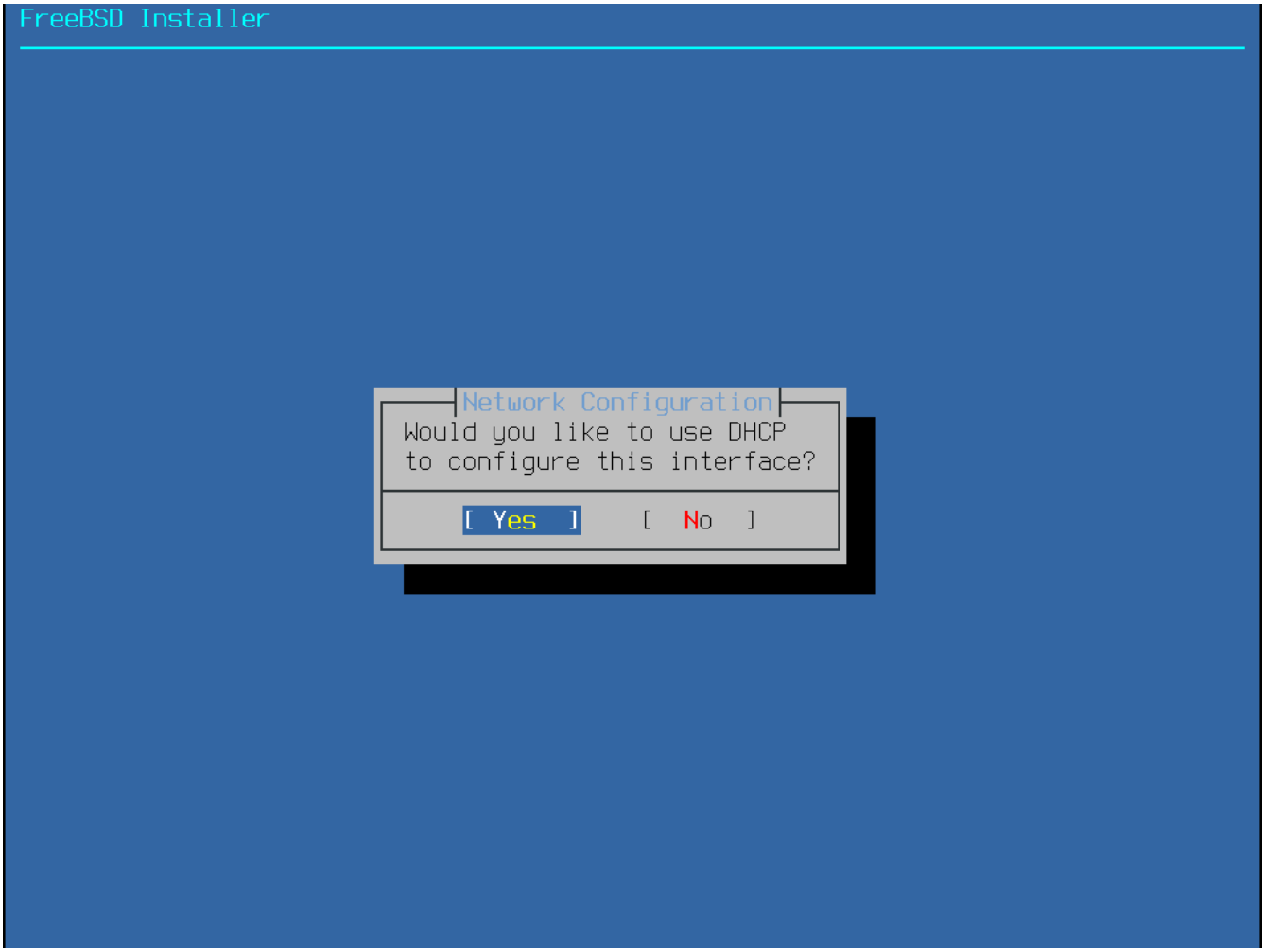


图 84. 选择 DHCP 配置 IPv4

3.9.2.2.2. 使用静态配置方式

网络接口的静态配置需要输入相关的 IPv4 配置信息。

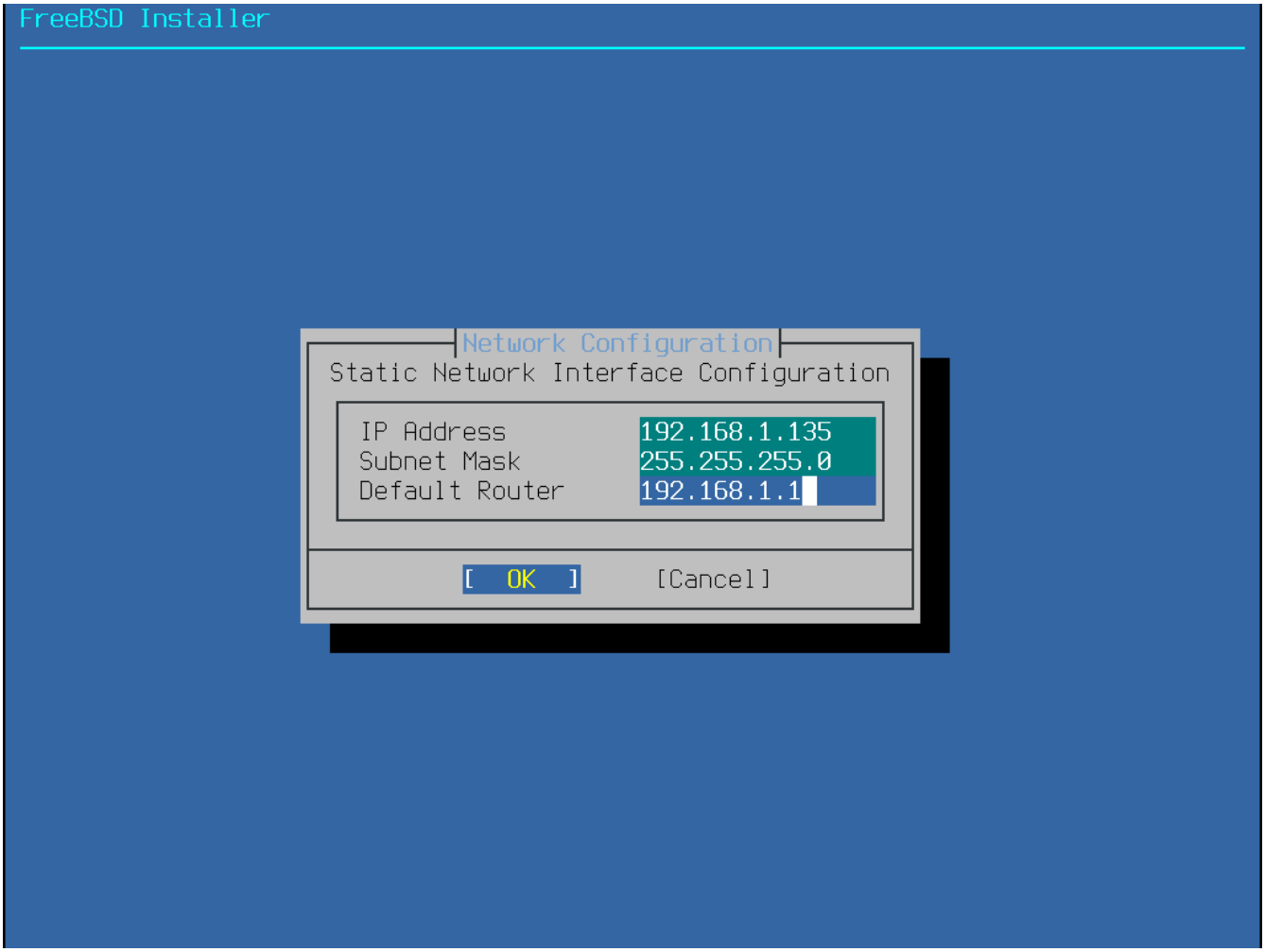


图 85. 静态配置 IPv4

- IP Address - IP 地址， 即给当前计算机手动分配的 IPv4 地址。 此地址必须是唯一的，并且在本地网络上还没有被其他设备使用。
- Subnet Mask - 子网掩码， 用于本地网络。 通常是 255.255.255.0。
- Default Router（默认路由） - 网络上默认路由的 IP 地址。 通常， 这是将本地网络连接至 Internet 的路由器或其他网络设备的地址。 也称作 default gateway （默认网关）。

3.9.2.3. 配置 IPv6 网络

IPv6 是一种新的网络配置方式。 如果您有可用的 IPv6 连接， 并需要使用它， 选择 [Yes] 来开始配置。

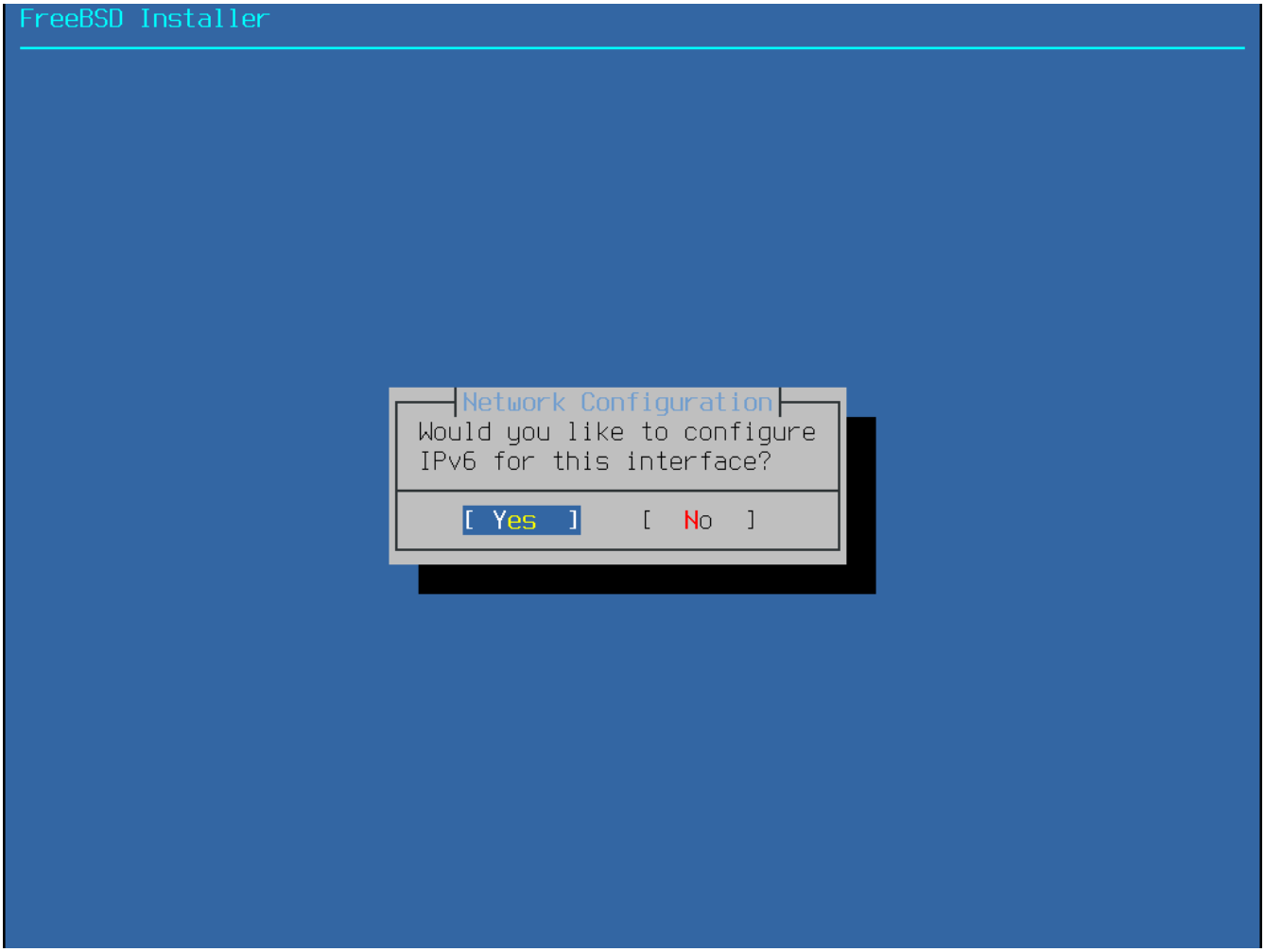


图 86. 选择 IPv6 网络

IPv6 也有两种配置方式。 SLAAC， 或 StateLess Address AutoConfiguration （无状态地址自动配置）方式能够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。

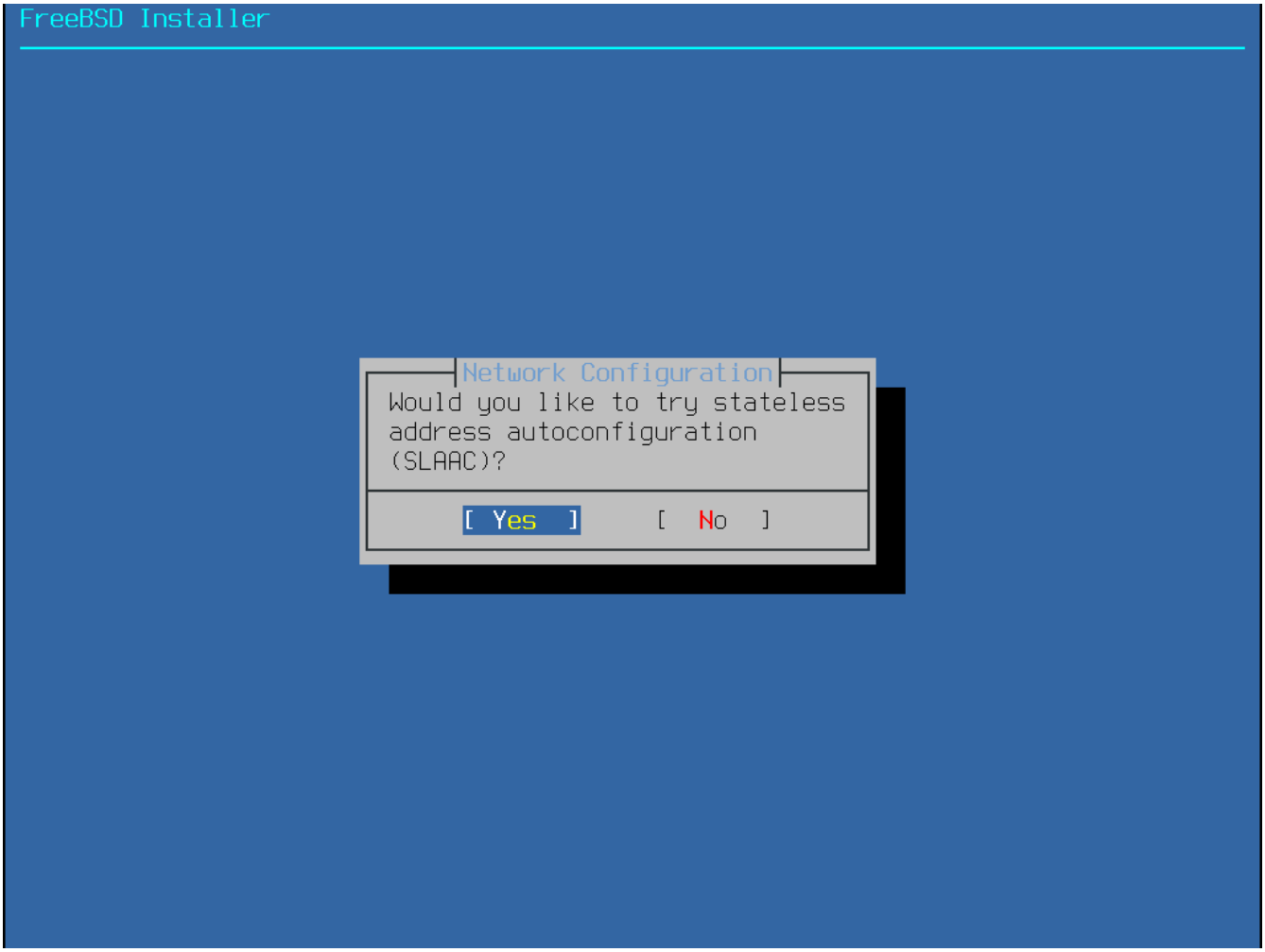


图 87. 选择 SLAAC 配置 IPv6

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。

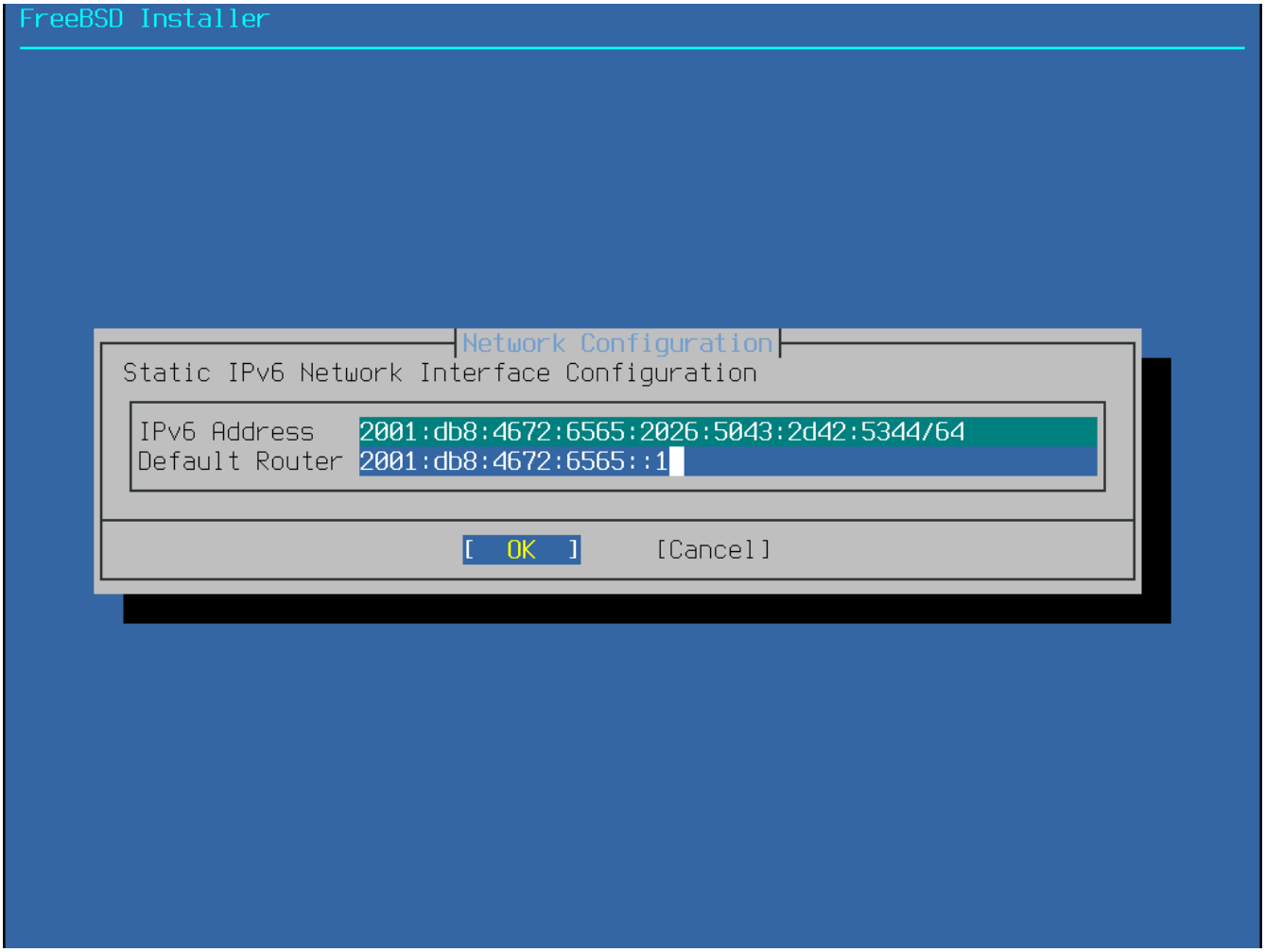


图 88. 静态配置 IPv6

- IPv6 Address （IPv6地址） - 为当前计算机手工分配的 IP 地址。 这个地址必须是唯一的，并且没有被其他本地网络设备使用。
- Default Router （默认路由） - 网络上默认路由的地址。 通常， 这是将本地网络连接至 Internet 的路由器或其他网络设备的地址。 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间的相互转换。 如果使用的是 DHCP 或 SLAAC， 那么其配置很可能已经存在； 否则， 请在 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。

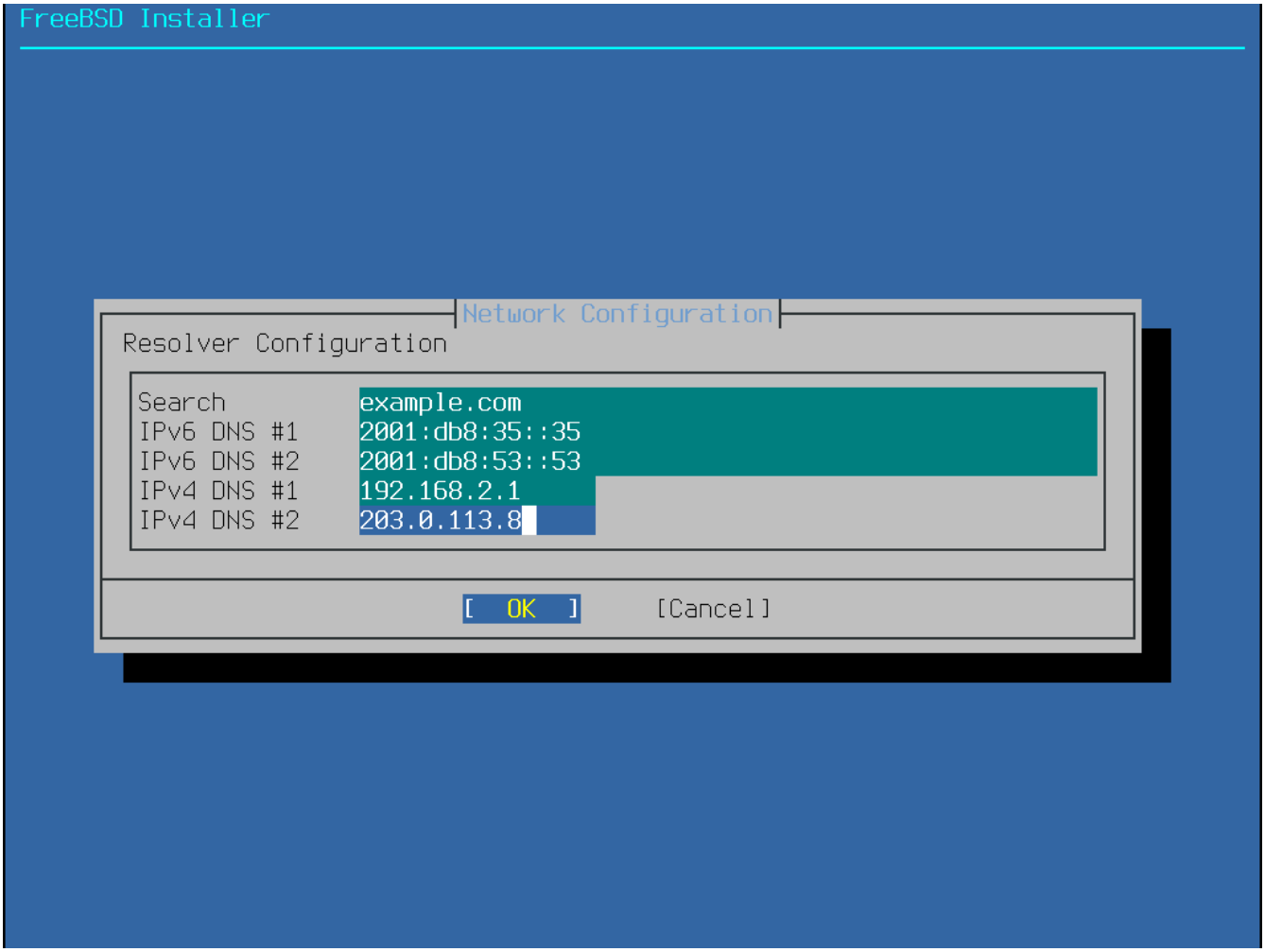


图 89. DNS 配置

3.9.3. 设置时区

为您的机器设置时区将允许其自动校时， 并正确执行一些与时区相关的操作。

示例中的机器位于美国东部时区。 根据所处的地理位置， 您的选择可能会有所不同。

图 90. 选择本地或 UTC 时钟

选择 [Yes] 或 [No] 以确定机器时钟的配置方式， 然后按 Enter 键。 如果您并不知道系统使用的是 UTC 还是本地时间， 请选择 [No] 以使用更为常见的本地时间。

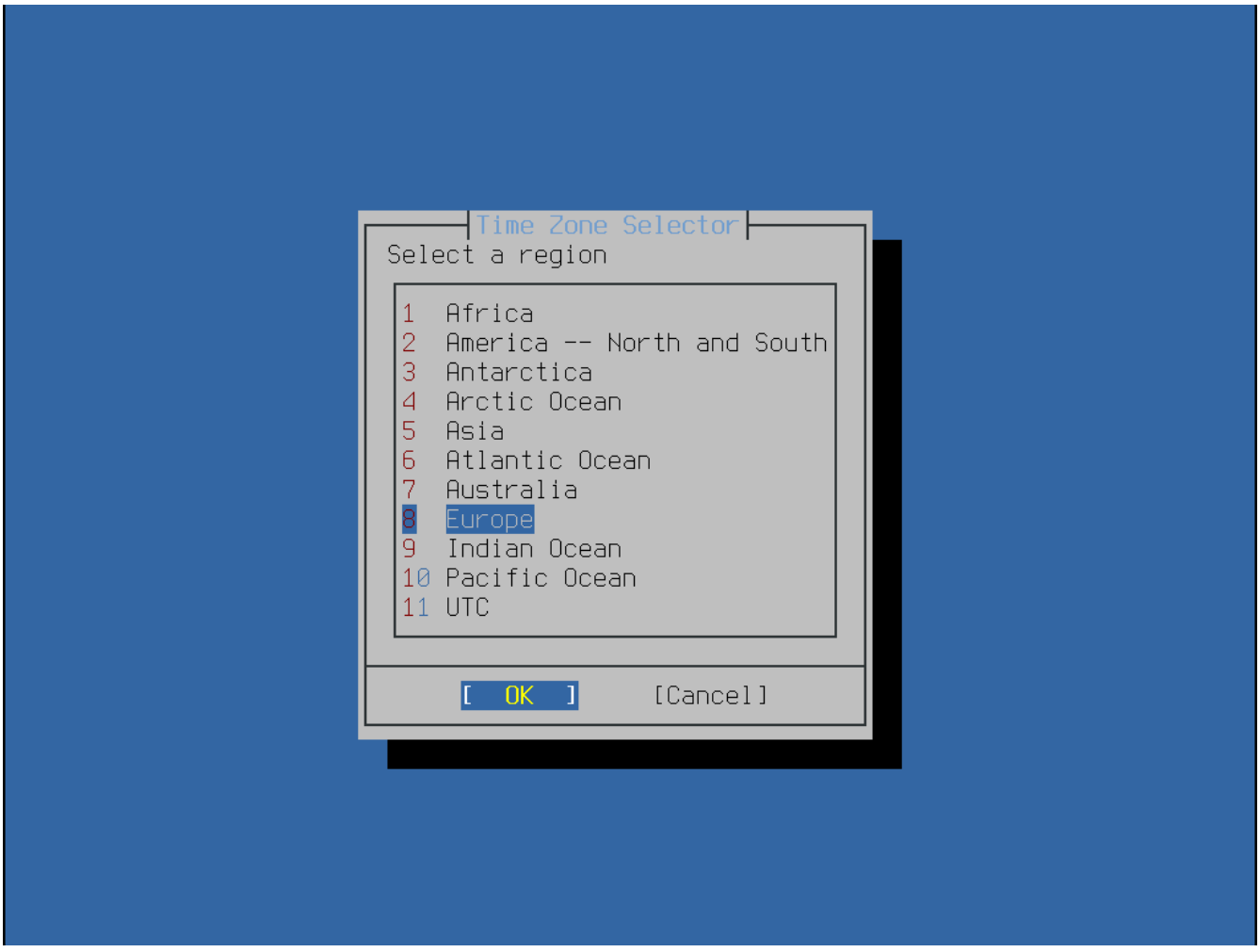


图 91. 选择地区

使用方向键选择合适的地区后按下 Enter 键。

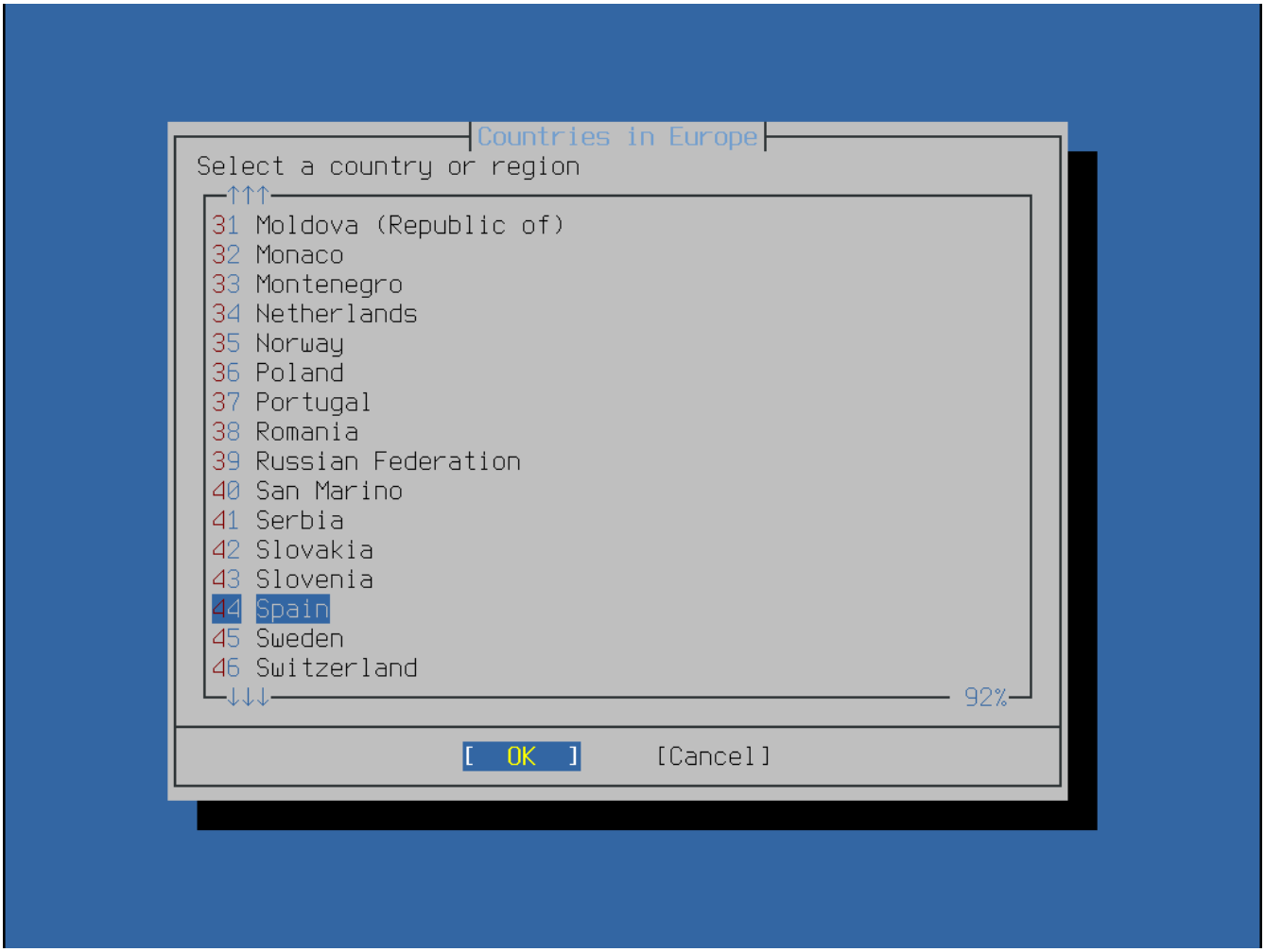


图 92. 选择国家

用方向键选择合适的国家后按下 Enter 键。

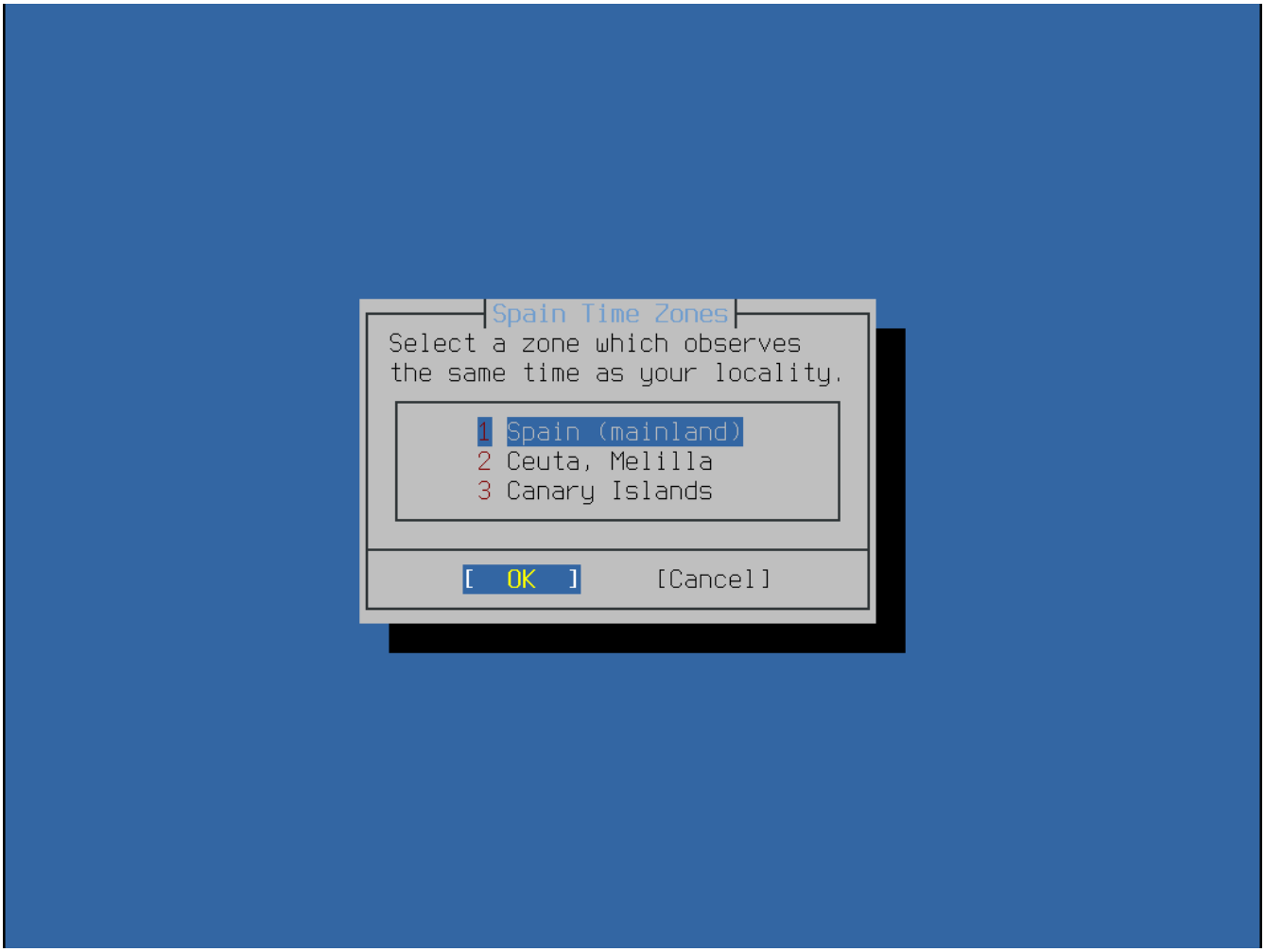


图 93. 选择时区

用方向键选择合适的时区后按下 Enter 键。

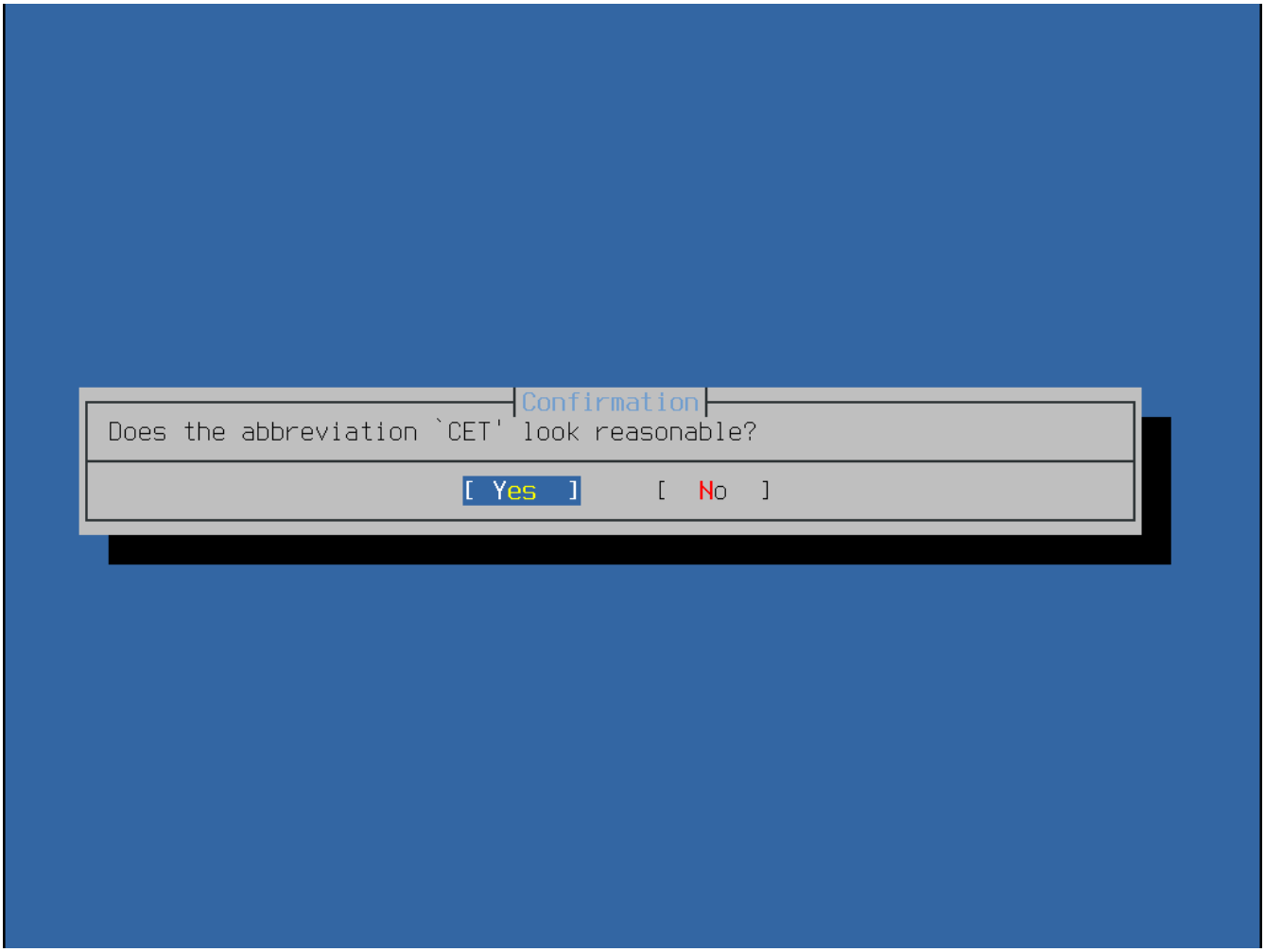


图 94. 确认时区选择

确认时区的缩写是正确的， 然后按 Enter 键以继续安装后的配置。

3.9.4. 选择需要开启的服务

可以开启额外的系统服务， 它们会在系统启动时自动运行。 所有这些服务都是可选的。

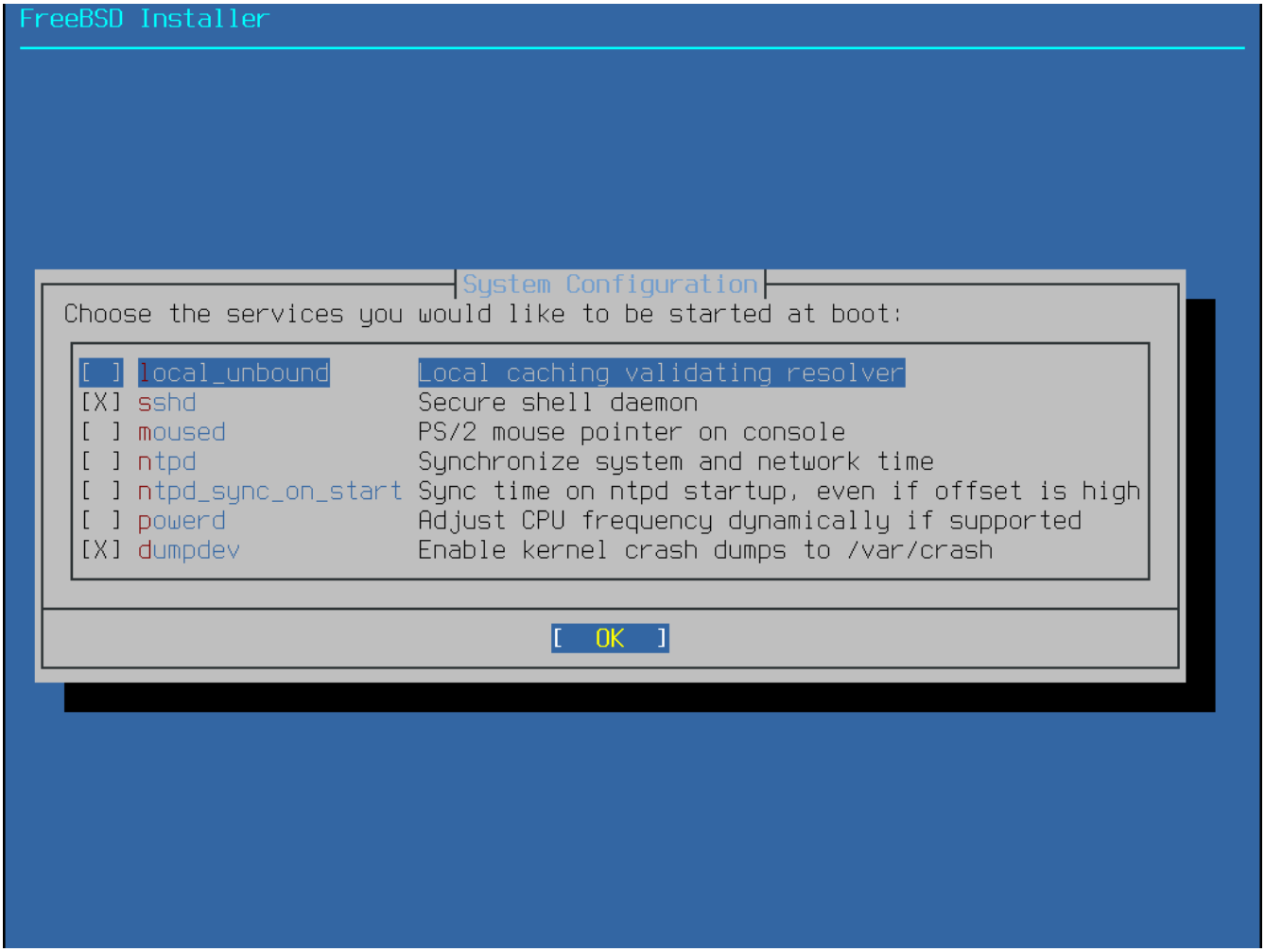


图 95. 选择需要开启的服务

额外的系统服务

- sshd - Secure Shell（即 SSH） 守护进程， 提供安全的远程访问。
- moused - 支持在系统控制台中使用鼠标。
- ntpd - Network Time Protocol（网络时间协议， 简称 NTP） 守护进程， 提供时钟自动同步。
- powerd - 系统电量控制程序， 用于控制电量及节能。

3.9.5. 启用崩溃转储

bsdinstall 将询问是否在目标系统上启用崩溃转储。 由于在调试系统时非常有用，
因此鼓励用户尽可能地启用崩溃转储。 选择 [Yes] 以启用崩溃转储， 或选择 [No] 以不启用崩溃转储。

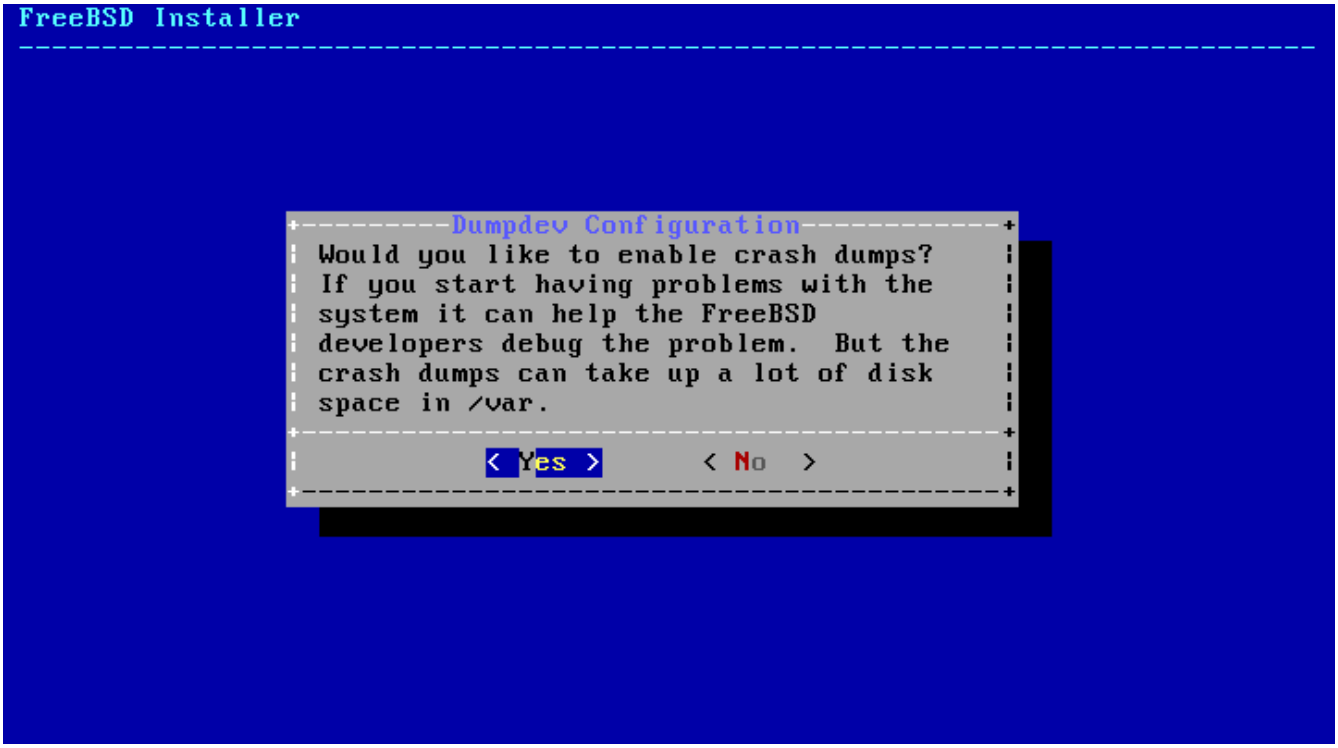


图 96. 启用崩溃转储

3.9.6. 添加用户

在安装过程中， 应至少添加一位普通用户， 而不要始终以 root 身份登入。 当以 root 身份登入系统时，系统几乎不会对其操作提供任何限制或保护。 以普通用户身份登录更为安全。

选择 [Yes] 来添加新用户。

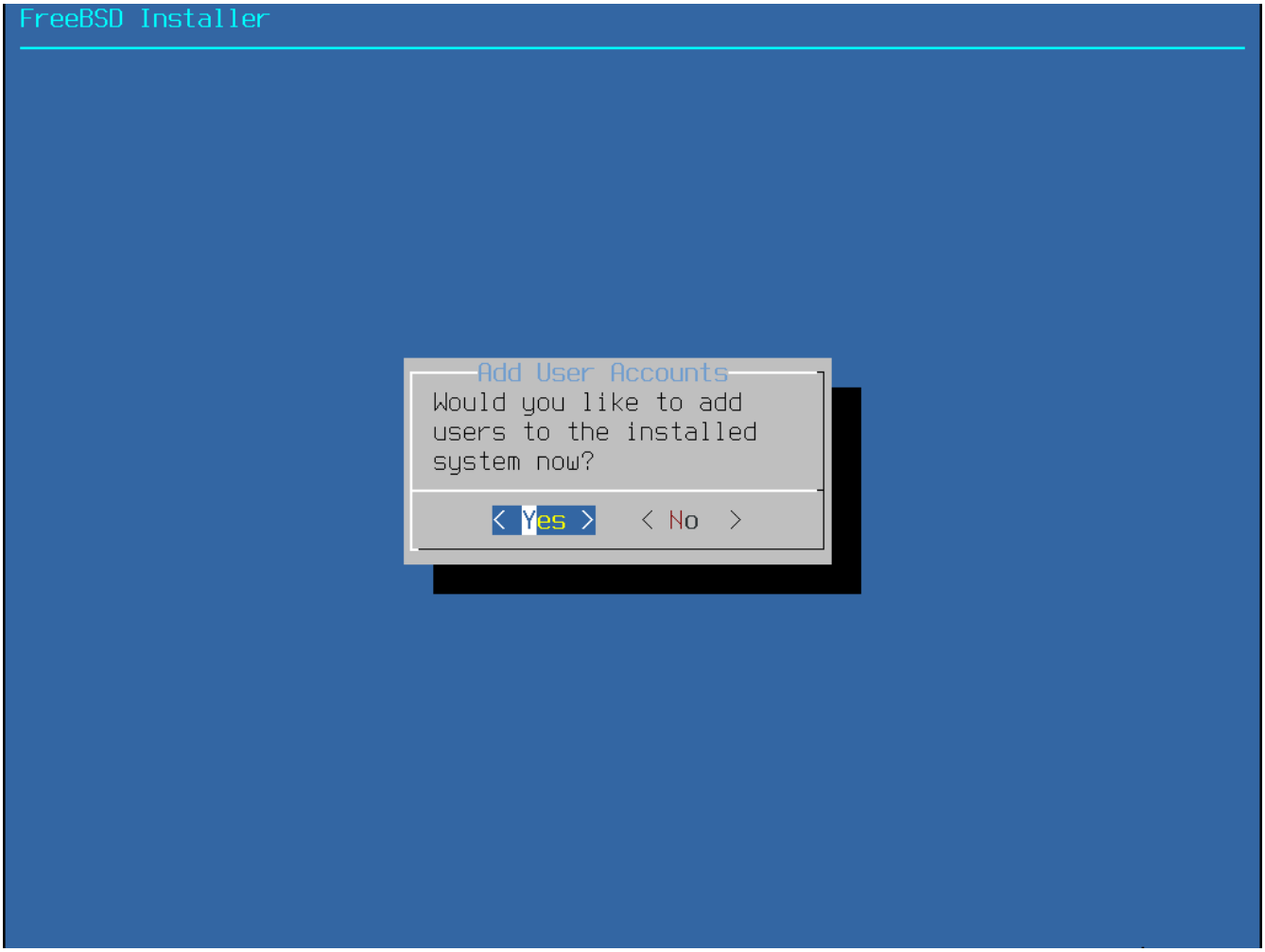


图 97. 添加用户帐号

为需要添加的用户输入信息。

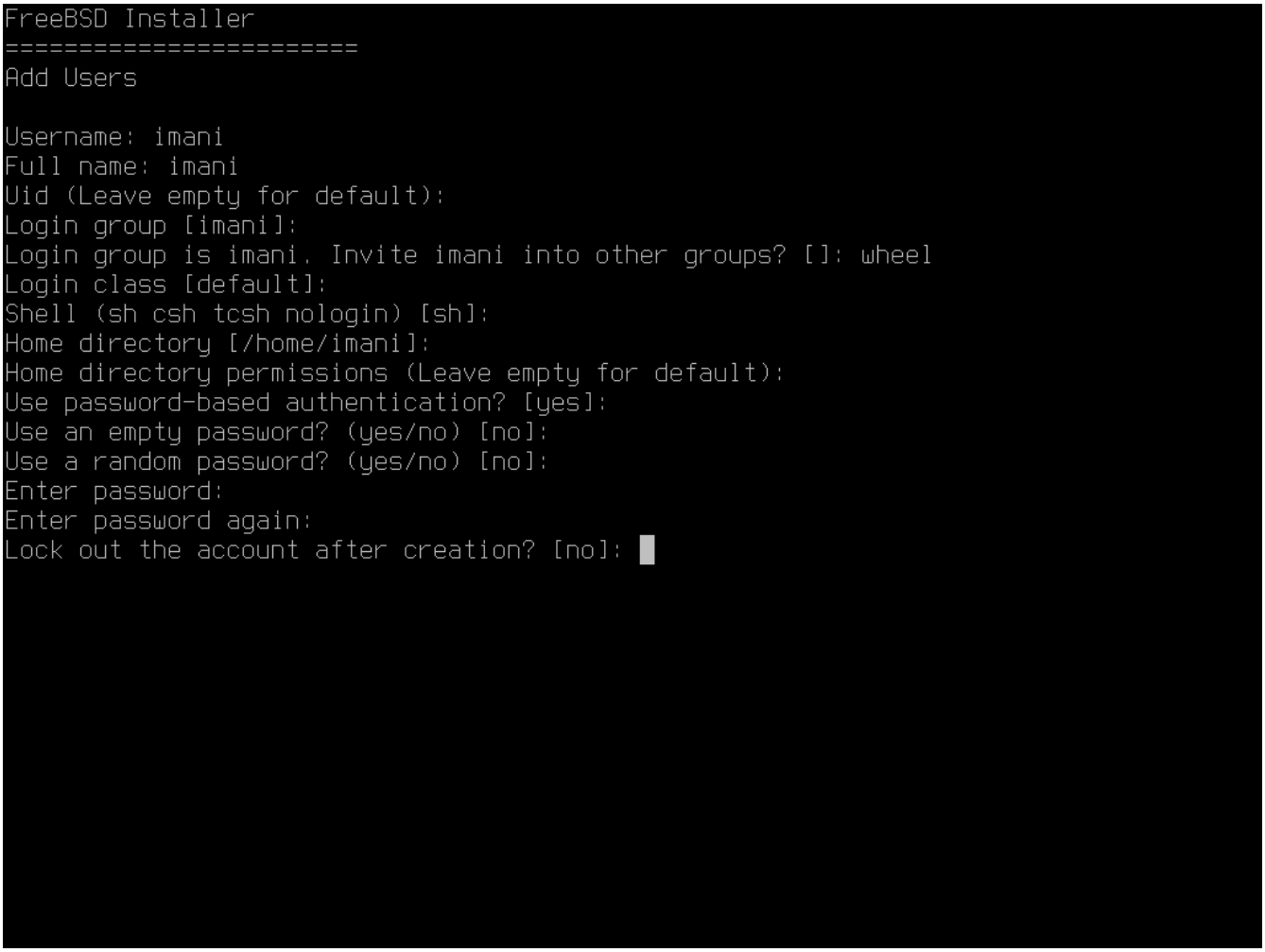


图 98. 输入用户信息

用户信息

- Username - 用户名， 即登入时用户所输入的名称。 通常是名的首字母加姓的组合。
- Full name - 用户的全名。
- Uid - 用户 ID。 通常留空以自动分配。
- Login group - 用户组。 通常留空以接受默认取值。
- Invite user into other groups? - 是否同时将用户加入其他权限组？ 如果需要， 请输入权限组名称。
- Login class - 登录类别。 通常留空以接受默认取值。
- Shell - 用户 shell。 在本例中选择的是 csh(1)。
- Home directory - 用户主目录。 通常留空以接受默认取值。
- Home directory permissions - 用户主目录的权限。 通常留空以接受默认取值。
- Use password-based authentication? - 是否使用基于密码的认证？ 通常为 “yes”。
- Use an empty password? - 是否使用空密码？ 通常为 “no”。
- Use a random password? - 是否使用随机密码？ 通常为 “no”。
- Enter password - 用户的实际密码。 输入的字符不会在屏幕上显示。
- Enter password again - 必须再次输入密码以进行验证。
- Lock out the account after creation? - 创建后锁定帐号？ 通常为 “no”。

全部信息输入完成后， 系统会显示摘要并询问是否正确。 如果发现了错误， 可以输入 no 后进行修改；如果没有错误， 请输入 yes 以创建新用户。

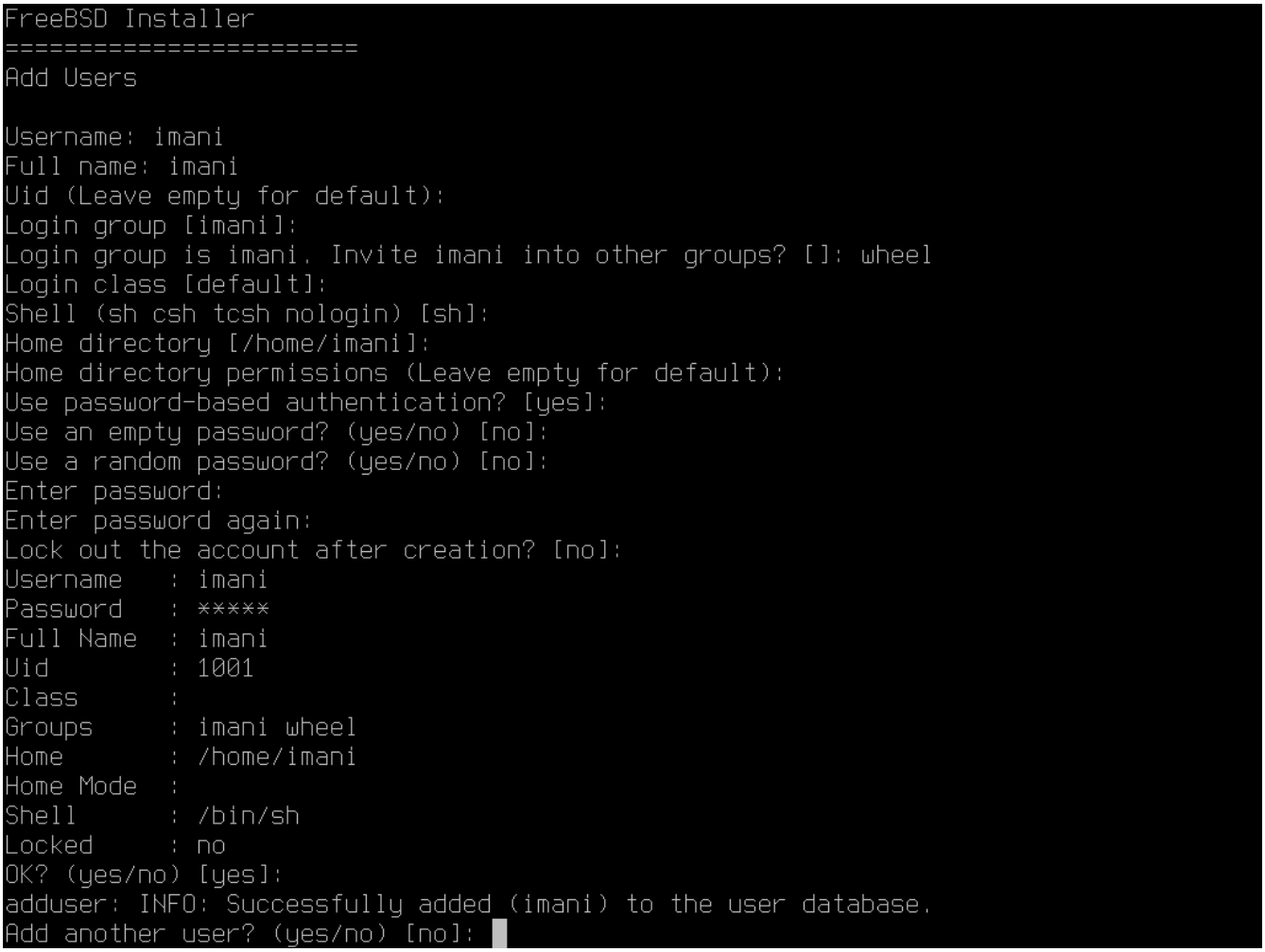


图 99. 退出用户与组管理

若需添加更多用户， 请在问题"`Add another user?"后输入 `yes； 输入 no 以完成用户添加并继续安装。

更多有关用户添加及管理的信息， 请参见 用户和基本的帐户管理。

3.9.7. 最终配置

所有的安装及配置完成后， 仍有机会对其进行修改。

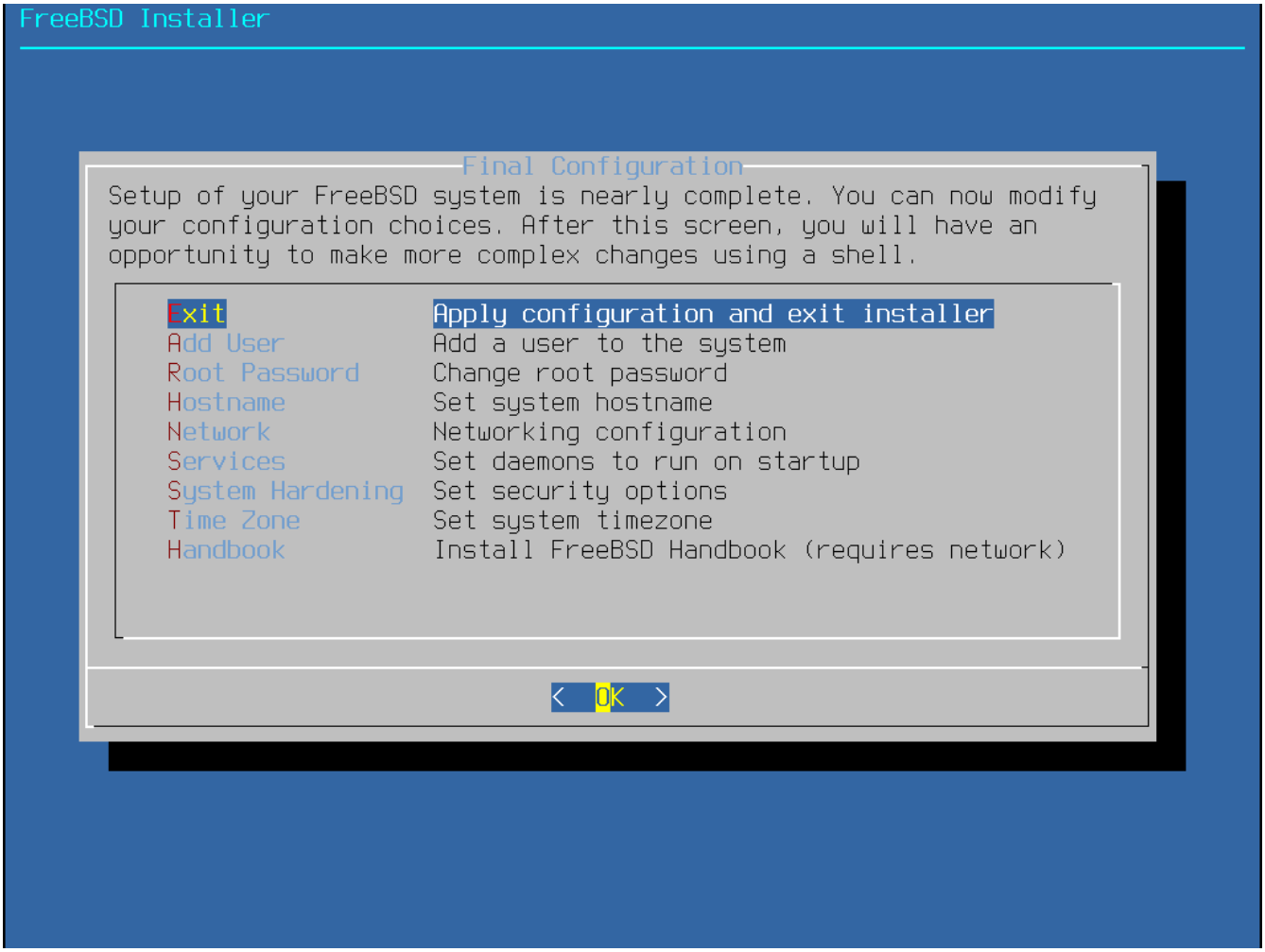


图 100. 最终的配置菜单

使用此菜单， 可以在完成安装前添加或修改任何配置。

最终的配置选项

- Add User - 添加用户， 详见 添加用户。
- Root Password - root 密码， 详见 设置 root 密码。
- Hostname - 主机名， 详见 设置主机名。
- Network - 网络， 详见 配置网络接口。
- Services - 服务， 详见 选择需要开启的服务.
- Time Zone - 时区， 详见 设置时区。
- Handbook - 手册，将下载并安装 FreeBSD 使用手册（即本书）。

完成了最终配置后， 请选择 [Exit] 以继续安装。

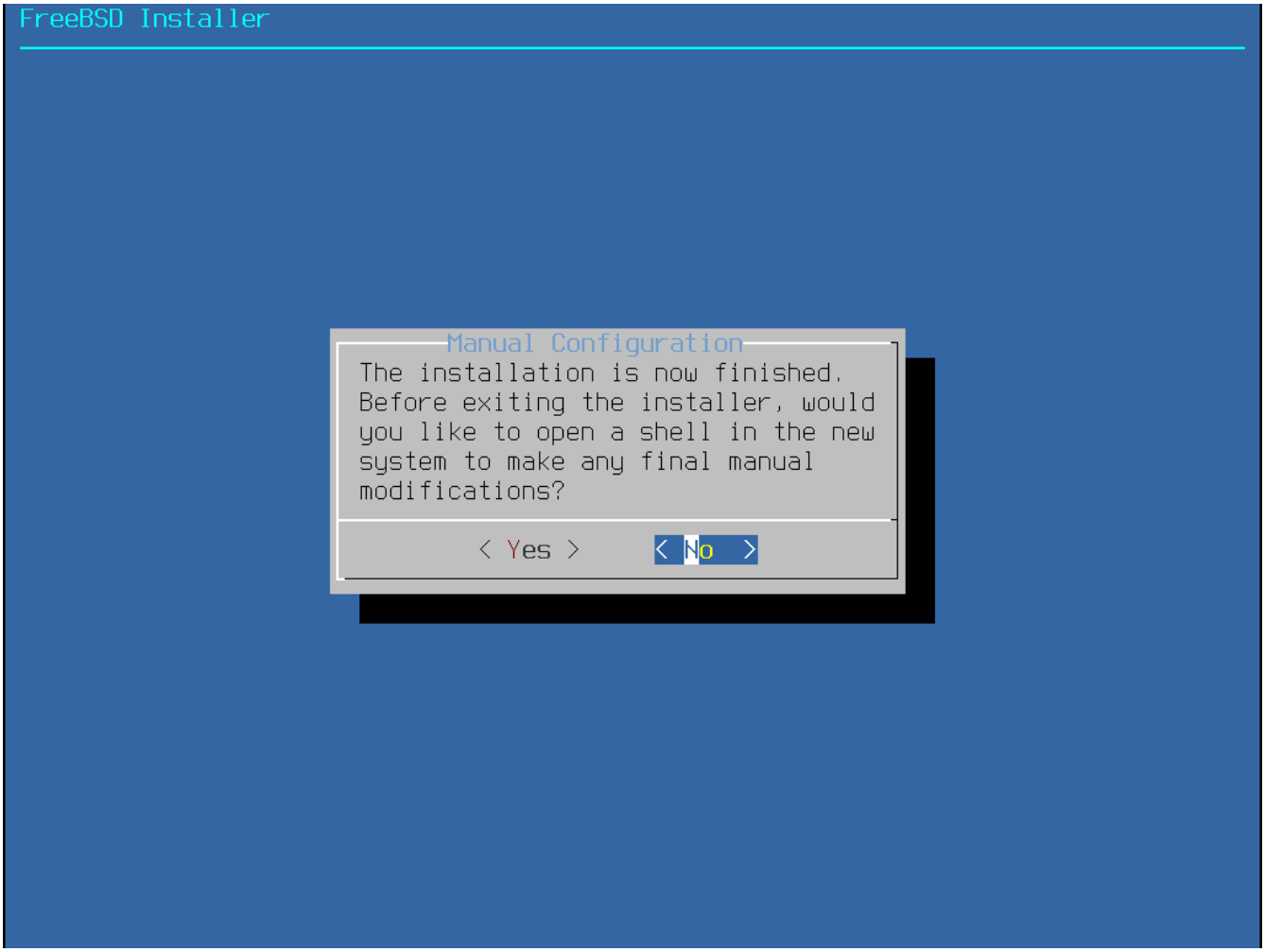


图 101. 手动配置

bsdinstall 会询问重启前是否还需要额外的配置： 选择 [Yes] 进入 shell 做这些配置， 选择 [No] 以执行安装的最后一步。

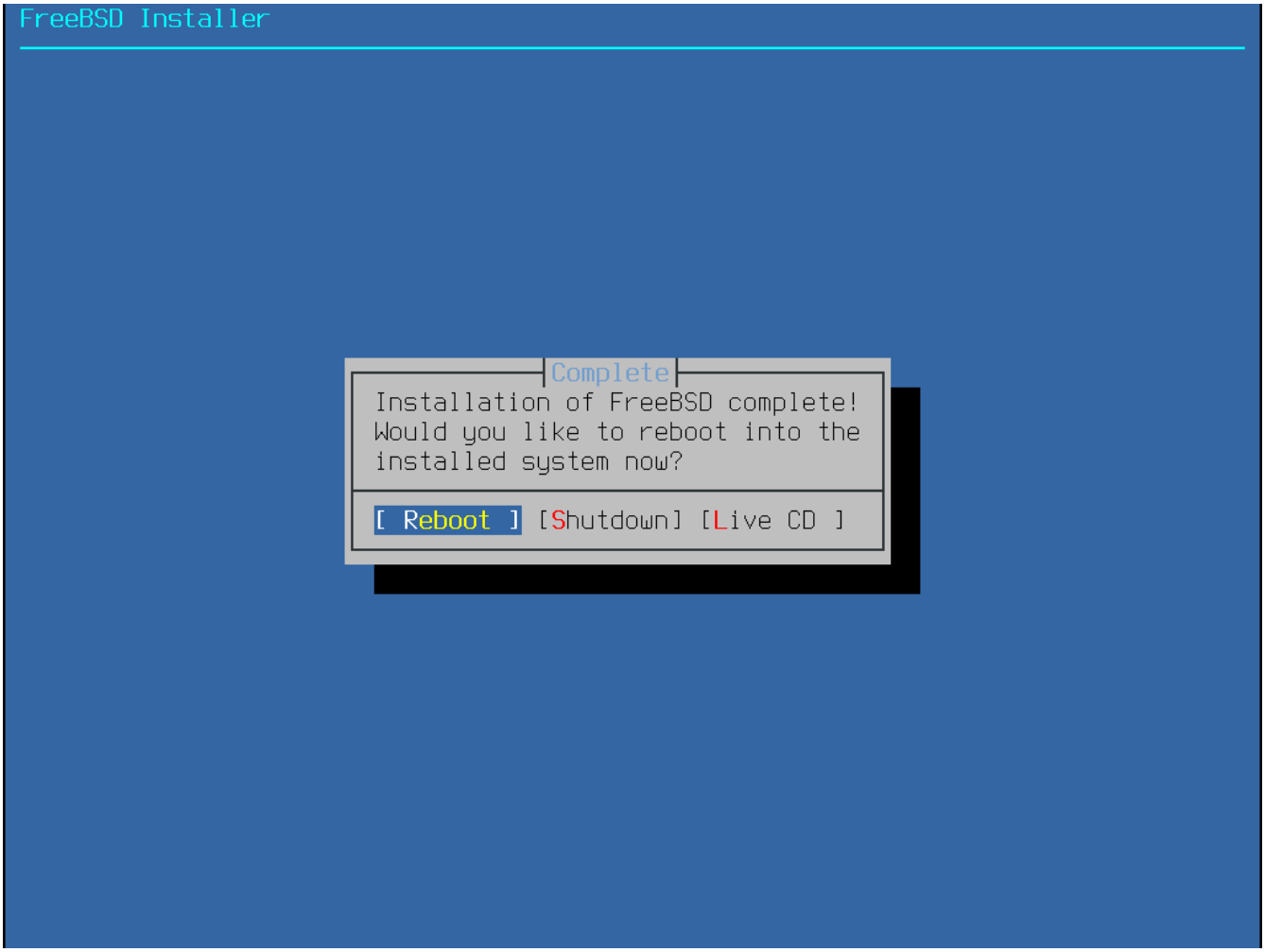


图 102. 完成安装

如果需要进一步的配置或特殊的设置， 可以选择 [Live CD] 来进入安装介质的 Live CD 模式。

安装完成后，选择 [Reboot] 重启计算机， 并开始使用全新的 FreeBSD 系统。 请不要忘记移除 FreeBSD 的安装 CD、 DVD 或 USB 记忆棒， 否则计算机可能会再次从这些介质启动。

3.9.8. FreeBSD 的启动与关闭

3.9.8.1. FreeBSD/i386 的启动

FreeBSD 启动时会显示许多相关信息， 正常情况下屏幕会不断滚动，
而启动完成后则会显示一个登录提示符。 如果需要查看启动时的相关信息， 可以按下 Scroll-Lock 键开启 scroll-back buffer（回滚缓存）， 然后使用 PageUp 键、 PageDown 键与方向键行翻阅； 再次按下 Scroll Lock 键将关闭回滚缓存并返回正常的屏幕。

在 login: 提示符处输入安装时添加的用户名来登录系统， 本例中是 asample。 除非有必要，否则请勿作为 root 登录。

上述的回滚缓存大小有限， 因而未必全部可见。 登入系统后， 在提示符处输入 dmesg | less，
能够查看到绝大部分的启动信息， 查看后按 q 键返回命令行。

典型的启动信息（此处略去了版本信息）：

```
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
=0x783fbff<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
usbus0: <OHCI (generic) USB controller> on ohci0
pci0: <bridge> at device 7.0 (no driver attached)
```

```
acpi_acad0: <AC Adapter> on acpi0
atkbdc0: <Keyboard controller (i8042)> port 0x60,0x64 irq 1 on acpi0
atkbd0: <AT Keyboard> irq 1 on atkbdc0
kbd0 at atkbd0
atkbd0: [GIANT-LOCKED]
psm0: <PS/2 Mouse> irq 12 on atkbdc0
psm0: [GIANT-LOCKED]
psm0: model IntelliMouse Explorer, device ID 4
attimer0: <AT timer> port 0x40-0x43,0x50-0x53 on acpi0
Timecounter "i8254" frequency 1193182 Hz quality 0
Event timer "i8254" frequency 1193182 Hz quality 100
sc0: <System console> at flags 0x100 on isa0
sc0: VGA <16 virtual consoles, flags=0x300>
vga0: <Generic ISA VGA> at port 0x3c0-0x3df iomem 0xa0000-0xbffff on isa0
atrtc0: <AT realtime clock> at port 0x70 irq 8 on isa0
Event timer "RTC" frequency 32768 Hz quality 0
ppc0: cannot reserve I/O port range
Timecounters tick every 10.000 msec
pcm0: measured ac97 link rate at 485193 Hz
em0: link state changed to UP
usbus0: 12Mbps Full Speed USB v1.0
ugen0.1: <Apple> at usbus0
uhub0: <Apple OHCI root HUB, class 9/0, rev 1.00/1.00, addr 1> on usbus0
cd0 at ata1 bus 0 scbus1 target 0 lun 0
cd0: <VBOX CD-ROM 1.0> Removable CD-ROM SCSI-0 device
cd0: 33.300MB/s transfers (UDMA2, ATAPI 12bytes, PIO 65534bytes)
cd0: Attempt to query device size failed: NOT READY, Medium not present
ada0 at ata0 bus 0 scbus0 target 0 lun 0
ada0: <VBOX HARDDISK 1.0> ATA-6 device
ada0: 33.300MB/s transfers (UDMA2, PIO 65536bytes)
ada0: 12546MB (25694208 512 byte sectors: 16H 63S/T 16383C)
ada0: Previously was known as ad0
Timecounter "TSC" frequency 3007772192 Hz quality 800
Root mount waiting for: usbus0
uhub0: 8 ports with 8 removable, self powered
Trying to mount root from ufs:/dev/ada0p2 [rw]...
Setting hostuuid: 1848d7bf-e6a4-4ed4-b782-bd3f1685d551.
Setting hostid: 0xa03479b2.
Entropy harvesting: interrupts ethernet point_to_point kickstart.
Starting file system checks:
/dev/ada0p2: FILE SYSTEM CLEAN; SKIPPING CHECKS
/dev/ada0p2: clean, 2620402 free (714 frags, 327461 blocks, 0.0% fragmentation)
```

```
Mounting local file systems:.
vboxguest0 port 0xd020-0xd03f mem 0xf0400000-0xf07fffff,0xf0800000-0xf0803fff irq 20 at
device 4.0 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: usbus0.
DHCPREQUEST on em0 to 255.255.255.255 port 67
DHCPACK from 10.0.2.2
bound to 192.168.1.142 -- renewal in 43200 seconds.
add net ::ffff:0.0.0.0: gateway ::1
add net ::0.0.0.0: gateway ::1
add net fe80::: gateway ::1
add net ff02::: gateway ::1
ELF ldconfig path: /lib /usr/lib /usr/lib/compat /usr/local/lib
32-bit compatibility ldconfig path: /usr/lib32
Creating and/or trimming log files.
Starting syslogd.
No core dumps found.
Clearing /tmp (X related).
Updating motd:.
Configuring syscons: blanktime.
Generating public/private rsa1 key pair.
Your identification has been saved in /etc/ssh/ssh_host_key.
Your public key has been saved in /etc/ssh/ssh_host_key.pub.
The key fingerprint is:
10:a0:f5:af:93:ae:a3:1a:b2:bb:3c:35:d9:5a:b3:f3 root@machine3.example.com
The key's randomart image is:
+--[RSA1 1024]----+
|   o..           |
```

```
| o . .     |
| . o       |
|   o       |
| o S       |
| + + o     |
|o . + *    |
|o+ ..+ .   |
|==o..o+E   |
+-----------------+
Generating public/private dsa key pair.
Your identification has been saved in /etc/ssh/ssh_host_dsa_key.
Your public key has been saved in /etc/ssh/ssh_host_dsa_key.pub.
The key fingerprint is:
7e:1c:ce:dc:8a:3a:18:13:5b:34:b5:cf:d9:d1:47:b2 root@machine3.example.com
The key's randomart image is:
+--[ DSA 1024]----+
|    ..   . .|
|   o . . .+ |
|  . ..  . E .|
|  . . o o . . |
|   + S = .   |
|  + . = o    |
|   + . * .   |
|  . . o .    |
|   .o. .     |
+-----------------+
Starting sshd.
Starting cron.
Starting background file system checks in 60 seconds.

Thu Oct  6 19:15:31 MDT 2011

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

login:
```

在较慢的机器上， 生成 RSA 和 DSA 密钥可能需要一些时间。 这种情况只会在开启了 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。 请参阅 X Window 系统 了解详情。

3.9.9. 关闭 FreeBSD

正常关闭 FreeBSD 有助于保护数据及系统硬件不受损坏。 不要直接关闭电源。 如果用户是 wheel 组的成员， 首先在命令行中输入 su 后键入 root 密码成为超级用户。 此外， 也可作为 root 登录，

然后使用命令 shutdown -p now。 这样系统将安全地自行关闭。

虽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， 例如用户经常报告的问题。

3.10.1. 遇到错误时该如何处理

由于 PC 架构的各种限制， 硬件检测不可能 100% 地可靠探测， 然而， 当此类现象发生时，
您有可能可以通过一些操作来自行解决它们。

首先应该根据所安装的 FreeBSD 版本核对 硬件兼容说明 文档， 以确保其支持您的硬件。

如果使用被支持的硬件时仍遇到了死机或其他问题， 请联编一个 自定义内核， 这样即可为那些 GENERIC
内核中不存在的设备提供支持。 引导盘上的内核假定绝大多数硬件的 IRQ、 IO 地址和 DMA
通道均为出厂设置， 如果您的硬件被重新配置过， 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 侦测这些硬件。

还可能出现一种情况， 检测某个不存在的设备会导致稍后对其他存在的设备检测失败。 在这种情况下，
应该禁止检测引起冲突的设备所对应的驱动程序。

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拥有提供升级和相关信息的网站。

制造商通常建议， 除非有类似关键更新这种必要的原因， 否则应避免升级主板 BIOS。
升级过程一旦出现错误， 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. 在启动时， 我的系统在检测硬件时挂起， 或在安装过程中行为异常。

在 i386、 amd64 和 ia64 平台的启动过程中， FreeBSD 广泛使用了 ACPI 服务来检测系统配置， 不幸的是
ACPI 驱动和主板 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"
```

这一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， 请参见 概述。

Chapter 4. UNIX 基础

4.1. 概述

下列章节的命令和功能适用于FreeBSD操作系统。 同时这里许多内容和一些 类-UNIX® 操作系统相关。
假如您已经熟悉这些内容可跳过不阅读。 假如您是FreeBSD新手，
那您应该认真详细地从头到尾读一遍这些章节。

读取这些内容，您将了解：

- 怎样在FreeBSD使用 "虚拟控制台"。
- 在 UNIX® 中文件权限如何运作， 以及理解 FreeBSD 中的文件标志。
- FreeBSD 默认文件系统的架构。
- FreeBSD磁盘架构。
- 怎样挂接或卸下文件系统。
- 什么是进程、守护进程、信号。
- 什么是shell，应当怎样去改变登录进入的默认环境。
- 怎样使用基本的文本编辑器。
- 什么是设备，什么是设备节点。
- FreeBSD 下，使用的是什么可执行文件格式。
- 怎样使用 man 手册并取得更多资讯。

4.2. 虚拟控制台和终端

可以用多种不同的方式使用 FreeBSD， 在文本终端输入命令是其中之一。 通过使用这种方式，
您可以容易地使用 FreeBSD 来获得 UNIX® 操作系统的灵活而强大的功能。 这一节将介绍 "终端" 和
"控制台"， 以及如何在 FreeBSD 中使用它们。

4.2.1. 控制台

假如您没有设置 FreeBSD 在启动期间开启图形登录界面，
那么系统将在引导和启动脚本正确运行完成后，给您一个登录的提示。 您会看到类似这样的界面:

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

Fri Sep 20 13:01:06 EEST 2002

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

login:
```

这些信息可能和您的系统稍微有点不同，但不会有很大差别。 最后两行是我们感兴趣的， 理解这一行:

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

这一行是您刚才启动的系统信息其中一块， 您所看到的是一个"FreeBSD"控制台，
运行在一个Intel或兼容的x86体系架构上面。 这台计算机的名字 (每台 UNIX® 计算机都有自己的名字) 叫
pc3.example.org， 就是现在这个系统控制台-这个 ttyv0 终端的样子。

在最后，最后一行一直保持这样:

```
login:
```

这里， 您将可以输入用户名 "username" 并登录到 FreeBSD 系统中。 接下来的一节，
将介绍如何登录系统。

4.2.2. 进入FreeBSD

FreeBSD是一个多用户多任务的系统， 换句话来说就是一个系统中可以容纳许多不同的用户，
而这些用户都可以同时在这台机器中运行大量的程序。

每一个多用户系统都必须在某方面去区分 "user"， 在 FreeBSD 里 (以及 类-UNIX® 操作系统)，
完成这方面工作是有必要的， 因而， 每位使用者在运行程序之前都必须首先 "登录"，
而每位用户都有与之对应的用户名 ("username") 和密码 ("password")。 FreeBSD
会在用户进入之前作出询问这两项信息。

当 FreeBSD 引导并运行完启动脚本之后， ， 它会给出一个提示， 并要求输入有效的用户名：

```
login:
```

举个例子更容易理解，我们假设您的用户名叫 john。 在提示符下输入 john 并按 Enter，
此时您应该看到这个提示 "password"：

```
login: john
Password:
```

现在输入 john的密码并按下 Enter。 输入密码时是 不回显的! 不必为此担心， 这样做是出于安全考虑。

假如您输入的密码是正确的， 这时你应该已进入 FreeBSD， 并可以开始尝试可用的命令了。

您应该看见 MOTD 或者出现一个命令提示符 (#、$ 或 % 字符). 这表明您已成功登录进入FreeBSD。

4.2.3. 多个控制台

在一个控制台运行 UNIX® 命令虽说很好， 但 FreeBSD 具有一次运行 多个程序的能力。
仅使用一个控制台只会浪费 FreeBSD 同时运行多任务的能力。 而 "虚拟控制台" 在这方面发挥强大的功能。

FreeBSD 能配置出满足您不同需求的虚拟控制台， 在键盘上您用一组键就能从各个虚拟控制台之间切换。
各个控制台有自己的传输通道， 当您在各个控制台切换时 FreeBSD
会切换到合适的键盘传输通道和显示器传输通道。

FreeBSD 各个控制台之间可利用特殊组键切换并保留原有控制台 ，您可这样做: Alt + F1， Alt + F2， 一直到
Alt + F8 在FreeBSD里切换到其中一个虚拟控制台。

同样地, 您正在从其中某个控制台切换到另一个控制台的时候, FreeBSD
会保存正在使用和恢复将要使用屏幕传输通道。 这种结果形成一种 "错觉"，
您拥有许多"虚拟"屏幕和键盘可以输入很多的命令。
这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，
它继续运行而您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为8个，但并不是硬性设置，
您可以很容易设置虚拟控制台的个数增多或减少。 虚拟控制台的的编号和设置在 /etc/ttys 文件里。

您可以使用 /etc/ttys 文件在 FreeBSD 下配置虚拟控制台。
文件里每一未加注释的行都能设置一个终端或虚拟控制台 (当行里含有 # 这个字符时不能使用) 。 FreeBSD 默认配置是配置出9个虚拟控制台而只能启动8个， 以下这些行是 ttyv 一起启动:

```
# name  getty                 type    status      comments
#
ttyv0   "/usr/libexec/getty Pc"       cons25  on  secure
# Virtual terminals
ttyv1   "/usr/libexec/getty Pc"       cons25  on  secure
ttyv2   "/usr/libexec/getty Pc"       cons25  on  secure
ttyv3   "/usr/libexec/getty Pc"       cons25  on  secure
ttyv4   "/usr/libexec/getty Pc"       cons25  on  secure
ttyv5   "/usr/libexec/getty Pc"       cons25  on  secure
ttyv6   "/usr/libexec/getty Pc"       cons25  on  secure
ttyv7   "/usr/libexec/getty Pc"       cons25  on  secure
ttyv8   "/usr/X11R6/bin/xdm -nodaemon"  xterm   off secure
```

如果要了解这个文件中每一列的详细介绍， 以及虚拟控制台上所能使用的配置， 请参考联机手册 ttys(5)。

4.2.5. 单用户模式的控制台

关于 "单用户模式" 详细介绍在 单用户模式 这里可以找到。 当您运行单用户模式时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 文件设置，
可在这行找到要启动的控制台:

```
# 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 已经注释掉, 您可编辑这行把 secure 改为 insecure。 这样，
当用单用户进入 FreeBSD 时， 它仍然要求提供 root 用户的密码。

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

4.2.6. 改变控制台的显示模式

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

```
options VESA
options SC_PIXEL_MODE
```

在内核用这2个选项编译完成后，你就可以使用 vidcontrol(1) 工具来测定你的硬件支持何种显示模式了。 以 root 身份在控制台键入以下命令来获得一份所支持的显示模式列表。

```
# vidcontrol -i mode
```

这个命令的输出是一份你的硬件所支持的显示模式列表。 你可以在以 root 身份在控制台上键入 vidcontrol(1) 命令来改变显示模式：

```
# vidcontrol MODE_279
```

如果你对于新的显示模式满意，那么可以把它加入到 /etc/rc.conf 使机器在每次启动的时候都能生效，我们使用了上一个例子中的模式：

```
allscreens_flags="MODE_279"
```

4.3. 权限

FreeBSD，是 BSD UNIX® 的延续， 并基于几个关键的 UNIX® 观念。 从一开始就多处提到 FreeBSD 是一个多用户的操作系统， 它能分别处理几个同时工作的用户所分配的毫无关联任务。 并负责为每位用户的硬件设备、 外设、 内存和 CPU 处理时间作出合理安排。

因为系统有能力支持多用户， 在每一方面系统都会作出谁能读、 写和执行的资源权力限制。 这点权限以三个八位元的方式储存着， 一个是表示文件所属者， 一个是表示文件所属群组， 一个是表示其他人。 这些数字以下列方式表示：

数值	权限	目录列表
0	不能读，不能写，不能执行	---
1	不能读，不能写，可执行	--x
2	不能读，可写，不能执行	-w-
3	不能读，可写，可执行	-wx
4	可读，不能写，不能执行	r--
5	可读，不能写，可执行	r-x
6	可读，可写，不能执行	rw-
7	可读，可写，可执行	rwx

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

```
% 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 在每行的开始出现了：

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

从左边起的第一个字，告诉我们这个文件是一怎样的文件: 普通文件?目录?特殊设备?socket?或是设备文件?
在这个例子， - 表示一个普通文件。 接下来三个字是 rw- 是文件拥有者的权限。 再接下来的三个字是 r--
是文件所属群组的权限。 最後三个字是 r-- 是其他人的权限。
以这一个文件为例，他的权限设定是拥有者可以读写这个文件、群组可以读取、
其他使用者也能读取这个文件。 根据上面的表格， 用数字表示这个文件其三部分的权限应该是 644。

这样很好，但系统怎样对设备进行权限控制的? 事实上 FreeBSD 将大部份硬件设备当作一个文件看待，
用程序能打开、读取、写入数据就如其他的文件一样。 而设备文件放在 /dev 目录。

目录也视为一种文件，也有读取、写入、执行的权限。 但目录的执行权限意义并不与普通文件相同，
实际上执行权限是进入权限。 当一个目录是被标示可以执行的时， 表示可以进入它， 或者换言之， 利用
"cd" (改变当前目录) 进入它。 此外， 这也表示有权进入目录的用户， 可以访问其下的已知名字的文件
(当然目录下的文件也受到访问限制)。

详细方面，想读取一个目录的列表就必须设为可读权限，
同时想删除一个已知的文件，就必须把目录下这个文件设为可写 和 执行权限。

还有更多权限设定， 但是他们大多用在特殊状况下如一个setuid的执行文件和粘贴性目录，
如果想要得知有关文件权限和如何设定的更多资讯，请看手册chmod(1)。

4.3.1. 权限的符号化表示

权限符号，某些时候就是指符号表达式， 使用八进制的字符给目录或文件分配权限。 权限符号的使用语法是
(谁) (作用) (权限)。 看看下列数值的在那些地方所起什么样的作用:

选项	字母	介绍
(谁)	u	用户
(谁)	g	所属群体
(谁)	o	其他人
(谁)	a	所有人 ("全部")
(作用)	+	增加权限
(作用)	-	减少权限
(作用)	=	确定权限
(权限)	r	可读
(权限)	w	可写
(权限)	x	执行
(权限)	t	粘贴位
(权限)	s	设置 UID 或 GID

这些数值 chmod(1) 以习惯标定的。 举个例子，用以下命令阻止其他人访问 FILE文件:

```
% chmod go= FILE
```

如果需要对文件一次进行多项变动， 则可用逗号分开， 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， 并为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之上， FreeBSD 还支持使用 "文件标志"。
这些标志为文件提供了进一步的安全控制机制， 但这些控制并不适用于目录。

这些文件标志提供了针对文件的进一步控制， 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
上应用系统禁删标志， 应使用下述命令：

```
# chflags sunlink file1
```

要禁用系统禁删标志， 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， 应使用命令 ls(1) 的 -lo 参数：

```
# ls -lo file1
```

输出结果应类似于：

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

许多标志只可以由 root 用户来增加， 而另一些， 则可以由文件的所有者来增加。 建议管理员仔细阅读
chflags(1) 和 chflags(2) 联机手册， 以对其加深理解。

4.3.3. setuid、 setgid 和 sticky 权限

除了前面已经讨论过的那些权限之外， 还有三个管理员应该知道的权限配置。 它们是 setuid、 setgid 和
sticky。

这些配置对于一些 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 是拥有或启动进程的用户 UID。 生效 UID 是进程以其身份运行的用户 ID。 举例来说，
passwd(1) 工具通常是以发起修改密码的用户身份启动， 也就是说其进程的真实用户 ID 是那个用户的 ID；
但是， 由于需要修改密码数据库， 它会以 root 用户作为生效用户 ID 的身份运行。 这样，
普通的非特权用户就可以修改口令， 而不是看到 Permission Denied 错误了。

mount(8) 的 nosuid 选项可以令系统在不给出任何错误提示的情况下不执行这些程序。
另一方面， 这个选项并不是万无一失的， 正如 mount(8) 联机手册所提到的那样，
如果系统中安装了绕过 nosuid 的封装程序， 那么这种保护就可以被绕过了。

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 的权限应该如下面这样：

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

您会注意到， 在原先的属主执行权限的位置变成了 s。 这样， 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， 以普通用户身份启动 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
```

正如前面所说的那样， passwd 是以普通用户的身份启动的， 但其生效 UID 是 root。

与此对应， setgid 权限的作用， 与 setuid 权限类似， 只是当应用程序配合这一设定运行时， 它会被授予拥有文件的那个组的权限。

如果需要在文件上配置 setgid 权限， 可以在权限数值前面增加数字二 (2) 来运行 chmod 命令， 如下面的例子所示：

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

可以用与前面类似的方法来检视新设定的生效情况， 在组权限的地方的 s 表示这一配置已经生效：

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

在这些例子中， 尽管 shell 脚本也属于可执行文件的一种， 但它们不会以您配置的 EUID 或生效用户 ID 的身份运行。 这是因为 shell 脚本可能无法直接呼叫 setuid(2) 调用。

我们已经讨论了两个特殊权限位 (setuid 和 setgid 权限位)， 它们让用户在使用程序时能够用到更高的权限， 有时这会削弱系统的安全性。 除了这两个之外， 还有第三个特殊权限位： sticky bit， 它能够增强安全性。

当在目录上设置了 sticky bit 之后， 其下的文件就只能由文件的所有者删除了。 这个权限设置能够防止用户删除类似 /tmp 这样的公共目录中不属于他们的文件。 要应用这种权限， 可以在权限设置前面加上数字一 (1)。 例如：

```
# chmod 1777 /tmp
```

现在， 可以用 ls 命令来查看效果：

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

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

这里的结尾的 t 表示了 sticky bit 权限。

4.4. 目录架构

理解 FreeBSD 的目录层次结构对于建立对系统整体的理解十分重要的基础。 其中，
最重要的概念是根目录， "/"。 这个目录是系统引导时挂接的第一个目录，
它包含了用以准备多用户操作所需的操作系统基础组件。
根目录中也包含了用于在启动时转换到多用户模式之前挂接其他文件系统所需的挂接点。

挂接点 (mount point) 是新增的文件系统在接入现有系统时的起点位置 (通常是根目录)。 在 磁盘组织 对此进行了详细的阐述。 标准的挂接点包括 /usr、 /var、 /tmp、 /mnt， 以及 /cdrom。
这些目录通常会在 /etc/fstab 文件中提及。 /etc/fstab 是一张包含系统中各个文件系统及挂接点的表。 在 /etc/fstab 中的绝大多数文件系统都会在启动时由 rc(8) 脚本自动挂接， 除非特别指定了 noauto 选项。
更多细节请参考 fstab 文件。

您可以通过 hier(7) 来了解完整的文件系统层次说明。 现在，
让我们先来看一看绝大多数的常见的目录以供参考。

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/	在操作系统在启动加载期间所用的程序和配置。
/boot/defaults/	默认每步引导启动的配置内容，请查阅loader.conf(5)。
/dev/	设备节点，请查阅 intro(4)。
/etc/	系统启动的配置和脚本。
/etc/defaults/	系统默认的启动配置和脚本，请参考 rc(8) 。
/etc/mail/	关系到邮件系统运作的配置， 请参考 sendmail(8)。
/etc/namedb/	named 配置文件，请参考 named(8)。
/etc/periodic/	每天、每星期和每月周期性地运行的脚本， 请通过 cron(8)查阅 periodic(8)。
/etc/ppp/	ppp配置文件，请查阅ppp(8)。
/mnt/	由管理员习惯使用挂接点的临时空目录。
/proc/	运行中的文件系统，请参阅 procfs(5) 和 mount_procfs(8)。
/rescue/	用于紧急恢复的一组静态联编的程序； 参见 rescue(8)。
/root/	root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实用目录。

目录	介绍
/tmp/	临时文件。 /tmp 目录中的内容， 一般不会在系统重新启动之后保留。 通常会将基于内存的文件系统挂在 /tmp 上。 这一工作可以用一系列 tmpmfs 相关的 rc.conf(5) 变量来自动完成。 (或者， 也可以在 /etc/fstab 增加对应项； 参见 mdmfs(8))。
/usr/	存放大多数用户的应用软件。
/usr/bin/	存放实用命令，程序设计工具，和应用软件。
/usr/include/	存放标准 C include 文件.
/usr/lib/	存放库文件。
/usr/libdata/	存放各种实用工具的数据文件。
/usr/libexec/	存放系统实用或后台程序 (从另外的程序启动执行)。
/usr/local/	存放本地执行文件， 库文件等等， 同时也是 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， 请查阅 hier(7)。 但 man 目录例外， 它们是直接放在 /usr/local 而不是 /usr/local/share 下的， 而 ports 说明文档在 share/doc/port。
/usr/obj/	通过联编 /usr/src 得到的目标文件。
/usr/ports/	存放 FreeBSD 的 Ports Collection (可选)。
/usr/sbin/	存放系统后台程序 和 系统工具 (由用户执行)。
/usr/shared/	存放架构独立的文件。
/usr/src/	存放 BSD 或者本地源码文件。
/usr/X11R6/	存放 X11R6 可执行文件、 库文件、 配置文件等的目录(可选)。
/var/	多用途日志、 临时或短期存放的， 以及打印假脱机系统文件。 有时会将基于内存的文件系统挂在 /var 上。 这一工作可以通过在 rc.conf(5) 中设置一系列 varmfs 变量 (或在 /etc/fstab 中加入一行配置； 参见 mdmfs(8)) 来完成。
/var/log/	存放各种的系统记录文件。
/var/mail/	存放用户mailbox(一种邮件存放格式)文件。
/var/spool/	各种打印机和邮件系统spooling(回环)的目录。
/var/tmp/	临时文件。 这些文件在系统重新启动时通常会保留， 除非 /var 是一个内存中的文件系统。
/var/yp/	NIS 映射。

4.5. 磁盘组织

FreeBSD 查找文件的最小单位是文件名。 而文件名区分大小写，这就意味着 readme.txt 和 README.TXT 是两个不相同的文件。 FreeBSD 不凭文件扩展名 (.txt) 去识别这个文件是 程序、 文档，
或是其他格式的数据。

各种文件存放在目录里。 一个目录可以为空， 也可以含有多个的文件。一个目录同样可以包含其他的目录，允许您在一个目录里建立多个不同层次的目录。 这将帮助您轻松地组织您的数据。

文件或目录是由文件名或目录名，加上斜线符号 /， 再根据需要在目录名后面加上其他目录的名称。
如果您有一个名为 foo 的目录， 它包含另一个目录 bar， 后者包括一个叫 readme.txt 的文件， 则全名，
或者说到文件的 路径 就是 foo/bar/readme.txt。

在文件系统里目录和文件的作用是存储数据。 每一个文件系统都有且只有一个顶级目录 根目录，
这个根目录则可以容纳其他目录。

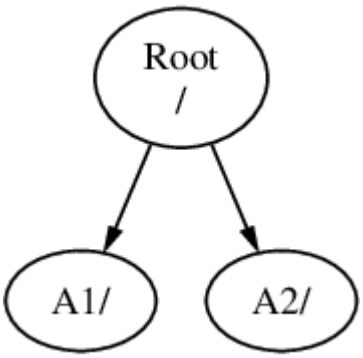
您也许在其他的一些操作系统碰到类似这里的情况， 当然也有不同的情况。 举些例子， MS-DOS® 是用 \
分隔文件名或目录名， 而 Mac OS® 则使用:。

FreeBSD在路径方面不使用驱动器名符号或驱动器名称， 在FreeBSD里您不能这样使用:
c:/foo/bar/readme.txt。

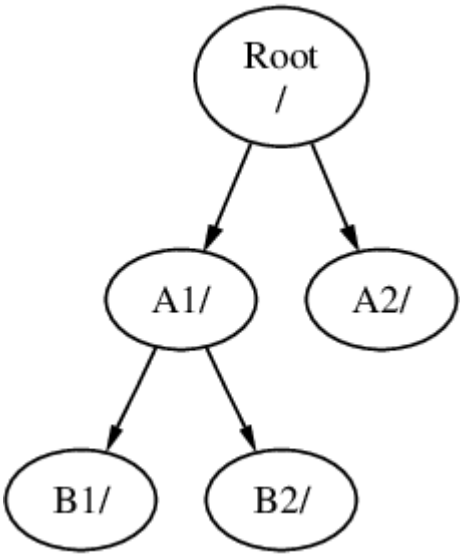
为了代替(驱动器名符号)， 一个文件系统会指定 根 文件系统， 根文件系统的根目录是 /。
其他每一个文件系统 挂接在根文件系统下。 无论有多少磁盘在FreeBSD 系统里，
每个磁盘都会以目录的方式加上。

假设您有三个文件系统， 名为 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，
名为 A1, A2 (B1, B2 和 C1, C2)。

看看 A 这个根文件系统。 假如您用 ls 命令来查看这个目录您会见到两个子目录: A1 和 A2。
这个目录树是这个样子:



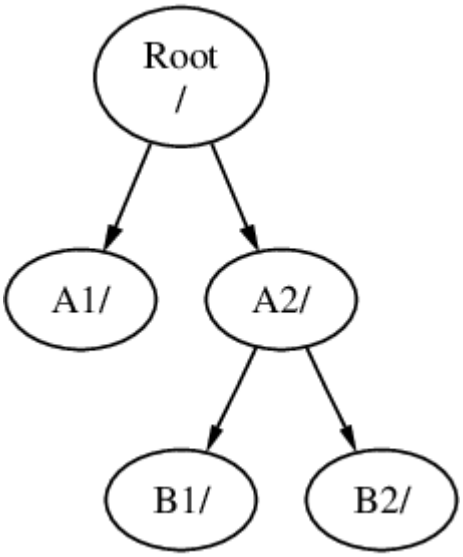
一个文件系统必须挂到另一个文件系统的某一目录， 所以现在假设把 B 文件系统挂到 A1目录， 那 B
根目录因此代替 了 A1，而显示出 B 目录(的内容):



无论B1 或 B2 目录在那里而延伸出来的路径必须为 /A1/B1 或 /A1/B2。 而在 /A1 里原有的文件会临时隐藏。
想这些文件再出现把 B 从 A 挂接释放。

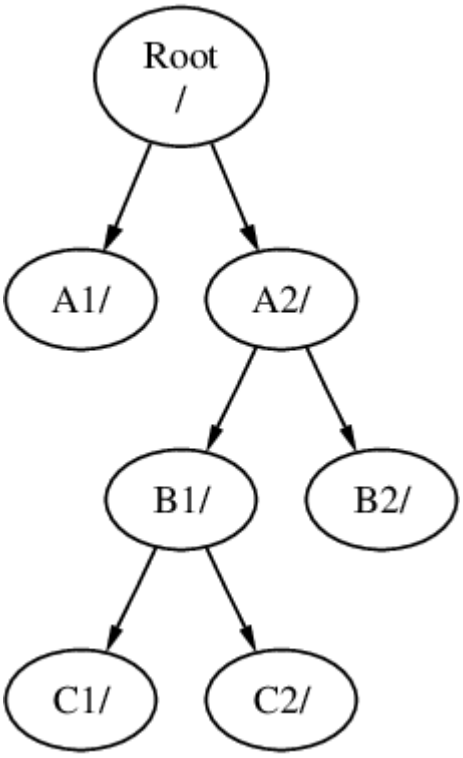
所有在B1 或 B2 目录里的文件都可以通过 /A1/B1 或 /A1/B2 访问。而在 /A1
中原有的文件会被临时隐藏，直到 B 从 A 上被卸载 (unmout) 为止。

把 B 挂接在 A2 那图表的样子就是这样子:

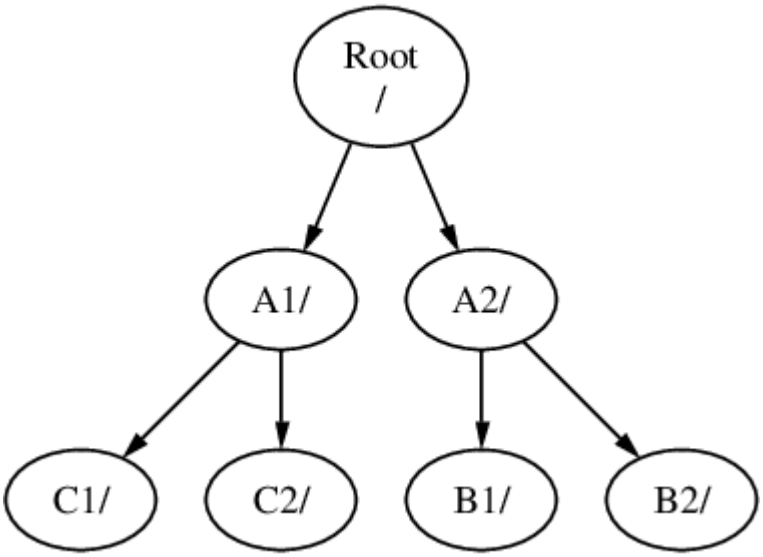


这个路径分别是 /A2/B1 和 /A2/B2 。

文件系统能把顶部挂接在另一个文件系统上。 继续这个例子， 把 C 文件系统挂接在 B 文件系统里的 B1 目录， 排列如下:



或者把 C 文件系统挂接在 A 文件系统里的A1目录：



假如您熟悉 MS-DOS® 并知道 join 命令， 尽管不相同，其实功能是相似的。

这方面不是普通知识而且涉及到您自己所关心的， 当您安装FreeBSD并在以后添加新磁盘时，
您必须知到该如何新建文件系统和挂接上。

(FreeBSD系统)它有一个主要的根文件系统， 不需要另外新建立，
但当需要手工处理时，这是一个有用的知识。

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。 举些例子， 仔细想一下， 根文件系统能用只读的方式挂接上，
  防止不经意删除或编辑到一个危险的文件。 把各用户能写入的文件系统分开， 像/home这样，
  由另外的文件系统分别用 nosuid 参数挂接，这个参数防止 suid/guid
  在执行这个文件系统中的文件时生效， 从而缓解了一些安全问题。
- FreeBSD 能根据一个文件系统使用的情况自动优化 这个文件系统上的文件布局。
  所以对一个存储了大量小文件并会被频繁写入文件系统的优化与一个存储了少量大文件的优化是不同的
  。 而在一个大的单一文件系统上则无法体现这样的优化。
- FreeBSD 的文件系统能够在断电时尽可能避免损失。 然而，
  在关键点时的电源失效仍然可能会破坏文件系统的结构。 将您的文件系统分成多个有助于分散风险，
  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时新建一个文件系统并设定一个大小，
  您会在稍后发觉到必须去建一个大的分区。 如果配置不当， 则需要备份、 重新创建文件系统，
  然后再恢复数据。

FreeBSD 提供了 growfs(8) 命令。 这使得能够实时地调整文件系统的大小，
因而不再受其限制。

文件系统是和分区一一对应的。 这里的分区和常用的术语分区 (例如， MS-DOS® 分区) 的意思并不一样，
这是由于 FreeBSD 的 UNIX® 传统造成的。 每一个分区使用一个从 a 到 h 的字母来表示。
每个分区只能包含一个文件系统， 这意味着文件系统通常可以由它们在文件系统目录结构中的挂接点，
或对应的分区字母来表示。

FreeBSD 的 交换分区 也需要使用磁盘空间。 交换分区是给 FreeBSD 作 虚拟内存 使用的，
这样能令您的计算机有更多的内存可使用， 当FreeBSD在运行而内存不够的时候，
它会把其他一些可转移的数据转移到交换分区， 空出内存的位置以供使用。

某些 partitions 的用途是确定的。

分区	约定
a	通常指定为根文件系统
b	通常指定为交换分区
c	通常它和所在的 slice 大小相同。 c 分区上工作时必定会影响到事整个 slice (举个例子，坏块扫描器)。 您通常不愿意在这个partition建立文件系统。
d	分区 d 曾经有特殊的含义， 不过这种意义在现时的系统上已不再适用， 因此 d 可以和任何其它普通的分区一样使用了。

每一个包含了文件系统的分区被保存在 FreeBSD 称为 slice 的部分上。 Slice 是一个 FreeBSD 术语，通常被叫做分区， 再次强调， 这是由于 FreeBSD 的 UNIX® 背景。 Slices 有其编号， 从1到4。

Slice 编号在设备名后面， 并有一个 s 前缀， 从 1 开始。 因此 "da0s1" 是第一个 SCSI 驱动器的第一个 slice。 每个磁盘上只能有四个物理的 slices， 但您可以在物理 slice 中使用适当的类型来创建逻辑 slice。 这些扩展 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。 文件系统所使用的设备应该占满 slice。

Slices, "专用指定" 物理驱动器， 和其他驱动器都包含 partitions， 那几个的 partitions 都是用字母从 a 到 h 来标定的， 而这些字母都在驱动器名字之后，所以 "da0a" 是指首个da设备的 a partition， 而那个就是 "专项指定"。 "ad1s3e" 是指IDE磁盘上第三个slice的第五个partition。

最终，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标识的，它不像slices，磁盘的编号是由0开始的。 对应代码请看这里所列出的磁盘设备的代码。

当在 FreeBSD 中指定 partition 名字时， 必须同时包含这个分区的 slice 和磁盘的名字； 类似地， 在指定 slice 时， 也应该给出包含该 slice 的磁盘名字。 可这样列出： 磁盘名称，s，slice 编号，和partition标定字母。 例子请看 样例磁盘, Slice, 和 Partition 它们的命名。

一个磁盘的布局 这里显示了一个磁盘的布局，有更清楚的帮助。

在安装FreeBSD时，您首先要配置好磁盘slices， 然后在FreeBSD使用的slice上建立partitions。 并在每个partition上建立一个文件系统(或交换分区)， 和指定文件系统的挂接位置。

表 6. 磁盘设备的代码

代码	说明
ad	ATAPI (IDE) 磁盘
da	SCSI 直接存取磁盘
acd	ATAPI (IDE) 光驱
cd	SCSI 光驱
fd	软驱

例 6. 样例磁盘, Slice, 和 Partition 它们的命名

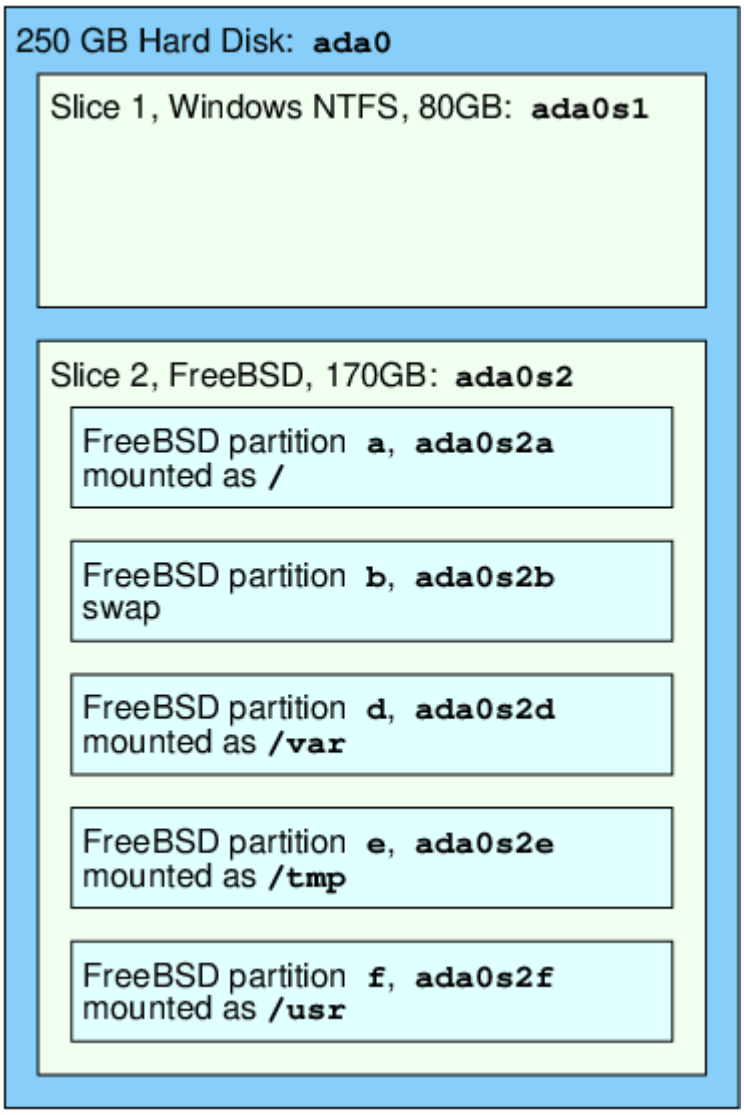
命名	说明
ad0s1a	在首个IDE磁盘(ad0)上的 第一个slice (s1)里的 第一个partition (a)。
da1s2e	在第二个SCSI磁盘(da1)上的 第二个slice(s2)里的 第五个partition(e)。

例 7. 一个磁盘的布局

从在系统里的首个IDE磁盘图表可以显示出FreeBSD的见解。 假设磁盘大小为4

GB，它里面包含了两个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和另一个交换分区。

这三个partitions各自控制一个文件系统。 partitiona 用于根文件系统， partitione 用于 /var 目录层， partitionf 用于 /usr 目录层。



4.6. 文件系统的挂接和卸下

这种文件系统就像一棵树那样用/确立根部， 是比较理想的文件系统。 而/dev、 /usr 和其他目录就是根目录的分枝， 另外这些目录可以再分枝，例如/usr/local。

应该考虑给某些目录一些空间从而分散文件系统。 /var 之下包含目录 log/，目录spool/，和不同类型的临时文件，很可能把它塞满。 把什么都塞进根文件系统不是一个好主意， 好的做法是应该把 /var 从 /分离出去。

另一个要考虑的是，给物理设备或虚拟磁盘这些自带空间的文件系统确定目录结构树。 例如 网络文件系统 或光驱的挂接。

4.6.1. fstab 文件

在 引导过程 期间， 自动挂上/etc/fstab所列出的文件系统。 (除非他们注明为noauto 选项)。

/etc/fstab 文件包含的各行的列表格式如下:

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

device
: 设备名称(设备必须存在)， 说明在 设备命名.

mount-point
: 目录 (目录必须存在)， 用在那个挂接上的文件系统上。

fstype
: 文件系统类型，请通过mount(8)查阅。 默认的FreeBSD文件系统类型是ufs。

options
: 设为可读写文件系统的rw选项， 或设为只读文件系统的ro选项， 或其他一些选项，可随意选一个。
一个常用的选项 noauto 用在不需在引导过程期间挂接的文件系统。 其他的选项在 mount(8)
手册里列出。

dumpfreq
: dump(8) 使用这项去决定那个文件系统必须移贮。 假如缺少这项，默认的数值为0。

passno
: 这一项决定文件系统的检查顺序， 文件系统想跳过检查应将passno设为0。
根文件系统(那个是在每方面开始之前必须检查的) 应该将它的 passno 设为1， 其他文件系统的 passno
必须把数值设到大于1。假如多个文件系统的passno的值相同， 那么 fsck(8)
在允许的情况下将尝试并行地去检查文件系统。

请参阅 fstab(5) 联机手册， 以获得关于 /etc/fstab 文件格式， 以及其中所包含的选项的进一步信息。

4.6.2. mount 命令

这个 mount(8) 命令是挂接文件系统的基本运用。

使用最多的基本格式:

```
# mount device mountpoint
```

它的选项非常多，而mount(8) 手册同样提及， 但常用的都在这里:

挂接的各种选项

-a
: 挂接/etc/fstab里所有列出的文件系统。 除非标记为 "noauto" 或作了排除在外的 -t
类型标记，或者在这之前已挂上。

-d
: 除了实际上系统调用以外，可以完成任何事情，这个选项是和 -v
参数一起连在一块使用，可以决定mount(8)所做的事情。

-f
: 强制去挂接一个未知的文件系统(会有危险)，
或当把一个文件系统挂接状态由可读写降为只读时，强制撤消可写通道。

-r
: 以只读方式挂接文件系统。 这和在指定了 -o 选项配合 ro 参数的效果是一样的。

-t fstype
根据给出的文件系统类型挂接文件系统， 假如给于-a选项，仅挂接这个类型的文件系统。

"ufs" 是默认的文件系统类型。

-u
在文件系统上修改挂接选项。

-v
版本模式。

-w
以可读写方式挂接文件系统。

The -o 选项采用一个逗号分开以下多个选项:

noexec
不允许文件系统上的二进制程序执行。这也是一个有用的安全选项。

nosuid
不允许文件系统上的 setuid 或 setgid 标记生效。这也是一个有用的安全选项。

4.6.3. umount 命令

umount(8) 命令同样采用一个参数、一个挂接点、一个设备名。 或采用-a选项，又或采用-A选项。

所有格式都可采用 -f 去强行卸下， 或采用-v 用那适当的版本。 但警告，采用 -f并不是一个好主意，
强行卸下文件系统可能损坏计算机或破坏文件系统上的数据。

-a 和 -A 会卸下所有已挂接的文件系， 可能通过-t后面列出的文件系统进行修改， 但无论如何，-
A都不会尝试去卸下根文件系统。

4.7. 进程

FreeBSD 是一个多任务操作系统。 这就意味着好像一次可以运行一个以上的程序。
每个占用一定时间运行的程序就叫 进程 (process)。
你运行的每一个命令会至少启动一个新进程，还有很多一直运行着的系统进程， 用以维持系统的正常运作。

每个进程用来标识的一个编号就叫 进程 ID， 或叫 PID。
而且，就像文件那样，每个进程也有所属用户和所属群体。
所属用户和所属群体使用在这方面:确定这个进程可以打开那些文件和那些设备，
从而在初期使用文件的权限。 多数的进程都有一个父进程， 而进程是依靠父进程来启动的。
例如，假如您把命令输入到shell里那shell是一个进程，而您运行的各个命令同样是进程，
那么，shell就是您各个运行进程的父进程。 而这方面有一个例外的进程就叫init(8)。
init始终是首个进程,，所以他的PID始终是1， 而init在FreeBSD起动时由内核自动启动。

在系统上，有两个命令对进程观察非常有用:ps(1) 和 top(1)。 这个ps命令作用是观察当前运行进程的状态，
显示他们的PID，使用了多少内存，它们启动的命令行。 而
top命令则是显示所有运行进程，并在以秒计的短时内更新数据。 您能交互式的观察您计算机的工作。

默认情况下， ps仅显示出您自己所运行的命令。 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/netscape-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
```

在这个例子里您可看到，从 ps(1) 输出的每一列是有规律的。 PID 就是进程ID，这个较早前已讨论过了。PID号的分配由 1一直上升直到99999， 当您运行到超过限制时，这些编号会回转分配 (仍在使用中的 PID 不会分配给其他进程)。 TT这一列显示了程序运行所在的终端， 目前可以安全地忽略。 STAT 显示程序的状态，也可以安全地被忽略。 TIME是程序在CPU处理时间-运行的时间量，
并不是指您程序启动到现在的所用的时间。
许多程序碰巧遇到某方面在他们之前要花费大量CPU处理时间时，他们就必须等候。 最后， COMMAND 是运行程序时使所用的命令行。

ps(1)支持使用各种选项去改变显示出来的内容， 最有用的一个就是auxww。
a选项显示出所有运行进程的内容， 而不仅仅是您的进程。
u选项显示出进程所归属的用户名字以及内存使用， x 选项显示出后台进程。 而 ww 选项表示为 ps(1) 把每个进程的整个命令行全部显示完， 而不是由于命令行过长就把它从屏幕上截去。

下面和从top(1)输出是类似的，一个示例式对话就象这样子:

```
% 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
...
```

这个输出分成两部份。 前面部份(起始前五行) 显示了:运行于最后进程的PID、 系统负载均衡

(那个是指系统繁忙时的调节方式)、 系统正常运行时间 (指从启动算起所用的时间) 和当前时间。
前面部份另外的图表 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。

在那下面一连串的纵列和从ps(1)输出的的内存是相似的。 如以前
ps(1)一样，您能见到:PID、用户名、CPU处理时间合计、运行的命令。
top(1)默认是显示您的进程所用内存空间的合计。
内存空间这里分成两列，一列为总体大小，另一列是必须请求驻留大小是多少内存-总体大小。
而驻留大小实际上是瞬间使用的多少。 在以上那个例子，您会看到那getenv(3)总计需要30 MB内存，
但实际只用了9 MB。

top(1) 每两秒自动刷新一次，您可以用s改变刷新的秒数。

4.8. 守护进程，信号和杀死进程

当您运行一个编辑器时它是很容易控制的，告诉它去加载文件它就加载。
您之所以能这样做，是因为编辑器提供这样便利去这样做，和因为有编辑器去附上的终端。
一些程序在运行中不需要连续的用户输入，一有机会就从终端里分离到后台去。
例如，一个web系统整天都在作web请求的响应，他不需要您输入任何东西就能完成，
这个类别的另一个例子就是把email的传送。

我们把那些程序叫 守护进程。 守护神是希腊神话中的一些人物，非正非邪，他们是些守护小精灵，
大体上为人类作出贡献。 许多类似web服务或mail服务的系统对于今天仍有用途，
这就是为什么在那么长的时间里，BSD的吉祥物保持为一双鞋加一把钢叉的守护神模样。

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 (而实际执行的程序名称则是
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。
这只是一种惯例，不是标准或硬性规定。 例如，为Sendmail而应用的主要mail守护进程就叫sendmail，
却不叫maild，这和您推测的一样。

有时可能会需要与守护进程进行通讯。 而 信号 则是其中的一种通讯机制。 可以发送信号给守护进程
(或相关的另一些进程) 来与它进行通信， 不同的信号都有自己的数字编号-其中一些有特殊的含义，
其它的则可以被应用程序自己进行解释， 而一般来说， 应用程序的文档会告诉哪些信号会被如何处理。
您只能给所属于您的进程发信号，假如您给其他人的进程发信号， 进程就会用kill(1) 或
kill(2)权限进行拒绝。 当然,root 用户会例外，它能把各种信号发送给每个进程。

在某些情况下，FreeBSD也会向应用软件发送信号。
假如一个应用软件含有恶意写入并试图去访问内存，那是不可想象的，FreeBSD会向那个进程发送 段式违规
信号 (SIGSEGV)。 假如一个应用软件使用alarm(3)系统去进行周期性调用闹钟功能，每当达到时间时，
FreeBSD会向应用软件发送闹钟信号(SIGALRM)。

有两个信号可以停止进程:SIGTERM 和 SIGKILL。 SIGTERM比较友好，进程能捕捉这个信号，
根据您的需要来关闭程序。在关闭程序之前，您可以结束打开的记录文件和完成正在做的任务。
在某些情况下， 假如进程正在进行作业而且不能中断，那么进程可以忽略这个 SIGTERM信号。

对于SIGKILL信号，进程是不能忽略的。 这是一个 '"我不管您在做什么,立刻停止"'的信号。
假如您发送SIGKILL信号给进程， FreeBSD就将进程停止在那里。.

您可能会去使用 SIGHUP、 SIGUSR1 和 SIGUSR2信号。 这都是些通用的信号，各种应用程序都可以应用
在各方面的信号发送。

假如您改变了web系统的配置文件-并想web系统去重读它的配置， 您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，那样是不受欢迎的。几乎所有的守护进程在编写时，都会指定对SIGHUP
信号进行响应从而重读配置文件。 所以， 最好的方法， 就不是杀死并重启 httpd， 而是发一个 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，所以不了解时应读一下守护进程的文档。

发送信号可用kill(1) 命令， 请参考kill(1)所列出的例子。

Procedure: 发送一个信号给进程

这个例子显示了怎样去发一个信号给inetd(8)。 inetd配置文件是/etc/inetd.conf， 如果想inetd去重读文件系统的话，可以给它发一个SIGHUP 信号。

1. 寻找您要发送信号的进程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，搜索您指定的字符串。
   这命令是由普通用户来执行的，而inetd(8)是root用户运行的， 所以必须给ps(1)带上ax选项。

   ```
   % ps -ax | grep inetd
   198  ??  IWs    0:00.00 inetd -wW
   ```

   得出 inetd(8) PID号是198。 有时 grep inetd 命令也出现在输出中， 这是因为在这方面 ps(1) 也是寻找列表中运行进程。

2. 使用 kill(1) 去发送信号。 因为 inetd(8) 是由 root启动的， 您必须使用 su(1) 去 变为 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, 就不会给出任何消息。
   假如您发送信号给一个不属于您的进程， 您会看到 kill: PID: Operation not permitted.
   假如输错了PID号，把信号发送到其他进程，那是坏事。
   或者您侥幸，把信号发送到不存在的进程， 您会看见 kill: PID: No such process.

为什么使用 /bin/kill?
许多shell提供了内建 kill 命令， 这样， shell就能直接发送信号，而不是运行 /bin/kill。 这点非常有用， 但不同shell有不同的语法来指定发送信号的名字， 与其试图把它们学完倒不如简单地直接使用 /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。可以运行 /bin/kill -s KILL 1 命令来让系统迅速关机。
当您按下 Return （回车）键之前， 一定要 详细检查您运行 kill(1) 时所指定的参数。

4.9. Shells

在FreeBSD里，每日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。
许多的shell同样能帮我们完成内建的每日功能，例如:文件管理、文件寻找、命令行编辑、
宏指令和环境变量。FreeBSD内含了一些shell，例如:sh、Bourne Shell、 tcsh和改良过的C-shell。
另外也有些shell也可在FreeBSD的Ports得到，例如:zsh和bash。

您想使用哪一种shell取决于您的喜好， 假如您是C程序设计师，您可能选择一个C-like shell例如tcsh。
假如您是从Linux过来的或是一个命令行的新手，您可能会试一下bash。
这一点告诉我们每一个shell都有各自的特性，可能适用于您的工作环境，也可能不适用于您的工作环境。

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 键，就能靠shell的自动补全功能得出 命令或文件名。这里有一个例子，假设您有两个文件叫 foobar 和foo.bar，而您想删除 foo.bar， 可这样在键盘上输入 rm fo[Tab].[Tab]。

那么shell就会输出 rm foo[BEEP].bar。

这个[BEEP] 是这控制台铃声， 那个是告诉我们它不能完成文件名补全，因为有多个文件名符合。 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。

另一个特点就是shell利用环境变量运行。环境变量是贮存在shell环境空间上相对应的键和可变值，这个空间能够补程序从shell里读出，而且包含了许多程序的配置。
这个一个常用环境变量列和其含义的列表：

变量	说明
USER	当前登录进入的用户名。
PATH	搜索程序路径，以两点的冒号分隔开。
DISPLAY	假如有这个变量的话，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，通常用在确定终端的能力。
TERMCAP	各种终端功能所用终端分离编码的基本数据项目。
OSTYPE	操作系统类型，默认是FreeBSD。
MACHTYPE	是指系统上运行的CPU体系结构。
EDITOR	用户首选的文本编辑器。
PAGER	用户首选的文本页面调度程序 。
MANPATH	搜索联机手册路径，以两点的冒号分隔开。

不同的shell设置环境变量也不相同。举个例子， 在如tcsh 和 csh这样的C-Style shell， 您必须使用 setenv去设置环境变量。 而在如sh和bash这样的Bourne shell， 您必须使用export去设置当前环境变量。
再举个例子，要去设置或改变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"
```

您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。 举个例子，用echo $TERM 就会显示出$TERM的设定值， 其实就是shell取得$TERM并传给echo来显示的。

shell里有许多特别的字符代表着特别的资料，我们把叫做meta-characters。 最常用的就是 *字符，它可代表文件名的任何字符。 这些特别字符应用到文件名全域方面。假如，输入 echo *和输入 ls的效果是相同的，其实就是 shell 取得了全部符合 *的文件名，并传给 echo 在命令行下显示出来。

为了防止shell去分析这些特别字符， 我们可在它之前加一个 \字符去说明它只是普通字符。 echo $TERM就会显示出您的终端情况， 而 echo \$TERM 就会显示出 $TERM 这几个字。

4.9.1. 改变您用的Shell

改变您的Shell的最简单方法是使用 chsh 命令。 执行 chsh 将根据您设定的EDITOR 环境变量进入到那个编辑器，假如没有设定，就会进入vi编辑器。 请改变"Shell:"这行对应值。

您可使用chsh 的-s选项， 这样就能设置您的shell却又不用编辑器。假如您想把shell改为bash 可用下面的技巧。

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

您使用的shells必须 在/etc/shells 文件里列出。 假如您从 ports里装一个shell，那就不用做这步了。 假如您手工装一个shell，那就要手工添加进去。

举个例了子，假如您手工把 bash装到 /usr/local/bin里，您还要进行这一步:

```
# echo "/usr/local/bin/bash" >> /etc/shells
```

然后运行chsh。

4.10. 文本编辑器

FreeBSD 的很多配置都可以通过编辑文本文件来完成。 因此， 最好能熟悉某种文本编辑器。 FreeBSD 基本系统中提供了一些， 您也可以从 Ports Collection 安装其它编辑器。

最容易学的而又简单的编辑器是 ee编辑器， 是个标准的简易编辑器。 要启动 ee，首先就要在命令行输入 ee filename， filename 是一个要编辑的文件名。 例如，要编辑 /etc/rc.conf就要输入 ee /etc/rc.conf，在 ee的控制内， 编辑器所有功能的操作方法都显示在最上方。 这个^ 字符代表 键盘上的 Ctrl 键， 所以^e 就是 Ctrl + e 组合键。 假如想离开ee， 按 Esc 键，就可选择离开编辑器。
当您修改了内容的时候，编辑器会提示您保存。

FreeBSD本身也带许可多有强大功能的文本编辑器， 例如 vi。还有其他在FreeBSD Ports里几种， 像 emacs 和 vim。 这些编辑器有着强大的功能，但同时学习起来比较复杂。
不管怎样，假如您从事文字编辑方面的工作， 学习如vim 或 emacs 这些有强大功能的编辑器用法，在长时间工作里会帮您节省不少的时间。

很多需要修改文件或打字输入的应用程序都会自动打开一个文本编辑器。 更改默认使用的编辑器， 请设置 EDITOR 环境变量。 参阅 shells 以获取更多详细信息。

4.11. 设备和设备节点

在一个系统里，硬件描述通常用法就是一个设备对应一个术语，包括磁盘、打印机、显卡和键盘。 当 FreeBSD 启动过程中，大多数的设备都能探测到并显示出来， 您也可以查阅/var/run/dmesg.boot，引导时所有信息都在里面。

例如， acd0 就是 首个 IDE 光盘设备， 而 kbd0 则代表键盘。

在UNIX®操作系统里，大多数设备存在的特殊访问文件就是叫做设备节点， 他们都定位在/dev目录里。

4.11.1. 建立设备节点

当在系统中添加新设备或将附加设备的支持编译进内核之后， 都必须为其建立设备节点。

4.11.1.1. DEVFS (DEVice 文件系统)

这个设备文件系统， 或叫 DEVFS， 为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节点， DEVFS只是为您的特别文件系统进行维护。

请参见 devfs(5) 联机手册以了解更多细节。

4.12. 二进制文件格式

要理解为什么 FreeBSD 使用 elf(5) 格式， 您必须首先了解一些 UNIX® 系统中的 三种 "主要" 可执行文件格式的有关知识:

- a.out(5)

  是最古老和"经典的" UNIX® 目标文件格式， 这种格式在其文件的开始处有一个短小而又紧凑的首部，

该首部带有一个魔幻数字，用来标识具体的格式(更多详情参见a.out(5))。
这种格式包含3个要装载入内存的段：.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。其文件头现在包括一个区段表(section table)，
  因此除了.text，.data，和.bss区段以外，您还可以包含其它的区段。

- elf(5)

  COFF 的后继， 其特点是可以有多个区段， 并可以使用32位或64位的值。 它有一个主要的缺点： ELF
  在其设计时假设每个系统体系结构只有一种 ABI。 这种假设事实上相当错误，
  甚至在商业化的SYSV世界中都是错误的 (它们至少有三种ABI: SVR4, Solaris, SCO)。

  FreeBSD试图在某种程度上解决这个问题，它提供一个工具，可以 对一个已知的ELF可执行文件
  标识它所遵从的ABI的信息。 更多这方面的知识可以参见手册页brandelf(1)

FreeBSD从"经典"阵营中来，因此使用了a.out(5)格式，
众多BSD版本的发行(直到3.X分支的开始)也证明了这种格式的有效性。
虽然在那以前的某段时间，在FreeBSD系统上创建和运行ELF格式
的二进制可执行文件(和内核)也是可能的，但FreeBSD一开始并不积极"进步"
到使用ELF作为其缺省的格式。为什么？噢，当Linux阵营完成了
转换到ELF格式的痛苦历程后，却发现并不足以由此而放弃
a.out可执行文件格式，因为正是由于它们不灵活的，
基于跳转表的共享库机制，使得销售商和开发者们构建共享库非常困难。
直到已有的ELF工具提供了一种解决共享库问题的办法，
并被普遍认为是"前进方向"以后，迁徙的代价在FreeBSD界才被接受，
并由此完成了迁徙。FreeBSD的共享库机制其基础更类似于Sun SunOS™的共享库机制，
并且正因为此，其易用性很好。

那么，为什么会有这么多不同的格式呢?

回溯到蒙昧和黑暗的过去，那时只有简单的硬件。这种简单的硬件支撑了一个简单
和小型的系统。在这样的简单系统上(PDP-11)a.out格式
足以胜任表达二进制文件的任务。当人们将UNIX®从这种简单的系统中移植出来的时候，
a.out格式被保留了下来，因为对于早期将UNIX®移植到 Motorola
68k，VAXen等系统来说，它还是足够可用的。

然后，一些聪明的硬件工程师认为，如果可以让软件完成一些简单的聪明操作，
那么他们就可以在硬件设计中减少若干门电路，并可以让CPU核心运行得更快。 当
a.out格式用于这种新型的硬件系统时(现在我们叫它
RISC)，显得并不合适。因此，人们设计了许多新的格式 以便在这样的硬件系统上能获得比简单的
a.out格式更优越 的性能。诸如COFF，ECOFF，还有其它
一些晦涩难懂的格式正是在这个阶段被发明出来的，人们也研究了这些格式的局限性，
慢慢地最终落实到ELF格式。

同时，程序的大小变得越来越大，磁盘空间(以及物理内存)相对来说却仍然较小，
因此共享库的概念便产生了。VM系统也变得越来越复杂了。当所有这些进步都建立在
a.out格式的基础上的时候，它的可用性随着每个新特性
的产生就受到了严重考验。并且，人们还希望可以在运行时动态装载某些东西，或者
在初始化代码运行以后可以丢弃部分程序代码，以便节约主存储器和交换区。编程语言
也变得越来越复杂，人们希望可以在main()函数执行之前自动执行某些代码。为了实现
所有这些功能，人们对a.out格式作了很多改动(hack)，
他们在某个阶段里基本也是可行的。随着时间的推移，a.out格式
不得不增加大量的代码和复杂度来满足这些需求。虽然ELF格式
解决了许多这样的问题，但是从一个可用的系统迁移到另一个系统却是痛苦的。因此 直到继续保留
a.out格式的代价比迁移到ELF格式 的代价还大的时候，人们才会最终转换到ELF格式。

然而，随着时间的推移，FreeBSD系统本身的编译工具(特别是汇编器和装载器)
赖以派生的编译工具，其发展却形成了两个平行的分支。FreeBSD这个分支增加了共享库，
并修改了一些错误。而原先编写了这些工具的GNU人则重写了这些工具，并对交叉编译提供了
更简化的支持，还随意插入了不同格式的支持，等等。虽然很多人希望创建针对FreeBSD的
交叉编译器，但他们却并未如愿以偿，因为FreeBSD的as 和ld的源代码更为老旧，所以无法完成这个任务。

新的GNU工具链(binutils)则确实支持交叉编译，ELF
格式，共享库，C++扩展，等等。并且，由于很多供应商都发布ELF格式的
二进制文件，因而让FreeBSD能够运行它们将是一个很好的事情。

ELF格式比a.out格式开销要大些，同时也
允许基础系统有更好的扩展性。ELF格式的有关工具有着更好的维护，
并且提供交叉编译支持，这对许多人来说是很重要的。ELF格式可能会稍微
慢一些，但很难测量出来。另外，在这两者之间，有许多细节也是不同的，比如它们映射页面的方式，
处理初始化代码的方式，等等。所有这些都不太重要，但这也确实是不同之处。在将来的适当时候，
GENERIC内核将不再支持a.out格式，并且， 当不再需要运行遗留的
a.out格式程序时，内核也将不再提供对其的支持。

4.13. 取得更多的资讯

4.13.1. 联机手册

最详细的使用说明文档莫过于 FreeBSD 里的联机手册了。 几乎每一个程序都会附上一份简短说明，
以介绍这个程序的的基本功能以及参数的用法。 我们能通过 man 命令来阅读这些说明， 而使用 man
命令却是简单的事情:

```
% man command
```

command 就是您要了解的命令命称。 举个例子，想了解 ls 命令就输入:

```
% man ls
```

这些在线手册分下列章节:

1. 用户命令。
2. 系统调用以及错误代码。
3. C 库文件里的函数说明。
4. 设备驱动程序。
5. 文件格式。
6. 游戏以及其他娱乐。
7. 各种资讯。
8. 系统维护以及命令。
9. 内核开发情况。

在某些情况下，同样的主题也会出现在在线手册的不同章节。 举个例子，系统里有
chmod这个用户命令，而又有个 chmod() 系统调用。 在这种情形下，您应当向 man 命令指定需要的内容:

```
% man 1 chmod
```

这样就会显示出手册里的用户 chmod 命令。
传统上，我们在写入文档时把特定详细参考内容在在线手册括号里注明。 所以 chmod(1) 是指 chmod
用户命令， 而 chmod(2) 是指系统调用。

如果您已经知道命令的名字，只是不知道要怎样使用的话，那就比较好办。
但您连名字都不知道呢?这个时候您就可以利用 man 的搜寻功能，
它会在手册的介绍部份找寻您要搜寻的关键字，它的选项是 -k:

```
% man -k mail
```

当您使用这个命令的时候，man会把介绍里含有"mail"关键字 的命令列出来，实际上这和apropos命令的功能是相同的。

有时您会看到/usr/bin 下有许多命令但不知他们的用途， 您只需这样做:

```
% cd /usr/bin
% man -f *
```

或者这样做

```
% cd /usr/bin
% whatis *
```

两个命令是一样的。

4.13.2. GNU Info 文件

FreeBSD许多应用软件以及实用工具来自Free软件基金会(FSF)。
作为手册的扩充，这些程序提供了一种更具有活力的超文档说明info， 您可用info命令来阅读他们。
假如您装上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
```

想得到简单介绍， 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 将许多系统工具捆绑作为基本系统的一部分。 然而， 要完成实际的工作，
可能还需要安装更多的第三方应用。 FreeBSD 提供了两种补充的技术，
用以在您的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
(用以从预编译的二进制版本安装)。 这两种方法都可以用于从本地介质，
或从网上直接安装您喜欢的应用程序的最新版本。

读完这章，您将了解到:

- 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。
- 如何改动Ports Collection里面的一些参数，定制软件使用。
- 如何找到您需要的软件包。
- 如何升级您的应用软件。

5.2. 软件安装预览

如果您以前使用过 UNIX® 系统，那典型的第三方软件安装的步骤是像下面描述的：

1. 下载这个软件，软件的发行版可能是源代码格式，或是一个二进制包。
2. 解开软件(其中代表性的是用 compress(1), gzip(1), 或 bzip2(1) 压缩过的tar包)。
3. 阅读相关文档，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。可能需要编辑一个 Makefile文件, 或运行
   configure脚本，和其他的一些工作。
5. 测试和安装软件。

如果一切顺利的话，就这么简单。如果您在安装一个软件包时发生一些错误，
您可能需要编辑一下它的代码，以使它能正常工作。

您可以继续使用 "传统的"方式安装软件。 然而， FreeBSD 提供了两种技术： packages 和 ports。
就在写这篇文章的时候， 已经有超过 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。这个 FreeBSD
package包含了编译好的的副本， 还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。
可以使用一个简单的命令安装一个新的应用程序。

一个FreeBSD的port是一个可以自动从源代码编译成应用程序的文件集合。

记住，如果您自己来编译的话，需要执行很多步的操作 (解压， 补丁， 编译， 安装)。 这些整理 port
的文件集合包含了系统需要完成这个工作的必需信息。 您可以运行一些简单的命令，
那些源代码就可以自动地下载， 解开， 打补丁， 编译， 直至安装完成。

实际上，ports 系统也能做出被 pkg_add 的程序包和不久就要讲到的其他包管理命令来安装的软件包。

Packages 和 ports 是互相 依赖 的。 假设您想安装一个依赖于已经安装的特定库的应用程序。
应用程序和那个库都已经应用于 FreeBSD ports 和 packages。 如果您使用 命令或 ports
系统来添加应用程序， 两个都必须注意库是否被安装， 如果没有， 它会自动先安装库。

这里给出的两种技术是很相似的，您可能会奇怪为什么 FreeBSD 会弄出这两种技术。 其实， packages 和 ports 都有它们自己的长处， 使用哪一种完全取决于您自己的喜好。

Package Benefits

- 一个压缩的 package 通常要比一个压缩的包含源代码的应用程序小得多。
- package 不需要进行额外的编译。 对于大型应用程序如 Mozilla， KDE 或 GNOME 来说这显得尤为重要， 特别是在您的系统资源比较差的情况下。
- package不需要您知道如何在FreeBSD上编译软件的详细过程。

Ports Benefits

- package 在编译时通常使用比较保守的选项， 这是为了保证它们能够运行在大多数的系统上。 通过从 port 安装， 您可以细微调整编译选项来产生适合于处理器的代码 (针对于 Pentium 4 或 AMD 的 Athlon CPU)。
- 一些软件包已经把与它们相关的能做和不能做的事情的选项都编译进去了。 例如， Apache 可能就配置了很多的选项。 从 port 中安装时， 您不一定要接受默认的选项， 可以自己来设置。

  在一些例子中，一个软件有不同的配置存在多个package。 例如， Ghostscript存在 ghostscript package 和 ghostscript-nox11 package两个配置package， 这取决于您是否安装了X11服务器。 这样的调整对package是可能的， 但如果一个应用程序有超过一个或两个不同的编译时间选项时， 就不行了。

- 一些软件的许可条件禁止采用二进制形式发行。 它们必须带上源代码。
- 一些人不信任二进制发行形式。 至少有了源代码， (理论上) 可以亲自阅读它，寻找潜在的问题。
- 如果您要自己对软件打补丁，您就需要有源代码。
- 一些人喜欢整天围着源代码转， 所以他们喜欢亲自阅读源代码， 修改源代码等等。

保持更新 ports， 订阅邮件列表 FreeBSD ports 邮件列表 和递交错误报告 FreeBSD ports bugs 邮件列表。

安装任何应用程序之前， 应首先检查 http://vuxml.freebsd.org/ 上是否有关于您所安装的应用程序的安全问题报告。

您也可以安装 ports-mgmt/portaudit， 它能够自动地检查已经安装的应用程序的漏洞; 此外， 在您安装程序之前它也会首先检查是否存在已知的漏洞。 另外， 您也可以使用 portaudit -F -a 这个命令在安装了某个软件包之后作出检查。

这章的其余部分将介绍在 FreeBSD 上如何使用 packages 和 ports 来安装和管理第三方软件。

5.3. 寻找您要的应用程序

在您安装任何应用程序之前，需要知道您需要什么，那个应用程序叫什么。

FreeBSD中可用的应用程序正在不断地增长着。幸运的是， 有许多方法可以找到您所需要的程序:

- 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/。 它分很多种类，您既可以通过程序的名称来搜索(如果您知道名字)， 也可以在分类中列出所有可用的应用程序。
- Dan Langille 维护着网站 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时，他们就会发 email 提醒您。
- 如果您不知道您想要的应用程序的名字，可以通过 (http://www.freshmeat.net/) 网站来查找， 如果找到了应用程序， 您可以回 FreeBSD 的主站去看一下这个应用程序是否已经被 port 进去了。
- 如果您知道一个port的准确名字， 但需要知道在哪个类别里面能找到它，您可以使用 whereis(1) 这个命令。简单地输入 whereis file， file 就是您想安装的程序名字。 如果系统找到了它， 您将被告知在它在哪里， 例如:

```
# whereis lsof
lsof: /usr/ports/sysutils/lsof
```

结果告诉我们这个命令lsof (一个系统配置程序)可以在 /usr/ports/sysutils/lsof目录中找到。

- 你可以使用简单的 echo(1) 语句来查找某个 port 是否存在于 ports 树中。 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.

请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。

- 还有另外的一个寻找您需要的port的方法—是用ports collecton 内嵌的搜索机制。要使用这个搜索, 您需要先到 /usr/ports目录下面。 在那个目录里面， 运行make search name=program-name， program-name 就是您想寻找的程序名字。 举个例子， 如果您想找 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:
```

在输出的内容里面您要特别注意包含 "Path:" 的这行将告诉您在哪里可以找到这个 port。 如果要安装此 port， 那其他输出的信息不是必须的， 但是还是显示输出了。

为了更深入的搜索，您还可以用 make search key=string， string就是您想搜索的部分内容。 它将搜索port的名字、 注释， 描述和从属关系， 如果您不知道您想搜索的程序名字， 可以利用它搜索一些关键主题来找到您需要的。

上面说的这些方法， 搜索的关键字没有大小写区分的。 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 系统上有几种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经安装的预编译软件包的任务。 如欲了解进一步信息， 请参阅 安装预编译的软件包 (package)。
- 这一节余下的部分将介绍用于管理预编译软件包的命令行工具。

5.4.1. 一个 package 的安装

您可以用 pkg_add(1) 这个命令从本地文件或网络上的服务器来安装一个 FreeBSD 软件包。

例 8. 在本地手动下载一个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
```

如果您没有本地package的安装盘 (如 FreeBSD CD-ROM)， 可以执行 pkg_add(1) 命令并加上 -r 选项。 这将迫使程序自动决定目标文件的正确格式和版本， 然后自动从一个 FTP 站点寻找和安装 package。

```
# pkg_add -r lsof
```

上面的例子将下载正确的package， 而不需要用户的干预就可以安装。 如果您想指定 FreeBSD package 的镜像站点， 替换主站点， 就必须相应地设置 PACKAGESITE 这个环境变量， 覆盖原来的设置。 pkg_add(1) 使用 fetch(3) 下载文件， 可以使用多种环境变量， 包含 FTP_PASSIVE_MODE、 FTP_PROXY， 和 FTP_PASSWORD。 如果您使用 FTP/HTTP 代理或在防火墙后面， 您可能需要设置这些环境变量。 详细的列表请参考 fetch(3)。上述例子中用 lsof 替代了 lsof-4.56.4。 当使用远程安装 Package 的时候软件名字不需要包含版本号。 pkg_add(1) 将自动的找到这个软件最新的版本。

如果您使用 FreeBSD-CURRENT 或 FreeBSD-STABLE版本的FreeBSD， pkg_add(1) 将下载您的应用软件的最新版本。 如果您使用 -RELEASE 版本的 FreeBSD， 它将会获得与您的版本相应的软件包版本。 您可以通过修改环境变量 PACKAGESITE 来改变这一行为。 例如， 如果您运行 FreeBSD 8.1-RELEASE 系统， 默认情况下 pkg_add(1) 将尝试从 ftp://ftp.freebsd.org/pub/FreeBSD/ports/i386/packages-8.1-release/Latest/ 下载预编译的软件包。 如果您希望强制 pkg_add(1) 下载 FreeBSD 8-STABLE 的软件包， 则可以将 PACKAGESITE 设置为 ftp://ftp.freebsd.org/pub/

FreeBSD/ports/i386/packages-8-stable/Latest/。

软件包采用 .tgz 和 .tbz 两种格式。您可以在 ftp://ftp.FreeBSD.org/pub/FreeBSD/ports/packages/ 下面或从 FreeBSD 的发行光盘找到， 它在每一个 4CD 的 FreeBSD 发行版的 /packages目录中。
软件包的设计规划与 /usr/ports 树一致。 每个分类都有自己的目录， 所有的软件包可以在目录 All中找到。

软件包系统的目录结构与ports的设计规划一致； 它们共同构成了整个 package/port。

5.4.2. 软件包的管理

pkg_info(1) 是用于列出已安装的所有软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)是一个用来统计所有安装的软件包版本的工具。 它可以用来比较本地 package 的版本与 ports 目录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关时间和本地ports目录树中可用的版本。

符号	含义
=	在本地ports树中与已安装的软件包版本相匹配。
<	已安装的版本要比在ports树中的版本旧。
>	已安装的版本要比在ports树中的版本新 (本地的port树可能没有更新)。
?	已安装的软件包无法在ports索引中找到。 (可能发生这种事情，举个例子， 您早先安装的一个 port 从 port 树中移出或改名了)
*	软件包有很多版本。
!	已安装的软件包在索引中存有记录， 但是由于某些原因 pkg_version 无法比较已安装的软件包与索引中相对应的版本号。

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； 如果您只是指定了类似 xchat 而不是 xchat-1.7.1 这样的名字， 则它将拒绝执行操作。 不过， 您可以使用 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\*
```

这时， 所有名字以 xchat 开头的 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关文件中找到。

5.5. 使用Ports Collection

下面的几个小节中， 给出了关于如何使用 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

在您能使用 ports 之前， 您必须先获得 Ports Collection - 本质上是 /usr/ports 目录下的一堆 Makefile、补丁和描述文件。

在您安装 FreeBSD 系统的时候， sysinstall 会询问您是否需要安装 Ports Collection。 如果您选择 no，那您可以用下面的指令来安装 Ports Collection：

Procedure: CVSup 方法

保持您本地 Ports 套件最新的一种快捷的方法， 是使用 CVSup 协议来进行更新。
如果您希望了解更多关于 CVSup 的细节， 请参见 使用 CVSup。

在 FreeBSD 系统里对 CVSup 的实现叫作 csup。

在首次运行 csup 之前， 务必确认 /usr/ports 是空的！ 如果您之前已经用其他地方安装了一份 Ports 套件， 则 csup 可能不会自动删除已经在上游服务器上删除掉的补丁文件。

1. 运行 csup:

```
# csup -L 2 -h cvsup.FreeBSD.org /usr/shared/examples/cvsup/ports-supfile
```

将 cvsup.FreeBSD.org 改为离您较近的 CVSup 服务器。 请参见 CVSup 镜像 (CVSup 站点) 中的镜像站点完整列表。

有时可能希望使用自己的 ports-supfile， 比如说，
不想每次都通过命令行来指定所使用的 CVSup 服务器。

a. 这种情况下， 需要以 root 身份将 /usr/shared/examples/cvsup/ports-supfile 复制到新的位置， 例如 /root 或您的主目录。
b. 编辑 ports-supfile。
c. 把 CHANGE_THIS.FreeBSD.org 修改成离您较近的 CVSup 服务器。 可以参考 CVSup 镜像 (CVSup 站点) 中的镜像站点完整列表。
d. 接下来按如下的方式运行 csup：

```
# csup -L 2 /root/ports-supfile
```

2. 此后运行 csup(1) 命令将下载最近所进行的改动， 并将它们应用到您的 Ports Collection 上，
不过这一过程并不重新联编您系统上的 ports。

Procedure: Portsnap 方式

Portsnap 是用于发布 Ports 套件的另一套系统。 请参阅 使用 Portsnap 以了解关于 Portsnap
功能更详细的介绍。

1. 下载压缩的 Ports 套件快照到 /var/db/portsnap。 您可以根据需要在这之后关闭 Internet 连接。

```
# portsnap fetch
```

2. 假如您是首次运行 Portsnap， 则需要将快照释放到 /usr/ports：

```
# portsnap extract
```

如果您已经有装好的 /usr/ports 而您只想更新， 则应执行下面的命令：

```
# portsnap update
```

Procedure: Sysinstall 方式

这种方法需要使用 sysinstall 从安装介质上安装 Ports 套件。 注意， 安装的将是发布发行版时的旧版
Ports 套件。 如果您能访问 Internet， 应使用前面介绍的方法之一。

1. 以 root 身份运行 sysinstall：

```
# sysinstall
```

2. 用光标向下选择 Configure， 并按 Enter。
3. 向下并选择 Distributions， 按 Enter。
4. 选择 ports， 并按 Space。
5. 选择 Exit， 并按 Enter。
6. 选择所希望的安装介质， 例如 CDROM、 FTP， 等等。
7. 选择 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"。 简单地说， port skeleton
是让一个程序在 FreeBSD 上简洁地编译并安装的所需文件的最小组合。 每个 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。这个文件包括这些信息： 这些文件用来对下载后的文件校验和进行检查 (使用
sha256(1))， 来确保在下载过程中文件没有被破坏。

- 一个 files 目录。 这个目录包括在 FreeBSD 系统上编译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， 指出特定文件作了哪些修正。
  它们都是纯文本的的格式，基本上是这样的 "删除第 10 行" 或 "将第 26 行改为这样 …"，
  补丁文件也被称作 "diffs"， 他们由 diff(1) 程序生成。

  这个目录也包含了在编译 port 时要用到的其它文件。
- 一个 pkg-descr 文件。 这是一个提供更多细节，有软件的多行描述。
- 一个 pkg-plist 文件。 这是即将被安装的所有文件的列表。它告诉 ports
  系统在卸载时需要删除哪些文件。

一些ports还有些其它的文件， 例如 pkg-message。 ports 系统在一些特殊情况下会用到这些文件。
如果您想知道这些文件更多的细节以及 ports 的概要， 请参阅 FreeBSD Porter’ s Handbook。

port里面包含着如何编译源代码的指令， 但不包含真正的源代码。 您可以在网上或 CD-ROM
上获得源代码。 源代码可能被开发者发布成任何格式。 一般来说应该是一个被 tar 和 gzip 过的文件，
或者是被一些其他的工具压缩或未压缩的文件。 ports中这个程序源代码标示文件叫 "distfile"， 安装
FreeBSD port的方法还不止这两种。

您必须使用 root 用户登录后安装 ports。

在安装任何 port 之前， 应该首先确保已经更新到了最新的 Ports Collection， 并检查
http://vuxml.freebsd.org/ 中是否有与那个 port 有关的安全问题。

在安装应用程序之前， 可以使用 portaudit 来自动地检查是否存在已知的安全问题。
这个工具同样可以在 Ports Collection (ports-mgmt/portaudit) 中找到。 在安装新的
port 之前， 可以考虑先运行一下 portaudit -F 来抓取最新的漏洞数据库。
在每天的周期性系统安全检察时， 数据库会被自动更新， 并且会在这之后实施安全审计。
欲了解进一步的情况，请参阅 portaudit(1) 和 periodic(8)。

Ports 套件假定您有可用的 Internet 连接。 如果您没有， 则需要将 distfile 手工放到 /usr/ports/distfiles
中。

要开始操作， 首先进入要安装 port 的目录：

```
# cd /usr/ports/sysutils/lsof
```

一旦进入了 lsof 的目录，您将会看到这个port的结构。 下一步就是 make，或说 "联编" 这个 port。
只需在命令行简单地输入 make 命令就可轻松完成这一工作。 做好之后，您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，一旦编译完成，您就会回到命令行。 下一步安装 port， 要安装它只需要在 make 命令后跟上一个单词 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.
#
```

一旦您返回到提示符，您就可以运行您刚刚安装的程序了。因为 lsof 是一个赋予特殊权限的程序，因此显示了一个安全警告。 在编译和安装 ports 的时候， 您应该留意任何出现的警告。

删除工作目录是个好主意， 这个目录中包含了全部在编译过程中用到的临时文件。这些文件不仅会占用宝贵的磁盘空间， 而且可能会给升级新版本的 port 时带来麻烦。

```
# make clean
===>  Cleaning for lsof-4.57
#
```

使用 make install clean 可以一步完成 make、 make install 和 make clean 这三个分开的步骤的工作。

一些 shell 会缓存环境变量 PATH 中指定的目录里的可执行文件，以加速查找它们的速度。 如果您使用的是这类 shell， 在安装 port 之后可能需要执行 rehash 命令， 然后才能运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， 对应的命令是 hash -r。 请参见您的 shell 的文档以了解进一步的情况。

某些第三方 DVD-ROM 产品， 如 FreeBSD Mall 的 FreeBSD Toolkit 中包含了 distfiles。 这些文件可以与 Ports 套件配合使用。 将 DVD-ROM 挂接到 /cdrom。 如果您使用不同的挂接点， 则应设置 make 变量 CD_MOUNTPTS。 如果盘上有需要的 distfiles， 则会自动使用。

请注意， 少数 ports 并不允许通过 CD-ROM 发行。
这可能是由于下载之前需要填写注册表格， 或者不允许再次发布， 或者有一些其它原因。
如果您希望安装在 CD-ROM 上没有的 port， 就需要在线操作了。

ports 系统使用 fetch(1) 去下载文件， 它有很多可以设置的环境变量， 其中包括 FTP_PASSIVE_MODE、FTP_PROXY， 和 FTP_PASSWORD。 如果您在防火墙之后，或使用 FTP/HTTP代理，
您就可能需要设置它们。 完整的说明请看 fetch(3)。

当使用者不是所有时间都能连接上网络， 则可以利用 make fetch。 您只要在顶层目录 (/usr/ports) 下运行这个命令， 所有需要的文件都将被下载。 这个命令也同样可以在下级类别目录中使用， 例如: /usr/ports/net。 注意， 如果一个port有一些依赖的库或其他 port， 它将 不 下载这些依赖的 port 的 distfile 文件， 如果您想获取所有依赖的 port 的所有 distfile， 请用 fetch-recursive 命令代替 fetch命令。

您可以在一个类别或在顶级目录编译所有的 port， 或者使用上述提到的 make fetch命令。 这样是非常危险的， 因为有一些port不能并存。 或者有另一种可能，
一些port会安装两个不同的文件， 但是却是相同的文件名。

在一些罕见的例子中， 用户可能需要在除了 MASTER_SITES 以外的一个站点(本地已经下载下来的文件)去获得一个文件包。 您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
```

在这个例子中，我们把 MASTER_SITES这个选项改为了 ftp.FreeBSD.org/pub/FreeBSD/ports/distfiles/。

一些 port 允许 (或甚至要求) 您指定编译选项来 启用/禁用 应用程序中非必需的功能，
一些安全选项， 以及其他可以订制的内容。 具有代表性的包括 www/mozilla、security/gpgme、 以及 mail/sylpheed-claws。 如果存在这样的选项，
通常会在编译时给出提示。

5.5.2.1. 改变默认的 Ports 目录

有时， 使用不同的工作临时目录和目标目录可能很有用 (甚至是必要的)。 可以用 WRKDIRPREFIX 和 PREFIX 这两个变量来改变默认的目录。 例如:

```
# make WRKDIRPREFIX=/usr/home/example/ports install
```

将在 /usr/home/example/ports 中编译 port 并把所有的文件安装到 /usr/local。

```
# make PREFIX=/usr/home/example/local install
```

将在 /usr/ports 编译它并安装到 /usr/home/example/local。

当然，

```
# make WRKDIRPREFIX=../ports PREFIX=../local install
```

将包含两种设置 (没有办法在这一页把它写完， 但您应该已经知道怎么回事了)。

另外， 这些变量也可以作为环境变量来设置。 请参考您的 shell

的联机手册上关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 (这是 X Window 系统的一部分) 不能正常地配合 PREFIX， 它们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。 让这些 port 尊重 PREFIX 是困难甚至是不可能的事情。

5.5.2.3. 重新配置 Ports

当你在编译某些 ports 的时候，可能会弹出一个基于 ncurses 的菜单来让你来选择一些编译选项。
通常用户都希望能够在一个 port 被编译安装了以后还能再次访问这份菜单以添加删除或修改这些选项。
实际上有很多方法来做这件事情。 一个方法进入那个 port 的目录后键入 make config，
之后便会再次显示出菜单和已选择的项目。 另一个方法是用 make showconfig，
这会给你显示出所有的配置选项。还有一个方法是执行 make rmconfig， 这将删除所有已选择的项目。
有关这些选项更详细的内容请参阅 ports(7)。

5.5.3. 卸载已经安装的 Ports

现在您已经了解了如何安装 ports， 并希望进一步了解如何卸载， 特别是在错误地安装了某个 port 之后。我们将卸载前面例子 (假如您没有注意的话， 是 lsof) 中安装的 port。 Ports 可以同 packages 以完全相同的方式 (在 Packages 一节 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

在您更新了 Ports 套件之后， 在升级 port 之前， 应查看 /usr/ports/UPDATING。
这个文件中介绍了在升级时用户应注意的问题， 以及一些可能需要进行的操作。 这可能包括更改文件格式、配置文件位置的变动， 以及与先前版本的兼容性等等。

如果 UPDATING 与本书中介绍的内容不同， 请以 UPDATING 为准。

5.5.4.2. 使用 Portupgrade 来更新 Ports

portupgrade 工具是设计来简化升级已安装的 port 的操作的。 它通过 ports-mgmt/portupgrade port 来提供。 您可以像其它 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， 并修正其所报告的不一致。 在每次升级之前，
有规律地执行它是个好主意。

运行 portupgrade -a 时， portupgrade 将开始并升级系统中所安装的所有过时的 ports。
如果您希望在每个升级操作时得到确认， 应指定 -i 参数。

```
# portupgrade -ai
```

如果您只希望升级某个特定的应用程序， 而非全部可用的 port， 应使用 portupgrade pkgname。 如果 portupgrade 应首先升级指定应用程序的话， 则应指定 -R 参数。

```
# portupgrade -R firefox
```

要使用预编译的 package 而不是 ports 来进行安装， 需要指定 -P。 如果指定了这个选项， portupgrade 会搜索 PKG_PATH 中指定的本地目录， 如果没有找到， 则从远程站点下载。 如果本地没有找到， 而且远程站点也没有成功地下载预编译包， 则 portupgrade 将使用 ports。 要禁止使用 port， 可以指定 -PP。

```
# portupgrade -PP gnome2
```

如果只想下载 distfiles (或者， 如果指定了 -P 的话， 是 packages) 而不想构建或安装任何东西， 可以使用 -F。 要了解更多细节， 请参考 portupgrade(1)。

5.5.4.3. 使用 Portmanager 来升级 Ports

Portmanager 是另一个用以简化已安装 port 升级操作的工具。 它可以通过 ports-mgmt/portmanager port 安装：

```
# cd /usr/ports/ports-mgmt/portmanager
# make install clean
```

可以通过这个简单的命令来升级所有已安装的 port：

```
# portmanager -u
```

如果希望 Portmanager 在进行每步操作之前都给出提示， 应使用 -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，它会在联编和安装您所选择的 port 之前升级所有依赖包。

```
# portmanager x11/gnome2
```

如果关于所选 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， 有问题的 port 也将被重新构建。

```
# portmanager graphics/gimp -f
```

要了解更多信息， 请参见 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/ 中的信息来检测出需要升级的 ports。你可以在 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)

你可以使用 -L 选项列出所有已安装的 ports 和查找存在更新的 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
```

可以使用这个简单的命令升级所有已安装的 ports：

```
# portmaster -a
```

Portmaster 默认在删除一个现有的 port 前会做一个备份包。如果新的版本能够被成功安装， Portmaster 将删除备份。 使用 -b 后 Portmaster 便不会自动删除备份。加上 -i 选项之后 Portmaster 将进入互动模式，在升级每个 port 以前提示你给予确认。

如果你在升级的过程中发现了错误，你可以使用 -f 选项升级/重新编译所有的 ports：

```
# portmaster -af
```

同样你也可以使用 Portmaster 往系统里安装新的 ports，升级所有的依赖关系之后并安装新的 port：

```
# portmaster shells/bash
```

更多的详细信息请参阅 portmaster(8)

5.5.5. Ports 和磁盘空间

使用 Ports 套件会最终用完磁盘空间。 在通过 ports 联编和安装软件之后，您应记得清理临时的 work 目录， 其方法是使用 make clean 命令。 您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， 您可能会在 distfiles 目录中积累下大量源代码文件。 您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， 也可以用下列命令删去目前安装的 port 没有使用的源码包文件：

```
# portsclean -DD
```

这个 portsclean 工具是 portupgrade 套件的一部分。

不要忘记删除那些已经安装， 但已不再使用的 ports。 用于自动完成这种工作的一个好工具是 ports-mgmt/pkg_cutleaves port。

5.6. 安装之后还要做点什么?

通常，您通过port安装完一个软件后,可以阅读它带的一些文档(如果它包含文档的话)，
或需要编辑它的配置文件，来确保这个软件的运行，
或在机器启动的时候启动(如果它是一个服务的话)，等等。

对于不同的软件有着不同的配置步骤。不管怎样，
如果您装好了一个软件，但是不知道下一步怎么办的时候， 这些小技巧可能可以帮助您:

- 使用 pkg_info(1) 命令，它能找到安装了哪些文件，以及装在哪里。 举个例子，如果您安装了 FooPackage version 1.0.0, 那么这个命令

  ```
  # pkg_info -L foopackage-1.0.0 | less
  ```

  将显示这个软件包安装的所有文件，您要特别注意在man/目录里面的文件， 它们可能是手册，etc/目录里面的配置文件，以及 doc/目录下面更多的文档。

  如果您不确定已经安装好的软件版本，您可以使用这样的命令

```
# pkg_info | grep -i foopackage
```

它将会找到所有已安装的软件包名字中包含foopackage 的软件包。 对于其他的查找，
您只需要在命令行中替换 foopackage。

- 一旦一些软件手册已被您确认安装，您可以使用 man(1) 查看它。
  同样的，如果有的话，您还可以完整的查看一遍配置文件的示例，以及任何额外的文档。
- 如果应用软件有网站, 您还可以从网站上找到文档，常见问题的解答，或其他更多。
  如果您不知道它们的网站地址，请使用下面的命令

```
# pkg_info foopackage-1.0.0
```

一个 WWW: 行, 如果它存在， 它将提供一个这个应用程序的网站URL.

- Ports 如果需要在服务器启动时运行(就像互联网服务器)， 它通常会把一个脚本的样例放入
  /usr/local/etc/rc.d 目录。为了保证正确性， 您可以查看这个脚本， 并编辑或更改这个脚本的名字。
  详情请看启动服务。

5.7. 如何处理坏掉的 Ports

如果您发现某个 port 无法正常工作， 有几件事值得尝试， 包括:

1. 在 问题报告数据库 中查找是否有尚未提交的修正。 如果有， 可以使用所提议的修正。
2. 要求 port 的监护人 (maintainer) 提供帮助。 输入 make maintainer 或阅读 Makefile
   查找监护人的电子邮件地址。 请记得把 port 的名字和版本写在邮件里 (Makefile 中的
   $FreeBSD:这一行) 并把错误输出的头几行发给 maintainer。

某些 ports 并非一个人维护， 而是写了一个 邮件列表。 许多， 但并非所有 port，
使用类似 freebsd-listname@FreeBSD.org 这样的地址。
请在提出问题时考虑这一点。

特别地， 由 ports@FreeBSD.org 监护的 port， 实际上并没有人维护。
订阅这个邮件列表的人们会感谢您提供的修正和支持。 我们一直都需要更多志愿者!

如果您没有得到回应， 则可以使用 send-pr(1) 来提交问题报告 (请参见 如何撰写 FreeBSD 问题报告)。

3. 修正它! Porter 手册 中提供了关于 "Ports" 基础设施的详细信息， 通过了解这些内容，
   您就能修正偶然坏掉的 port， 或甚至提交自己的 port 了!
4. 从较近的 FTP 站点下载一个编译好的安装包。 "中央的" package collection 在 ftp.FreeBSD.org 的
   packages 目录中， 但 在此之前 请 事先 检查一下是否存在较近的 镜像网站!
   通常情况下这些安装包都可以直接使用， 而且应该比自行编译快一些。 安装过程本身可以通过
   pkg_add(1) 来完成。

Chapter 6. X Window 系统

6.1. 概述

FreeBSD 使用 X11 来为用户提供功能强大的图形用户界面。 X11 是一种可以免费使用的 X 视窗系统，其实现包括 Xorg FreeBSD 中默认使用并受官方支持的 X11 实现即是 Xorg， 它是由 X.Org 基金会开发的 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 所支持的显示卡等硬件， 请访问 Xorg 网站。

在阅读完这一章后，您将会了解：

- X 视窗系统的不同组件，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。
- 如何为您的系统设置图形登录 (XDM)。

在阅读这一章之前，您应该：

- 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 可能会感觉很惊讶。

通常您并不需要深入了解各种 X 组件的作用以及它们之间的相互影响， 不过，了解一些关于它们的基础知识， 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 不是第一个为 UNIX® 而开发的视窗系统， 但它是最流行的。 X 的原始开发团队在开发 X 之前就已经在另外一个视窗系统上工作了。 那个系统的名字叫做 "W" (就是 "Window")。X 只是罗马字母中 W 后面 的一个。

X 可以被叫做 "X", "X Window 系统", "X11", 等等。把 X11 称做 "X Windows" 可能会冒犯某些人； 查看 X(7) 可以了解更多的信息。

6.2.2. X 客户机/服务器模型

X 一开始就是针对网络而设计的，所以 采用了 "client-server" 模型。在 X 模型中， "X server" 运行在有键盘，显示器，鼠标的计算机上。 服务器用来管理显示信息，处理来自键盘和鼠标的输入信息，并与其他输入输出设备交互 (比如作为输入设备的 "tablet"， 或者作为输出设备的投影仪)。 每一个 X 应用程序 (比如 XTerm, 或者 getenv(3)) 就是一个 "客户程序 (client)"。 客户程序给服务器发送信息，如 "请在这些坐标上画一个窗口"， 而服务器则返回处理信息， 如 "用户刚刚点击了 OK 按钮"。

如果您家或办公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， 如果您有很多运行 FreeBSD 的机器， 您可以在您的桌面计算机上运行 X server，而在比较高档的服务器上运行 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络来进行。

这可能会让一些人感到困惑， 因为 X 的术语和他们料想的有些不同。 他们以为 "X server" 是运行在功能强大的大型机上的，而 "X client" 是运行在他们桌面上的计算机上的。

记住，X server 是有键盘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种操作系统上， 它们甚至无需在同一种类型的计算机上运行。 在

Microsoft® Windows® 或 Apple 公司的 Mac OS® 上运行 X server 也是可以的，
在它们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 的设计哲学很像 UNIX® 的设计哲学， "tools, not policy"。这就意味着 X 不会试图去规定任务应该如何去完成，而是，只给用户提供一些工具，至于决定如何使用这些工具是用户自己的 事情。

这套哲学扩展了 X，它不会规定窗口在屏幕上应该是什么样子，要如何移动鼠标， 应该用什么键来切换窗体 (比如， Alt + Tab 按键，在 Microsoft® Windows® 环境中的作用), 每个窗口的工具条应该
看起来像什么，他们是否应该有关闭按钮等等。

实际上，X 行使了一种叫做 "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观感；其中一些还支持 "虚拟桌面"；有一些允许您可以定制一些键来管理您的桌面；一些有"开始" 按钮，或者其他类似的设计；一些是 "可定制主题的(themeable)"， 通过安装新的主题，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 分类目录里找到。

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， 而另外一些则可以使用 GUI 工具来完成大部分的配置任务， 举例而言， (Sawfish) 就使用 Lisp 语言书写配置文件。

焦点策略

窗口管理器的另一个特性是鼠标的 "focus policy"。
每个窗口系统都需要有一个选择窗口的方法来接受键盘的输入信息，以及当前
哪个窗口处于可用状态。

您通常比较熟悉的是一个叫做 "click-to-focus" 的焦点策略。 这是 Microsoft® Windows® 使用的典型焦点策略，也就是您在一个窗口上点击
一下鼠标，这个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。确切地说，窗口管理器控制着在什么时候哪个窗口拥有焦点。不同的窗口管理器支持不同的焦点方案。它们都支持点击即获得焦点，而且它们中的大多数都支持好几种方案。

最流行的焦点策略：

focus-follows-mouse
: 鼠标指示器下面的窗口就是获得焦点的窗口。
这个窗口不一定位于其他所有窗口之上。
通过将鼠标移到另一个窗口就可以改变焦点， 而不需要在它上面点击。

sloppy-focus
: 这种方式是对 focus-follows-mouse 策略的一个小小扩展。对于 focus-follows-mouse， 如果您把鼠标移到了根窗口（或桌面背景）上，
则所有的其它窗口都会失去焦点， 而相关的全部键盘输入也会丢失。 如果选择了 sloppy-focus， 则只有当指针进入新窗口时， 窗口焦点才会发生变化，
而当退出当前窗口时是不会变化的。

click-to-focus
: 当前窗口由鼠标点击来选择。窗口被"突出显示" ，
出现在所有其他窗口的前面。即使指针被移向了另一个窗口，所有的键盘输入仍会被这个窗口接收。

许多窗口管理器支持其他的策略，与这些相比又有些变化。您可以看具体
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 方法使得在每个应用程序屏幕上看到的窗口部件得到了 大大的扩展。

"Widget" 只是针对用户接口中所有列举项目的一个术语，它
可以用某种方法来点击或操作；如按钮，复选框，单选按钮，图标，列表框等等。 Microsoft® Windows®
把这些叫做"控件"。

Microsoft® Windows® 和苹果公司的 Mac OS® 都有一个严格的窗口部件策略。
应用程序开发者被建议确保他们的应用程序共享一个普通的所见即所得的用户界面。 对于
X，它并不要求一个特殊的图形风格或一套相结合的窗口部件集。

这样的结果是您不能期望 X 应用程序只拥有一个普通的所见即所得的界面。
有很多的流行的窗口部件集设置，包括来自于 MIT 的 Athena， Motif® (模仿 Microsoft® Windows®
的窗口风格， 所有部件都具有斜边和3种灰色度)， OpenLook， 等等。

如今， 绝大多数比较新的 X 应用程序采用一组新式的窗口设计， 这包括 KDE 所使用的 Qt， 以及 GNOME
所使用的 GTK+。 在这样一种窗口系统下，UNIX® 桌面的一些所见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 系统实现中的
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 编译和安装 Xorg：

```
# cd /usr/ports/x11/xorg
# make install clean
```

要完整地编译 Xorg 则需要至少 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。
如果从远程下载和安装， 在使用 pkg_add(1) 时请不要指定版本号。 pkg_add(1)
会自动地下载最新版本的安装包。

想要从 package 安装 Xorg， 简单地输入下面的命令：

```
# 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， 您也可以安装 x11/xorg-minimal。

这一章余下的部分将会讲解如何配置 X11, 以及如何设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， 您需要了解所安装的系统的下列信息：

- 显示器规格

- 显示卡的芯片类型
- 显示卡的显存容量

显示器的规格被 X11 用来决定显示的分辨率和刷新率。 这些规格通常可以从显示器所带的文档中，以及制造商的网站找到。 需要知道两个数字范围： 垂直刷新率和水平刷新率。

显示卡的芯片类型将决定 X11 使用什么模块来驱动图形硬件。 尽管系统能自动检测出绝大多数的硬件，但事先了解在自动检测出错的时候还是很有用处的。

显示卡的显存大小决定了系统支持的分辨率和颜色深度。 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，在提示符下键如下命令：

```
% startx
```

从 Xorg 7.4 开始， 可以使用 HAL 自动检测键盘和鼠标。Ports sysutils/hal 和 devel/dbus 将被作为 x11/xorg 所依赖的包安装进系统。 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， 需要运行这些服务 (手工启动或者重启机器)。

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。

诸如 GNOME， KDE 或 Xfce 之类的桌面环境，
大多都提供了一些允许用户非常易用的工具， 来设置像分辨率这样的显示参数。
所以如果你觉得默认的配置并不适合， 而且你打算安装一个这样的桌面环境，
那么就请继续完成桌面环境的安装， 并使用适合的显示设置工具。

配置 X11 需要一些步骤。 第一步是以超级用户的身份建立初始的配置文件：

```
# Xorg -configure
```

这会在 /root 中生成一个叫做 xorg.conf.new 的配置文件 (无论您使用 su(1) 或直接登录， 都会改变默认的 $HOME 目录变量)。 X11 程序将尝试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。

下一步是测试现存的配置文件， 以确认 Xorg 能够同系统上的图形设备正常工作。 对于 Xorg 7.3 或者之前的版本， 键入：

```
# Xorg -config xorg.conf.new
```

从 Xorg 7.4 和更高的版本开始， 这个测试将显示出一个黑色的屏幕，对于判断 X11 是否能正常工作会造成一些困扰。 可以通过 retro 选项使用旧的模式：

```
# Xorg -config xorg.conf.new -retro
```

如果看到黑灰的格子以及 X 型鼠标指针， 就表示配置成功了。 要退出测试， 需要同时按下 Ctrl + Alt + Fn

来切换到用于启动 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， 应使用 Ctrl + Alt + Backspace 组合键来强制退出 Xorg。 如果需要在 7.4 和之后的版本中启用这个组合键， 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
```

或者为 hald 创建一个叫作 x11-input.fdi 的键盘配置文件并保存至 /usr/local/etc/hal/fdi/policy 目录。 这个文件需包含以下这些：

```
<?xml version="1.0" encoding="utf-8"?>
<deviceinfo version="0.2">
 <device>
  <match key="info.capabilities" contains="input.keyboard">
   <merge key="input.x11_options.XkbOptions"
type="string">terminate:ctrl_alt_bksp</merge>
  </match>
 </device>
</deviceinfo>
```

你可能需要重启你的机器来使得 hald 重新读取这个文件。

此外， 还需要在 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， 在继续深入之前需要先配置它。 参阅 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， 这有助于自动检测硬件设备。 可以在这个文件中的 ServerLayout 或者 ServerFlags 加入以下选项使用旧的模式：

```
Option "AutoAddDevices" "false"
```

输入设备连同其他需要的选项 (比如， 键盘布局切换) 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 守护进程默认自动检测你的键盘。 可能检测出你的键盘布局或型号有差异， 在桌面环境中， 比如 GNOME， KDE 或者 Xfce 提供了工具来配置键盘。 另一方面， 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键盘的属性。

举例来说， 如果某人想要使用一个 PC 102 键法语布局的键盘， 我们就需要为 hald 创建一个配置文件， 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这些：

```
<?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>
```

如果这个文件已经存在， 只要把键盘配置相关的部分拷贝加入即可。

你需要重启你的机器使 hald 读入此文件。

也可以在 X 模拟终端或一个脚本中使用以下的命令达到相同的效果:

```
% setxkbmap -model pc102 -layout fr
```

/usr/local/shared/X11/xkb/rules/base.lst 列出了各种不同的键盘，
布局和可用的选项。

接下来是调整 xorg.conf.new 配置文件并作测试。 用文本编辑器如 emacs(1) 或 ee(1) 打开这个文件。 要做的第一件事是为当前系统的显示器设置刷新率。 这些值包括垂直和水平的同步频率。 把它们加到 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。 如果是这样的话， 就只能手动添加， 并在 HorizSync 和 VertRefresh 后面设置合适的数值了。 在上面的例子中， 给出了相应的显示器的参数。

X 能够使用显示器所支持的 DPMS (能源之星) 功能。 xset(1) 程序可以控制超时时间，
并强制待机、挂起或关机。 如果希望启用显示器的 DPMS 功能， 则需要把下面的设置添加到 monitor 节中：

```
    Option     "DPMS"
```

关闭 xorg.conf.new 之前还应该选择默认的分辨率和色深。 这是在 "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 关键字描述了要运行的默认色深。 这可以通过 Xorg(1) 的 -depth 命令行开关来替代配置文件中的设置。 Modes 关键字描述了给定颜色深度下屏幕的分辨率。 需要说明的是， 目标系统的图形硬件只支持由 VESA 定义的标准模式。 前面的例子中， 默认色深是使用 24位色。 在采用这个色深时， 允许的分辨率是 1024x768。

最后就是将配置文件存盘， 并使用前面介绍的测试模式测试一下。

在发现并解决问题的过程中， 包含了与 X11 服务器相关的各个设备的信息的 X11 日志文件会为您发现和排除问题有所帮助。 Xorg 日志的文件名是 /var/log/Xorg.0.log 这样的格式。 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。 您可以在 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
```

现在已经完成了 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 显示芯片组

配置Intel i810芯片组的显示卡需要有针对 X11 的能够用来驱动显示卡的 agpgart AGP程序接口。 请参见 agp(4) 驱动程序的联机手册了解更多细节。

这也适用于其他的图形卡硬件配置。 注意如果系统没有将 agp(4) 驱动程序编译进内核，尝试用 kldload(8) 加载模块是无效的。 这个驱动程序必须编译进内核或者使用 /boot/loader.conf 在启动时加载进入内核。

6.4.3.2. 添加宽屏平板显示器

这一节假定您了解一些关于高级配置的知识。 如果使用前面的标准配置工具不能产生可用的配置， 则在日志文件中提供的信息应该足以修正配置使其正确工作。 如果需要的话， 您应使用一个文本编辑器来完成这项工作。

目前的宽屏 (WSXGA、 WSXGA+、 WUXGA、 WXGA、 WXGA+， 等等) 支持 16:10 和 10:9 或一些支持不大好的显示比例。 常见的一些 16:10 比例的分辨率包括：

- 2560x1600
- 1920x1200
- 1680x1050
- 1440x900
- 1280x800

有时， 也可以简单地把这些分辨率作为 Section "Screen" 中的 Mode 来进行配置， 类似下面这样：

```
Section "Screen"
Identifier "Screen0"
Device    "Card0"
Monitor   "Monitor0"
DefaultDepth 24
SubSection "Display"
  Viewport  0 0
  Depth    24
  Modes    "1680x1050"
EndSubSection
EndSection
```

Xorg 能够自动地通过 I2C/DDC 信息来自动获取宽屏显示器的分辨率信息，
并处理显示器支持的频率和分辨率。

如果驱动程序没有对应的 ModeLines， 就需要给 Xorg 一些提示了。 使用 /var/log/Xorg.0.log
能够提取足够的信息， 就可以写一个可用的 ModeLine 了。 这类信息如下所示：

```
(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
```

这些信息称做 EDID 信息。 从中建立 ModeLine 只是把这些数据重新排列顺序而已：

```
ModeLine <name> <clock> <4 horiz. timings> <4 vert. timings>
```

如此， 本例中的 Section "Monitor" 中的 ModeLine 应类似下面的形式：

```
Section "Monitor"
Identifier    "Monitor1"
VendorName    "Bigname"
ModelName     "BestModel"
ModeLine      "1680x1050" 146.2 1680 1784 1960 2240 1050 1053 1059 1089
Option        "DPMS"
EndSection
```

经过简单的编辑步骤之后， X 就可以在您的宽屏显示器上启动了。

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，看起来很不专业， 并且， 在 getenv(3) 中，小字体简直无法看清。 有好几种免费、 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，但它们中的绝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：请参见后面的 TrueType® 字体一节。

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 或其他的字库和上面所说的大体类似。 为了让 X 服务器能够检测到这些字体， 需要在 X 服务器的配置文件 (/etc/X11/xorg.conf) 中增加下面的配置：

```
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
```

这样会起作用，但是当 X 会话结束后就会丢失， 除非它被添加到启动文件 (~/.xinitrc 中， 针对一个寻常的 startx 会话，或者当您通过一个类似 XDM 的图形登录管理器登录时添加到 ~/.xsession 中)。第三种方法是使用新的 /usr/local/etc/fonts/local.conf 文件： 查看 anti-aliasing 章节。

6.5.2. TrueType® 字体

Xorg 已经内建了对 TrueType® 字体的支持。有两个不同的模块能够启用这个功能。 在这个例子中使用 freetype 这个模块，因为它与其他的字体描绘后端 是兼容的。要启用 freetype 模块，只需要将下面这行添加到 /etc/X11/xorg.conf 文件的 "Module" 部分。

```
Load "freetype"
```

现在，为 TrueType® 字体创建一个目录 (比如， /usr/local/lib/X11/fonts/TrueType) 然后把所有的 TrueType® 字体复制到这个目录。记住您不能直接从 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。 一旦您已经将这些文件复制到了这个目录，就可以用 ttmkfdir 来创建 fonts.dir 文件， 以便让X字体引擎知道您已经安装了这些新文件。 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
```

现在把 TrueType® 字体目录添加到字体路径中。 这和上面 Type1 字体的步骤是一样的， 那就是，使用

```
% xset fp+ /usr/local/lib/X11/fonts/TrueType
```

```
% xset fp rehash
```

或者把 FontPath 这行加到 xorg.conf 文件中。

就是这样。现在 getenv(3), Gimp, StarOffice™ 和其他所有的 X 应用程序 应该可以认出安装的 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) 现在看起来已经很好了。

6.5.3. Anti-Aliased 字体

对于所有支持 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走样支持。 绝大多数较新的程序都提供了 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， 创建(或者编辑，如果文件已经存在的话)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这个文件来调节； 这一部分只描述几种最简单的情况。要了解更多的细节，请查看 fonts-conf(5).

这个文件一定是 XML 格式的。注意确保所有的标签都完全的关闭掉。 这个文件以一个很普通的 XML 头开始， 后跟一个 DOCTYPE 定义， 接下来是 <fontconfig> 标签：

```
<?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/ 目录下的所有字体已经可以被支持 Xft 的应用程序使用了。如果您想添加这两个目录以外的其他路径， 简单的添加下面这行到 /usr/local/etc/fonts/local.conf文件中：

```
<dir>/path/to/my/fonts</dir>
```

添加了新的字体，尤其是添加了新的字体目录后， 您应该运行下面的命令重建字体缓存：

```
# fc-cache -f
```

Anti-aliasing 会让字体边缘有些模糊，这样增加了非常小的文本的可读性， 并从大文本字体中删除 "锯齿"。 但如果使用普通的文本， 则可能引起眼疲劳。 要禁止 14磅 以下字体的反走样， 需要增加如下配置：

```
  <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 来间隔一些等宽字体也是不适当的。 这似乎是 KDE 的一个问题。
要修复这个问题需要确保每个字体之间的间距保持在100。 加入下面这些行：

```
  <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复的字体作为 "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 的时候可能存在问题。
通常的表现为字体本身似乎被垂直的切成两半。 糟糕的时候，还可能导致应用程序崩溃。
为了避免这样的现象，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>
```

一旦您完成对 local.conf 文件的编辑，确保您使用了 </fontconfig> 标签来结束文件。
不这样做将会导致您的更改被忽略。

最后，用户可以通过他们个人的 .fonts.conf 文件来添加自己的设定。 要完成此项工作，
用户只需简单地创建 ~/.fonts.conf 并添加相关配置。 此文件也必须是 XML 格式的。

最后：对于LCD屏幕， 可能希望使用子像素的取样。 简单而言， 这是通过分别控制 (水平方向分开的)
红、绿、蓝 像素， 来改善水平分辨率； 这样做的效果一般会非常明显。 要启用它， 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>
```

随您显示器的种类不同， 可能需要把 rgb 改为 bgr、 vrgb 或 vbgr：
试验一下看看那个更好。

6.6. X 显示管理器

6.6.1. 概要

X 显示管理器(XDM) 是一个X视窗系统用于进行登录会话管理的可选项。
这个可以应用于多种情况下，包括小 "X Terminals"， 桌面，大网络显示服务器。既然 X
视窗系统不受网络和协议的限制， 那对于通过网络连接起来的运行 X 客户端和服务器端的不同机器，
就会有很多的可配置项。 XDM 提供了一个选择要连接到哪个显示服务器的图形接口，
只要键入如登录用户名和密码这样的验证信息。

您也可以把 XDM 想象成与 getty(8) 工具一样(see 配置 for
details)。为用户提供了同样功能。它可以完成系统的登录任务， 然后为用户运行一个会话管理器
(通常是一个 X 视窗管理器)。接下来 XDM
就等待这个程序退出，发出信号用户已经登录完成，应当退出屏幕。 这时， XDM
就可以为下一个登录用户显示登录和可选择屏幕。

6.6.2. 使用 XDM

如果希望使用 XDM 来启动， 首先需要安装 x11/xdm port (在较新版本的 Xorg 中它并不是默认安装的)。

XDM 服务程序位于 /usr/local/bin/xdm。 任何时候都可以 root 用户的身份来运行它，
以令其管理本地系统的 X 显示。 如果希望让 XDM 在系统每次启动过程中自动运行，
比较方便的做法是把它写到 /etc/ttys 的配置中。 有关这个文件的具体格式和使用方法请参阅
添加一个记录到/etc/ttys。 在默认的 /etc/ttys 文件中已经包含了在虚拟终端上运行 XDM 服务的示范配置：

```
ttyv8   "/usr/local/bin/xdm -nodaemon"  xterm   off secure
```

默认情况下，这个记录是关闭的，要启用它， 您需要把第5部分的 off 改为 on 然后按照
重新读取/etc/ttys来强制init 的指导 重新启动 init(8)。第一部分，这个程序将管理的终端名称是
ttyv8。这意味着 XDM 将运行在第9个虚拟终端上。

6.6.3. 配置 XDM

XDM 的配置目录是在 /usr/local/lib/X11/xdm中。在这个目录中， 您会看到几个用来改变 XDM
行为和外观的文件。您会找到这些文件：

文件	描述
Xaccess	客户端授权规则。
Xresources	默认的X资源值。
Xservers	远程和本地显示管理列表。
Xsession	用于登录的默认的会话脚本。
Xsetup_*	登录之前用于加载应用程序的脚本。
xdm-config	运行在这台机器上的所有显示的全局配置。
xdm-errors	服务器程序产生的错误。
xdm-pid	当前运行的 XDM 的进程 ID。

当 XDM 运行时， 在这个目录中有几个脚本和程序可以用来设置桌面。
这些文件中的每一个的用法都将被简要地描述。 这些文件的更详细的语法和用法在 xdm(1)
中将有详细描述。

默认的配置是一个矩形的登录窗口，上面有机器的名称， "Login:" 和
"Password:"。如果您想设计您自己个性化的 XDM 屏幕，这是一个很好的起点。

6.6.3.1. Xaccess

用以连接由 XDM 所控制的显示设备的协议， 叫做 X 显示管理器连接协议 (XDMCP)。
这个文件是一组用以控制来自远程计算机的 XDMCP 连接的规则。 除非您修改 xdm-config
使其接受远程连接， 否则其内容将被忽略。 默认情况下， 它不允许来自任何客户端的连接。

6.6.3.2. Xresources

这是一个默认的用来显示选项和登录屏幕的应用程序文件。
您可以在这个文件中对登录程序的外观进行定制。 其格式与 X11 文档中描述的默认应用程序文件是一样的。

6.6.3.3. Xservers

这是一个选择者应当提供的作为可选的远程显示列表。

6.6.3.4. Xsession

这是一个用户登录后针对 XDM 的默认会话脚本。通常，在 ~/.xsession
中每个用户将有一个可定制的会话脚本。

6.6.3.5. Xsetup_*

在显示选择者或登录接口之前，这些将被自动运行。 这是一个每个显示都要用到的脚本，叫做 Xsetup_，后面会跟一个本地显示的数字(比如 Xsetup_0)。典型的，这些脚本将在后台 (如 xconsole)运行一个或两个程序。

6.6.3.6. xdm-config

此文件以应用程序默认值的形式， 提供了在安装时所使用的普适的显示设置。

6.6.3.7. xdm-errors

这个文件包含了 XDM 正设法运行的的 X 服务器 的输出。 如果 XDM 正设法运行的显示由于某种原因被挂起， 那这是一个寻找错误信息的好地方。 这些信息会在每一个会话的基础上被写到用户的 ~/.xsession-errors 文件中。

6.6.4. 运行一个网络显示服务器

对于其他客户端来说， 如果希望它们能连接到显示服务器，您就必须编辑访问控制规则， 并启用连接侦听。 默认情况下， 这些都预设为比较保守的值。 要让 XDM 能侦听连接， 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。 记住默认应用程序文件的注释以"!" 字母开始，不是"\#"。 您需要设置严格的访问控制 - 看看在 Xaccess 文件中的实例， 并参考 xdm(1) 的联机手册， 以了解进一步的细节。

6.6.5. 替换 XDM

有几个替换默认 XDM程序的方案。 其中之一是 上一节已经描述过的kdm (与 KDE捆绑在一起)。 kdm 提供了许多视觉上的改进和局部的修饰， 同样能让用户在启动时能选择他们喜欢的窗口管理器。

6.7. 桌面环境

这节描述了 FreeBSD 上用于 X 的不同桌面环境。 "桌面环境" 可能仅仅是一个简单的窗口管理器， 也可能是一个像 KDE 或者 GNOME这样的完整桌面应用程序套件。

6.7.1. GNOME

6.7.1.1. 有关 GNOME

GNOME 是一个用户界面友好的桌面环境， 能够使用户很容易地使用和配置他们的计算机。 GNOME 包括一个面板(用来启动应用程序和显示状态)， 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协议集。 其他操作系统的用户在使用 GNOME提供的强大的图形驱动环境时会觉得很好。 更多的关于 FreeBSD 上 GNOME 的信息 可以在 FreeBSD GNOME Project 的网站上找到。 此外， 这个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

这个软件可以很容易地通过预编译包或 Ports 套件来安装：

要从网络安装GNOME， 只要键入：

```
# pkg_add -r gnome2
```

从源代码编译GNOME，可以使用 ports树：

```
# cd /usr/ports/x11/gnome2
# make install clean
```

GNOME 需要挂载 /proc 文件系统才能正常运作。添加如下

```
proc        /proc     procfs  rw  0   0
```

到 /etc/fstab 以便在系统启动时自动挂载 procfs(5)。

一旦装好了 GNOME， 就必须告诉 X server 启动 GNOME 而不是默认的窗口管理器。

最简单的启动 GNOME 的方法是使用 GDM， GNOME 显示管理器。 随 GNOME 桌面一同安装的 GDM 尽管默认是禁用的。 可以在 /etc/rc.conf 中加入以下这行启用：

```
gdm_enable="YES"
```

这样在你重启机器的时候， GDM 将自动运行。

通常我们希望在 GDM 启动时， 同时启用所有的 GNOME 服务， 可以将如下这行加入 /etc/rc.conf：

```
gnome_enable="YES"
```

GNOME 也可以通过适当地配置名为 .xinitrc 的文件来启动。 如果已经有了自定义的 .xinitrc， 将启动当前窗口管理器的那一行改为启动 /usr/local/bin/gnome-session 就可以了。 如果还没有， 那么只需简单地：

```
% echo "/usr/local/bin/gnome-session" > ~/.xinitrc
```

接下来输入 startx， GNOME 桌面环境就启动了。

如果之前使用了一些旧式的显示管理器， 例如 XDM， 则这样做是没用的。 此时应建立一个可执行的 .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 取代现有的窗口管理器：

```
% echo "#!/bin/sh" > ~/.xsession
% echo "/usr/local/bin/gnome-session" >> ~/.xsession
% chmod +x ~/.xsession
```

还有一种做法， 是配置显示管理器， 以便在登录时提示您选择窗口管理器； 在 KDE 细节 环节中介绍了关于如何为 kdm （KDE 的显示管理器）进行这样的配置。

6.7.2. KDE

6.7.2.1. 有关 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：

- 一个美丽的现代的桌面。
- 一个集合了完美网络环境的桌面。
- 一个集成的帮助系统，能够方便、高效地帮助您使用 KDE 桌面和它的应用程序。
- 所有的KDE应用程序具有一致的所见即所得界面。
- 标准的菜单和工具栏，键盘布局，颜色配置等。
- 国际化：KDE 可以使用超过40种语言。
- 集中化、 统一的对话框驱动的桌面配置
- 许多有用的 KDE应用程序。

KDE 附带了一个名为 Konqueror 的 web 浏览器， 它是其他运行于 UNIX® 系统上的 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， 可以访问 KDE 网站。 与 FreeBSD 相关的 KDE 信息和资源， 可以在 FreeBSD 上的 KDE 团队 的网站找到。

FreeBSD 上提供了两种版本的 KDE。 版本 3 已经推出了很长时间， 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。 这两种版本甚至能够并存。

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预编译包或 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 则需要：

```
# pkg_add -r kde4
```

pkg_add(1) 就会自动的下载最新版本的应用程序。

要从源代码编译 KDE3， 可以使用 ports 树：

```
# cd /usr/ports/x11/kde3
# make install clean
```

而从 ports 提供的源代码编译 KDE4， 对应的操作则是：

```
# cd /usr/ports/x11/kde4
# make install clean
```

安装好 KDE 之后， 还需要告诉X server 启动这个应用程序来代替默认的窗口管理器。 这可以通过编辑 .xinitrc 文件来完成：

对于 KDE3：

```
% echo "exec startkde" > ~/.xinitrc
```

对于 KDE4：

```
% echo "exec /usr/local/kde4/bin/startkde" > ~/.xinitrc
```

现在，无论您什么时候用 startx进入 X 视窗系统， KDE 就将成为您的桌面环境。

如果使用一个像 XDM这样的显示管理器， 那配置文件可能有点不同。需要编辑一个 .xsession 文件，有关 kdm 的用法会在这章的后面介绍。

6.7.3. 有关 KDE 的更多细节

现在 KDE 已经被安装在系统中了。 通过帮助页面或点击多个菜单可以发现很多东西。 Windows® 或 Mac® 用户会有回到家的感觉。

有关 KDE 的最好的参考资料是 它的在线文档。KDE 拥有它自己的 web 浏览器 Konqueror，
还有很多其他的应用程序和丰富文档。
这节的余下部分将讨论一些很难用走马观花的方法来学习的技术项目。

6.7.3.1. KDE 显示管理器

如果在同一系统上有多个用户， 则管理员通常会希望使用图形化的登录界面。 前面已经提到， 使用 XDM 可以完成这项工作。 不过， KDE 本身也提供了另一个选择， 即 kdm， 它的外观更富吸引力，
而且提供了更多的登录选项。 值得一提的是， 用户还能通过菜单很容易地选择希望使用的桌面环境 (KDE、GNOME 或其它)。

要启用 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 ttyv8 项需被改写成如下的形式：

```
ttyv8 "/usr/local/bin/kdm -nodaemon" xterm on secure
```

对于 KDE4， 你需要将如下这行加入 /etc/rc.conf：

```
local_startup="${local_startup} /usr/local/kde4/etc/rc.d"
kdm4_enable="YES"
```

6.7.4. Xfce

6.7.4.1. 有关Xfce

Xfce是以被GNOME 使用的 GTK+ 工具包为基础的桌面环境，
但是更加轻巧，适合于那些需要一个易于使用和配置并且简单而高效的桌面的人。
看起来，它非常像使用在商业UNIX®系统上的 CDE环境。 Xfce的主要特性有下面这些：

- 一个简单，易于使用的桌面。
- 完全通过鼠标的拖动和按键来控制等。
- 与CDE 相似的主面板，菜单，applets和应用launchers。
- 集成的窗口管理器，文件管理器，声音管理器， GNOME 应用模块等等。
- 可配置界面的主题。(因为它使用GTK+)

- 快速，轻便，高效：对于比较老的/旧的机器或带有很少内存的机器仍然很理想。

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(在写作的时候)。要安装的话，执行下面的命令：

```
# pkg_add -r xfce4
```

另外， 也可以使用 Ports Collection 从源代码联编：

```
# cd /usr/ports/x11-wm/xfce4
# make install clean
```

现在，要告诉X服务器在下次X启动时执行 Xfce。 只要执行下面的命令：

```
% echo "/usr/local/bin/startxfce4" > ~/.xinitrc
```

接下来就是启动 X， Xfce将成为您的桌面。 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，就像有关 GNOME 的那节描述的，
使用/usr/local/bin/startxfce4 命令，或者，配置显示管理器允许在启动时选择一个桌面， 就像有关
kdm的那节描述的。

Part II: 常见的任务

前面已经介绍了必要的基础知识， 手册的这一部分将讨论 FreeBSD 的一些最常用的功能。 这些章节包括：

- 向您介绍流行和实用的桌面应用程序： 浏览器、产品工具、文档察看程序，等等。
- 向您介绍一系列可以在 FreeBSD 上使用的多媒体工具。
- 介绍联编定制的 FreeBSD 内核以启用附加功能的方法。
- 详细介绍包括桌面和网络打印机在内的打印系统设置。
- 向您展示如何在 FreeBSD 上运行 Linux 应用程序。

某些章节希望您首先阅读过其他部分， 在这些章的开头部分也会给出类似的提示。

Chapter 7. 桌面应用

7.1. 概述

FreeBSD 可以运行种类繁多的桌面应用程序， 这包括像浏览器和字处理这样的软件。
绝大多数这样的程序都可以通过 package 来安装， 或者从 Ports Collection 自动地构建。
许多新用户希望能够在它们的系统中找到这样的应用程序。 这一章将向您展示如何轻松地使用 package 或者 Ports Collection 中安装这样的软件。

需要注意的是从 ports 安装意味着要编译源码。 根据编译的 ports 和电脑速度的不同，
这可能需要花费相当长的时间。 若是您觉得编译源码太过耗时的话， 绝大多数 ports
也有预编译的版本可供安装。

因为 FreeBSD 提供的二进制兼容 Linux 的特性， 许多原本为 Linux 开发的程序都可以直接用在您的桌面。
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。 当您在寻找特定的 ports 时， 可以使用 whereis(1)。 一般来说， 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 应用程序前已经开启了 Linux 二进制兼容功能。

本章涵盖以下种类应用程序：

- 浏览器 (例如 Firefox、 Opera、 Konqueror)
- 办公、图象处理 (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)

阅读这章之前，您应该：

- 知道如何安装额外的第三方软件(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)。

想要获得更多的有关多媒体环境的信息，请阅读 多媒体。如果您想要建立和使用电子邮件， 请参考 电子邮件。

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。如果您没有时间一一编译它们 (有些时候这可能需要花费相当长的时间) 大部分都有 package 可用。

KDE 和 GNOME 已经提供 HTML 浏览器。 请参考桌面环境得到更多完整的有关设定这些桌面环境的信息。

如果您要找小型的浏览器， 可以试试看 www/dillo2、 www/links 或 www/w3m。

这一节涉及如下程序：

程序名称	资源需求	安装时间	主要依赖
Firefox	中等	长	Gtk+
Opera	少	轻松	同时有可用的 FreeBSD 和 Linux 版本。 Linux 版本需要使用 Linux 二进制兼容模块和 linux-openmotif。
Firefox	中等	长	Gtk+
Konqueror	中等	长	需要 KDE 库

7.2.1. Firefox

Firefox 是一个现代， 自由， 开放源代码稳定的浏览器， 并完全移植到了 FreeBSD 上：
它的特性包括有一个非常标准的 HTML 显示引擎， 标签式浏览， 弹出窗口阻止， 扩展插件，
改进的安全性， 等等。 Firefox 是基于 Mozilla 的代码。

您可以通过输入下面的命令来安装预编译的包：

```
# pkg_add -r firefox
```

这将会安装 Firefox 7.0， 如果希望运行 Firefox 3.6， 则应使用下面的命令：

```
# pkg_add -r firefox36
```

如果你希望从源代码编译的话， 可以通过 Ports Collection 安装：

```
# cd /usr/ports/www/firefox
# make install clean
```

对于 Firefox 3.6， 对应的命令中的 firefox 应改为 firefox36。

7.2.2. Firefox 与 Java™ 插件

在这一节和接下来的两节中， 我们均假定您已经安装了 Firefox。

通过 Ports 套件来安装 OpenJDK 6， 输入下面的命令：

```
# 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
```

请确认在编译上述 port 时使用的是系统预设的配置。

启动浏览器并在地址栏中输入 about:plugins 然后按 Enter。
浏览器将会呈现一个列出所有已安装插件的页面； Java™ 插件应在其中出现。

如果浏览器找不到插件， 则用户可能必须运行下面的命令， 并重启浏览器：

```
% ln -s /usr/local/lib/IcedTeaPlugin.so \
  $HOME/.mozilla/plugins/
```

7.2.3. Firefox 与 Adobe® Flash™ 插件

Adobe® Flash™ 插件并没有直接提供其 FreeBSD 版本。 不过， 我们有一个软件层 (wrapper) 可以用来运行

Linux 版本的插件。 这个 wrapper 也支持 Adobe® Acrobat®、 RealPlayer 和很多其他插件。

根据你 FreeBSD 版本的不同选择相应的安装步骤：

1. FreeBSD 7.X

   安装 www/nspluginwrapper port。 这个 port 需要安装一个较大的emulators/linux_base-fc4 port。

   下一步是安装 www/linux-flashplugin9 port。 这将会安装 Flash™ 9.X， 此版本目前能在 FreeBSD 7.X 上正常运行。

   在比 FreeBSD 7.1-RELEASE 更旧版本的系统上， 你必须安装 www/linux-flashplugin7 并跳过以下 linprocfs(5) 的部份。

2. FreeBSD 8.X

   安装 www/nspluginwrapper port。 这个 port 需要安装一个较大的emulators/linux_base-f10 port。

   下一步是安装 www/linux-f10-flashplugin10 port。 这将会安装 Flash™ 10.X， 此版本目前能在 FreeBSD 8.X 上正常运行。

   这个版本需要创建一个符号链接：

   ```
   # ln -s /usr/local/lib/npapi/linux-f10-flashplugin/libflashplayer.so \
    /usr/local/lib/browser_plugins/
   ```

   如果系统中没有 /usr/local/lib/browser_plugins 目录， 则应手工创建它。

按照 FreeBSD 版本， 在安装了正确的 Flash™ port 之后， 插件必须由每个用户运行 nspluginwrapper 安装：

```
% nspluginwrapper -v -a -i
```

如果希望播放 Flash™ 动画的话，Linux® 的进程文件系统， linprocfs(5) 必须挂载于 /usr/compat/linux/proc。 可以通过以下的命令实现：

```
# mount -t linprocfs linproc /usr/compat/linux/proc
```

这也可以在机器启动时自动挂载， 把以下这行加入 /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。
这将显示目前可用的插件列表。

7.2.4. Firefox and Swfdec Flash™ Plugin

Swfdec 是一个用以解码和渲染 Flash™ 动画的库。 Swfdec-Mozilla 是一个使用了 Swfdec 库让 Firefox 能播放 SWF 文件的插件。它目前仍处于开发状态。

如果你不能或者不想编译安装，可以通过网络安装二进制包：

```
# pkg_add -r swfdec-plugin
```

如果二进制包还不可用，你可以通过 Ports Collection 编译安装：

```
# cd /usr/ports/www/swfdec-plugin
# make install clean
```

然后重启你的浏览器使得这个插件生效。

7.2.5. Opera

Opera 是一个功能齐全， 并符合标准的浏览器。 它还提供了内建的邮件和新闻阅读器、 IRC 客户端，RSS/Atom feed 阅读器以及更多功能。 除此之外， Opera 是一个比较轻量的浏览器， 其速度很快。它提供了两种不同的版本： "native" FreeBSD 版本， 以及通过 Linux 模拟运行的版本。

要使用 Opera 的 FreeBSD 版本来浏览网页，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 专用的版本。 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
```

然后可以检查插件是否可用了： 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 为 Firefox 添加插件 的方法相同。

7.2.6. Konqueror

Konqueror 是 KDE 的一部分，不过也可以通过安装 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。

也有种类丰富的插件能够配合 Konqueror 一起使用， 您可以通过 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、图象处理

当需要进行办公或者进行图象处理时， 新用户通常都会找一些好用的办公套件或者字处理软件。
尽管目前有一些 桌面环境， 如 KDE 已经提供了办公套件， 但目前这还没有一定之规。
无论您使用那种桌面环境， FreeBSD 都能提供您需要的软件。

这节涉及如下程序：

软件名称	资源需求	安装时间	主要依赖
KOffice	少	多	KDE
AbiWord	少	少	Gtk+ 或 GNOME
The Gimp	少	长	Gtk+
OpenOffice.org	多	长	JDK™、 Mozilla
LibreOffice	较重	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，这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 是演示文档制作管理程序、 Kontour是矢量绘图软件。

安装最新的 KOffice 之前，先确定您是否安装了最新版的 KDE。

使用 package 来安装 KOffice，安装细节如下：

```
# pkg_add -r koffice
```

如果没有可用的 package，您可以使用 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。
它适合用来打印文件、信函、报告、备忘录等等， 它非常快且包含许多特性，并且非常容易使用。

AbiWord 可以导入或输出很多文件格式， 包括一些象 Microsoft® .doc 这类专有格式的文件。

AbiWord 也有 package 的安装方式。您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 编译。ports collection 应该是最新的。它的安装方式如下：

```
# cd /usr/ports/editors/abiword
# make install clean
```

7.3.3. GIMP

对图象的编辑或者加工， GIMP 是一个非常精通图象处理的软件。
它可以被用来当作简单的绘图程序或者一个专业的照片处理套件。
它支持大量的插件和具有脚本界面的特性。 GIMP 可以读写众多的文件格式， 支持扫描仪和手写板。

您可以用下列命令安装：

```
# pkg_add -r gimp
```

如果您在 FTP 站点没有找到这个 package，您也可以使用 Ports Collection 的方法安装。ports 的 graphics 目录也包含有 Gimp 手册。 以下是安装它们的方法：

```
# cd /usr/ports/graphics/gimp
# make install clean
# cd /usr/ports/graphics/gimp-manual-pdf
# make install clean
```

Ports 中的 graphics 目录也有开发中的 GIMP 版本 graphics/gimp-devel。 HTML 版本的 Gimp 手册 可以在 graphics/gimp-manual-html 找到。

7.3.4. OpenOffice.org

OpenOffice.org 包括一套完整的办公套件： 字处理程序、 电子表格程序、 演示文档管理程序和绘图程序。
它和其它的办公套件的特征非常相似，它可以导入输出不同的流行的文件格式。 它支持许多种语言 - 国际化已经渗透到了其界面、 拼写检查和字典等各个层面。

OpenOffice.org 的字处理程序使用 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库界面。 OpenOffice.org 已经可以平稳的运行在 Windows®、Solaris™、Linux、FreeBSD 和 Mac OS® X 等各种操作系统下。 更多的有关 OpenOffice.org 的信息可以在 OpenOffice.org 网页 找到。 对于特定的 FreeBSD 版本的信息，您可以在直接在 FreeBSD OpenOffice 移植团队的页面下载。

安装 OpenOffice.org 方法如下：

```
# pkg_add -r openoffice.org
```

如果您正在使用 FreeBSD 的 -RELEASE 版本， 一般来说这样做是没问题的。
如果不是这样， 您就可能需要看一看 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。
最新的发布版本和开发版本都可以在那里找到。

装好 package 之后， 您只需输入下面的命令就能运行 OpenOffice.org 了：

```
% openoffice.org
```

在第一次运行时， 将询问您一些问题， 并在您的主目录中建立一个 .openoffice.org 目录。

如果没有可用的 OpenOffice.org package，您仍旧可以选择编译 port。然而，
您必须记住它的要求以及大量的磁盘空间和相当长的时间编译。

```
# cd /usr/ports/editors/openoffice.org-3
# make install clean
```

如果希望联编一套进行过本地化的版本， 将前述命令行改为：

```
# make LOCALIZED_LANG=your_language install clean
```

您需要将 your_language 改为正确的 ISO-代码。 所支持的语言代码可以在 files/Makefile.localized 文件中找到， 这个文件位于 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。 这是 OpenOffice.org 的一个贴牌的分支版本，
包含了完整办公效率套件中必备的应用： 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。 它提供了许多不同语言的支持 - 国际化支持除了界面之外，
还包括了拼写检查器和字典。

LibreOffice 的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灵活性。
电子表格程序提供了一种可以与外部数据库交互的宏语言支持。 LibreOffice 目前已经可以稳定运行于 Windows®、 Linux、 FreeBSD 和 Mac OS® X。 关于 LibreOffice 的更多信息可以在 LibreOffice 网站 找到。

如果希望通过预编译的二进制包安装 LibreOffice， 执行：

```
# pkg_add -r libreoffice
```

如果运行的是 FreeBSD 的 -RELEASE 版本， 这个命令应该不会遇到任何问题。

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， 系统会询问一系列问题， 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， 您还是可以通过 port 安装。 不过，
请注意编译它需要相当多的磁盘空间和时间。

```
# cd /usr/ports/editors/libreoffice
# make install clean
```

如果希望编译本地化的版本， 把前面的命令换成：

```
# make LOCALIZED_LANG=your_language install clean
```

您需要把 your_language 换成正确的语言 ISO 代码。 可用的代码可以在 port 的 Makefile 中的 pre-fetch target 中找到。

完成联编和安装之后， 就可以用下面的命令运行 LibreOffice 了：

```
% libreoffice
```

7.4. 文档查看器

UNIX® 系统出现以来， 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统内。 本节中， 我们将了解如何安装它们。

这节涵盖如下应用程序:

软件名称	资源需求	安装时间	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®

现在许多文档都用 PDF 格式， 根据"轻便小巧文档格式"的定义。一个被建议使用的查看器是 Acrobat Reader®，由 Adobe 所发行的 Linux 版本。因为 FreeBSD 能够运行 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， 我们不提供预编译的版本。

7.4.2. gv

gv 是 PostScript® 和 PDF 文件格式查看器。它源自 ghostview 因为使用 Xaw3d 函数库让它看起来更美观。 它很快而且界面很干净。gv 有很多特性比如象纸张大小、刻度或者抗锯齿。 大部分操作都可以只用键盘或鼠标完成。

安装 gv package，如下：

```
# pkg_add -r gv
```

如果您无法获取预编译的包， 则可以使用 Ports Collection：

```
# cd /usr/ports/print/gv
```

```
# make install clean
```

7.4.3. Xpdf

如果您想要一个小型的 FreeBSD PDF 查看器， Xpdf 是一个小巧并且高效的查看器。
它只需要很少的资源而且非常稳定。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 不可用或者您宁愿使用 Ports Collection，如下：

```
# cd /usr/ports/graphics/xpdf
# make install clean
```

一旦安装完成，您就可以启动 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。 您可以单击鼠标来观看一个文件、开启一个外部编辑器、
使用预览和更多的功能。它也有幻灯片播放模式和一些基本的文件操作。
您可以管理采集的图片并且很容易找到重复的。 GQview 可以全屏幕观看并且支持国际化。

如果您想要安装 GQview package，如下：

```
# pkg_add -r gqview
```

如果您没有可用的 package 或者您宁愿使用 Ports Collection，如下：

```
# cd /usr/ports/graphics/gqview
# make install clean
```

7.5. 财务

假如，基于任何的理由，您想要在 FreeBSD Desktop
管理您个人的财政，有一些强大并且易于使用的软件可以被您选择安装。
它们中的一些与流行的文件格式兼容象 Quicken 和 Excel 文件。

本节涵盖如下程序：

软件名称	资源需求	安装时间	主要依赖
GnuCash	少	长	GNOME
Gnumeric	少	长	GNOME
Abacus	少	少	Tcl/Tk
KMyMoney	少	长	KDE

7.5.1. GnuCash

GnuCash 是 GNOME 的一部分，GNOME 致力于为最终用户提供用户友好且功能强大的软件。使用 GnuCash，您可以关注您的收入和开支、您的银行帐户， 或者您的股票。它的界面特性看起来非常的专业。

GnuCash 提供一个智能化的注册、帐户分级系统、
很多键盘快捷方式和自动完成方式。它能分开一个单个的处理到几个详细的部分。 GnuCash 能导入和合并 Quicken QIF 文件格式。 它也支持大部分的国际日期和流行的格式。

在您的系统中安装 GnuCash 所需的命令如下：

```
# pkg_add -r gnucash
```

如果 package 不可用，您可以使用 Ports Collection 安装：

```
# cd /usr/ports/finance/gnucash
# make install clean
```

7.5.2. Gnumeric

Gnumeric 是一个电子表格程序， GNOME 桌面环境的一部分。
它以通过元素格式和许多片断的自动填充系统来方便的自动"猜测"用户输入而著称。
它能导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 支持图表。 它有大量的嵌入函数和允许所有通常比如象、数字、货币、日期、 时间等等的一些单元格式。

以 package 方式安装 Gnumeric 的方法如下：

```
# pkg_add -r gnumeric
```

如果 package 不可用，您可以使用 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统计学、财务和数学方面很有帮助。 它能导入和输出 Excel 文件格式。
Abacus 可以产生 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，您可以使用 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
旨在提供并融合各种商业财务管理软件所有的重要特性。 它也同样注重易用性和特有的复式记帐功能。
KMyMoney 能从标准的 Quicken Interchange Format (QIF) 文件导入数据，
追踪投资，处理多种货币并能提供一个财务报告。 另有可用的插件支持导入 OFX 格式的数据。

以 package 的方式安装 KMyMoney 的方法如下：

```
# pkg_add -r kmymoney2
```

如果 package 不可用，您可以使用 Ports Collection 安装：

```
# cd /usr/ports/finance/kmymoney2
# make install clean
```

7.6. 总结

尽管 FreeBSD 由于其高性能和可靠性而获得了许多 ISP 的信赖， 但它也完全可以用于桌面环境。
拥有数以千计的 packages 和 ports 能够帮您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简要回顾：

软件名称	Package 名称	Ports 名称
Opera	opera	www/opera
Firefox	firefox	www/firefox
KOffice	koffice	editors/koffice-kde3
AbiWord	abiword	editors/abiword
The GIMP	gimp	graphics/gimp
OpenOffice.org	openoffice	editors/openoffice.org-3
LibreOffice	libreoffice	editors/libreoffice
Acrobat Reader®	acroread	print/acroread8
gv	gv	print/gv
Xpdf	xpdf	graphics/xpdf
GQview	gqview	graphics/gqview
GnuCash	gnucash	finance/gnucash
Gnumeric	gnumeric	math/gnumeric
Abacus	abacus	deskutils/abacus
KMyMoney	kmymoney2	finance/kmymoney2

Chapter 8. 多媒体

8.1. 概述

FreeBSD 广泛地支持各种声卡， 让您可以从容地享受来自您的计算机的高保真输出。 这包括了录制和播放 MPEG Audio Layer 3 (MP3)、 WAV、 以及 Ogg Vorbis 等许多种格式声音的能力。 FreeBSD 同时也包括了许多的应用程序，让您可以录音、 增加声音效果以及控制附加的MIDI设备。

要是乐于动手， FreeBSD 也能支持播放一般的视频文件和 DVD。 对各种视频媒体进行编码、转换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， 在撰写这章时， FreeBSD Ports Collection 中还没有类似 audio/sox 那样好的重编码工具能够用来在不同的格式之间转换。 不过，这个领域的软件研发进展是很快的。

本章将介绍配置声卡的必要步骤。 X11 的安装和配置 (X Window 系统) 里已经考虑到了您显卡的问题，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，您将知道：

- 如何配置系统识别声卡。
- 测试声卡是否正常工作的方法。
- 如何排除声卡安装中的问题。
- 如何播放和编码MP3以及其它格式的音频。
- X 服务器如何支持视频。
- 哪些好的视频播放/压缩"ports"。
- 如何播放 DVD、 .mpg 以及 .avi 文件。
- 如何从 CD 和 DVD 中提取文件。
- 怎样配置电视卡。
- 如何配置图像扫描仪。

在读本章这前，您应该：

- 知道如何配置、安装一个新的内核 (配置FreeBSD的内核)

用mount(8) 命令去装载CD光盘，至少会产生一个错误， 更糟的情况下会产生 kernel panic。 这种媒体所用的编码与通常的ISO文件系统是不同的。

8.2. 安装声卡

8.2.1. 配置系统

在开始之前，您应该清楚声卡类型、所用的芯片以及它是 PCI 还是 ISA 卡。 FreeBSD 支持种类繁多的 PCI 和 ISA 卡。检查 硬件兼容说明 中支持的音频设备列表看看是否支持您的声卡，硬件兼容说明也会说明支持您声卡的是哪个驱动程序。

要使用声卡， 就应装载正确的驱动程序。完成的方式有两种： 最简单的是使用命令 kldload(8) 来装载一个内核模块，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"
```

上边实例用于 Creative SoundBlaster® Live! 声卡。 其它可装载的模块列在文件 /boot/defaults/loader.conf 里边。 如果不知道应该使用哪个驱动， 您可以尝试加载 snd_driver module:

```
# kldload snd_driver
```

这是个 meta 驱动，一次加载了最常见的设备驱动。 这会提高搜索正确驱动的速度。也可以通过 /boot/loader.conf 工具来加载所有的声卡驱动。

如果希望在加载了 snd_driver meta 驱动之后了解到底选择了哪种声卡， 可以通过使用 cat /dev/sndstat 来查询 /dev/sndstat 文件。

另外，您也可以把支持您声卡的代码静态地编译到内核里去。 下一节就采用这种方式支持硬件给出提示。 关于重新编译内核，请参考 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驱动 sound(4) 到内核中，
您需要添加下面这行到内核配置文件中：

```
device sound
```

接下来就是加入对我们所用声卡的支持了。 首先需要确定我们的声卡需要使用哪一个驱动。 您可以参考 硬件兼容列表 所列出的音频设备， 以确定您声卡的驱动。 例如， Creative SoundBlaster® Live! 声卡由 snd_emu10k1(4) 驱动来支持。 要添加它， 需要在内核编译配置文件中加入下面一行：

```
device snd_emu10k1
```

一定要阅读驱动的联机手册了解如何使用它们。 关于内核配置文件中声卡驱动的具体写法， 也可以在 /usr/src/sys/conf/NOTES 文件中找到。

非即插即用的 ISA 卡可能需要您为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， 并将其中的配置传给内核。 例如， 旧式的 Creative SoundBlaster® 16 ISA 非即插即用卡需要使用 snd_sbc(4) 驱动并配合 snd_sb16(4)。
您可以在内核编译配置文件中增加如下配置：

```
device snd_sbc
device snd_sb16
```

还有下面这些到 /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"
```

这样，声卡使用 0x220 I/O 端口和 IRQ 5。

在 /boot/device.hints 文件中所使用的语法， 在 sound(4)
联机手册中以及所用的具体声卡驱动的联机手册中， 会进行进一步的讲解。

上面所展示的是默认的配置。 有时候， 您可能需要更改 IRQ 或其他配置， 以适应声卡的实际情况。 查看 snd_sbc(4) 联机手册了解更多信息。

8.2.2. 测试声卡

用修改过的内核重起，或者加载了需要的模块之后， 声卡将会出现在您的系统消息缓存中 (dmesg(8))，就像这样：

```
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>
```

声卡的状态可以通过 /dev/sndstat 文件来查询：

```
# 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)
```

您系统的输出可能与此不同。如果没有看到 pcm 设备，回顾并检查一下前面做的。
重新检查您的内核配置文件并保证选择了正确的设备。 常见问题列在 常见问题 一节。

如果一切正常，您现在应该拥有一个多功能声卡了。 如果您的 CD-ROM 或者 DVD-ROM
驱动器的音频输出线已经与声卡连在一起， 您可以把 CD 放入驱动器并用 cdcontrol(1) 来播放：

```
% cdcontrol -f /dev/acd0 play 1
```

许多应用程序，比如 audio/workman 可以提供一个友好的界面。 您可能想要安装一个应用程序比如 audio/mpg123 来听 MP3 音频文件。

另一种快速测试声卡的方法， 是将数据发送到 /dev/dsp， 像这样做：

```
% cat filename > /dev/dsp
```

这里 filename 可以是任意文件。 这行命令会产生一些噪音，证明声卡果真在工作。

设备节点 /dev/dsp* 会在需要的时候自动产生。 如果没有使用它们， 则它们不会出现在 ls(1) 的输出中。

声卡混音级别可以通过 mixer(8) 命令更改。 更多细节可以在 mixer(8) 联机手册中找到。

8.2.2.1. 常见问题

错误信息	解决方法
sb_dspwr(XX) timed out	I/O端口没有设置正确。
bad irq XX	IRQ设置不正确。确信设定的IRQ和声卡的IRQ是一样的。
xxx: gus pcm not attached, out of memory	没有足够的内存空间供设置使用。
xxx: can’ t open /dev/dsp!	使用命令 `fstat

另一个问题是许多新式的显卡本身包含它们自己的声音驱动， 用以配合 HDMI 这样的设备使用。
这个声音设备有时会在真正的声卡之前被探测到， 从而成为默认的回放设备， 而使真正的声卡无法发声。
要检查这种情况， 运行 dmesg 并观察 pcm。 其输出类似下面这样：

```
...
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
...
```

此处显卡 (NVidia) 先于真正的声卡 (Realtek ALC889) 被探测到。 要使用声卡作为默认的回放设备， 将 hw.snd.default_unit 改为对应的设备编号：

```
# sysctl hw.snd.default_unit=n
```

这里的 n 是希望使用的声音设备编号， 在这个例子中是 4。 您可以在 /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， 会希望多个音源能够同时播放， 例如， esound 或者 artsd 就可能不支持与其它程序共享音频设备。

FreeBSD 可以通过 虚拟声道(Virtual Sound Channels) 来达到这样的效果， 它可以用 sysctl(8) 来启用。

虚拟的声道可以能过在内核里混合声音来混合声卡里播放的声道。

使用三条sysctl命令来设置虚拟声道的数目。 如果您是 root 用户， 执行下面的操作：

```
# sysctl dev.pcm.0.play.vchans=4
# sysctl dev.pcm.0.rec.vchans=4
# sysctl hw.snd.maxautovchans=4
```

上面的实例设定了4个虚拟声道，这也是实际上所使用的数目。 dev.pcm.0.play.vchans=4 和 dev.pcm.0.rec.vchans=4 是 pcm0 用来播放与录音的虚拟声道数， 一当链接上一个设备它就可配置了。 hw.snd.maxautovchans 是分配给新的音频设备的虚拟声道数， 此时这个设备要用 kldload(8) 来链接。 因为 pcm 模块可以独立装载许多硬件驱动程序， 因此 hw.snd.maxautovchans 也就可以存储分配给以后链接到的设备的虚拟声道数。 可参阅 pcm(4) 手册页义获取更多细节。

您不能在使用某个设备的时候改变其虚拟通道数。 首先需要关闭所有使用该设备的程序， 如音乐播放器或声音服务。

当应用程序请求 /dev/dsp0 时， 系统会自动为其分配正确的 pcm 设备。

8.2.4. 如何设置混音器通道值

不同的混音通道的默认音量是硬编码进 pcm(4) 驱动程序的。 同时，
也有很多应用或服务程序提供了允许用户直接设置并记住这些值的功能。
不过这并不是一个很好的解决方案， 您可能希望在驱动一级有一个可以设置的默认值。 这可以通过在 /boot/device.hints 定义适当的值来实现。 例如：

```
hint.pcm.0.vol="50"
```

这将在 pcm(4) 模块加载时， 将通道音量设置为默认的 50。

8.3. MP3音频

MP3 (MPEG Layer 3 Audio)达到过CD音质的效果，FreeBSD工作站没理由会缺少这样的好东东。

8.3.1. MP3播放器

目前为止， 最为流行的 X11 MP3 播放器是 XMMS (X 多媒体系统)。 Winamp 的肤面可以直接用于 XMMS， 因为它的 GUI 几乎和 Nullsoft 的 Winamp 完全一样。 另外， XMMS 也提供了内建的插件支持。

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。 如果您熟悉 Winamp， 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 可以在执行时通过命令行指定声音设备和要播放的 MP3 文件， 假设你的声音设备是 /dev/dsp1.0 并且你想要播放的 MP3 文件为 Foobar-GreatestHits.mp3 你可以键入以下的命令：

```
# 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. 抓取CD音轨

在对CD或CD音轨编码成MP3之前， CD上的音频数据应先抓到硬盘里。
这个可以通过复制原始的CDDA(CD数字音频)数据成为波形(WAV)文件。

工具 cdda2wav 是 sysutils/cdrtools 套件的一部份，可用来从CD中获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` 用户) 把整张 CD 分割成单个 (每个音轨) 的WAV文件：

```
# cdda2wav -D 0,1,0 -B
```

cdda2wav 支持 ATAPI (IDE)光驱。 从IDE光驱中抓取音轨， 需要用设备名称代替SCSI的单元号。 例如，
想从 IDE 光驱中抓取第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 的输出相对应。

抓取单轨，要使用选项 -t，如下所示：

```
# cdda2wav -D 0,1,0 -t 7
```

这个实例用于抓取第七个音轨。要抓取一定范围的音轨，如从1到7：

```
# cdda2wav -D 0,1,0 -t 1+7
```

利用dd(1)也可以从ATAPI光驱中抓取音轨，从 复制音频 CD 可以了解更多。

8.3.3. MP3 编码

现今，可选的MP3编码器是 lame。 Lame 可以从ports树里的 audio/lame 处找到。

利用抓取的WAV文件，下边的命令就可以把 audio01.wav 转换成 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)。 许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)" 模式的开关。 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。
其它的编码选项可以查询 lame 的联机手册。

8.3.4. MP3 解码

要把MP3歌曲刻录成音乐CD，就需要把它转换成非压缩的波形(WAV)格式。 XMMS 和 mpg123 都支持把MP3输出成非压缩格式文件。

在 XMMS 中输出到磁盘：

1. 启动 XMMS.
2. 在窗口里右击鼠标，弹出 XMMS 菜单。
3. 在 选项(Options) 里选择 设定(Preference)。
4. 改变输出插件成 "写磁盘插件(Disk Writer Plugin)"。
5. 按 配置(Configure)。
6. 输入或选择一个目录用于存放解压的文件。
7. 象平常一样，把MP3文件装入到 XMMS 里边， 把音量调节到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，记得把默认的输出插件设回原来的值。

用 mpg123 进行标准输出：

1. 执行 ` mpg123 -s audio01.mp3 > audio01.pcm`

XMMS 输出的文件是波形(WAV)格式， 而 mpg123 则把MP3转换成无压缩的PCM 音频数据。两种格式都支持用 cdrecord 刻录成音乐CD。 使用 burncd(8) 您就必须使用无压缩的PCM。
如果选择波形格式， 就要注意在每道开始时的一小点杂音， 这段声音是波形文件的头部份。 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
```

阅读 创建和使用光学介质(CD) 这部份可以了解到更多在 FreeBSD 里刻盘的信息。

8.4. 视频回放

视频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处音频那么顺利。

在开始之前，您要了解显卡的类型以及它所用的芯片的类型。 尽管 Xorg 支持大量的显卡，但能达到好的回放效果的却寥寥无几。 在X11运行时，您可以使用命令 xdpyinfo(1) 获得使用您的显卡的X服务器所支持的扩展列表。

为了评估各种播放器和设置，您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 里去找DVD文件， 因此， 您会发现建立符号链接到恰当的设备会很有用：

```
# ln -sf /dev/acd0 /dev/dvd
# ln -sf /dev/acd0 /dev/rdvd
```

注意：由于 devfs(5) 本身的原因， 像这样手工建立的链接在重启后将不会存在。
想要无论什么时候您启动系统都能自动建立符号链接， 那就把下边这行加到 /etc/devfs.conf 里边：

```
link acd0 dvd
link acd0 rdvd
```

另外，DVD解密要求调用专用的DVD-ROM函数，要求把许可定到DVD设备里。

为了改善 X11 界面使用共享内存的能力， 建议提高一些 sysctl(8) 变量的值：

```
kern.ipc.shmmax=67108864
kern.ipc.shmall=32768
```

8.4.1. 测定视频的性能

在X11下有几种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，
下边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 的每一个版本， 都会有很大的突破。

常见图像接口列表：

1. X11: 一般性的使用共享内存的X11输出。
2. XVideo: 一种X11接口扩展，支持任何X11图像的可拖拉。
3. SDL: 简单直接媒体层。
4. DGA: 直接图片存取。
5. SVGAlib: 低层次掌控图片层。

8.4.1.1. XVideo

Xorg 有种扩展叫做 XVideo (或称Xvideo, Xv, xv)，
它可以通过一个特殊的加速器直接把图像显示在可拖拉的对象里。 即使在低端的计算机 (例如我的PIII 400 Mhz膝上电脑)， 这个扩展也提供了很好的播放质量。

要了解这一扩展是否在正常工作， 使用 xvinfo 命令：

```
% xvinfo
```

如果显示结果如下，那您的显卡就支持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
```

同时注意：列出来的格式(YUV2, YUV12, 等等) 并不总是随着
XVdieo的每一次执行而存在。没有它们可能会迷惑某些人。

如果结果看起来是这样：

```
X-Video Extension version 2.2
screen #0
no adaptors present
```

那么您的显卡可以就不支持XVideo功能。

如果您的卡不支持XVideo， 则只是说明您的显示器在满足刷新图像的计算要求上存在更大的困难。
尽管显卡和处理器很重要，您仍然会有个不错的显示效果。 此外， 您也可以参考我们提供的文献， 在
进一步了解 中有所介绍。

8.4.1.2. 简单直接媒体层

简单直接媒体层(SDL)，原意是做为 Microsoft® Windows®、BeOS 以及 UNIX®
之间的端口层，允许跨平台应用发展，更高效地利用声卡和图形卡。SDL 层可以在低层访问硬件，
有时这样做就比 X11 接口层更为高效。

关于 SDL， 可以参考 devel/sdl12。

8.4.1.3. 直接图形存取

直接图形存取 (Direct Graphics Access) 是一种 X11 扩展， 通过它， 应用程序能够绕过 X 服务，
并直接修改画面缓存 (framebuffer)。 由于它依赖一种底层的内存映射来实现其功能，
因此使用它的程序必须以 root 身份来执行。

DGA 扩展可以通过 dga(1) 来完成测试和性能测量。 运行 dga 时， 它将随按键改变现实的颜色。 按 `q`
退出这个程序。

8.4.2. Ports 和 包(Packages) 对视频的解决

这部份主要讨论在 FreeBSD Ports 集中提供的可用于视频回放的软件。
视频回放在软件发展中是个很活跃的领域， 并且各种不同程序的功能可能与这里的描述不尽相同。

首先要弄清楚的重要一点是在 FreeBSD 上使用的视频程序其发展与在 Linux 里使用的是一样的。
大部份程序都还处在β阶段。使用 FreeBSD 的包可能面对的问题：

1. 一个应用程序不能播放其它程序制作的文件。
2. 一个应用程序不能播放其自已制作的文件。
3. 不同机上的同样的程序，各自重新建立(rebuild)了一次， 播放同一个文件结果也会有不同。
4. 一个看起来没什么的过滤器， 如图像尺寸的调整， 也有可能因为一个调整例程的问题变得很不象样。
5. 应用程序频繁地留下垃圾(dumps core)。
6. 没有随 port 一起安装的文档可以在网上或者 port 的 work 目录中找到。

这些程序中许多也体现了 "Linux主义"。即， 有些问题来自于(程序)使用的标准库存在于Linux的发行版中，
或者有些是 Linux 内核的功能， 而该程序的作者事先所假定了的是 Linux内核。这些问题并不总是被 port
编护人员注意到或处理过， 这也就可能导致如下问题：

1. 使用/proc/cpuinfo去检测处理器的特性。
2. 滥用线程可能导致一个程序悬挂完成，而不是完全中止。
3. 软件还不属于FreeBSD Ports集，而又与其它程序经常地一起使用。

现在，这些程序的开发人员也已同 port 的维护人员进行了联合， 以减少制作port时出错。

8.4.2.1. MPlayer

MPlayer 是近来开发的同时也正迅速发展着的一个视频播放器。 MPlayer 团队的目标是在 Linux 和其它
UNIX 系统中的速度和机动性能。 在团队的创始人实在受不了当时可用的播放器的性能时，
这个计划就开始了。 有人也许会说图形接口已经成为新型设计的牺牲品。
但是一旦您习惯了命令行选项和按键控制方式，它就能表现得很好。

8.4.2.1.1. 创建MPlayer

MPlayer 可以从 multimedia/mplayer 找到。 MPlayer 在联编过程中会进行许多硬件检测，
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
完成联编而不是安装预编译的包就很重要。 另外， 在 make 命令行还可以指定许多选项， 在 Makefile
中有所描述， 接下来我们开始联编：

```
# 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 选项对于绝大多数用户来说是够用了。 不过， 如果您需要 XviD 编解码器， 则必须指定 WITH_XVID 这个命令行选项。 默认的 DVD 设备也可以用 WITH_DVD_DEVICE 选项来定义， 其默认值是 /dev/acd0。

撰写这一章的时候， MPlayer port 的联编过程包括了 HTML 文档和两个可执行文件， mplayer 和 mencoder， 后者是一个视频再编码工具。

MPlayer 的 HTML 文档提供了丰富的内容。 如果读者发现本章中缺少关于视频硬件的一些信息， 则 MPlayer 的文档将是十分详尽的补充。 如果您正在找关于 UNIX® 中的视频支持的资料，
您绝对应该花一些时间来阅读 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。
输入下边的内容来建立这个必须的子目录：

```
% cd /usr/ports/multimedia/mplayer
% make install-user
```

在 mplayer 的手册里列出了它的命令选项。 HTML文档里有更为详细的信息。 这部份里，
我们只是描述了很少的常见应用。

要播放一个文件，如 testfile.avi， 可以通过各种视频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
```

所有这些选项都是值得一试的， 因为它们的性能依赖很多因素，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 是要播放的节目编号， 而 DEVICE 则是 DVD-ROM 的设备节点。 例如， 要播放 /dev/dvd 的第三个节目：

```
# mplayer -vo xv dvd://3 -dvd-device /dev/dvd
```

可以在编译 MPlayer 时， 通过 WITH_DVD_DEVICE 来指定默认的 DVD 设备。
系统内定的默认设备是 /dev/acd0。 更多细节， 请参考 port 的 Makefile。

要停止、暂停、前进等等，可以参考设定的按键---这些可以通过 mplayer -h 得到或查看手册。

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`。

为了让 mplayer 的命令行不是太长，使用者可以通过建立一个文件 .mplayer/config 来设定如下默认选项：

```
vo=xv
fs=yes
zoom=yes
```

最后，mplayer 可以把DVD题目(title)抓取成为 .vob 文件。为了从DVD中导出第二个题目，请输入：

```
# mplayer -dumpstream -dumpfile out.vob dvd://2 -dvd-device /dev/dvd
```

输出文件 out.vob 将是 MPEG 并且可以被这部份描述的其它 "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 文档则帮助不大。 有无数种方法来提高视频品质、 降低比特率、 修改格式，而这些技巧可能会影响性能。 下面是几个例子， 第一个是简单地复制：

```
% mencoder input.avi -oac copy -ovc copy -o output.avi
```

不正确的命令选项组合可能使生成的文件不能被 mplayer 播放。因此，如果您只是想抓取文件， 一定在 mplayer 里使用 “-dumpfile” 。

转换 input.avi 成为带有MPEG3音频编码 (要求 audio/lame) 的MPEG4编码：

```
% mencoder input.avi -oac mp3lame -lameopts br=192 \
   -ovc lavc -lavcopts vcodec=mpeg4:vhq -o output.avi
```

这样就产生了可被 mplayer 和 `xine`播放的输出。

input.avi 可以换成 dvd://1 -dvd-device /dev/dvd 并以 root 的身份来执行， 以重新对 DVD 节目进行编码。 由于您第一次做这样的工作时很可能会对结果不太满意， 建议您首先把节目复制成文件，然后对它进行操作。

8.4.2.2. xine视频播放器

xine 视频播放器是一个关注范围很广的项目， 它不仅看准多合一的视频解决，而且出品了一个可再用的基本库和一个可扩展插件的可执行模块。 发行有 "包" 和port版本--multimedia/xine。

xine 播放器仍然很粗糙， 但这很显然与好开头无关。实际上 xine 要求你有快速的 CPU 和快速的显卡来运行，或者需要支持 XVideo 扩展。 图形界面(GUI)可以使用，但很勉强。

到写这章时，还没有可用于播放CSS编码的DVD文件的输入模块随同 xine 一起发行。第三方的建造(builds)里内建有这样的模块， 但都不属于FreeBSD Ports 集。

与MPlayer 相比， xine 为用户考虑得更多， 但同时，对用户来说也少了很多有条理的控制方式。 xine 播放器在XVideo接口上做得不错。

默认情况下，播放器 xine 启动的时候会使用图形界面。那么就可以使用菜单打开指定的文件：

```
% xine
```

另外，没有图形界面也可以使用如下命令立即打开播放文件：

```
% xine -g -p mymovie.avi
```

8.4.2.3. 使用transcode

transcode 这个软件并不是播放器， 而是一系列用于对视频和音频文件进行重新编码的工具。 通过使用 transcode， 就可以拥有使用带 stdin/stdout 接口的命令行工具来合并视频文件，
以及修复坏损文件的能力。

在联编 multimedia/transcode port 时可以指定大量选项， 我们建议使用下面的命令行来构建 transcode：

```
# make WITH_OPTIMIZED_CFLAGS=yes WITH_LIBA52=yes WITH_LAME=yes WITH_OGG=yes
\
WITH_MJPEG=yes -DWITH_XVID=yes
```

对于多数用户而言， 前述配置已经足够了。

为了说明 transcode 的功能， 下面的例子展示了如何将 DivX 转换为 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。 您甚至可以直接将这个文件刻录到 CD-R 介质上来创建 Video CD， 如果希望这样做的话， 需要安装 multimedia/vcdimager 和 sysutils/cdrdao 这两个程序。

transcode 提供了联机手册， 但您仍应参考 transcode wiki 以了解更多信息和例子。

8.4.3. 进一步了解

FreeBSD里不同的视频软件包正迅速发展中。 很可能在不久的将来，这里所谈到的问题都将得到解决。
同时，有些人想超越FreeBSD的音/像(A/V)能力， 那他们就不得不从一些FAQ和指南里学知识，
并使用一些不同的应用程序。 这里就给这些读者指出一些补充信息。

MPlayer 文档 是很技术性的。 这些文档可以给那些希望获得关于UNIX®视频高级技术的人们提供参考。
MPlayer 邮件列表很不喜欢没耐心阅读文档的人， 如果您发现什么问题想报告给他们，请首先RTFM。

xine HOWTO 里边有一章是关于提高性能的，对所有的播放器都很适应。

最后是一些很有前途的程序，读者可以试一下：

- Avifile，它就是 multimedia/avifile port。
- Ogle 它就是 multimedia/ogle port。
- Xtheater
- multimedia/dvdauthor， 一个制作 DVD 节目的源码开放包。

8.5. 安装电视卡

8.5.1. 介绍

电视卡可以让您在您的计算机里观看到无线或有线电视。 许多卡是通过RCA或S-video输入接收复合视频，而且有些卡还带有调频广播接收器。

FreeBSD 通过bktr(4)驱动程序，提供了对基于PCI的电视卡的支持， 要求这些卡使用的是Brooktree Bt848/849/878/879 或 Conexant CN-878/Fusion 878a视频采集芯片。
您还要确保这个板上带的有被支持的调谐器， 参考bktr(4)手册查看所支持的调谐器列表。

8.5.2. 增加驱动程序

要使用您的卡，您就要装载bktr(4)驱动程序。 这个可以通过往 /boot/loader.conf 里边添加下边一行来实现。象这样：

```
bktr_load="YES"
```

另外，您也可以把这个驱动编译进内核， 要是这样的话，就把下边几行加到内核配置里去：

```
device  bktr
device  iicbus
device  iicbb
device  smbus
```

这些附加的设备驱动程序是必须的， 因为卡的各组成部分是能过一根I2C总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您的系统里，您须要重启系统。
在启动过程中，您的电视卡应该显示为up(启动)，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，这些信息可能因您的硬件不同而有所区别。 但是您应该能检查那个调制器是否被正确检测到了，可能要忽略一些检测到的同sysctl(8) MIB（管理系统库）和内核配置文件选项一起的参数。
例如，如果您想强制使用Philips(飞利浦) SECAM制式的调谐器， 您就应把下列行加到内核配置文件里：

```
options OVERRIDE_TUNER=6
```

或者，您直接使用sysctl(8)：

```
# sysctl hw.bt848.tuner=6
```

请参见 bktr(4) 手册和 /usr/src/sys/conf/NOTES 文件， 以了解更多详细关于可用选项的资料。

8.5.3. 有用的应用程序

要使用您的电视卡，您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 也是一款电视应用程序，功能同 fxtv 一样。
- misc/alevt 解码和显示Videotext/Teletext。
- audio/xmradio， 一款用于一些电视卡的调频电台调谐器的程序。
- audio/wmtune， 一款用于电台调谐器的便捷的桌面程序。

更多的程序在FreeBSD Ports Collection(Ports 集)里。

8.5.4. 问题解决

如果您的电视卡遇到了什么问题， 您应该首先检查一下您的视频采集芯片和调谐器是不是真正的被bktr(4) 驱动程序支持，并且是不是使用了正确的配置选项。 想得到更多支持和关于您的电视卡的各种问题，您可以接触和使用FreeBSD 多媒体应用邮件列表 邮件列表的压缩包。

8.6. 图象扫描仪

8.6.1. 介绍

在 FreeBSD 中， 访问扫描仪的能力， 是通过 SANE (Scanner Access Now Easy) API 提供的。 SANE 也会使用一些 FreeBSD 设备驱动来访问扫描仪硬件。

FreeBSD 支持 SCSI 和 USB 扫描仪。 在做任何配置之前请确保您的扫描仪被 SANE 支持。 SANE 有一个 支持的设备 列表， 可以为您提供有关扫描仪的支持情况和状态的信息。 在 FreeBSD 8.X 之前版本的系统中， uscanner(4) 手册页也提供了系统支持的 USB 扫描仪列表。

8.6.2. 内核配置

上面提到 SCSI 和 USB 接口都是支持的。 取决于您的扫描仪接口， 需要不同的设备驱动程序。

8.6.2.1. USB 接口

默认的 GENERIC 内核包含了支持 USB 扫描仪需要的设备驱动。 如果您决定使用一个定制的内核，确保下面在您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， 还需要下面这行配置：

```
device uscanner
```

在这些 FreeBSD 版本中， 是通过设备驱动程序 uscanner(4) 来提供对 USB 扫描仪的支持的。 从 FreeBSD 8.0 开始， 这些支持则直接由 libusb(3) 函数库提供。

使用正确的内核重新引导系统之后， 插入 USB 扫描仪。 系统消息缓冲区 (使用 dmesg(8) 查看) 中会出现下面的信息， 表示检测到了扫描仪：

```
ugen0.2: <EPSON> at usbus0
```

或者， 对于 FreeBSD 7.X 系统而言：

```
uscanner0: EPSON EPSON Scanner, rev 1.10/3.02, addr 2
```

随 FreeBSD 版本不同， 这些信息表示扫描仪设备位于设备节点 /dev/ugen0.2 或 /dev/uscanner0。
在这个例子中， 我们使用的是 EPSON Perfection® 1650 USB 扫描仪。

8.6.2.2. SCSI 接口

如果您的扫描仪是 SCSI 接口的， 重要的是要知道您使用哪种 SCSI 控制器。 取决于所使用的 SCSI 芯片，
您需要调整内核配置文件。 GENERIC 的内核支持最常用的 SCSI 控制器。 请阅读 NOTES
文件并在您的内核配置文件中添加正确的行。 除了 SCSI 适配器驱动之外，
您还需要在内核配置文件中增加下述配置：

```
device scbus
device pass
```

在正确地联编并安装了内核之后， 就应该可以在系统启动时， 从系统消息缓冲中看到这些设备：

```
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
```

如果您的扫描仪没有在系统启动的时候加电， 很可能还需要强制手动检测一下，用 camcontrol(8)
命令执行一次 SCSI 总线扫描：

```
# 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 设备列表里：

```
# 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 设备的更多细节， 可查看 scsi(4) 和 camcontrol(8) 手册页。

8.6.3. SANE 配置

SANE 系统分为两部分： 后端 (graphics/sane-backends) 和前端 (graphics/sane-frontends)。
后端部分提供到扫描仪自身的访问。 SANE 的
支持设备列表详细说明了哪一个后端可以支持您的图象扫描仪。
如果您想使用您的设备，就必须为您的扫描仪选定正确的后端。 前端部分提供图形化的扫描界面
(xscanimage)。

要做的第一步就是安装 graphics/sane-backends port 或者 package。然后，使用 sane-find-scanner
命令来检查 SANE 系统做的扫描仪检测：

```
# sane-find-scanner -q
found SCSI scanner "AGFA SNAPSCAN 600 1.10" at /dev/pass3
```

输出显示了扫描仪的接口类型和扫描仪连接到系统上的设备节点。
生产厂家和产品型号可能没有显示，不过不重要。

一些 USB 扫描仪需要您加载固件，后端的手册页中有这方面的解释。 您也应该阅读
sane-find-scanner(1) 和 linprocfs(7) 手册页。

现在我们需要检查扫描仪是否可以被扫描前端识别。 默认情况下， SANE 后端自带一个叫做 sane(1)
的命令行工具。 这个命令允许您列出设备以及从命令行执行图片扫描。 -L 选项用来列出扫描仪设备：

```
# 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 扫描仪：

```
# 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'
给出了扫描仪所使用的后台名字 (epson2) 和设备节点 (/dev/ugen0.2)。

如果没有输出任何信息， 或提示没有识别到扫描仪， 则说明 sane(1) 无法识别它。
如果发生这种情况， 您就需要修改扫描仪支持后端的配置文件，
并定义所使用的扫描设备。 /usr/local/etc/sane.d/ 目录中包含了所有的后端配置文件。
这类识别问题经常会在某些 USB 扫描仪上发生。

linkend="scanners-kernel-usb"> 中所使用的 USB 扫描仪， `sane-find-scanner` 会给出下面的信息：

例如， 对于在 USB 接口， 在 FreeBSD 8.X 中，
扫描仪已经被很好地识别并能够正常工作了； 而对于更早版本的 FreeBSD 而言 (使用
uscanner(4) 驱动程序) sane-find-scanner 则会给出这样的信息：

```
# sane-find-scanner -q
found USB scanner (UNKNOWN vendor and product) at device
/dev/uscanner0
```

扫描仪被正确的探测到了，它使用 USB 接口，连接在 /dev/uscanner0 设备节点上。
我们现在可以检查看看扫描仪是否被正确的识别：

```
# 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).
```

由于扫描仪没有识别成功， 我们就需要编辑 /usr/local/etc/sane.d/epson2.conf 文件。
所用的扫描仪型号是 EPSON Perfection® 1650， 这样我们知道扫描仪应使用 epson
后端。确保阅读后端配置文件中的帮助注释。
改动非常简单：注释掉导致您的扫描仪使用错误接口的所有行
(在我们这种情况下，我们将注释掉从 scsi 开始的所有行，因为我们的扫描仪使用 USB
接口)，然后在文件的结尾添加指定的接口和所用的设备节点。 这种情况下，
添加下面这行：

```
usb /dev/uscanner0
```

请确保阅读后端配置文件提供的注释以及后端手册页了解更多细节，
并使用正确的语法。我们现在可以检验扫描仪是否被识别到了：

```
# scanimage -L
device `epson:/dev/uscanner0' is a Epson GT-8200 flatbed scanner
```

我们的 USB 扫描仪被识别到了。 此时如果商标和型号与扫描仪的实际情况不符，
并不会带来太大的麻烦。 您需要关注的是 `epson:/dev/uscanner0' 字段，
这个给了我们正确地后端名称和正确的设备节点。

一旦 scanimage -L 命令可以看到扫描仪， 配置就完成了。设备现在准备好等待扫描了。

sane(1) 允许我们从命令行执行图片扫描， 相比之下使用图形用户界面来执行图片扫描会更好。 SANE
提供了一个简单但实用的图形界面： xscanimage (graphics/sane-frontends)。

Xsane (graphics/xsane)是另一个流行的图形扫描前端。 这个前端提供了一些高级特性，
比如多样的扫描模式(photocopy，fax，等。)， 色彩校正，批量扫描，等等。这两个程序都可以作为 GIMP
的插件使用。

8.6.4. 授权其他用户访问扫描仪

前面所有的操作都是用 root 权限来完成的。 然而您可能需要让其他的用户也可以访问扫描仪。
用户需要有扫描仪所用的设备节点的读和写权限。 比如，我们的 USB 扫描仪使用设备节点 /dev/ugen0.2
实际上只是到实际设备节点 /dev/usb/0.2.0 的符号连接 (可以通过查看 /dev 目录的内容来确认这一点)。
设备节点本身和这个符号连接分别属于 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仪了， 不过， 出于显而易见的安全方面的原因， 在将用户加到特定的用户组，
特别是 wheel 组时， 无疑需三思而后行。 更好的解决方法是创建一个专门用于访问 USB 设备的组，
并让这个组的成员能够访问 USB 设备。

这里作为示例， 我们将会使用名为 usb 的组。 第一步是借助 pw(8) 命令来创建它：

```
# pw groupadd usb
```

接下来， 令 /dev/ugen0.2 符号连接和 /dev/usb/0.2.0 设备节点能够以 usb 组的身份来访问，
具体而言是配置正确的写权限 (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 文件中添加如下的设置来实现的：

```
[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为使用与之对应的 /dev/uscanner0：

```
[system=5]
add path uscanner0 mode 660 group usb
```

随后您还需要在 /etc/rc.conf 中添加下面的内容并重新启动：

```
devfs_system_ruleset="system"
```

关于这些配置的进一步细节请参考联机手册 devfs(8)。

现在， 只需将用户添加到 usb 组， 就可以使用扫描仪了：

```
# pw groupmod usb -m joe
```

更多详情， 请参见联机手册 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。 它负责管理内存、 执行安全控制、 网络、 磁盘访问等等。 尽管 FreeBSD 可以动态修改的现在已经越来越多， 但有时您还是需要重新配置和编译您的内核。

读完这章，您将了解：

- 为什么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创建和联编新的内核。
- 如何安装新内核。
- 如何处理出现的问题。

这一章给出的命令应该以 root 身份执行， 否则可能会不成功。

9.2. 为什么需要建立定制的内核?

过去， FreeBSD 采用的是被人们称作 "单片式" 的内核。 这种内核本身是一个大的程序，
它支持的设备不能够动态地加以改变， 而当希望改变内核的行为时， 就必须编译一个新的内核，
并重新启动计算机才可以使用它。

如今， FreeBSD 正在迅速地迁移到一种新的模型，
其特点是将大量内核功能放进可以动态加载和卸载的内核模块来提供。 这使得内核能够适应硬件的调整
(例如笔记本计算机中的 PCMCIA 卡)， 以及为内核引入新的功能， 而无需在编译内核时就将其添加进去。
这种做法称为模块化内核。

尽管如此， 仍然有一些功能需要静态地联编进内核。 有时，
这是由于这些功能与内核的结合非常紧密而无法实现动态加载，
还有一些情况是暂时没有人将这些功能改写为可动态加载的模块。

联编定制的内核是成为高级 BSD 用户所必须经历的一关。 尽管这一过程需要花费一些时间，
但它能够为您的 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对于 您 PC 硬件的支持。 这样做有很多好处， 例如：

- 更快地启动。 因为内核只需要检测您系统上的硬件， 启动时所花费的时间将大大缩短。
- 使用更少的内存。 由于可以删去不需要的功能和设备驱动， 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，
  从而使应用程序能够用到更多的内存。 正因为这样， 对 RAM 较小的系统来说定制内核就更为重要了。
- 支持更多的硬件。 定制的内核允许您增加类似声卡这样的 GENERIC 内核没有提供内建支持的硬件。

9.3. 发现系统硬件

在尝试配置内核以前，比较明智的做法是先获得一份机器硬件的清单。 当 FreeBSD
并不是主操作系统时，通过查看当前操作系统的配置可以很容易的
创建一份机器硬件的配置清单。举例来说， 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 系统 图标会指明 设备管理器 的位置。

如果机器上并不存在其他的操作系统， 系统管理员只能手动寻找这些信息了。其中的一个方法是使用 dmesg(8) 工具以及 man(1) 命令。FreeBSD 上大多数的驱动程序都有一份手册页（manual page）列出了所支持的硬件， 在系统启动的时候，被发现的硬件也会被列出。举例来说，
下面的这几行表示 psm 驱动找到了一个鼠标：

```
psm0: <PS/2 Mouse> irq 12 on atkbdc0
psm0: [GIANT-LOCKED]
psm0: [ITHREAD]
psm0: model Generic PS/2 mouse, device ID 0
```

这个驱动需要被包含在客户制定的内核配置文件里， 或着使用 loader.conf(5) 加载。

有时，dmesg 里只会显示来自系统消息的数据，
而不是系统启动时的检测信息。在这样的情况下，你可以查看文件 /var/run/dmesg.boot。

另一个查找硬件信息的方法是使用 pciconf(8) 工具， 它能提供更详细的输出，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
```

这个片断取自于 pciconf -lv 命令的输出，显示 ath 驱动找到了一个无线以太网设备。输入命令 man ath
就能查阅有关 ath(4) 的手册页（manual page）了。

还可以传给 man(1) 命令 -k 选项， 同样能获得有用的信息。例如：

```
# man -k Atheros
```

能得到一份包含特定词语的手册页（manual page）：

```
ath(4)              - Atheros IEEE 802.11 wireless network driver
ath_hal(4)            - Atheros Hardware Access Layer (HAL)
```

手头备有一份硬件的配置清单， 那么编译制定内核的过程就显得不那么困难了。

9.4. 内核驱动，子系统和模块

在编译一个制定的内核之前请三思一下这么做的理由， 如果仅是需要某个特定的硬件支持的话，
那么很可能已经存在一个现成的模块了。

内核模块存放在目录 /boot/kernel 中，并能由 kldload(8) 命令加载入正在运行的内核。
基本上所有的内核驱动都有特定的模块和手册页。比如， 下面提到的 ath 无线以太网驱动。
在这个设备的联机手册中有以下信息：

```
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" 则能在机器启动的时候动态加载这个模块。

某些情况下，则没有相关的模块。通常是一些子系统和非常重要的驱动， 比如，快速文件系统 (FFS) 就是一个内核必需的选项。 同样的还有网络支持 (INET)。不幸的是，
分辨一个驱动是否必需的唯一方法就是检查测试以下那个模块本身。

去除某个驱动的支持或某个选项会非常容易得到一个坏掉的内核。 举例来说，如果把 ata(4) 驱动从内核配置文件中去掉， 那么一个使用 ATA 磁盘设备的系统可能就变得无法引导，除非有在 loader.conf 中加载。当你无法确定的时候， 请检查一下那个模块并把它留在你的内核配置中。

9.5. 建立并安装一个定制的内核

首先对内核构建目录做一个快速的浏览。 这里所提到的所有目录都在 /usr/src/sys 目录中； 也可以通过 /sys 来访问它。 这里的众多子目录包含了内核的不同部分， 但对我们所要完成的任务最重要的目录是 arch/conf， 您将在这里编辑定制的内核配置； 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 (在日本比较流行的另一种 PC 硬件开发分支)。 在特定硬件架构目录中的文件只和特定的硬件有关； 而其余代码则是与机器无关的，
则所有已经或将要移植并运行 FreeBSD 的平台上都共享这些代码。 文件目录是按照逻辑组织的，
所支持的硬件设备、 文件系统， 以及可选的组件通常都在它们自己的目录中。

这一章提供的例子假定您使用 i386 架构的计算机。 如果您的情况不是这样，
只需对目录名作相应的调整即可。

如果您的系统中 没有/usr/src/sys 这样一个目录， 则说明没有安装内核源代码。
安装它最简单的方法是通过以 root 身份运行 sysinstall， 选择 Configure， 然后是 Distributions、 src， 选中其中的 base 和 sys。 如果您不喜欢 sysinstall 并且有一张 "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 -xzvf -
# cat /cdrom/src/sbase.[a-d]* | tar -xzvf -
```

接下来， 进入 arch/conf 目录下面， 复制 GENERIC 配置文件， 并给这个文件起一个容易辨认的名称，
它就是您的内核名称。例如：

```
# cd /usr/src/sys/i386/conf
# cp GENERIC MYKERNEL
```

通常，这个名称是大写的，如果您正维护着多台不同硬件的FreeBSD机器，
以您机器的域名来命名是非常好的主意。我们把它命名为 MYKERNEL就是这个原因。

将您的内核配置文件直接保存在 /usr/src 可能不是一个好主意。 如果您遇到问题， 删掉 /usr/src 并重新开始很可能是一个诱人的选择。 一旦开始做这件事，
您可能几秒钟之后才会意识到您同时会删除定制的内核配置文件。 另外， 也不要直接编辑 GENERIC， 因为下次您 更新代码 时它会被覆盖， 而您的修改也就随之丢失了。

您也可以考虑把内核配置文件放到别的地方， 然后再到 i386 目录中创建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 身份执行这些和接下来命令， 否则就会得到 的错误提示。

现在就可以用您喜欢的文本编辑器来编辑 MYKERNEL 了。 如果您刚刚开始使用 FreeBSD，
唯一可用的编辑器很可能是 vi， 它的使用比较复杂， 限于篇幅， 这里不予介绍， 您可以在 参考书目
一章中找到很多相关书籍。 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，
这很可能是一个不错的选择。 您可以修改配置文件中的注释以反映您的配置， 或其他与 GENERIC
不同的地方。

如果您在SunOS™或者其他BSD系统下定制过内核，那这个文件中的绝大部分将对您非常熟悉。
如果您使用的是诸如DOS这样的系统，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。

如果您和 FreeBSD project 进行了 代码同步， 则一定要在进行任何更新之前查看
/usr/src/UPDATING。
这个文件中描述了更新过的代码中出现的重大问题或需要注意的地方。
/usr/src/UPDATING 总是和您的 FreeBSD 源代码对应，
因此能够提供比手册更具时效性的新内容。

现在应该编译内核的源代码了。

Procedure: 联编内核

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
```

使用这种方法联编内核时， 需要安装完整的 FreeBSD 源代码。

默认情况下， 在联编您所定制的内核时， 全部 内核模块也会同时参与构建。
如果您希望更快地升级内核， 或者只希望联编您所需要的模块， 则应在联编之前编辑
/etc/make.conf：

```
MODULES_OVERRIDE = linux acpi sound/sound sound/driver/ds1 ntfs
```

这个变量的内容是所希望构建的模块列表。

```
WITHOUT_MODULES = linux acpi sound ntfs
```

这个变量的内容是将不在联编过程中编译的顶级模块列表。
如果希望了解更多与构建内核有关的变量， 请参见 make.conf(5) 联机手册。

新内核将会被复制到 /boot/kernel 目录中成为 /boot/kernel/kernel 而旧的则被移到 /boot/kernel.old/kernel。 现在关闭系统， 然后用新的内核启动计算机。 如果出现问题， 后面的一些故障排除方法 将帮您摆脱困境。 如果您的内核 无法启动， 请参考那一节。

其他与启动过程相关的文件， 如 loader(8) 及其配置， 则放在 /boot。
第三方或定制的模块也可以放在 /boot/kernel，
不过应该注意保持模块和内核的同步时很重要的， 否则会导致不稳定和错误。

9.6. 配置文件

配置文件的格式是非常简单的。 每一行都包括一个关键词， 以及一个或多个参数。 实际上，
绝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 中的顺序介绍所有关键字。 如果需要与平台有关的选项和设备的详细列表， 请参考与 GENERIC 文件在同一个目录中的那个 NOTES， 而平台无关的选项， 则可以在 /usr/src/sys/conf/NOTES 找到。

配置文件中还可以使用 include 语句。 这个语句能够在内核配置文件中直接引用其他配置文件的内容，
使得您能够使用较小的、 仅包含相对于现存配置的变动而减少维护所需的工作。 例如， 如果您只需对 GENERIC 内核进行少量定制， 在其中添加几个驱动程序和附加选项， 则只要维护相对于 GENERIC 的变化就可以了：

```
include GENERIC
ident MYKERNEL

options     IPFIREWALL
options     DUMMYNET
options     IPFIREWALL_DEFAULT_TO_ACCEPT
options     IPDIVERT
```

许多系统管理员会发现， 这种方法与先前从头开始写配置文件的方法相比， 可以带开相当多的好处：
本地采用的配置文件只表达与 GENERIC 内核的差异， 这样， 在升级的时候往往就不需要做任何改动，
而新加入 GENERIC 的功能就会自动加入到本地的内核， 除非使用 nooptions 或 nodevice 语句将其排除。
这一章余下的部分将着重介绍典型的配置文件， 以及内核选项和设备的作用。

如果您需要一份包含所有选项的文件， 例如用于测试目的， 则应以 root 身份执行下列命令：

```
# 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释的注释。 这个例子应该和您复制的 /usr/src/sys/i386/conf/GENERIC 非常接近。

```
machine    i386
```

这是机器的架构， 它只能是 amd64, i386, ia64, pc98, powerpc, 或 sparc64 中的一种。

```
cpu     I486_CPU
cpu     I586_CPU
cpu     I686_CPU
```

上面的选项指定了您系统中所使用的 CPU 类型。 您可以使用多个 CPU 类型 (例如， 您不确定是应该指定 I586_CPU 或 I686_CPU)。 然而对于定制的内核， 最好能够只指定您使用的那种 CPU。
如果您对于自己使用的 CPU 类型没有把握， 可以通过查看 /var/run/dmesg.boot 中的启动信息来了解。

```
ident       GENERIC
```

这是内核的名字。 您应该取一个自己的名字， 例如取名叫 MYKERNEL，
如果您一直在按照前面的说明做的话。 您放在 ident 后面的字符串在启动内核时会显示出来，
因此如果希望能够容易区分常用的内核和刚刚定制的内核， 就应该采取不同的名字 (例如，
您想定制一个试验性的内核)。

```
#To statically compile in device wiring instead of /boot/device.hints
#hints      "GENERIC.hints"     # Default places to look for devices.
```

device.hints(5) 可以用来配置设备驱动选项。 在启动的时候 loader(8) 将会检查缺省位置
/boot/devicehints。 使用 hints 选项您就可以把这些 hints 静态编译进内核。 这样就没有必要在
/boot下创建devicehints。

```
makeoptions    DEBUG=-g       # Build kernel with gdb(1) debug symbols
```

一般的 FreeBSD 联编过程， 在所联编的内核指定了 -g 选项时， 由于此选项将传递给 gcc(1)
表示加入调试信息， 因此会将调试符号也包含进来。

```
options     SCHED_ULE      # ULE scheduler
```

这是 FreeBSD 上使用的默认系统调度器。 请保留此选项。

```
options     PREEMPTION      # Enable kernel thread preemption
```

允许内核线程根据优先级的抢占调度。 这有助于改善交互性， 并可以让中断线程更早地执行，
而无须等待。

```
options     INET         # InterNETworking
```

网络支持，即使您不打算连网，也请保留它，大部分的程序至少需要回环网络（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
```

这将打开IPv6连接协议。

```
options         FFS             # Berkeley Fast Filesystem
```

这是最基本的硬盘文件系统，如果打算从本地硬盘启动，请保留它。

```
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。
尽管这项功能是由内核直接提供的， 但仍然需要在每个磁盘上启用它。 请检查 mount(8) 的输出，
以了解您系统中的磁盘上是否已经启用了 Soft Updates。 如果没有看到 soft-updates 选项， 则需要使用
tunefs(8) (对于暨存系统) 或 newfs(8) (对于新系统) 命令来激活它。

```
options         UFS_ACL         # Support for access control lists
```

这个选项将启用内核中的访问控制表的支持。 这依赖于扩展属性以及 UFS2， 以及在 文件系统访问控制表
中所介绍的那些特性。 ACL 默认是启用的， 并且如果已经在文件系统上使用了这一特性，
就不应再关掉它， 因为这会去掉文件的访问控制表， 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
```

通过使用额外的内存，这个选项可以加速在大目录上的磁盘操作。
您应该在大型服务器和频繁使用的工作站上打开这个选项，而在磁盘操作不是很重要的
小型系统上关闭它，比如防火墙。

```
options         MD_ROOT         # MD is a potential root device
```

这个选项将打开以基于内存的虚拟磁盘作为根设备的支持。

```
options         NFSCLIENT       # Network Filesystem Client
options         NFSSERVER       # Network Filesystem Server
options         NFS_ROOT        # NFS usable as /, requires NFSCLIENT
```

网络文件系统。 如果您不打算通过 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。 只要您不打算在启动时挂接由 DOS 格式化的硬盘分区， 就可以把它注释掉。
如前面所介绍的那样， 在您第一次挂接 DOS 分区时， 内核会自动加载需要的模块。 此外，
emulators/mtools 软件提供了一个很方便的功能， 通过它您可以直接访问 DOS
软盘而无需挂接或卸下它们 (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 (因为在首次使用数据
CD 时会自动加载)， 就可以把它注释掉。 音乐 CD 并不需要这个选项。

```
options     PROCFS      # Process filesystem (requires PSEUDOFS)
```

进程文件系统。 这是一个挂接在 /proc 的一个 "假扮的" 文件系统， 其作用是允许类似 ps(1)
这样的程序给出正在运行的进程的进一步信息。 多数情况下， 并不需要使用 PROCFS，
因为绝大多数调试和监控工具， 已经进行了一系列修改， 使之不再依赖 PROCFS：
默认安装的系统中并不会挂接这一文件系统。

```
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 上编译的使用旧式接口的应用程序， 就需要加入这一选项。 一般来说，
推荐在所有的 i386™ 系统上启用这个选项， 因为难免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， 则不需要这个选项。

```
options     COMPAT_FREEBSD5   # Compatible with FreeBSD5
```

如果希望支持在 FreeBSD 5.X 版本上编译， 且使用 FreeBSD 5.X 系统调用接口的应用程序，
则应加上这个选项。

```
options     COMPAT_FREEBSD6   # Compatible with FreeBSD6
```

如果希望支持在 FreeBSD 6.X 版本上编译， 且使用 FreeBSD 6.X 系统调用接口的应用程序，
则应加上这个选项。

```
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 版本上编译， 且使用 FreeBSD 7.X
系统调用接口的应用程序， 则应加上这个选项。

```
options     SCSI_DELAY=5000  # Delay (in ms) before probing SCSI
```

这将让内核在探测每个 SCSI 设备之前等待 5 秒。 如果您只有 IDE 硬盘驱动器， 就可以不管它，
反之您可能会希望尝试降低这个数值以加速启动过程。 当然， 如果您这么做之后 FreeBSD 在识别您的 SCSI
设备时遇到问题， 则您还需要再把它改回去。

```
options     KTRACE      # ktrace(1) support
```

这个选项打开内核进程跟踪，在调试时很有用。

```
options     SYSVSHM     # SYSV-style shared memory
```

提供System V共享内存(SHM)的支持，最常用到SHM的应该是X Window的XSHM延伸，
不少绘图相关程序会自动使用SHM来提供额外的速度。如果您要使用X Window，您最好加入这个选项。

```
options     SYSVMSG     # SYSV-style message queues
```

支持 System V 消息。 这只会在内核中增加数百字节的空间占用。

```
options     SYSVSEM     # SYSV-style semaphores
```

支持System V 信号量， 不常用到， 但只在kernel中占用几百个字节的空间。

ipcs(1) 命令的 -p 选项可以显示出任何用到这些 System V 机制的进程。

```
options     _KPOSIX_PRIORITY_SCHEDULING # POSIX P1003_1B real-time extensions
```

在 1993 年 POSIX® 添加的实时扩展。 在 Ports Collection 中某些应用程序会用到这些
（比如StarOffice™）。

```
options     KBD_INSTALL_CDEV  # install a CDEV entry in /dev
```

这个选项是在 /dev下建立键盘设备节点必需的。

```
options     ADAPTIVE_GIANT   # Giant mutex is adaptive.
```

内核全局锁 (Giant) 是一种互斥机制 (休眠互斥体) 的名字， 它用于保护许多内核资源。 现在，
这已经成为了一种无法接受的性能瓶颈， 它已经被越来越多地使用保护单个资源的锁代替。
ADAPTIVE_GIANT 选项将使得内核全局锁作为一种自适应自旋锁。 这意味着，
当有线程希望锁住内核全局锁互斥体， 但互斥体已经被另一个 CPU 上的线程锁住的时候， 它将继续运行，
直到那个线程释放锁为止。 一般情况下， 另一个线程将进入休眠状态并等待下一次调度。
如果您不确定是否应该这样做的话， 一般应该打开它。

请注意在 FreeBSD 8.0-RELEASE 及以后的版本，所有的互斥体默认都是自适应的，
除非在编译时使用 NO_ADAPTIVE_MUTEXES 选项，
明确的指定为非自适应。因此，内核全局锁（Giant）目前默认也是自适应的, 而且
ADAPTIVE_GIANT 选项已经从内核配置文件中移出。

```
device      apic        # I/O APIC
```

apic 设备将启用使用 I/O APIC 作为中断发送设备的能力。 apic 设备可以被 UP 和 SMP 内核使用， 但 SMP
内核必须使用它。 要支持多处理器， 还需要加上 options SMP。

只有在 i386 和 amd64 平台上才存在 apic 设备， 在其他硬件平台上不应使用它。

```
device      eisa
```

如果您的主机板上有EISA总线，加入这个设置。使用这个选项可以自动扫描并设置所有连接在EISA总线上的设备。

```
device      pci
```

如果您的主板有PCI总线，就加入这个选项。使用这个选项可以自动扫描PCI卡，并在PCI到ISA之间建立通路。

```
# Floppy drives
device      fdc
```

这是软驱控制器。

```
# ATA and ATAPI devices
device      ata
```

这个驱动器支持所有ATA和ATAPI设备。您只要在内核中加入device ata选项，
就可以让内核支持现代计算机上的所有PCI ATA/ATAPI设备。

```
device      atadisk         # ATA disk drives
```

这个是使用 ATAPI 硬盘驱动器时必须加入的选项。

```
device      ataraid         # ATA RAID drives
```

这个选项需要 device ata， 它用于 ATA RAID 驱动。

```
device      atapicd          # ATAPI CDROM drives
```

这个是ATAPI CDROM驱动器所必须的。

```
device      atapifd         # ATAPI floppy drives
```

这个是ATAPI 软盘驱动器所必须的。

```
device      atapist         # ATAPI tape drives
```

这个是ATAPI 磁带机驱动器所必须的.

```
options      ATA_STATIC_ID       # Static device numbering
```

这指定对控制器使用其静态的编号； 如果没有这个选项， 则会动态地分配设备的编号。

```
# 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。可以注释掉您系统中没有的设备。 如果您只有IDE设备，您可以把这些一起删掉。
*_REG_PRETTY_PRINT 这样的配置， 则是对应驱动程序的调试选项。

```
# 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)
```

SSCSI外围设备。也可以像上面一样操作。

目前系统提供的 USB umass(4) 以及少量其它驱动使用了 SCSI 子系统，
尽管它们并不是真的 SCSI 设备。 因此， 如果在内核配置使用了这类驱动程序，
请务必不要删除 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。如果您没有这些，可以把它们注释掉或是删掉。

```
# atkbdc0 controls both the keyboard and the PS/2 mouse
device      atkbdc   # AT keyboard controller
```

键盘控制器（atkbdc）提供AT键盘输入以及PS/2指针设备的I/O服务。 键盘驱动程序（
atkbd）与PS/2鼠标驱动程序（psm）需要这个控制器，所以不要删除它。

```
device      atkbd    # AT keyboard
```

atkbd驱动程序，与atkbdc控制器一起使用， 提供连接到AT键盘控制器的AT
84键盘与AT加强型键盘的访问服务。

```
device      psm      # PS/2 mouse
```

如果您的鼠标连接到PS/2鼠标端口，就使用这个设备驱动程序。

```
device      kbdmux      # keyboard multiplexer
```

针对键盘多路选择器的基本支持。 如果您不打算使用多个键盘， 则可以放心地删除这一行。

```
device      vga      # VGA video card driver
```

显卡驱动。

```
device      splash    # Splash screen and screen saver support
```

启动时的 splash 画面！ 屏幕保护程序也需要这一选项。

```
# syscons is the default console driver, resembling an SCO console
device      sc
```

sc 是默认的控制台驱动程序， 类似 SCO 控制台。 由于绝大部分全屏幕程序都通过类似 termcap
这样的终端数据库函数库赖访问控制台， 因此无论您使用这个或与 VT220 兼容的 vt 都没有什么关系。
如果您在运行这种控制台时使用全屏幕程序时发生问题， 请在登录之后将 TERM 变量设置为 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
```

这是一个兼容 VT220 的控制台驱动， 它同时能够向下兼容 VT100/102。 在同 sc
硬件不兼容的一些笔记本上它能够运行的很好。 当然， 登录系统时请把 TERM 变量设置为 vt100 或
vt220。 此驱动在连接网络上大量不同的机器时也被证明非常有用， 因为此时 termcap 或 terminfo
通常没有可用的 sc 设备 - 而 vt100 则几乎每种平台都支持。

```
device      agp
```

如果您的机器使用 AGP 卡， 请把上面一行加入配置。 这将启用 AGP， 以及某些卡上的 AGP GART 支持。

```
# 电源管理支持 (参见 NOTES 了解更多选项)
#device      apm
```

高级电源管理支持。对笔记本有用，不过在 GENERIC 里默认禁用。

```
# 增加 i8254 的 挂起/恢复 支持。
device      pmtimer
```

用于电源管理事件， 例如 APM 和 ACPI 的时钟设备驱动。

```
# 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。如果您使用膝上型计算机，您需要这个。

```
# 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 而您另有一个串口在 COM2，则必须把调制解调器的 IRQ 改为 2 (由于晦涩的技术原因，IRQ2 = IRQ 9) 才能够在 FreeBSD 中访问它。如果有多口的串口卡，请参考 sio(4) 以了解需要在 /boot/device.hints 中进行的设置。某些显卡 (特别是基于 S3 芯片的卡) 使用形如 0x*2e8 的 IO 地址，而许多廉价的串口卡不能够正确地对 16-位 IO 地址空间进行解码，因此它们会产生冲突，并造成 COM4 实际上无法使用。

每一个串口都需要有一个唯一的 IRQ (除非您使用支持中断分享的串口卡)，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)
```

提供并行总线的支持。

```
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
```

这是针对并行网络接口的驱动器。

```
device      ppi     # Parallel port interface device
```

普通用途的I/O ("geek port") + IEEE1284 I/O.

```
#device      vpo     # Requires scbus and da
```

这是针对Iomega Zip驱动器的。它要求scbus和da的支持。 最好的执行效果是工作在EPP 1.9模式。

```
#device      puc
```

如果您有由 puc(4) 支持的 "哑" 串行或并行 PCI 卡， 则应去掉这一行的注释。

```
# 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)
```

多种PCI网卡驱动器。注释或删除您系统中没有的设备.

```
# PCI Ethernet NICs that use the common MII bus controller code.
# NOTE: Be sure to keep the 'device miibus' line in order to use these NICs!
device      miibus    # MII bus support
```

MII总线支持对于一些PCI 10/100 Ethernet NIC来说是必需的。

```
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MII总线控制器代码的驱动器。

```
# 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 以太网卡驱动。 参见 /usr/src/sys/i386/conf/NOTES
以了解关于哪个驱动程序能够驱动您的网卡的细节。

```
# Wireless NIC cards
device      wlan        # 802.11 support
```

通用 802.11 支持。 这行配置是无线网络所必需的。

```
device      wlan_wep    # 802.11 WEP support
device      wlan_ccmp   # 802.11 CCMP support
device      wlan_tkip   # 802.11 TKIP support
```

针对 802.11 设备的加密支持。 如果希望使用加密和 802.11i 安全协议， 就需要这些配置行。

```
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.
```

用以支持多种无线网卡。

```
# Pseudo devices
device  loop        # Network loopback
```

这是 TCP/IP 的通用回环设备。 如果您 telnet 或 FTP 到 localhost (也就是 127.0.0.1)
则将通过这个设备回到本机。 这个设备是 必需的。

```
device  random      # Entropy device
```

Cryptographically secure random number generator.

```
device  ether       # Ethernet support
```

ether 只有在使用以太网卡时才需要。 它包含了通用的以太网协议代码。

```
device  sl          # Kernel SLIP
```

sl 用以提供 SLIP 支持。 目前它几乎已经完全被 PPP 取代了， 因为后者更容易配置，
而且更适合调制解调器之间的连接， 并提供了更强大的功能。

```
device  ppp         # Kernel PPP
```

这一选项用以提供内核级的 PPP 支持， 用于拨号连接。 也有以用户模式运行的 PPP 实现， 使用 tun
并提供包括按需拨号在内的更为灵活的功能。

```
device  tun         # Packet tunnel.
```

它会被用户模式的 PPP 软件用到。 参考本书的 PPP 以了解更多的细节。

```
device  pty         # Pseudo-ttys (telnet etc)
```

这是一个 "pseudo-terminal" 或模拟登入端口。 它用来接收连入的 telnet 以及 rlogin 会话、 xterm，
以及一些其它程序如 Emacs 等。

```
device  md          # Memory disks
```

内存盘伪设备。

```
device  gif         # IPv6 and IPv4 tunneling
```

它实现了在 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 设备是 "自动克隆" 的， 它会根据需要自动创建设备节点。

```
device   faith      # IPv6-to-IPv4 relaying (translation)
```

这个伪设备能捕捉发给它的数据包，并把它们转发给 IPv4/IPv6 翻译服务程序。

```
# 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
```

这是 Berkeley 包过滤器。这个伪设备允许网络接口被置于混杂模式， 从而，截获广播网 (例如，以太网) 上的每一个数据包。 截获的数据报可以保存到磁盘上，也可以使用 tcpdump(1) 程序来分析。

bpf(4) 设备也被用于 dhclient(8) 来获取默认路由器(网关)的 IP 地址。如果使用DHCP，就不要注释掉这行。

```
# 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
```

支持各类 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!)
```

支持各类火线设备。

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 提供的一种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， 并通过 PAE 选项来启用这个能力。
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。

为了让内核支持PAE，只要增加下面这一行到配置文件：

```
options     PAE
```

PAE在FreeBSD里面现在只能支持 Intel® IA-32处理器。
同时，还应该注意，FreeBSD的PAE支持没有经过广泛的测试，和其他稳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：

- 进程不能接触大于4GB的VM空间。
- 没有使用 bus_dma(9) 接口的设备驱动程序在打开了 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 会把所有在打开了 PAE 的内核上不能工作的驱动程序排除在外。
- 一些系统打开了探测系统内存资源使用能力的功能，因为打开了 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选项的总量（看上面）。KVA_PAGES选项 可以用来增加KVA空间。

为了稳定和高性能，建议查看tuning(7)手册页。pae(4)手册页包含 FreeBSD’ sPAE支持的最新信息。

9.7. 如果出现问题怎么办

在定制一个内核时，可能会出现四种问题。它们是：

config失败：
如果 config(8) 在给出您的内核描述时失败， 则可能在某些地方引入了一处小的错误。 幸运的是， config(8) 会显示出它遇到问题的行号， 这样您就能够迅速地定位错误。 例如， 如果您看到：

```
config: line 17: syntax error
```

可以通过与 GENERIC 或其他参考资料对比， 来确定这里的关键词是否拼写正确。

make失败：
如果 make 命令失败， 它通常表示内核描述中发生了 config(8) 无法找出的的错误。 同样地，

仔细检查您的配置， 如果仍然不能解决问题， 发一封邮件到 FreeBSD 一般问题邮件列表
并附上您的内核配置， 则问题应该很快就能解决。

内核无法启动：

如果您的内核无法启动， 或不识别您的设备， 千万别慌！ 非常幸运的是， FreeBSD
有一个很好的机制帮助您从不兼容的内核恢复。 在 FreeBSD
启动加载器那里简单地选择一下要启动的内核就可以了。 当系统在引导菜单的 10 秒倒计时时进入它，
方法是选择 "Escape to a loader prompt" 选项， 其编号为 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。 当重新配置内核时，
保持一个已经证明能够正常启动的内核永远是一个好习惯。

当使用好的内核启动之后您可以检查配置文件并重新尝试编译它。 比较有用的资源是
/var/log/messages 文件， 它会记录每次成功启动所产生的所有内核消息。 此外， dmesg(8)
命令也会显示这次启动时产生的内核消息。

如果在编译内核时遇到麻烦， 请务必保留一个 GENERIC 或已知可用的其他内核，
并命名为别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， 尽快把可用的内核挪到 /boot/kernel 否则类似 ps(1)
这样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

如果您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， 许多用于检查系统状态的工具如 ps(1) 和 vmstat(8) 都将无法正常使用。 您应该
重新编译一个和内核版本一致的系统。
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 可以支持众多种类的打印机，
从最古老的针式打印机到最新的激光打印机以及它们之间所有类型的打印机，
令您运行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，
而且能够对机器或用户提交打印任务的情况进行统计并找到其中用量最多的人，
以及生成用于标识打印任务属于哪位用户的 "标签" 页等等。

在读完这章后，您将知道：

- 怎样配置 FreeBSD 后台打印。
- 怎样安装打印过滤器来对特殊的打印任务做特殊的处理，
  包括把传来的文档转换成打印机能理解的格式。
- 怎样在打印输出上开启报头或者横幅页功能。
- 怎样打印到连接在其他计算机上的打印机。
- 怎样打印到直接连接在网络上的打印机。
- 怎样控制打印机的限制， 包括限制打印任务的大小和阻止某些用户打印。
- 怎样记录打印机统计表和使用情况。
- 怎样解决打印故障。

在读这章之前， 您应该：

- 知道怎样配置并安装新内核 (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。 这章介绍 LPD 后台打印系统， 在接下来将简称为 LPD， 并且将指导您完成其配置。

如果您已经熟悉了 LPD 或者其他后台打印系统， 则可以跳到 设置后台打印系统 这部分。

LPD 完全控制一台计算机上的打印机。 它负责许多的事情：

- 它控制本地和连接在网络上其他计算机上打印机的访问。
- 它允许用户提交要打印的文件; 这些通常被认为是任务。
- 它为每个打印机维护一个 队列 来防止多个用户在同一时刻访问一台打印机。
- 它可以打印报头(也叫做banner或者 burst页使用户可以轻松的从一堆打印输出中找到它们打印的任务。
- 它来设置连接在串口上的打印机的通讯参数。
- 它能通过网络将任务发送到另外一台计算机的 LPD后台打印队列中。
- 它可以根据不同种类的打印机语言和打印机的性能运行特殊的过滤器来格式化任务。
- 它记录打印机的使用情况。

通过配置文件 (/etc/printcap)和提供的特殊过滤程序， 您可以使LPD
系统在众多种类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

如果您是系统唯一的用户， 您可能会奇怪为什么要在您不需要访问控制，
报头页或者打印机使用统计时为后台打印费心。 它可以设置成允许直接访问打印机，
但您还是应该使用后台打印， 因为：

- LPD在后台打印任务； 您不用被迫等待数据被完全副本到打印机的时间。
- LPD可以可以方便的通过过滤器给任务加上日期/ 时间的页眉或者把一种特殊的文件格式 (比如TeX DVI 文件) 转换成一种打印机可以理解的格式。 您不必去手动做这些步骤。
- 许多提供打印功能的免费和商业程序想要和您计算机上的的后台打印系统通讯。
  通过设置后台打印系统， 您将更轻松的支持其他以后要添加的或者现有的软件。

10.3. 基本设置

从 FreeBSD 8.0 起， 串口对应的设备名由 /dev/ttydN 变为 /dev/ttyuN。 FreeBSD 7.X 用户应将这篇文档的示例中的设备名改为原先的样子。

要想在 LPD后台打印系统上使用打印机， 您需要设置打印机硬件和 LPD软件。 这个
文档描述了这两级设置：

- 参见简单打印机 设置来了解怎样连接一个打印机， 告诉 LPD怎样与 它通讯， 并且打印纯文本到 打印机。
- 参见 高级打印机设置 来了解怎样打印多种 特殊格式的文件， 怎样打印报头页， 怎样通过网络 打印， 怎样控制打印机的访问权限， 并且学会为打印 作业记帐统计。

10.3.1. 简单打印机设置

这部分讲解怎样配置打印机硬件和 LPD使之与打印机配合。 讲解的基础知识有：

- 硬件 设置部分将讲解怎样把一台打印机连接到 您计算机的一个端口上。
- 软件 设置部分将讲解怎样配置 LPD后台打印的配置 文件 (/etc/printcap)。

如果您正在设置一台通过网络协议 接收数据来打印而不是通过串口或者并口的打印机， 参见
使用网络数据流界面的打印机。

尽管这部分叫"简单打印机 设置"， 但还是相当复杂的。 使打印机 配合 LPD
后台打印系统在计算机上正常运转是最难的 部分。 一旦您的打印机可以正常工作后，那些高级选项，
比如报文页和记帐， 是相当简单的。

10.3.1.1. 硬件设置

这部分讲述了打印机连接到计算机的多种 途径。 主要讨论了多种接口和 连接线， 还有允许 FreeBSD 与打印机通讯所需的 内核配置。

如果您已经连接好了您的打印机而且已经 用它在另外一个操作系统下成功的打印， 您
或许可以跳到这个部分软件设置。

10.3.1.1.1. 端口和连接电缆

现在所出售的在 PC 上使用的打印机通常至少有 以下三种接口中的一个：

- 串口， 也叫 RS-232 或者 COM 口， 使用您计算机上的串口来发送数据到打印机。
  串口在计算机上已经非常普遍， 而且电缆也非常容易买到且容易制作。 串口有时需要特殊的电缆，
  而且可能需要您去配置稍微有点儿复杂的通讯选项。 大多数 PC 的串口的最高传输速度只有 115200 bps， 这使得打印很大的图像需要的时间很长。
- 并口 使用计算机上的并口来发送数据到打印机。 并口在计算机上也已经非常普遍， 而且速度高于 RS-232 串口。 电缆非常容易买到， 但很难手工制作。 并口通常没有通讯选项， 这使得配置它相当简单。

并口按打印机上的接头来命名也叫做 "Centronics"接口。

- USB 接口， 即通用串行总线， 可以达到比并口和串口高很多的速度。 其电缆既简单又便宜。 USB 用来打印比串口和并口更有优势， 但 UNIX® 系统不能很好的支持它。 避免这个问题的方法就是购买一台像大多数打印机一样的既有 USB 接口又有并口的 打印机。

一般来说并口只提供单向通讯 (计算机到打印机)， 而串口和 USB 则可以提供双向通讯。 新的并口 (EPP 和 ECP) 及打印机在使用了 IEEE-1284 标准的电缆之后， 可以在 FreeBSD 下双向通讯。

与打印机通过并口双向通讯通常由这两种方法中的一种来完成。 第一个方法是使用为 FreeBSD 编写的可以通过打印机使用的语言与打印机通讯的驱动程序。 这通常用在喷墨打印机上， 且可以用来报告剩余墨水多少和其他状态信息。 第二种方法使用在支持 PostScript® 的打印机上。

PostScript® 任务事实上由程序发送给打印机； 但它并不进行打印而是直接将结果返回给计算机。 PostScript® 也采取双向通讯来将打印中的问题报告给计算机， 比如 PostScript® 程序中的错误或者打印机卡纸。 这些信息对于用户来说也许是非常有价值的。 此外， 最好的在支持 PostScript® 的打印机上记帐的方法需要双向通讯： 询问打印机打印总页数 (打印机从出厂一共打印过多少页)， 然后发送用户的任务， 之后再次查询总打印页数。 将打印前后得到的两个值相减就可以得到该用户要付多少纸钱。

10.3.1.1.2. 并口

用并口连接打印机需要用 Centronics 电缆把打印机与计算机连接起来。 具体说明指导在打印机， 计算机的说明书上应该有， 或者干脆两个上面都有。

记住您用的计算机上的哪个并口。 第一个并口在 FreeBSD 上叫 /dev/ppc0； 第二个叫 /dev/ppc1， 依此类推。打印机设备也用同样的方法命名： /dev/lpt0 是接在第一个并口上的打印机， 依此类推。

10.3.1.1.3. 串口

用串口连接打印机需要用 合适的串口电缆把打印机与计算机连接起来。 具体 说明指导应该在打印机， 计算机的说明书上有， 或者 同样干脆两个上面都有。

如果您不确定什么样儿的电缆才是 " 合适的串口 电缆 " ， 您可以尝试以下几种不同的 电缆：

- 调制解调器 电缆每一端的 每一根引脚都直接连接到另一端 相应的引脚 上。 这种电缆也叫做 "DTE-to-DCE" 电缆。
- 非调制解调器电缆上每一端的有些引脚 是与另一端相应引脚直接连接的， 而有一些则是交叉连接的 (比如， 发送数据引脚连接到 接收数据引脚)， 还有一些引脚直接在电缆连接头儿内 短接。 这种电缆也叫做 "DTE-to-DTE" 电缆。
- 一些特殊的打印机需要的串口打印机 电缆， 是一种和非调制解调器电缆类似的电缆， 只是一些信号还是送到了另一端， 而 不是直接在连接头儿内短路。

当然， 您还得为打印机设置通讯参数。 一般是通过打印机面板上的按钮或者 DIP 开关进行设置。 在计算机和打印机上都选择它们所支持的最高 波特 (每秒多少比特， 有时也叫 波特率) 的传输速率。 选择7或者8个数据位； 选择不校验， 偶校验或者奇校验； 选择1个或2个停止位。 还要选择流量控制协议： 无， XON/XOFF (也叫做 "in-band" 或 "软件") 流量控制。 记住您的软件配置中的参数也要设成上面的数值。

10.3.1.2. 软件设置

这部分描述了要使用 FreeBSD 系统中的 LPD 后台打印系统进行打印所需的软件设置。

包括这几个步骤：

1. 在需要的时候配置内核来允许您连接 打印机的端口； 配置内核 部分会告诉您 需要做什么。
2. 如果您使用并口， 则需要设置一下 并口的通讯模式; 设置 并口通讯模式 部分会告诉您具体的细节。
3. 测试操作系统是否能够发送数据到打印机。 检测打印机 联机状况 部分会告诉您要怎样 做。

4. 为 LPD 设置与打印机匹配的参数则 通过修改 /etc/printcap 这个文件来完成。 这章后面的部分将讲解如何来完成设置。

10.3.1.2.1. 配置内核

操作系统的内核为了使某些特殊设备工作需要重新 编译。 打印机所用的串口、 并口就属于那些特殊设备。因此， 可能需要 添加对串口或并口的支持， 如果内核并没有配置它们的话。

要想知道您现在使用的内核是否支持串口， 输入：

```
# grep sioN /var/run/dmesg.boot
```

其中 N 是串口的 编号， 从0开始。 如果您看到 类似下面的输出：

```
sio2 at port 0x3e8-0x3ef irq 5 on isa
sio2: type 16550A
```

则说明您现在使用的内核支持串口。

要想知道您现在使用的内核是否支持并口， 输入：

```
# grep ppcN /var/run/dmesg.boot
```

其中 N 是并口的 编号， 同样从0开始。 如果得到类似 下面的输出：

```
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
```

那么您现在使用的内核支持并口。

您可能必须为了使操作系统支持您打印机需要的串口或 并口而 重新配置内核。

要增加对串口的支持， 参见 内核配置这部分。 要增加对并口的支持， 除了参见 上面提到的那部分之外，还要 参见下面的 部分。

10.3.1.3. 设置并口的通讯模式

在使用并口时， 您可以选择 让 FreeBSD 用中断方式还是轮询方式来 与打印机通讯。 在 FreeBSD 上，通用的打印机驱动 (lpt(4)) 使用 ppbus(4) 系统， 它利用 ppc(4) 驱动来控制端口芯片。

- 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线来检测打印机是否已经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线。 一些新出的 HP 打印机不能正常的工作在中断模式下， 是由于一些定时问题 (还没正确的理解) 造成的。这些打印机需要使用轮询方式。 您应该使用 任何一种方式， 只要它能正常工作就行。一些打印机虽然在两种模式下都可以 工作， 但在中断模式下会慢的要命。

您可以用以下两种方法设定通讯模式： 通过 配置内核或者使用 lptcontrol(8) 这个程序。

要通过配置内核的方法设置 通讯模式：

1. 修改内核配置文件。 找到 一个叫 ppc0 的记录。 如果您想要设置的是 第二个并口， 那么用 ppc1 代替。 使用第三个并口的时候用 ppc2 代替， 依此类推。
   - 如果您希望使用中断驱动模式， 则应编辑下面的配置：

     ```
     hint.ppc.0.irq="N"
     ```

     它在 /boot/device.hints 这个文件中， 其中 N 用正确的中断 编号代替。 同时，
     核心配置文件也必须 包括 ppc(4) 的驱动：

     ```
     device ppc
     ```

   - 如果您想要使用轮询方式， 只需要把 /boot/device.hints 这个文件中的下面这行删除掉：

     ```
     hint.ppc.0.irq="N"
     ```

     在 FreeBSD 下， 有时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) 驱动造成的， 它可以自动侦测到设备并将其挂载到系统上， 但也因此，
     它控制着打印机端口的访问模式. 您需要检查 acpi(4) 的配置来解决这个问题。
2. 保存文件。 然后配置， 建立， 并安装刚配置的内核， 最后重新启动。 参见 内核配置 这章来获得更多细节。

使用 lptcontrol(8) 设置通讯模式：

1. 输入：

   ```
   # lptcontrol -i -d /dev/lptN
   ```

   将 lptN 设置成中断方式。
2. 输入：

   ```
   # lptcontrol -p -d /dev/lptN
   ```

   将 lptN 设置成轮询方式。

您可以把这些命令加入到 /etc/rc.local 这个文件中， 这样每次启动系统 时都会设置成您想要的方式。 参见 lptcontrol(8) 来获得 更多信息。

10.3.1.4. 检测打印机的通讯

在设置后台打印系统之前， 您应该确保您的计算机可以把数据 发送到打印机上。
分别独立调试打印机的通讯和后台打印系统会更简单。

我们为了测试打印机，将发送一些文本给它。 一个叫 lptest(1) 的程序能胜任这项工作，
它可以让打印机立即打印出程序发给它的 字符： 它在每行打出 可以打印的 96 个 ASCII 字符。

当我们使用的是一台 PostScript® (或者以其他语言为基础的) 打印机， 那么 需要更仔细的检测。
一段小小的 PostScript® 程序足以完成检测的任务， 比如下面这段程序：

```
%!PS
100 100 moveto 300 300 lineto stroke
310 310 moveto /Helvetica findfont 12 scalefont setfont
(Is this thing working?) show
showpage
```

可以把上面这段 PostScript® 代码写进一个文件里， 并且像下面部分的例子里那样 使用。

上面的小程序是针对 PostScript® 而不是惠普的 PCL 写的。 由于 PCL
拥有许多其他打印机没有的强大功能， 比如它支持在打印纯文本的同时夹带特殊的命令，
而 PostScript® 则不能直接打印纯文本， 所以需要对这类打印机语言进行特殊的处理。

10.3.1.4.1. 检测并口打印机

这部分内容将指导您怎样检测 FreeBSD 是否可以与一台已经连接在并口上的打印机通讯。

要测试并口上的打印机：

1. 用 su(1) 命令转换到 root 用户。
2. 发送数据到打印机。
   - 如果打印机可以直接打印纯文本， 可以用 lptest(1)。 输入：

     ```
     # lptest > /dev/lptN
     ```

     其中 N 是并口的编号， 从0开始。
   - 如果打印机支持 PostScript® 或其他打印机语言， 可以发送一段小程序到打印机。 输入：

     ```
     # cat > /dev/lptN
     ```

     然后， 一行一行地 输入 输入这段程序。 因为在按下 换行 或者 回车 之后，
     这一行就不能再修改了。 当您输入完这段程序之后， 按 CONTROL+D，
     或者其他表示文件结束的键。

     另外一种办法， 您可以把这段程序写在一个文件里， 并输入：

     ```
     # cat file > /dev/lptN
     ```

     其中 file 是包含这您要发给打印机程序的文件名。

之后， 您应该看到打印出了一些东西。 如果打印出的东西看起来并不正确， 请不要着急；
我们将在后面指导您如何解决这类问题。

10.3.1.4.2. 检测串口打印机

这部分将告诉您如何检测 FreeBSD 是否可以与连接在串口上的打印机通讯。

要测试连接在串口上的打印机：

1. 通过 su(1) 命令转为 root 用户。
2. 修改 /etc/remote 这个文件。 增加下面这些内容：

```
printer:dv=/dev/port:br#bps-rate:pa=parity
```

其中 port 是串口的设备节点 (ttyu0、 ttyu1， 等等)， bps-rate 是与打印机通讯时使用的速率，而 parity 是通讯时打印机要求的校验方法 (应该是 even、 odd、 none， 或 zero 之一)。

这儿有一个串口打印机的例子， 它连接在第三个串口上， 速度为 19200 波特， 不进行校验：

```
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， 则要再次修改 /etc/remote 这个文件， 并且试试用 /dev/cuaaN 代替 /dev/ttydN。

4. 发送数据到打印机。
   - 如果打印机可以直接打印纯文本， 则用 lptest(1)。 输入：

     ```
     % $lptest
     ```

   - 如果打印机支持 PostScript® 或者其他 打印机语言， 则发送一段小程序到 打印机。一行一行的输入程序， 必须 非常仔细 因为像退格 或者其他编辑键也许对打印机来说有它的意义。 您同样也需要按一个特殊的 文件结束键， 让打印机知道它已经 接收了整个程序。 对于 PostScript® 打印机， 按 CONTROL+D。

     或者， 您同样也可以把程序存储在一个文件里 并输入：

     ```
     % >file
     ```

     其中 file 是 包含要发送程序的文件名。 在 tip(1) 发送这个文件之后， 按代表 文件结束的键。

您应该看到打印出了一些东西。 如果它们看起来并不正确也不要着急；
我们将在稍后的章节中介绍如何解决这类问题。

10.3.1.5. 启用后台打印： 文件 /etc/printcap

目前， 您的打印机应该已经连好了线， 系统内核也为与打印机联机而重新配置好 (如果需要的话)，而且您也已经可以发送一些简单的数据到打印机。 现在， 我们要配置 LPD 来使其控制您的打印机。

配置 LPD 要修改 /etc/printcap 这个文件。 由于 LPD 后台打印系统在每次使用后台打印的时候，都会读取这个文件， 因此对这个文件的修改会立即生效。

printcap(5) 这个文件的格式很简单。 您可以用您最喜欢的文本编辑器来修改 /etc/printcap 这个文件。这种格式和其他的像 /usr/shared/misc/termcap 和 /etc/remote 这类文件是一样的。

要得到关于这种格式的详尽信息， 请参阅联机手册 cgetent(3)。

简单的后台打印配置包括下面的几步：

1. 给打印机起一个名字 (记忆和使用的别名)， 然后把它们写进文件 /etc/printcap； 参见 如何为打印机命名 这章来得到更多的关于起名的帮助。
2. 通过增加 sh 项关掉报头页 (它默认是启用的)； 参见 如何禁用报头页 部分来得到更多信息。
3. 建立一个后台打印队列的目录， 并且通过 sd 项目指定它的位置； 您可参见 创建后台打印队列目录 一节了解更多信息。
4. 在 /dev 下设置打印机设备节点， 并且在写在 /etc/printcap 文件中 lp 项目里； 参见 识别打印机设备 这部分可以得到更多信息。 此外， 如果打印机连接在串口上， 通讯参数的设置需要写在 ms# 项中。 这些参数在 配置后台打印通讯参数 这在前面已经讨论过。
5. 安装纯文本过滤器； 详情请参见 安装文本过滤器 小节。
6. 用 lpr(1) 命令来测试设置。 想得到更多信息可以参见 测试 和 故障排除 部分。

使用打印机语言的打印机， 如 PostScript® 打印机， 通常是不能直接打印纯文本的。 前面提到， 并且将在后面继续进行介绍的简单的设置方法， 均假定您正在安装这种只能打印它能识别的文件格式的打印机。

用户通常会希望直接在系统提供的打印机上打印纯文本。 采用 LPD 接口的程序也通常是这样设计的。 如果您正在安装这样一台打印机， 并且希望它不仅能打印使用它支持的打印机语言的任务 而且 还能打印纯文本的任务的话， 那么强烈建议您在上面提到的简单设置的步骤上增加一步： 安装从自动纯文本到 PostScript® (或者其他打印机语言) 的转换程序。 更多的细节， 请参见 在 PostScript® 打印机上打印纯文本。

10.3.1.5.1. 打印机的命名

第一步 (简单) 就是给打印机起一个名字。 您是按功能起名字还是干脆起个古怪的名字都没有关系， 因为您可以给打印机设置许多的别名。

在 /etc/printcap 里至少有一个打印机必须指定， 别名是 lp。 这是默认的打印机名。 如果用户既没有 PRINTER 环境变量， 也没有在任何 LPD 命令的命令行中指定打印机名， 则 lp 将是默认要使用的打印机。

还有， 我们通常把最后一个别名设置成能完全描述打印机的名字， 包括厂家和型号。

一旦您选好了名字或者一些别名， 把它们放进文件 /etc/printcap 里。 打印机的名字应该从最左边的一列写起。 用竖杠来隔开每个别名， 并且在最后一个别名后面加上一个冒号。

在下面的例子中， 我们从一个基本的 /etc/printcap 开始， 它只定义了两台打印机 (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为 rattan 并且设置了 line， diablo, lp， 和 Diablo 630 Line Printer 这几个别名。 因为它被设置了 lp 这个别名， 所以它是默认打印机。 第二台 被命名为 bamboo， 并且设置了 ps， PS, S， panasonic， 和 Panasonic KX-P4455 PostScript v51.4 这几个别名。

10.3.1.5.2. 不打印报头页

LPD 后台打印系统默认 会为每个任务打印 报头页。 报头页 包含了发送这个任务的用户，
发送这个任务的计算机， 任务的名字， 并用大字母打出。 但不幸的是， 所有这些额外的文本，
都会给在对打印机进行最初的配置时排除故障带来困难， 所以我们将先不打印报头页。

要暂停打印报头页， 为打印机的记录增加 sh 标记， 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:
```

注意我们的正确格式: 第一行从最左边一列开始， 而后的每一行用 TAB 缩进一次。 一行写不下需要换行时，
在换行前打一个反斜杠。

10.3.1.5.3. 建立后台打印队列目录

下一步设置就是要建立一个 后台打印队列目录， 也就是在打印任务最终完成之前用于存放这些任务的目录，
这个目录中也会存放后台打印系统用到的其他一些文件。

由于后台打印队列目录的变量本质， 通常 把这些目录安排在 /var/spool 下。 您也没有必要去
备份后台打印队列目录里的内容。 重新建立它们只要简单的使用 mkdir(1) 命令。

通常， 我们习惯将目录名起成和 打印机一样的名字， 像下面 这样：

```
# mkdir /var/spool/printer-name
```

然而， 如果您有很多网络打印机， 您可能想要把这些后台打印的队列目录目录放在一个单独的专为使用
LPD 打印而准备的目录里。 我们将用我们的两台打印机 rattan 和 bamboo 作为例子：

```
# mkdir /var/spool/lpd
# mkdir /var/spool/lpd/rattan
# mkdir /var/spool/lpd/bamboo
```

如果担心用户任务的保密性， 可能会希望保护相应的后台打印队列目录，
使之不能被其他用户访问。 后台打印的队列目录的属主应该是 daemon 用户， 而
daemon 用户和 daemon 组拥有读写和搜索的权限，但其他用户没有。
接下来用我们的两台打印机作为例子:

```
# 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， 您需要通过/etc/printcap 文件告诉 LPD 这些目录。 您可以用 sd
标记来指定后台打印队列目录的路径:

```
#
#  /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， 其他记录每行都要用 TAB 键缩进一次，
写不开需要换行在最后加上反斜杠。

如果您没用 sd 标记指定后台打印队列目录， 后台打印系统会将 /var/spool/lpd 目录作为默认目录。

10.3.1.5.4. 识别打印机设备

在 Hardware Setup 一节中，我们说明了 FreeBSD 与打印机通讯将使用哪个端口和 /dev 目录下的节点。
我们要告诉 LPD 这些信息。 当后台打印系统有任务需要打印，它将为过滤程序 （负责传送数据到打印机）
打开指定的设备。

用 lp 标记在 /etc/printcap 里列出 /dev 下的设备节点。

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 要对
/etc/printcap 文件里增加的内容 :

```
#
#  /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:
```

如果您没在您的 /etc/printcap 文件中 用 lp 标记指定设备节点， LPD 将默认使用 /dev/lp 。 /dev/lp
目前在 FreeBSD 中不存在。

如果您正在安装的打印机是连接在 并口上的， 请跳到 安装文本 过滤器 这章。 如果不是的话，
还是最好按下面介绍的 步骤做。

10.3.1.5.5. 配置后台打印通讯参数

对于连在串口上的打印机， LPD 可以为发送数据到打印机的过滤程序设置好波特率， 校验，
和其他串口通讯参数 。 这是有利的， 因为:

- 它可以让您只需简单的修改 /etc/printcap 就能尝试不同的通讯 参数; 您并不需要去重新编译过滤器

程序。

- 它使得后台打印系统可以在 多台有不同串口通讯设置的打印机上使用 相同的过滤器程序。

下面这个 /etc/printcap 中 用 lp 标记来控制列出设备的 串口通讯参数 :

br#bps-rate
: 设置设备的通讯速度为 bps-rate， 这里 bps-rate 可以为 50， 75， 110, 134， 150， 200， 300， 600， 1200， 1800， 2400， 4800， 9600, 19200， 38400， 57600， or 115200 比特每秒。

ms#stty-mode
: 设置已打开的中端设备的选项 。 stty(1) 将详细 讲述可用的选项。

当 LPD 打开 用 lp 指定的设备时， 它会 将设备的特性设置成在 ms# 标记后指定的那样。 特别是 parenb, parodd， cs5, cs6， cs7, cs8， cstopb, crtscts， 和 ixon 这些模式， 它们在 stty(1) 手册中有详细说明。

我们举个例子来添加我们连在第6个串口上的 打印机。 我们将设波特为38400。 至于模式， 我们将用 -parenb 设置成不校验， 用 cs8 设置成8位字符， 用 clocal 设置成不要调制解调器控制， 用 crtscts 设置成硬件流量控制:

```
bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
    :sh:sd=/var/spool/lpd/bamboo:\
    :lp=/dev/ttyu5:ms#-parenb cs8 clocal crtscts:
```

10.3.1.5.6. 安装文本过滤器

我们现在准备告诉 LPD 使用什么文本过滤器 给打印机发送任务。 文本过滤器， 也叫 输入过滤器， 是一个在 LPD 有一个任务要发给 打印机时运行的程序。 当 LPD 为打印机运行文本过滤器时， 它设置过滤器的标准输入为要发给打印机的任务， 而标准输出为 用 lp 标记指定的打印机 。 过滤器先从标准输入读取 任务， 为打印机进行一些转换 ， 并将结果写到标准输出， 这些结果 将被打印。
想得到更多关于文本过滤器的信息， 见 过滤器 这节。

对于简单的打印机设置， 文本过滤器可以仅仅是一段 执行 /bin/cat 的 shell 脚本来 发送任务到打印机。 FreeBSD 还提供了一个叫做 lpf 的过滤器， 它可以处理退格和下划线来 使那些可能不能很好处理这类字符流的打印机正常工作。 而且， 当然， 您可以用任何其他的 您想用的过滤程序。 lpf 过滤器在 lpf: 一个文本 过滤器 这节将有详细描述。

首先， 我们来写一段叫做 /usr/local/libexec/if-simple 的简单 shell 脚本作为文本过滤器。 用您熟悉的文本编辑器将下面的内容放进 这个文件:

```
#!/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
```

使这个文件可以被执行:

```
# chmod 555 /usr/local/libexec/if-simple
```

然后用 if 标记在 /etc/printcap 里告诉 LPD 使用这个脚本。 我们将仍然为 一直作为例子的这两台打印机在 /etc/printcap 里增加这个标记：

```
#
#  /etc/printcap for host rose - added text filter
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
	:sh:sd=/var/spool/lpd/rattan:\ :lp=/dev/lpt0:\
	:if=/usr/local/libexec/if-simple:

bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
	:sh:sd=/var/spool/lpd/bamboo:\
	:lp=/dev/ttyu5:ms#-parenb cs8 clocal crtscts:\
	:if=/usr/local/libexec/if-simple:
```

if-simple 脚本的副本可以在 /usr/shared/examples/printing 目录中找到。

10.3.1.5.7. 开启 LPD

lpd(8) 在 /etc/rc 中被运行， 它是否被运行由 lpd_enable 这个变量控制。 这个 变量默认是 NO。 如果您还没有修改 ， 那么增加这行：

```
lpd_enable="YES"
```

到 /etc/rc.conf 文件当中， 然后既可以重启您的 机器， 也可以直接运行 lpd(8)。

```
# lpd
```

10.3.1.5.8. 测试

现在已经基本完成了 LPD 的基本设置。 但不幸的是， 还不是庆祝的时候， 因为我们还需要测试设置并且修正所有的 问题。 要测试设置， 尝试打印一些东西。 要 用 LPD 系统打印， 您可以 使用 lpr(1) 命令， 它可以提交一个任务来打印。

您可以联合使用 lpr(1) 和 lptest(1) 程序， 在 检查打印机 通讯 这节介绍怎样生成一些测试文本。

要测试简单 LPD 设置：

输入：

```
# lptest 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) 。 要测试默认 打印机， 输入 lpr(1) 不带任何 -P 选项。 同样， 如果您正在测试一台使用 PostScript® 的打印机， 发送一个 PostScript® 程序到打印机而不是 使用 lptest(1)。 您可以把程序放在一个 文件里， 然后输入： lpr file。

对于一台 PostScript® 打印机， 您应该得到那段程序的 结果。 而如果您使用的 lptest(1)， 则您得到的结果应该看起来像下面这样：

```
!"#$%&'()*+,-./01234
"#$%&'()*+,-./012345
#$%&'()*+,-./0123456
$%&'()*+,-./01234567
%&'()*+,-./012345678
```

要更进一步的测试打印机， 尝试下载一些大的 程序 (为基于特定语言的打印机) 或者运行 lptest(1) 并使用不同的参数。 比如， lptest 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 这节。

10.4. 高级设置

从 FreeBSD 8.0 起， 串口对应的设备名由 /dev/ttydN 变为 /dev/ttyuN。 FreeBSD 7.X 用户应将这篇文档的示例中的设备名改为原先的样子。

这部分将描述用来打印特别格式文件， 页眉， 通过网络打印， 以及对打印机使用限制和 记帐。

10.4.1. 过滤器

尽管 LPD 处理网络协议， 任务排队， 访问控制， 和打印的其他方面， 但大部分 实际 工作还是由 过滤器。 过滤器是 一种与打印机通讯并且处理设备依赖和特殊需要的 程序。 在简单打印机设置这节里， 我们安装了一个纯文本过滤器 - 一个应该可以用在大多数打印机上的极简单的过滤器 (安装文本过滤器)。

然而，为了进行格式转换， 打印记帐， 适应特殊的打印机， 等等， 您需要明白过滤器是怎样工作的。 在根本上过滤器负责处理这些方面。 但坏消息是大多数时候 您 必须自己提供过滤器。 好消息是很多过滤器通常都已经有了; 当没有的时候， 它们通常也是很好写的。

FreeBSD 也提供了一个过滤器， /usr/libexec/lpr/lpf， 可以让大多数可以打印纯文本的打印机工作。 (它处理文件里的退格和跳格，并且进行记帐， 但这基本就是它所有能做的了。) 这里还有几个过滤器和过滤器组件在 FreeBSD Ports Collection 里。

这是在这节里您将找到的内容：

- 在 过滤器是如何工作的 小节中将介绍在打印过程中过滤器的作用。 如果希望了解在 LPD 使用过滤器时， 在 "幕后" 发生的事情， 便应阅读这一小节。 了解这些知识能够帮助您在为打印机安装过滤器时更快地排查可能会遇到的各种问题。
- LPD 假定任何打印机在默认状态下均能打印纯文本内容。 对于不能直接打印纯文本的 PostScript® 打印机 (以及其他基于打印语言的打印机) 而言这会带来问题。 在 在 PostScript® 打印机上使用纯文本任务 这节中将会介绍如何解决这个问题的方法。 如果您使用 PostScript® 打印机， 就应阅读这节内容。
- PostScript® 对于许多程序来说都是一个非常受欢迎的输出格式。 一些人甚至直接写 PostScript® 代码。 但不幸的是， PostScript® 打印机非常昂贵。 模拟 PostScript® 在 非 PostScript® 打印机上 这节将告诉您怎样进一步修改 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 您没有 PostScript® 打印机， 那么您应该阅读这个小节。
- 转换过滤器 这节讲述了一个自动把指定格式文件， 比如图像或排版数据， 转换成您打印机可以理解的格式的方法。 在阅读了这节之后， 您就应该可以配置打印机， 让用户可以用 lpr -t 来打印 troff 数据、 用 lpr -d 来打印 TeX DVI 数据， 或用 lpr -v 来打印光栅图像数据等工作了。 建议您阅读这节。
- 输出 过滤器 这节讲述了这个不是经常使用的 LPD: 的功能－输出过滤器。 除非您要打印页眉 (见 页眉 这节)， 您或许可以完全跳过这节。

- lpf: 一个文本过滤器 描述了 lpf， 一个 FreeBSD 自带的相当完整而又简单的文本过滤器，
  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。
  如果您需要一个快速的方法来让打印机统计打印纯文本的工作量，
  或者您有一台遇到退格字符就冒烟的打印机， 您应该考虑 lpf。

您可以在 /usr/shared/examples/printing 目录中找到下面将提到的那些脚本的副本。

10.4.1.1. 过滤器是怎样工作的

前面说过， 过滤器是一个被 LPD 启动， 用来处理与打印机通讯过程中设备依赖的部分 的可执行程序。

当 LPD 想要打印 一个任务中的文件， 它启动一个过滤器 程序。
它把要打印的文件设置成过滤器的标准输入， 标准输出设置成打印机， 并且把错误信息定向到
错误日志文件 (在 lf 标识里指定， 默认在 /etc/printcap， 或者 /dev/console 文件里)。

过滤器被 LPD 启动， 并且过滤器的参数依赖于 /etc/printcap 文件中所列出的和用户为任务用 lpr(1)
命令所指定的。 例如， 如果用户输入 lpr -t， LPD 会启动 troff 过滤器， 即在目标打印机的 tf
标签里所列出的过滤器。 如果用户想要打印纯文本， 它将会启动 if 过滤器 (这是通常的情况： 参见
输出过滤器 来得到 细节)。

在 /etc/printcap 文件中， 您可以指定三种过滤器：

- The 文本过滤器， 在 LPD 文档中也叫做 输入过滤器 ， 处理 常规的文本打印。
  可以把它想象成默认过滤器。 LPD 假定每台打印机默认情况下都可以打印纯文本，
  而文本过滤器的任务就是来搞定退格、 跳格， 或者其他在某种打印机上容易错误的特殊字符。
  如果您所在的环境对打印机的使用情况进行记帐， 那么文本过滤器必须也对打印的页数进行统计，
  通常是根据打印的行数和打印机在每页上能打印的行数进行计算得出。 文本过滤器的启动命令为：

  filter-name [-c] -w width -l length -i indent -n login -h host acct-file

  这里

  -c
  : 当任务用 lpr -l 这个命令提交时出现

  width
  : 这里取您在 /etc/printcap 文件中指定的 pw (页 宽) 标签的值， 默认为 132。

  length
  : 这里取您的 pl (页 长) 标签的值， 默认为 66

  indent
  : 这里是来自 lpr -i 命令的总缩进量， 默认为 0

  login
  : 这里是正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 变量中指定的用于记帐的文件名。

- 转换过滤器 的功能是， 将特定格式的文件转换成打印机能够识别并打印的格式。 例如， ditroff
  格式的排版数据就是无法直接打印的， 但您可以安装一个转换过滤器来将 ditroff
  文件转换成一种打印机可以识别和打印的形式。 请参见 转换过滤器 这一节来了解更多细节。
  如果您需要对打印进行记帐， 那么转换过滤器也必须完成记帐工作。 转换过虑器的启动命令为：

  filter-name -x pixel-width -y pixel-height -n login -h host acct-file

这其中 pixel-width 的值来自 px 标签 (默认为 0)， 而 pixel-height 的值来自 py 标签 (默认为 0)。

- 输出过滤器 仅在没有文本过滤器时， 或者报头页被打开时使用。 就我们的经验而言，
  输出过滤器是很少用到的. 在 输出过滤器 这节中会介绍它们。 启动输出过滤器的命令行只有两个参数：
  filter-name -w width -l length

  它们的作用与文本过滤器的 -w 和 -l 参数是一样的。

过滤器也应该在 退出 时给出下面的几种退出状态:

exit 0
: 过滤器已经成功的打印了文件.

exit 1
: 过滤器打印失败了， 但希望 LPD 试着再打印一次。 如果过滤器返回了这个状态， LPD
将重新启动过滤器。

exit 2
: 过滤器打印失败并且不希望 LPD 重试。 这种情况下 LPD 会放弃这个文件。

文本过滤器随 FreeBSD 一起发布， 文件名为 /usr/libexec/lpr/lpf，
它利用页宽和页长参数来决定何时发送送纸指令， 并提供位打印记帐的方法。 它使用登录名、 主机名，
和记帐文件参数来生成记帐记录。

如果您想购买过滤器， 要注意它是否是与 LPD 兼容。 如果兼容的话，
则它们必须支持前面提到的那些参数。 如果您打算编写普通的过滤器程序，
则同样需要使之支持前面那些参数和退出状态码。

10.4.1.2. 在 PostScript® 打印机上打印纯文本任务

如果您是您的计算机和 PostScript® (或其他语言的) 打印机的唯一用户，
而且您不打算发送纯文本到打印机， 并因此不打算从应用程序程序直接将纯文本发到打印机的话，
就完全不需要再关心这节的内容了。

但是， 如果打印机同时需要接收 PostScript® 和纯文本的任务， 就需要对打印机进行设置了。
要完成这项工作， 我们需要一个文本过滤器来检测到达的任务是纯文本的还是 PostScript® 格式的。 所有
PostScript® 的任务必须以 %! (其他打印机语言请参见打印机的文档) 开头。
如果任务的头两个字符是这两个， 就代表这是 PostScript® 格式的， 并且可以直接略过任务剩余的部分。
如果任务开头的两个字符不是这两个， 那么过滤器将把文本转换成 PostScript® 并打印结果。

我们怎样去做?

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。 它用打印机传来的详细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的看到打印机处在什么样的状态 (比如 缺墨 或者 卡纸)。 但更重要的是，
它包含了一个叫做 psif 的程序， 它可以检测接收到的文件是否是纯文本的， 并且将使用 textps 命令 (
也是由 lprps 提供的程序) 转换文本到 PostScript®。 然后它会用 lprps 将任务发送到打印机。

lprps 可以在 FreeBSD Ports Collection (详见 The Ports Collection) 中找到。
你可以根据页面的尺寸选择安装 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， 这也是包含在 lprps 中的一个程序。 如果您已经用 ports 安装好了 lprps，
将下面的内容添加到 /etc/printcap 文件中 PostScript® 打印机的记录部分中：

```
:if=/usr/local/libexec/psif:
```

同时还需要指定 rw 标签来告诉 LPD 使用读-写模式打开打印机。

如果您有一台并口的 PostScript® 打印机 (因此不能与打印机进行 lprps 需要的双向通信)，
可以使用下面这段 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纯文本转换成 PostScript®。
您可以使用任何您喜欢的文本到 PostScript® 转换程序。 FreeBSD Ports Collection (详见 Ports Collection) 中包含了一个功能非常完整的文本到 PostScript® 的转换程序， 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 可以读取大多数 PostScript® 的文件并处理其中的页面交给多种设备，包括许多品牌的非 PostScript® 打印机。 通过安装 Ghostscript 并使用一个特殊的文本过滤器，则可以使一台非 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，有许多可用的版本， 比较常用的版本是 print/ghostscript-gpl。

要模拟 PostScript®， 文本过滤器要检测是否要打印一个 PostScript® 文件。 如果不是， 那么过滤器将直接将文件发送到打印机； 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: 下面的脚本是一个针对 Hewlett Packard DeskJet 500 打印机的文本过滤器。
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 来获得当前安装的 Ghostscript 所支持的设备列表。)

```
#!/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 标签完成:

```
:if=/usr/local/libexec/ifhp:
```

您可以输入 lpr plain.text 和 lpr whatever.ps， 它们都应该可以成功打印。

10.4.1.4. 转换过滤器

在完成了 打印机简单设置 这节中所描述的内容之后， 头一件事
恐怕就是为你喜爱的格式的文件安装转换过滤器了 (除了纯 ASCII 文本)。

10.4.1.4.1. 为什么安装转换过滤器?

转换过滤器使打印众多格式的文件变得很容易。 比如， 假设我们大量使用 TeX 排版系统， 并且有一台 PostScript® 打印机。 每次从 TeX 生成一个 DVI 文件， 我们都不能直接打印它直到我们将 DVI 文件转换成 PostScript®。 转换的命令应该是下面的样子:

```
% dvips seaweed-analysis.dvi
% lpr seaweed-analysis.ps
```

通过安装 DVI 文件的转换过滤器， 我们可以跳过每次手动转换这一步， 而让 LPD 来完成这个步骤。 现在，每次要打印 DVI 文件， 我们只需要一步就可以打印它:

```
% lpr -d seaweed-analysis.dvi
```

我们要 LPD 转换 DVI 文件是通过指定 -d 选项完成的。 格式和转换 选项 这一节列出了所有的转换选项。

对于每种想要打印机支持的转换， 首先要安装 f转换过滤器 然后在 /etc/printcap 中指定它的路径。
在简单打印设置中， 转换过滤器类似于文本过滤器 (详见 安装文本过滤器)
不同的是它不是用来打印纯文本， 而是将一个文件转换成打印机能够理解的格式。

10.4.1.4.2. 我应该安装哪个转换过滤器?

您应该安装您希望使用的转换过滤器。 如果要打印很多 DVI 数据， 就需要 DVI 转换过滤器； 如果有大量的 troff 数据， 就应该安装 troff 过滤器。

下面的表格总结了可以与 LPD配合 工作的过滤器， 以及它们在 /etc/printcap文件中的变量名， 还有如何在 lpr命令中调用它们:

文件类型	在/etc/printcap文件中的变量名	在lpr命令中调用使用的参数
cifplot	cf	-c
DVI	df	-d
plot	gf	-g
ditroff	nf	-n
FORTRAN text	rf	-f
troff	tf	-f
raster	vf	-v
plain text	if	none, -p, or -l

在例子中， lpr -d就是指 打印机需要在/etc/printcap文件中 df变量所指的过滤器。

不管别人怎么说， 像 FORTRAN 的文本 和 plot 这些格式已经基本不用了。 所以在您的机器上，
就可以安装其他的过滤器来替换这些参数原有的意义。 例如， 假设想要能直接打印 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 转换过滤器并且用 gf 变量指定它。 然后就可以告诉您的用户使用 lpr -g 就可以 "打印 Printerleaf 文件。"

10.4.1.4.3. 安装转换过滤器

以为安装的转换过滤器不是 FreeBSD 基本系统的一部分， 所以它们可能是在 /usr/local 目录下。 通常目录 /usr/local/libexec 是保存它们的地方， 因为它们通常是通过 LPD 运行的;
普通用户应该并不需要直接运行它们。

要启用一个转换过滤器， 只需要在 /etc/printcap
文件中为目标打印机中合适的变量赋上过滤器所在的路径。

在接下来的例子当中， 我们将为 一台叫做 bamboo 的打印机添加一个转换过滤器。 下面是这个例子的 /etc/printcap 文件， 其中使用新变量 df 来为打印机 bamboo 设置转换过滤器：

```
#
#  /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 crtscts:rw:\
    :if=/usr/local/libexec/psif:\
    :df=/usr/local/libexec/psdf:
```

这里的 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) 并从标准输入读取要打印的任务。 然后运行 PostScript® 文本过滤器 lprps (详见 在 PostScript® 打印机上打印纯文本任务 这一节)， 并且带着 LPD 传给脚本的全部参数。 lprps 工具将利用这些参数来为打印进行记帐。

10.4.1.4.4. 更多转换过滤器应用实例

因为安装转换过滤器的步骤并不是固定的， 所以这节介绍了一些可行的例子。
在以后的安装配置过程中可以以这些例子为参考。 甚至如果合适的话， 可以完全照搬过去。

这段例子中的脚本是一个 Hewlett Packard LaserJet III-Si 打印机的光栅格式数据 (实际上也就是 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 | ppmtopgm | 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。

下面是为打印机配置上上述过滤器的 /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 的这台 PostScript® 打印机上打印用 groff 排版软件生成的 troff 数据的打印过滤器：

```
#!/bin/sh
#
#  pstf - Convert groff's troff data into PS, then print.
#  Installed in /usr/local/libexec/pstf
#
exec grops | /usr/local/libexec/lprps "$@"
```

上面这段脚本还是用 lprps 来与打印机进行通讯。 如果打印机是接在并口上的，那么就应该使用下面的这段脚本：

```
#!/bin/sh
#
#  pstf - Convert groff's troff data into PS, then print.
#  Installed in /usr/local/libexec/pstf
#
exec grops
```

这里是我们要启用过滤器需要在 /etc/printcap 里增加的内容：

```
:tf=/usr/local/libexec/pstf:
```

下面的例子也许会让许多 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纯文本的打印机上使用。 我们将为打印机 teak 安装这个过滤器：

```
#!/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
```

然后我们要在 /etc/printcap 中为打印机能够 teak 启用这个过滤器添加下面的内容：

```
:rf=/usr/local/libexec/hprf:
```

最后， 再给出一个有些复杂的例子。 我们将给以前介绍过的 teak 这台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:
```

现在， 该困难的部分了： 编写过滤器。 为了实现过滤器， 我们需要一个 DVI-到-LaserJet/PCL 转换程序。
FreeBSD Ports Collection (详见 Ports Collection 这一节) 中有一个： print/dvi2xx。 安装这个 port
就会得到我们需要的程序， 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 变得十分复杂， 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 作为标准输入是有问题的。
我们可以通过连接 (符号连接) 来解决这个问题。 连接一个临时的文件名 (一个以 .dvi 结尾的文件名) 到
/dev/fd/0， 从而强制 dvilj2p 从标准输入读取。

现在迎面而来的是另外一个问题， 我们不能使用 /tmp 存放临时连接。 符号连接是被用户和组 bin 拥有的。
而过滤器则是以 daemon 用户运行的。 并且 /tmp 目录设置了 sticky 位。 所以过滤器只能建立符号连接，
但它不能在用完之后清除掉这些连接。 因为它们属于不同的用户。

所以过滤器将在当前工作目录下建立符号连接， 即后台打印队列目录 (用变量 sd 在 /etc/printcap 中指定)。
这是一个非常好的让过滤器完成它工作的地方， 特别还是因为 (有时) 这个目录比起 /tmp
来有更多的可用磁盘空间。

最后， 给出过滤器的代码：

```
#!/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 dvilj2p does not seem to be
#  reliable, so we ignore it.
```

```
#
dvilj2p -M1 -q -e- dfhp$$.dvi


#
#  Clean up and exit
#
cleanup
exit 0
```

10.4.1.4.5. 自动转换： 一种替代转换过滤器的方法

以上这些转换过滤器基本上建成了您的打印环境， 但也有不足就是必须由用户来指定 (在 lpr(1) 命令行中) 要使用哪一个过滤器。 如果您的用户不是对计算机很在行， 那么选用过滤器将是一件麻烦的事情。 更糟的是， 当过滤器设定的不正确时， 过滤器被用在了不它对应类型的文件上， 打印机也许会喷出上百张纸。

比只安装转换过滤器更好的方法， 就是让文本过滤器 (因为它是默认的过滤器) 来检测要打印文件的类型， 然后自动运行正确的转换过滤器。 像 file 这样的工具可以给我们一定的帮助。 当然， 要区分开 有些 文件的类型还是有困难的 - 但是， 当然， 您可以仅为它们提供转换过滤器。

FreeBSD 的 Ports 套件提供了一个可以自动进行转换的文本过滤器， 名字叫做 apsfilter (print/apsfilter)。 它可以检测纯文本、 PostScript®、 DVI 以及几乎任何格式的文件， 并在执行相应的转换之后完成打印工作。

10.4.1.5. 输出过滤器

LPD 后台打印系统还支持一种我们还没有讨论过的过滤器： 输出过滤器。 输出过滤器只是用来打印纯文本的， 类似于文本过滤器， 但简化了许多地方。 如果您正在使用输出过滤器而不是文本过滤器， 那么：

- LPD 为整个任务启动一个输出过滤器， 而不是为任务中的每个文件都启动一次。
- LPD 不会提供任务中文件开始和结束的信息给输出过滤器。
- LPD 不会提供用户名或者主机名给过滤器， 所以它是无法做打印记帐的。 事实上它只有两个参数：

  过滤器-名字 -w宽度 -l长度

  宽度 来自于 pw 变量， 而 length 来自于 pl 变量， 这些值都是实际问题中给打印机设置的。

不要让输出过滤器的简化所耽误。 如果想要输出过滤器完成让任务中的每个文件都重新开始一页打印是 不可能 的。 请使用文本过滤器 (也叫输入过滤器)； 详见 安装文本过滤器。 此外， 实际上， 输出过滤器 更复杂 ， 它要检查发给它的字节流中是否有特殊的标志字符， 并且给自己发送信号来代替 LPD 的。

可是， 如果打算要报头页或者需要发送控制字符或者其他的初始化字符串来完成打印报头页， 那么输出过滤器则是 必需的。 (但是它也是 无用的 如果打算对打印的用户计费， 因为 LPD 不会给输出过滤器任何用户或者主机的信息。)

在一台单个的打印机上， LPD 同时允许输出过滤器、 文本过滤器和其他的过滤器。 在某些情况下， LPD 将仅会启动输出过滤器来打印报头页 (详见 报头页)。 然后 LPD 会要求输出过滤器 自己停止运行 ， 它发送给过滤器两个字节： ASCII 031跟着一个 ASCII 001。 当输出过滤器看见这两个字节 (031, 001)， 它应该通过发送 SIGSTOP 信号来停止自己的运行。 当 LPD 已经运行好了其他的过滤器， 它会通过给输出过滤器发送 SIGCONT 信号来让输出过滤器重新运行。

如果仅有一个输出过滤器而 没有 文本过滤器， 并且 LPD 正在处理一个纯文本任务， LPD 会使用输出过滤器来完成这个任务。 像以前运行一样， 输出过滤器会按顺序打印任务中的文件， 而不会插入送纸或其他进纸的命令， 但这也许并 不是 您想要的结果。 在大多数情况下， 您还是需要一个文本过滤器。

lpf 这个我们前面介绍过的文本过滤器程序， 也可以用来做输出过滤器。
如果需要使用快速且混乱的输出过滤器， 但又不想写字节检测和信号发送代码， 那么试试 lpf。 lpf
也可以包含在一个 shell 脚本中来处理任何打印机可能需要的初始化代码。

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)。
它可以缩排输出 (用 lpr -i 命令提交的任务)， 可以打印控制字符禁止断页用 lpr -l 提交的任务)，
可以调整任务中退格和制表符打印的位置， 还可以对打印进行记帐。 它同样可以像输出过滤器一样工作。

lpf 适用于很多打印环境。 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执行 lpf 是很容易的。

为了让 lpf 可以正确的进行打印记帐， 那么需要 /etc/printcap 中的 pw 和 pl 变量都填入正确的值。
它用这些值来测定一页能打印多少文本， 并计算出任务有多少页。 想得到更多关于打印记帐的信息，
请参见 对打印机使用进行记帐。

10.4.2. 报头页

如果您有 很多 用户， 他们正在使用各式各样的打印机， 那么您或许要考虑一下把 报头页
当作无可避免之灾祸了。

报头页， 也叫 banner 或者 burst 页， 可以用来辨别打印出的文件是谁打印的。
它们通常用大号的粗体字母打印出来， 也可能用装饰线围绕四周， 所以在一堆打印出的文件中，
突出的显示了这个文件属于哪个用户的哪个任务。 这可以让用户快速的找到他们的任务。
而报头页一个明显的缺点就是， 在每个任务中都要有一张或者几张纸作为报头页印出来，
可是它们的有用的地方只发挥几分钟的作用， 最后它们会被放进回收站或者扔进垃圾堆。
(注意报头页只是一个任务一个， 而不是任务中的每个文件都有一个， 所以可能对纸张还不算很浪费。)

LPD 系统可以自动为您的打印提供报头页， 如果 您的打印机可以直接打印纯文本。 如果您的打印机是一台
PostScript® 打印机， 您将需要一个外部的程序来生成报头页； 详见 在 PostScript® 打印机上打印报头页。

10.4.2.1. 打开报头页

在 简单打印设置 这节， 我们通过在 /etc/printcap 文件中指定 sh ("禁止报头页") 来把报头页功能关掉了。
要重新为打印机开启报头页功能， 只需要删除掉 sh。

听起来很容易， 不是么?

是的。 您 可能 不得不让输出过滤器来给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。 参见 输出过滤器 这一节来得到更多信息。

下面是一个为我们以前介绍的叫做 teak 的打印机配置的 /etc/printcap 文件；
在配置当中我们开启了报头页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:
```

现在， 当用户再发任务给打印机 teak 的时候， 每个任务都会有一个报头页。
如果用户想要花时间来寻找他们自己打印的文件， 那么他们可以通过 lpr -h 命令来提交任务； 参考 报头页选项 这一节来得到更多关于 lpr(1) 的选项。

LPD 在报头页之后发出一个换纸字符。
如果您的打印机使用一个不同的字符或者字符串当作退纸指令， 在 /etc/printcap 中用 ff 变量指定即可。

10.4.2.2. 控制报头页

通过启用报头页， LPD 将生成出一个 长报头， 一整页的大字母， 标着用户， 主机和任务名。
下面是一个例子 (kelly 从主机 rose 打印了一个叫做 "outline" 的任务)：

```
k          ll    ll
k           l    l
k           l    l
k  k   eeee    l    l   y   y
k k    e   e   l    l   y   y
k k    eeeeee  l    l   y   y
kk k   e       l    l   y   y
k  k   e   e   l    l   y  yy
k   k  eeee   lll   lll   yyy y
                        y
                      y   y
                       yyyy

                 ll
           t     l    i
           t     l
 oooo   u   u  ttttt    l    ii   n nnn   eeee
o   o  u   u    t      l    i   nn  n  e   e
o   o  u   u    t      l    i   n   n  eeeeee
o   o  u   u    t      l    i   n   n  e
o   o  u  uu    t t    l    i   n   n  e   e
 oooo   uuu u    tt    lll   iii  n   n   eeee

r rrr    oooo    ssss    eeee
```

```
rr r o  o s  s e  e
r   o  o  ss   eeeeee
r   o  o   ss  e
r   o  o s  s e  e
r    oooo   ssss   eeee


                  Job:  outline
                  Date: Sun Sep 17 11:04:58 1995
```

LPD 会附加一个换页符在这段文本之后， 所以任务会在新的一页上开始 (除非设置了 sf (禁止换纸) 在 /etc/printcap 文件里目标打印机的记录中)。

如果您喜欢， LPD 可以生成一个 短报头； 指定 sb (短 banner) 在文件 /etc/printcap 中。
报头页就会看起来像下面这样：

```
rose:kelly  Job: outline  Date: Sun Sep 17 11:07:51 1995
```

同样是默认的， LPD 也是先打印报头页， 然后才是任务。 要想反过来， 在 /etc/printcap 中指定 hl (最后报头)。

10.4.2.3. 为带报头页的任务记帐

使用 LPD 内置的报头页会在进行打印记帐的时候产生一种特殊情况： 报头页肯定是 免费 的。

为什么?

因为输出过滤器是仅有的一个在打印报头页时能进行记帐的外部程序， 但却没有提供给它任何 用户或者主机 的信息或者记帐文件， 所以它无法知道谁应该为打印机的使用付费。 如果仅仅是 "增加一页计数" 给文本过滤器或者其他过滤器 (它们有用户和主机的信息) 是不够的， 因为用户可以用 lpr -h 命令跳过报头页。 他还是需要为自己并没有打印的报头页付钱。 基本上， lpr -h 是明知用户的首选， 但也不能强制让别人使用它。

让每个过滤器生成自己的报头页 (因此可以为它们计费) 是 仍然不够的。 如果用户想要用 lpr -h 命令禁止报头页， 它们将仍然印出报头页并且为它们付费。 因为 LPD 不会把 -h 这个参数传给任何过滤器。

这样， 您该怎么办呢?

您可以：

- 认可 LPD 的这个问题， 并且免费提供报头页打印。
- 安装一个替代 LPD 的软件， 比如 LPRng。 参考 替换标准的后台打印软件 来得到更多关于可以替代 LPD 的软件的信息。
- 写一个 聪明的 输出过滤器。 通常，
  输出过滤器不应该去完成除了初始化打印机或者进行一些简单字符转换以外的任何事情。
  它适合完成报头页和纯文本任务 (当没有文本 (输入) 过滤器时)。 但是，
  如果有文本过滤器为纯文本任务服务， 那么 LPD 将仅为打印报头页启动输出过滤器。 而且，
  这个输出过滤器可以理解报头页里 LPD 生成的信息， 然后决定哪位用户和主机应该为报头页付费。
  这种方法仅有的问题是输出过滤器仍然不知道应该使用什么记帐文件 (af 变量的内容并没有被传递过来)， 但是如果您有一个众所周知的记帐文件，
  就可以直接把文件名写进输出过滤器。 为了简化解释报头的步骤， 我们定义 sh (短报头) 变量在 /etc/printcap 文件中。 但这些还是太麻烦了，
  而且用户也更喜欢让他们免费打印报头页的慷慨的系统管理员。

10.4.2.4. 在 PostScript® 打印机上打印报头页

像上面描述的那样，LPD 可以生成一个纯文本的报头页来适应多种打印机。 当然， PostScript® 不能直接打印纯文本， 所以 LPD 没什么用-或者说大多时候是这样。

一个显而易见的方法来得到报头页就是让每个转换过滤器和文本过滤器都来生成报头页。 这些过滤器应该用用户名和主机的参数来生成一个相对应的报头页。 这种方法的缺点就是用户总是打印出报头页， 无论他们是否用 lpr -h 命令来提交的任务。

让我们来深入深入的研究一下这个方法。 下面的脚本输入三个参数 (用户登录名， 主机名， 和任务名) 然后生成一个简单的 PostScript® 报头页：

```
#!/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
```

现在， 每个转换过滤器和文本过滤器都能调用这段脚本来生成报头页， 然后打印用户的任务。
下面是我们早些时候在这个文档中提到的 DVI 转换过滤器， 被修改之后来生成一个报头页：

```
#!/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
```

过滤器是怎样解释参数列表来决定用户名和主机名的。 解释的方法对于其他转换过滤器来说也是一样的。
尽管文本过滤器需要输入的参数有些小的不同， (参见 过滤器是怎样工作的)。

像我们以前提到的那样， 上面的配置， 尽管相当简单， 关掉了 "禁止报头页" 的选项 (-h 选项) 在 lpr 中。
如果用户想要保护树木 (或者是几便士， 如果你对打印报头页收费的话)， 它还不能完成这件事情，
因为每个过滤器都要为每个任务打印一个报头页。

要允许用户对于每个任务都可以关闭报头页， 您需要使用在 为报头页记帐 这节中介绍的那种技巧：
写一个输出过滤器来解释 LPD- 生成的报头页并且生成一个 PostScript® 的版本。 如果用户用 lpr -h
命令提交任务， 那么 LPD 将不会生成报头页， 并且输出过滤器也不会生成报头页。 否则， 输出过滤器将从
LPD 读取文本， 然后发送适当的报头页的 PostScript® 编码给打印机。

如果您有的是一台连在串口上的 PostScript® 打印机， 您可以使用 lprps 里的一个输出过滤器， psof ，
它可以完成上述任务。 但注意 psof 不对报头页计费。

10.4.3. 网络打印

FreeBSD 支持网络打印： 发送任务给远程打印机。 网络打印通常指两种不同的方式：

- 访问一台连接在远程主机上的打印机。 在一台主机上安装一台常规的串口或并口打印机。 然后， 设置
  LPD 来通过网络访问其他主机上的打印机。 具体见 安装在远程主机上的打印机 这节。
- 访问一台直接连接在网络上的打印机。 打印机另有一个网络接口 (或者替代常规的串口或者并口)。
  这样的打印机可能像下面这样工作：
  - 它或许可以理解 LPD 的协议， 并且甚至可以接收远程主机发来的任务排进队列。 这样，
    它就像一个普通的主机运行着 LPD 一样。 做在 安装在远程主机上的打印机 里介绍的步骤，
    可以设置好这样的打印机。
  - 它或许支持网络数据流。 这样， 把打印机 "接" 在一台网络上的主机上，
    由这台主机负责安排任务并发送任务到打印机。 参见 带网络数据流接口的打印机
    这节来得到更多安装这类打印机的建议。

10.4.3.1. 安装在远程主机上的打印机

LPD 后台打印系统内建了对给其他也运行着 LPD (或者是与 LPD 兼容的) 的主机发送任务的功能。
这个功能使您可以在一台主机上安装打印机， 并让它可以在其他主机上访问。
这个功能同样适用在那些有网络接口并且可以理解 LPD 协议的打印机上。

要开启这种远程打印的功能， 首先在一台主机上安装打印机， 就是 打印服务器， 可以使用在
简单打印机设置 这节中简单设置的方法。 高级的设置可以参考 高级打印机设置 这节中你需要的部分。
一定要测试一下打印机， 看看它是不是所有您开启的 LPD 的功能都正常工作。 此外还需要确认 本地主机
允许使用 远程主机 上的 LPD 服务 (参见 限制远程主打印任务)。

如果您正在使用一台带网络接口并与 LPD 兼容的打印机， 那么我们那下面讨论中的 打印服务器
就是打印机本身， 而 打印机名 就是您为打印机配置的名字。 参考随打印机和/或者打印机-
网络接口供给的文档。

如果您正使用惠普的 Laserjet， 则打印机名 text 将自动地为您完成 LF 到 CRLF 的转换，
因而也就不需要 hpif 脚本了。

然后， 在另外一台你想要访问打印机的主机上的 /etc/printcap 文件中加入它们的记录， 像下面这样：

1. 可以随意给这个记录起名字。 简单起见， 您可以给打印服务器使用相同的名字或者别名。
2. 保留 lp 变量为空， (:lp=:)。
3. 建立一个后台打印队列目录， 并用 sd 变量指明其位置。 LPD 将把任务提交给打印服务器之前，
   会把这些任务保存在这里。
4. 在 rm 变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。

就是这样。 不需要列出转换过滤器， 页面大小， 或者其他的一些东西在 /etc/printcap 文件中。

这有一个例子。 主机 rose 有两台打印机， bamboo 和 rattan。 我们要让主机 orchid
的用户可以使用这两台打印机。 下面是 /etc/printcap 文件， 用在主机 orchid (详见 开启报头页) 上的。
文件中已经有了打印机 teak 的记录； 我们在主机 rose 上增加了两台打印机：

```
#
```

```
# /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， 我们只需要在主机 orchid 上建立一个后台打印队列目录：

```
# 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
```

现在， 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 上输入了：

```
% lpr -P bamboo -d sushi-review.dvi
```

LPD 系统在主机 orchid 上会复制这个任务到后台打印队列目录 /var/spool/lpd/bamboo 并且记下这是一个 DVI 任务。 当主机 rose 上的打印机 bamboo 的后台打印队列目录有空间的时， 这两个 LPD 系统将会传输这个文件到主机 rose 上。 文件将排在主机 rose 的队列中直到最终被打印出来。 它将被从 DVI 转换成 PostScript® (因为 bamboo 是一台 PostScript® 打印机) 在主机 rose。

10.4.3.2. 带有网络数据流接口的打印机

通常， 当您为打印机购买了一块网卡， 可以得到两个版本： 一个是模拟后台打印 (贵一些的版本)， 或者一个只发送数据给打印机就像在使用串口或者并口一样 (便宜一些的版本)。 这节讲述如何使用这个便宜一些的版本。 要得到贵一些版本的更多信息， 参见前面章节 安装在远程主机上的打印机。

/etc/printcap 文件的格式让您指定使用哪个串口或并口， 并且还要指定 (如果您正在使用串口)，
使用多快的波特， 是否使用流量控制， 为制表符延迟， 转换换行， 等等。
但是没有一种方法指定一个连接到一台正在监听 TCP/IP 的或者其他网络接口的打印机。

要发送数据到网络打印机， 就需要开发一个通讯程序， 它可以被文本或者转换过滤器调用。
下面是一些例子： 脚本 netprint 将标准输入的所有数据发送到一个连在网络上的打印机。
我们将打印机的名字作为第一个参数， 端口号跟在后面作为第二个参数， 传给 netprint。
注意它只支持单向通讯 (FreeBSD 到打印机)； 很多网络打印机支持双向通讯， 并且这是您可能利用到的
(得到打印机状态， 进行打印记帐， 等等的时候。)。

```
#!/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;
```

然后我们就可以在多种过滤器里使用这个脚本了。 加入我们有一台 Diablo 750-N 行式打印机联在网络上。
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。
这里是为这个打印机写的文本过滤器：

```
#!/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

这节将讲述关于限制打印机使用的问题。 LPD 系统让您可以控制谁可以访问打印机，
无论本地或是远程的， 是否他们可以打印机多份副本， 任务可以有多大， 以及打印队列的尺寸等。

10.4.4.1. 限制多份副本

LPD 系统能够简化用户在打印多份副本时的工作。 用户可以用 lpr -#5 (举例) 来提交打印任务，
则会将任务中每个文件都打印五份副本。 这是不是一件很棒的事情呢。

如果您感觉多份副本会对打印机造成不必要的磨损和损耗， 您可以屏蔽掉 lpr(1) 的 -# 选项， 这可以通过在 /etc/printcap 文件中增加 sc 变量来完成。 当用户用 -# 选项提交任务时， 他们将看到：

```
lpr: multiple copies are not allowed
```

注意当为一台远程打印机进行设置时 (参见 安装在远程主机上的打印机 这一节) 您还需要同时在远程主机的 /etc/printcap 文件中 增加sc 变量， 否则用户还是可以从其他主机上提交使用多份副本的任务。

下面是一个例子。 这个是 /etc/printcap 文件在主机 rose 上。 打印机 rattan 非常轻闲，
所以我们将允许多份副本， 但是激光打印机 bamboo 则有些忙， 所以我们禁止多份副本， 通过增加 sc 变量：

```
#
#  /etc/printcap for host rose - restrict multiple copies on bamboo
#
rattan|line|diablo|lp|Diablo 630 Line Printer:\
	:sh:sd=/var/spool/lpd/rattan:\
	:lp=/dev/lpt0:\
	:if=/usr/local/libexec/if-simple:

bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
	:sh:sd=/var/spool/lpd/bamboo:sc:\
	:lp=/dev/ttyu5:ms#-parenb cs8 clocal crtscts:rw:\
	:if=/usr/local/libexec/psif:\
	:df=/usr/local/libexec/psdf:
```

现在， 我们还需要增机 sc 变量在主机 orchid 的 /etc/printcap 文件中 (顺便我们也禁止打印机 teak 多份打印)：

```
#
#  /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:
```

通过使用 sc 变量， 我们阻止了 lpr -# 命令的使用， 但仍然没有禁止用户多次运行 lpr(1) ，
或者多次提交任务中同样的文件， 像下面这样：

```
% lpr forsale.sign forsale.sign forsale.sign forsale.sign forsale.sign
```

这里有很多种方法可以阻止这种行为 (包括忽略它)， 并且是免费的。

10.4.4.2. 限制对打印机的访问

您可以控制谁可以打印到哪台打印机通过 UNIX® 的组机制和文件 /etc/printcap 中的 rg 变量。
只要把可以访问打印机的用户放进适当的组中， 然后在 rg 变量中写上组的名字。

如果这组以外的用户 (包括 root) 试图打印到被限制的打印机，将会得到这样的提示：

```
lpr: Not a member of the restricted group
```

像使用 sc (禁止多份副本) 变量一样， 您需要指定 rg 在远程同样对打印机有访问限制的主机上，
如果您感觉合适的话 (参考 安装在远程主机上的打印机 这一节)。

比如， 我们将让任何人都可以访问打印机 rattan， 但只有在 artists 组中的人可以使用打印机 bamboo。
这里是类似的主机 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/ttyu5:ms#-parenb cs8 clocal crtscts:rw:\
        :if=/usr/local/libexec/psif:\
        :df=/usr/local/libexec/psdf:
```

Let us leave the other example /etc/printcap file (for the host orchid) alone. Of course, anyone on orchid can print to bamboo. It might be the case that we only allow certain logins on orchid anyway, and want them to have access to the printer. Or not.

这里每台仅能有一个限制的组。

10.4.4.3. 控制提交的任务大小

如果您有很多用户访问打印机， 可能需要对用户可以提交的文件尺寸设置一个上限。 毕竟，
文件系统中后台打印队列目录的空间是有限的， 您需要保证这里有空间来存放其他用户的任务。

LPD 允许通过使用 mx 变量来限制任务中文件的最大字节数， 方法是指定单位为块的 BUFSIZ 数， 每块表示 1024 字节。 如果在这个变量的值是 0， 则表示不进行限制； 不过， 如果不指定 mx 变量的话，
则会使用默认值 1000 块。

这个限制是对于任务中 文件 的， 而 不是 任务总共的大小。

LPD 不会拒绝比限制大小大的文件。 但它是将限制大小以内的部分排入队列， 并且打印出来的只有这些。
剩下的部分将被丢弃。 这个行为是否正确还需讨论。

让我们来为例子打印机 rattan 和 bamboo 增加限制。 由于那些 artists 的 PostScript® 文件可能会很大，
我们将限制大小为 5 兆字节。 我们将不对纯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/ttyu5:ms#-parenb cs8 clocal crtscts:rw:\
    :if=/usr/local/libexec/psif:\
    :df=/usr/local/libexec/psdf:
```

同样， 限制只对本地用户起作用。 如果设置了允许远程用户使用您的打印机，
远程用户将不会受到这些限制。 您也需要指定 mx 变量在远程主机的 /etc/printcap 文件中。 参见
安装在远程主机上的打印机 这一节来得到更多有关远程打印的信息。

除此之外， 还有另一种限制远程任务大小的方法； 参见 限制远程主机打印任务。

10.4.4.4. 限制远程主机打印任务

LPD 后台打印系统提供了多种方法来限制从远程主机提交的任务：

主机限制
: 您可以控制本地 LPD 接收哪台远程主机发来的请求， 通过 /etc/hosts.equiv 文件和 /etc/hosts.lpd 文件。 LPD 查看是否到来的任务请求来自被这两个文件中列出的主机。 如果没有， LPD

会拒绝这个请求。

这些文件的格式非常简单： 每行一个主机名。 注意 /etc/hosts.equiv 文件也被 ruserok(3) 协议使用，
并影响着 rsh(1) and rcp(1) 等程序， 所以要小心。

举个例子， 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 的请求。
如果任何其他的主机试图访问主机 rose 的 LPD， 任务将被拒绝。

大小限制

您可以控制后台打印队列目录需要保留多少空间。 建立一个叫做 minfree
的文件在后台打印队列目录下为本地打印机。 在这个文件中插入一个数字来代表多少磁盘块数 (512
字节) 的剩余空间来接收远程任务。

这让您可以保证远程用户不会填满您的文件系统。 您也可以用它来给本地用户一个优先：
他们可以在磁盘剩余空间低于 minfree 文件中的指定值后仍然可以提交任务。

比如， 让我们增加一个 minfree 文件为打印机 bamboo。 我们检查 /etc/printcap
文件来找到这个打印机的后台打印队列目录； 这里是打印机 bamboo 的记录：

```
bamboo|ps|PS|S|panasonic|Panasonic KX-P4455 PostScript v51.4:\
    :sh:sd=/var/spool/lpd/bamboo:sc:rg=artists:mx#5000:\
    :lp=/dev/ttyu5:ms#-parenb cs8 clocal crtscts:rw:mx#5000:\
    :if=/usr/local/libexec/psif:\
    :df=/usr/local/libexec/psdf:
```

后台打印队列目录在 sd 变量中给出。 我们设置 3 兆字节 (6144 磁盘块)
为文件系统上必须存在的总共剩余空间， 让 LPD 可以接受远程任务：

```
# echo 6144 > /var/spool/lpd/bamboo/minfree
```

用户限制

您可以控制哪些远程用户可以打印到本地打印机， 通过指定 rs 变量在 /etc/printcap 文件中。 当 rs
出现在一个本地打印机的记录中时， LPD 将接收来自远程主机 并
在本地有同样登录名的用户提交的任务。 否则， LPD 会拒绝这个任务。

这个功能在一个 (比如) 有许多部门共享一个网络的环境中特别有用，
并且有些用户可以越过部门的边界。 通过为他们在您的系统上建立帐号，
他们可以他们自己的部门的系统里使用您的打印机。 如果 只 允许他们您的打印机，
而不是您的计算机资源， 您可以给他们 "象征" 帐户， 不带主目录并且设置一个没用的 shell ， 比如
/usr/bin/false。

10.4.5. 对打印机使用记帐

当然， 你需要对打印付费。 为什么不？ 纸张和墨水都需要花钱的。 并且这里还有维护的费用 -
打印机是由很多部件组装成的， 并且零件会坏掉。 您可以检查您的打印机， 使用形式，
和维护费用来得出每页 (或者每尺， 每米， 或者每什么) 的费用。 现在， 您怎样启动打印记帐呢？

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。 记帐是一个对使用的打印机的种类，
打印的格式， 和 您的 在对打印机的使用计费的需求依赖性很高的。

要实现记帐， 您必须更改打印机的文本过滤器 (对纯文本任务记费) 和转换过滤器 (对其他格式的文件计费)，
要统计页数或者查询打印了多少页的话。 您不可以通过使用简单的输出过滤器来逃脱计费，
因为它不能进行记帐。 参见 过滤器 这节。

通常， 有两种方法来进行记帐：

- 定期记帐 是更常用的方法， 可能因为它更简单。 无论合适何人打印一个任务，
  过滤器都将记录用户名， 主机名， 和打印的页数到一个记帐文件。 每个月， 学期， 年，
  或者任何您想设定的时间段， 收集这些不同打印机上的记帐文件， 按用户对打印的页数进行结算，
  并对使用进行付费。 然后删掉所有记录文件， 开始一个新的计费周期。
- 实时记帐 不太常用， 可能因为它比较难。 这种方法让过滤器对用户的打印进行实时的记帐。
  像磁盘配额， 记帐是实时的。 您可以组织用户打印当他们的帐户超额的时候，
  并且可能提供一种方法让用户检查并调整他们的 "打印配额。"
  但这个方法需要一些数据库代码来跟踪用户和他们的配额。

LPD 后台打印系统对两种方法都支持且很简单： 所以您需要提供过滤器 (大多数时候)，
还要提供记帐代码。 但这好的方面是： 您可以有非常灵活的记帐方法。 比如，
您可以选择使用阶段记帐还是实时记帐。 您可以选择记录哪些信息： 用户名， 主机名， 任务类型，
打印页数， 使用了多少平方尺的纸， 任务打印了多长时间， 等等。
您可以通过修改过滤器来存储这些信息。

10.4.5.1. 快速并且混乱的打印记帐

FreeBSD 包含两个可以让您立刻可以建立起简单的阶段记帐的程序。 它们是文本过滤器 lpf， 在 lpf: 一个文本过滤器 这节中描述， 和 pac(8)， 一个收集并统计打印机记帐文件中记录的程序。

像在前面章节提到的过滤器一样 (过滤器)， LPD
启动文本或者转换过滤器并在过滤器命令行里带上记帐文件的名字。
过滤器可以使用这个参数知道该往哪写记帐记录。 这个文件的名字来自于 af 变量在 /etc/printcap 文件里，
并且如果没有指定绝对路径， 则默认是相对于后台打印队列目录的。

LPD 启动 lpf 带着页宽和页长的参数 (通过 pw 和 pl 变量)。 lpf 使用这些参数来判定将使用多少张纸。
在文件发送到打印机之后， 它就会在记帐文件中写入记录。 记录像下面这个样子：

```
2.00 rose:andy
3.00 rose:kelly
3.00 orchid:mary
5.00 orchid:mary
2.00 orchid:zhang
```

您应该让每个打印机都使用一个独立的记帐文件， 像 lpf 就没有内建文件锁逻辑， 这样两个 lpf
可能会发生彼此记录混合的情况， 如果它们同时要在同一个文件写入内容的时候。
一个最简单的保证每个打印机都使用一个独立的记帐文件的方法就是将 af=acct 写在 /etc/printcap
文件中。 然后， 每个打印机的记帐文件都会在这台打印机的后台打印队列目录中， 文件的名字叫做 acct。

当您准备对用户的打印进行收费时， 运行 pac(8) 程序。
只要转换到要收集信息的这台打印机的后台打印队列目录， 然后输入 pac。
您将会得到一个美元计费的摘要像下面这样：

```
 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
```

这些是 pac(8) 需要的参数：

-P 打印机
: 哪台 打印机 要结帐。 这个选项仅在用 af 变量在 /etc/printcap 文件中指定了绝对路径的情况下起作用。

-c
: 以金额来排序输出来代替以用户名字字母排序。

-m
: 忽略记帐文件中的主机名。 带上这个选项， 用户 smith 在主机 alpha 上与同样的用户 smith 在主机 gamma 上一样。 不带这个选项的话， 他们则是不同的用户。

-p 单价
: 使用 price 作为每页或每尺美元的单价来替代 pc 变量指定的单价在 /etc/printcap 文件中， 或者两分 (默认)。 price 可以用一个浮点数来指定。

-r
: 反向排序。

-s
: 建立一个记帐摘要文件， 并且截短记帐文件。

名字 …
: 只打印指定 名字 用户的记帐信息。

在 pac(8) 默认产生的摘要中， 可以看到在不同主机上的每个用户打印了多少页。 如果在您这里，
主机不考虑 (因为用户可以使用任何主机)， 运行 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
```

要以美元计算应付钱数， pac(8) 指定 pc 变量在 /etc/printcap 文件中 (默认是 200， 或者 2 分每页).
这个参数的单位是百分之一分， 在这个变量中指定每页或者每尺的价格。 您可以覆盖这个值当运行 pac(8)
带着参数 -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
```

设定每页的价格是 1 美元 5 美分。 您可以通过这个选项来达到目标利润。

最终， 运行 pac -s 将存储这些信息在一个记帐文件里， 文件名和打印机帐户的名字相同， 但是带着 _sum 的后缀。 然后截短记帐文件。 当您再次运行 pac(8) 的时候， 它再次读取记帐文件来得到初始的总计， 然后在记帐文件中增加信息。

10.4.5.2. 怎样对打印的页数进行计数?

为了进行远程的精确记帐， 需要判断一个任务将会消耗多少张纸。 这是打印记帐问题的关键。

对于纯文本任务， 这个问题不是太难解决:
对任务中的行数进行计数然后与打印机支持的每页行数进行比较。 别忘了也对添印的行，
或者很长的逻辑上的一行但在打印机上会折成两行的这类进行记帐。

文本过滤器 lpf (在 lpf: 一个文本过滤器 这节中介绍) 会在记帐时考虑这些问题。
如果正在编写一个可以进行记帐的文本过滤器， 您可能需要查看 lpf 的源代码。

怎样处理其他格式的文件?

好， 对于 DVI- 到 -LaserJet 或者 DVI- 到 -PostScript® 转换， 可以让您的过滤器输出诊断信息， 关于 dvilj 或者 dvips 命令， 并且看到多少页被转换了。 您也许可以对于其他类型的文件和转换程序进行类似操作。

但是这些方法的弱点就是事实上打印机并不是打印了所有的页。 比如， 卡纸， 缺墨， 或者炸掉了 - 但用户还是要为没有打印的部分付钱。

您该怎样做?

只有一条 肯定 的方法来进行 精确 的记帐。 购买一台可以告诉您它使用了多少纸的打印机，
并且将它连接到串口或者网络上。 几乎所有 PostScript® 打印机都支持这个小功能。
其他制造厂或其他型号也可以有这个功能 (比如 Imagen 激光网络打印机)。
为这些打印机更改过滤器使它在打印完每个任务之后接收纸张用量， 并 仅 基于这个值进行记帐。
不需要计算行数， 也不需要容易出错的文件检查。

当然， 您也总是可以大方的使打印免费。

10.5. 使用打印机

这节将讲述如何使用在 FreeBSD 下设置好的打印机。 下面是一个用户级命令的总览:

lpr(1)
: 打印任务

lpq(1)
: 检查打印队列

lprm(1)
: 从打印机的队列中移除任务

还有一个管理命令， lpc(8)， 在 管理打印机 一节中有所介绍， 它可以用于控制打印机及其队列。

lpr(1), lprm(1), and lpq(1) 这三个命令都接受 -P printer-name 选项来指定对哪个打印机 / 队列进行操作，
在 /etc/printcap 文件中列出的打印机。 这允许您提交， 删除， 并检查任务在多个打印机上。
如果您不使用 -P 选项， 那么这些命令会使用在 环境变量 PRINTER 中指定的打印机。 最终， 如果您也没有 PRINTER 这个环境变量， 这些命令的默认值是叫做 lp 的这台打印机。

从此以后， 术语 默认打印机 就是指 PRINTER 环境变量中指定的这台， 或者叫做 lp
的这一台当没有环境变量 PRINTER 的时候。

10.5.1. 打印任务

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
```

要选择一个指定的打印机， 输入：

```
% lpr -P printer-name filename ...
```

这个例子打印一个当前目录的长长的列表到叫做 rattan 的这台打印机：

```
% ls -l | lpr -P rattan
```

因为没有为 lpr(1) 命令列出文件， lpr 从标准输入读入数据， 在这里是 ls -l 命令的输出。

lpr(1) 命令同样可以接受多种控制格式的选项， 应用文件转换， 生成多份副本， 等等。 要得到更多信息，
参考 打印选项 这节。

10.5.2. 检查任务

当使用 lpr(1) 进行打印时， 您希望打印的所有数据被放在一起打包成了一个 "打印任务"， 它被发送到 LPD
后台打印系统。 每台打印机都有一个任务队列， 并且您的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规则打印这些任务。

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 输出的一个例子：

```
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， 用户 kelly 提交的， 标识 "任务编号" 9。
每个要打印的任务都会获得一个不同的任务编号。 大多时候可以忽略这个任务编号，
但在您需要取消任务时会用到这个号码； 参考 移除任务 这节得到更多信息。

编号为 9 的任务包含了两个文件； 在 lpr(1) 命令行中指定的多个文件被看作是一个单个的任务。
它是当前激活的任务 (注意这个词 激活 在 "Rank" 这列下面)， 意思是打印机当前正在打印那个任务。
第二个任务包含了标准输入传给 lpr(1) 命令的数据。 第三个任务来自用户 mary; ,

它是一个比较大的任务。 她要打印的文件的路径名太长了， 所以 lpq(1) 命令只显示了三个点。

lpq(1) 输出的头一行也很有用： 它告诉我们打印机正在做什么 (或者至少是 LPD
认为打印机应该正在做的)。

lpq(1) 命令同样支持 -l 选项来生成一个详细的长列表。 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. 移除任务

如果您对一个打印任务改变了主意， 可以用 lprm(1) 将任务从队列中删除。 通常， 您甚至可以用 lprm(1)
命令来移除一个当前激活的任务， 但是任务的一部分或者所有还是可能打印出来。

要从默认打印机中移除一个任务， 首先使用 lpq(1) 找到任务编号。 然后输入：

```
% lprm job-number
```

要从指定打印机中删除任务， 增加 -P 选项。 下面的命令会删除编号为 10 的任务从 bamboo 这台打印机：

```
% lprm -P bamboo 10
```

lprm(1) 命令有一些快捷方式：

lprm -
    删除所有属于您的任务 (默认打印机的)。

lprm user
    删除所有属于用户 user 的任务 (默认打印机的)。 超级用户可以删除用户的任务；
    您只可以删除自己的任务。

lprm
    命令行中不带任务编号， 任务名， 或者 - 选项， lprm(1) 会删除默认打印机上当前激活的任务，
    如果它属于你。 超级用户可以删除任务激活的任务。

使用参数 -P 和上面的快捷方式来用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户在打印机 rattan 队列中的所有任务：

```
% lprm -P rattan -
```

如果您正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时。 下面的命令证明了这个：

```
% 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. 超越纯文本：打印选项

lpr(1) 支持许多控制文本格式的参数， 转换图形和其他格式文件， 生成多份副本， 处理任务， 等等。
这一节将描述这些选项。

10.5.4.1. 格式与转换选项

下面的 lpr(1) 参数控制任务中文件的格式。 使用这些参数， 如果任务不含纯文本， 或者您想让纯文本通过 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) 文件名为 fish-report.dvi 到打印 bamboo：

```
% lpr -P bamboo -d fish-report.dvi
```

这些选项应用到任务中的每个文件， 所以您不能混合 (说) 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， 使用不同的转换选项给不同的任务。

所有这些选项除了 -p 和 -T 都需要转换过滤器安装给目标打印机。 例如， -d 选项需要 DVI 转换过滤器。 参考 转换过滤器 这节得到更多细节。

-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
缩进 number 列； 如果没有指定 number， 则缩进 8 列。 这个选项仅可以工作在某些过滤器上。

不要在选项 -i 和数字之间加入空格。

-l
打印文字数据， 包括控制字符。

-n
打印 ditroff (无设备依赖 troff) 数据。

-p
打印之前用 pr(1) 格式化纯文本。 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。 这个选项仅在使用 -p 选项时起作用。

-t
打印 troff 数据。

-v
打印 raster 数据。

下面是一个例子： 这个命令打印了一个很好的 ls(1) 联机手册到默认打印机：

```
% zcat /usr/shared/man/man1/ls.1.gz | troff -t -man | lpr -t
```

zcat(1) 命令解压缩 ls(1) 的手册并且将内容传给 troff(1) 命令， 它将格式化这些内容并且生成 GNU troff 输出给 lpr(1) ， 它提交任务到 LPD 后台打印。 因为使用了 -t 选项为 lpr(1) ， 后台打印将会转换 GNU troff 输出到默认打印机可以理解的格式当任务被打印时。

10.5.4.2. 任务处理选项

下面的 lpr(1) 选项告诉 LPD 对任务特殊处理：

-# copies
生成 copies 个副本给任务中的每个文件， 替代每个文件一份副本。
管理员可以禁止这个选项来减少打印机的浪费和鼓励复印机的使用。 参考 限制多份副本。

这个例子打印三份副本的文件 parser.c 跟着三份副本的文件 parser.h 到默认打印机：

```
% lpr -#3 parser.c parser.h
```

-m
打印完成后发信。 使用这个选项， LPD 系统将会发送邮件到您的帐户， 当它完成了处理您的任务后。 在信中， 它将会告诉您任务是否成功完成或者出现了错误， 并且 (通常) 指明是什么错误。

-s
不要复制文件到后台打印队列目录， 要使用符号连接。

如果您正在打印一个很大的任务， 您可能需要这个选项。 它节省后台打印队列目录的空间 (您的任务可能使后台打印队列目录所在的文件系统剩余空间超出)。 它同样也节省了时间， 因为 LPD 将不会副本任务的每个字节到后台打印队列目录。

这也有一个缺点： 因为 LPD 将直接指向源文件， 您不能修改或者删除它们直到它们被打印出来。

如果您打印到一台远程打印机， LPD 将最终将文件从本地主机副本到远程主机上，所以选项 -s 只能节省本地后台打印队列目录的空间， 而不是远程的。 虽然如此，但它还是很有用。

-r
: 移除任务中的文件在它们被复制到后台打印队列目录之后， 或者在用 -s 选项打印它们之后。
谨慎使用这个选项！

10.5.4.3. 报头页选项

这些 lpr(1) 的选项调整了通常出现在任务报头页上的文本。 如果报头页被跳过了在目标打印机上，这些选项将不会起作用。 参考 报头页 得到更多关于设置报头页的信息。

-C text
: 替换报头页上的主机名为 text。 主机名通常都是提交任务的主机名称。

-J text
: 替换报头页上的任务名为 text。 任务名通常是任务中头一个文件的名字， 或者 stdin
如果您正在打印标准输入。

-h
: 不打印任何报头页。

在某些地点， 这个选项可能无效， 与报头页的产生方法有关。 参考 报头页 得到详细信息。

10.5.5. 管理打印机

作为一个打印机的管理者， 您必须要安装， 设置， 并且测试它们。 使用 lpc(8) 命令，
您可以与打印机以更多的方式交流。 用 lpc(8) ， 您可以

- 启动或停止打印机
- 启用或禁止它们的队列
- 重新安排每个队列中的任务。

首先， 一个关于术语的解释： 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。
但用户还是可以提交任务， 它们会在队列中等待直到打印机被 启动 或者队列被清空。

如果一个队列被 禁止， 没有用户 (除了 root) 可以提交任务到打印机。 一个 启用 的队列允许任务被提交。
一个打印机可以被 启动 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务， 直到队列为空。

通常， 您必须有 root 权限来使用 lpc(8) 命令。 普通用户可以使用 lpc(8)
命令来获得打印机状态并且重启一台挂了的打印机。

这里是一个关于 lpc(8) 命令的摘要。 大部分命令带着一个 printer-name 参数来知道要对哪台打印机操作。
您可以用 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
: 取消当前任务并停止打印机。 用户仍然可以提交任务， 如果队列还是启用的。

clean printer-name
: 从打印机的后台打印队列目录移除旧的文件。 有时， 组成任务的文件没有被 LPD 正确的删除，
特别是在打印中出现错误或者管理活动比较多的时候。
这个命令查找不属于后台打印队列目录的文件并删除它们。

disable printer-name
: 禁止新任务入队。 如果打印机正在工作， 它将会继续打印队列中剩余的任务。 超级用户 (root)
总是可以提交任务， 甚至提交到一个禁止的队列。

这个命令在测试一台新打印机或者安装过滤器时非常有用： 禁止队列并提交以 root 提交任务。
其他用户将不能提交任务直到您完成了测试并用命令 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
: 为打印机开启队列。 用户可以提交任务到打印机但是在打印机启动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， 则打印可用命令的摘要。

restart printer-name
: 启动打印机。 普通用户可以使用这个命令， 当一些特别的环境导致 LPD 锁死时，
但他们不能启用一台使用 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务。

stop printer-name
: 停止打印机。 打印机将完成当前任务并且将不再打印队列中的任务任务。 尽管打印机被停用，
但用户仍然可以提交任务到一个开启的队列。

topq printer-name job-or-username
: 重新以 printer-name 安排队列， 通过将列出的 job 编号或者指定的所属 username
的任务放在队列的最前面。 对于这个命令， 您不可以使用 all 当作 printer-name。

up printer-name
: 打印机上线； 相对于 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。 如果您不输入任何命令， lpc(8) 则进入一个交互模式，
在这里您可以输入命令直到输入 exit， quit， 或者文件结束符。

10.6. 替换标准后台打印

如果您已经通读过了这个手册， 那么到现在您应该已经了解了关于 FreeBSD 包含的后台打印系统 LPD 的一切。 您可能发现了它很多的缺点， 它们很自然的让您提出这样的问题： "这里还有什么后台打印系统吗 (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 维护的主要负责人) 合作完成了 LPRng。 LPRng 的主站是 http://www.lprng.org/。

CUPS
: CUPS， 通用 UNIX 打印系统， 提供了一个轻便的打印层给 UNIX®-基础的操作系统。 它是由 Easy Software Products 开发的， 并且成为了 UNIX® 供应商和用户的标准打印解决方案。

  CUPS 使用 Internet 打印协议 (IPP) 作为管理打印任务和队列的基础。 行式打印机守护程序 (LPD) 服务器消息块 (SMB)， 和 AppSocket (a.k.a. JetDirect) 协议的部分功能也被支持。 CUPS 增加了基于浏览网络打印机和 PostScript 打印机描述 (PPD) 的打印选项来支持 UNIX® 下的真实打印。

  CUPS 的主站是 http://www.cups.org/。

HPLIP
: HPLIP， HP Linux® 成像及打印系统 (Imaging and Printing system)， 是一套由 HP 开发的用于支持 HP 的打印、 扫描和传真设备的工具。 这套程序利用 CUPS 打印系统作为后端来提供一些打印方面的功能。

HPLIP 的主页位于 http://hplipopensource.com/hplip-web/index.html。

10.7. 疑难问题

在使用 lptest(1) 进行简单的测试之后， 您可能得到了下面的结果， 而不是正确的结果：

过了一会儿， 它工作了； 或者， 它没有退出一整张纸。
打印机进行了打印， 但在这之前它呆了一段而且什么都没做。 事实上， 您可能需要按一下打印机上的 打印剩余 或者 送纸 按钮来让结果出现。

如果这是问题所在， 打印机可能在等待， 看看在打印之前， 您的任务是否还有更多的数据。
要修正这个问题， 您可以让文本过滤器发送一个送纸字符 (或者其他需要的) 到打印机。
这通常足够让打印机立即打印出内部缓存内剩余的文本。
它同样可以用来确保每个任务的结尾都占用一整张纸，
这样下一个任务才不会在前一个任务最后一张纸的中间开始。

接下来的 shell 脚本 /usr/local/libexec/if-simple
的脚本打印了一个送纸符在它发送任务到打印机之后：

```
#!/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
```

它的输出产生了 "楼梯效果"。
您可能在纸上得到下面这些：

```
!"#$%&'()*+,-./01234
        "#$%&'()*+,-./012345
                #$%&'()*+,-./0123456
```

您也成为了 楼梯效果 的受害者， 这是由对新行的标志字符的解释不一致造成的。 UNIX® 风格的操作系统使用一个单个字符： ASCII 码 10， 即换行 (LF)。 MS-DOS®， OS/2®， 和其他的系统使用一对儿字符， ASCII 码 10 和 ASCII 码 13 (回车 CR)。 许多打印机使用 MS-DOS® 的习惯来代表新行。

当您在 FreeBSD 上打印时， 您的文本仅用了换行字符。 打印机， 打印机看到换行字符后， 走一行纸， 但还光标位置还是在这张纸上要打印的下一个字符处。 这就是回车的作用： 将下一个要打印的字符的位置移到纸张的左边缘。

这里是 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

下面有几种完成这个的办法：

- 使用打印机的配置开关或者控制面板来更改它对这些字符的解释。
  查看打印机的手册来找到怎样更改。

如果您引导您的系统到其他除了 FreeBSD 之外的操作系统， 您可能不得不
重新配置 打印机使用 这个操作系统对 CR 和 LF 字符的解释。
您可能更喜欢下面这另一种解决方案。

- 让 FreeBSD 的串口驱动自动转换 LF 到 CR+LF。 当然， 这 仅仅 工作在串口打印机上。
  要开启这个功能， 定义 ms# 变量并 设置 onlcr 模式在 /etc/printcap 文件中相应打印机处。
- 发送一个 转义码 到打印机来让它临时对 LF 字符做不同的处理。
  参考您的打印机手册来了解您的打印机支持哪些转义码。 当您找到合适的转义码，
  修改文本过滤器让其先发送这个转义码， 然后再发送打印任务。

  这里是一个为懂得 Hewlett-Packard PCL 转义码的打印机编写的文本过滤器。
  这个过滤器使得打印机将 LF 作为一个 LF 和一个 CR 来对待； 然后它发送任务；
  最后发送一个送纸符弹出任务的最后一张纸。 它应该可以在几乎所有 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。
  它只有一台打印机连接在第一个并口上， 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:
```

行行覆盖。
打印机从来不进纸换行。 所有的文本都打印在头一行文本的上面。

这个问题是 "相反" 于楼梯效果， 像上面描述的那样， 并且更少见。 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纸的左边的字符对待。 而没有向下走纸一行。

使用打印机的配置开关或者控制面板来强制对 LF 和 CR 进行下面的转换：

打印机收到	打印机打印
CR	CR

打印机收到	打印机打印
LF	CR + LF

打印丢掉字符。

当打印时， 每行里打印机都丢掉一些字符没有打。 这个问题可能随着打印的进行越发严重，
丢掉越来越多的字符。

这个问题是由打印机跟不上计算机通过串口发送数据的速度造成的
(这个问题应该不会发生在并口打印机上)。 有两种方法能克服这个问题：

- 如果打印机支持 XON/XOFF 流量控制， 那就让 FreeBSD 使用它， 通过加入 ixon 模式在 ms# 变量里。
- 如果打印机支持请求/清除硬件握手信号 （通常时 RTS/CTS）， 指定 crtscts 模式在 ms# 变量里。 并且要确定连接打印机和计算机的线是支持硬件流量控制的。

它打印出垃圾。

打印机打印出的东西看起来是一些随机的字符， 而不是想要打印的东西。

这通常意味着另一种串口打印机通讯参数设置不正确的错误。 复查 br 变量中设定的波特， 和 ms# 中的校验设置； 确定打印机也在使用和 /etc/printcap 文件中相同的设置。

没有反应。

如果没有反应， 问题就可能出在 FreeBSD 而不是硬件上了。 增加日志文件 (lf) 变量到 /etc/printcap 文件里出现问题的打印机的记录处。 比如， 下面是打印机 rattan 的记录， 使用了 lf 变量：

```
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。 检查日志文件 (在我们的例子当中， 是 /var/log/rattan.log 这个文件)
来看是否有错误信息出现。 根据出现的信息， 试着来修正问题。

如果您没有指定 lf 变量， LPD 会使用 /dev/console 作为默认值。

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。 据说在某些情况下， 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启用虚拟 8086 模式 i386™ 特有的调用， 则无法在 FreeBSD 上运行。 另外， 目前还不支持 64-bit 的 Linux® 二进制程序。

读完这章，您将了解到：

- 如何在 FreeBSD 系统中启用 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® 兼容模式的实现细节。

在阅读这章之前，您应该知道：

- 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® 库并没有被安装而且 Linux® 二进制兼容模式也没有被启动。 Linux® 库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® 库的方法。 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
```

查看模块是否已经被加载：

```
% kldstat
Id Refs Address    Size     Name
 1    2 0xc0100000 16bdb8   kernel
 7    1 0xc24db000 d000     linux.ko
```

在 /etc/rc.conf 中加入以下这行后 Linux® 兼容模式便会在系统启动时自动开启：

```
linux_enable="YES"
```

想要在自制内核中静态链接 Linux® 二进制兼容支持的用户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 中所描述的方法编译并安装新内核。

11.2.1. 手动安装额外的库

在配置了 Linux® 兼容模式之后， 如果某个 Linux® 应用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 找出应用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 4.5pl26) => /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， 建立指向第一栏中文件名的符号链接。 这样在 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订号相同的 Linux® 共享库文件， 则不再需要复制最后那列文件， 现有的共享库应该可以正常使用。 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， 那么就可以删除旧版的了。

比如， FreeBSD 系统中现有这些共享库文件：

```
/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 4.5pl26) -> libc.so.4.6.29
```

既然现有文件最后的版本号只相差一到两个版本， 程序应该可以正常使用稍旧些的版本。 不管怎样， 使用新版本替换现有 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® 程序时需要寻找 Linux® 二进制程序所依赖的共享库文件。 在此之后， 系统里便会有足够多的 Linux® 共享库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 二进制程序有时需要额外的步骤。 当未被标记的 ELF 二进制程序被执行的时候，
会生成如下的错误信息：

```
% ./my-linux-elf-binary
ELF binary type not known
Abort
```

为了帮助 FreeBSD 内核分辨 FreeBSD ELF 二进制程序和 Linux® 二进制程序， 请使用 brandelf(1)：

```
% brandelf -t Linux my-linux-elf-binary
```

由于现在的 GNU 工具链能自动把适当的标记信息写入 ELF 二进制程序中，这个步骤通常不是必须做的。

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
```

如有必要的话使用 brandelf 标记安装好的 ELF 二进制程序。 注意此项安装将无法干净卸载。

11.2.4. 配置主机名解析器

如果 DNS 不能正常工作或是出现以下的错误信息：

```
resolv+: "bind" is an invalid keyword resolv+:
"hosts" is an invalid keyword
```

请参照此方法配置 /compat/linux/etc/host.conf：

```
order hosts, bind
multi on
```

这里指定了先查询 /etc/hosts 再查询 DNS。 如果 /compat/linux/etc/host.conf 不存在的话， Linux®
程序便会读取 /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. 高级主题

此章节将讲述是 Linux® 二进制兼容如何工作的， 内容基于 Terry Lambert tlambert@primenet.com
(Message ID: <199906020108.SAA07001@usr09.primenet.com>) 发表在 FreeBSD 闲聊邮件列表
的邮件。

FreeBSD 有一个叫 "execution class loader" 的抽象层。 它被嵌入进了 execve(2) 系统调用。

历史上 UNIX® 加载器会依靠查看魔数 （通常是文件的开头 4 至 8

个字节）来确认是否是系统已知的的二进制程序， 如果是的话， 就会调用二进制程序加载器。

如果它不是二进制类型的程序， execve(2) 调用会返回一个错误， shell 则会把它当作 shell 命令执行。
"不论当前是哪一种 shell" 都会默认做出此种假设。

随后， sh(1) 会检查开头的两个字符， 如果它们是 :\n， 那么就调用 csh(1)。

FreeBSD 有一份加载器列表而不是一个单一的加载器， 并能回退到 #! 加载器来运行 shell 解释器或者 shell 脚本。

为了支持 Linux® ABI， FreeBSD 看到了二进制 ELF 程序的魔数。 ELF 加载器会查找一个专用的 标记，
那是在 ELF 镜像中的一个注释部分， 此区域在 SVR4/Solaris™ ELF 二进制中并不存在。

要运行 Linux® 二进制程序， 必须先使用 brandelf(1) 命令 标记 为 Linux 类型：

```
# brandelf -t Linux file
```

当 ELF 加载器看到了 Linux 标记，便会替换 proc 结构中的一个指针。
所有的系统调用都通过此指针来索引。 除此以外， 进程被标记以便对 signal trampoline
代码的陷阱向量做特殊处理， 还有一些其他由 Linux® 内核模块来处理的（细微）修补。

Linux® 系统调用向量包含一个 sysent[] 记录的列表， 它的地址位于内核模块之中。

当一个系统调用被 Linux® 二进制程序调用时， 陷阱代码会把系统调用函数指针从 proc 解引用至 Linux®
而不是 FreeBSD 的系统调用入口。

Linux® 模式会动态地 reroots 查找。 这与 union 文件系统选项是等效的。 首先会试图在
/compat/linux/original-path 目录查找文件。 如果失败了， 就会在 /original-path 目录下查找。
这使得需要其它程序的程序得以运行。 例如，Linux® 工具链都可以在 Linux® ABI 的支持下运行。 也就是说
Linux® 二进制程序可以加载并执行 FreeBSD 二进制程序， 如果当前没有相应的 Linux® 二进制程序，
可以在 /compat/linux 目录树中放置一个 uname(1) 命令， 使 Linux® 程序不易察觉它们并没有运行在
Linux® 系统上。

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种底层功能在 FreeBSD
系统调用表的记录和 Linux® 系统调用表的记录是一样的： 文件系统操作， 虚拟内存操作， 信号发送， 和
System V IPC。 唯一的不同是 FreeBSD 会得到 FreeBSD 的 glue 功能， 而 Linux® 程序会得到 Linux® 的
glue 功能。 FreeBSD 的 glue 功能是静态链接入内核的， 而 Linux® 的 glue 功能可以静态链接，
或者通过内核模块访问。

严格说来其实并没有真正的模拟， 这是一种 ABI 的实现。 有时这被称为 "Linux® 模拟"
是因为在实现的时候还没有其他适合的词用来描述。 要说 FreeBSD 运行 Linux® 二进制程序并不确切，
因为当时代码并还没有被编译进去。

Part III: 系统管理

FreeBSD 手册中其余章节的内容都是关于系统管理。 每一章节都从描述这章将要介绍的内容开始，由浅入深对相关内容进行介绍。

这些章节在撰写时， 已经设计成了许多相互独立的部分， 如果您需要了解某部分内容，直接阅读这部分内容即可， 而无需按照顺序， 也不必在您开始使用 FreeBSD 之前完整地阅读它们。

Chapter 12. 设置和调整

12.1. 概述

使用 FreeBSD 的一个重要问题是系统配置。
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。 这章将解释一些 FreeBSD 的配置过程，包括一些可以调整的 FreeBSD 系统的一些参数。

读完本章， 您将了解：

- 如何有效地利用文件系统和交换分区。
- rc.conf 的基本设置以及 /usr/local/etc/rc.d 启动体系。
- 如何设置和测试网卡。
- 如何在您的网络设备上配置虚拟主机。
- 如何使用 /etc 下的各配置文件。
- 如何通过 sysctl 变量来对 FreeBSD 系统进行调优。
- 怎样调整磁盘性能和修改内核限制。

在阅读本章之前，您应该了解：

- 了解 UNIX® 和 FreeBSD 的基础知识 (UNIX 基础)。
- 熟悉内核配置编译的基础知识 (配置FreeBSD的内核)。

12.2. 初步配置

12.2.1. 分区规划

12.2.1.1. 基本分区

当使用 bsdlabel(8) 或者 sysinstall(8) 来分割您的文件系统的时候，
要记住硬盘驱动器外磁道传输数据要比从内磁道传输数据快。
因此应该将小的和经常访问的文件系统放在驱动器靠外的位置， 一些大的分区比如 /usr 应该放在磁盘比较靠里的位置。 以类似这样的顺序建立分区是一个不错的主意：root，swap， /var，/usr。

/var 分区的大小能反映您的机器使用情况。 /var 文件系统用来存储邮件， 日志文件和打印队列缓存，
特别是邮箱和日志文件可能会达到无法预料的大小，
这主要取决于在您的系统上有多少用户和您的日志文件可以保存多长时间。 大多数用户很少需要 /var 有 1GB 以上的闲置空间。

有时候 /var/tmp 需要很多的磁盘空间。 在使用 pkg_add(1) 安装新的软件时，包管理工具会在 /var/tmp 中解压出一份临时拷贝。 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 分区存储很多用来系统运行所需要的文件例如 ports(7) (建议这样做) 和源代码 (可选的)。 ports 和基本系统的源代码在安装时都是可选的， 但我们建议给这个分区至少保留 2GB 的可用空间。

当选择分区大小的时候，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，可能会导致出现一些错误。

一些用户会发现 sysinstall(8) 的 Auto-defaults 自动分区有时会分配给 /var 和 / 较小的分区空间。 分区应该精确一些并且大一些。

12.2.1.2. 交换分区

一般来讲，交换分区应该大约是系统内存 (RAM) 的两倍。 例如，如果机器有 128M 内存，交换文件应该是 256M。 较小内存的系统可以通过多一点地交换分区来提升性能。 不建议小于 256 兆的交换分区，并且扩充您的内存应该被考虑一下。 当交换分区最少是主内存的两倍的时候，内核的 VM (虚拟内存) 页面调度算法可以将性能调整到最好。如果您给机器添加更多内存， 配置太小的交换分区会导致 VM 页面扫描的代码效率低下。

在使用多块SCSI磁盘(或者不同控制器上的IDE磁盘)的大系统上，
建议在每个驱动器上建立交换分区(直到四个驱动器)。
交换分区应该大约一样大小。内核可以使用任意大小， 但内部数据结构则是最大交换分区的 4 倍。保持交换分区同样的大小， 可以允许内核最佳地调度交换空间来访问磁盘。
即使不太使用，分配大的交换分区也是好的，
在被迫重启之前它可以让您更容易的从一个失败的程序中恢复过来。

12.2.1.3. 为什么要分区?

一些用户认为一个单独的大分区将会很好， 但是有很多原因会证明为什么这是个坏主意。首先，
每个分区有不同的分区特性，因此分开可以让文件系统调整它们。 例如，根系统和 /usr 一般只是读取，写入很少。 很多读写频繁的被放在 /var 和 /var/tmp中。

适当的划分一个系统， 在其中使用较小的分区， 这样，
那些以写为主的分区将不会比以读为主的分区付出更高的代价。 将以写为主的分区放在靠近磁盘的边缘，
例如放在实际的大硬盘的前面代替放在分区表的后面，将会提高您需要的分区的 I/O 性能。现在可能也需要在比较大的分区上有很好的 I/O 性能，
把他们移动到磁盘外围不会带来多大的性能提升，反而把 /var 移到外面会有很好的效果。最后涉及到安全问题。
一个主要是只读的小的、整洁的根分区可以提高从一个严重的系统崩溃中恢复过来的机会。

12.3. 核心配置

系统的配置信息主要位于 /etc/rc.conf。
这个文件包含了配置信息很大的一部分，主要在系统启动的时候来配置系统，
这个名字直接说明了这点；它也是 rc* 文件的配置信息。

系统管理员应该在 rc.conf 文件中建立记录来覆盖 /etc/defaults/rc.conf 中的默认设置。
这个默认文件不应该被逐字的复制到 /etc —— 它包含的是默认值而不是一个例子。 所有特定的改变应该在 rc.conf 中。

在集群应用中，为了降低管理成本，
可以采用多种策略把涉及全站范围的设置从特定于系统的设置中分离出来。
推荐的方法是把系统范围的配置放到 /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 文件则保持不变。

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。
这使得系统管理员可以在其中添加一些逻辑， 从而创建能够适应非常复杂的场景的配置。
请参阅联机手册 rc.conf(5) 来了解关于这一话题的进一步信息。

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、语法等等。
从基本系统中分开他们是很重要的以至于他们可以容易的被 package 管理工具定位和管理

一般来说，这些文件被安装在 /usr/local/etc。这个例子中，
一个应用程序有很多配置文件并且创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， 配置文件示例也同样被安装了。它们通常用 .default 的后缀来标识。如果不存在这个应用程序的配置文件， 它们会通过复制 .default 文件来创建。

例如，看一下这个目下的内容 /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. 启动服务

许多用户会选择使用 Ports Collection 来在 FreeBSD 上安装第三方软件。
很多情况下这可能需要进行一些配置以便让这些软件能够在系统初始化的过程中启动。 服务， 例如 mail/postfix 或 www/apache13 就是这些需要在系统初始化时启动的软件包中的两个典型代表。
这一节解释了启动第三方软件所需要的步骤。

FreeBSD 包含的大多数服务，例如 cron(8)， 就是通过系统启动脚本启动的。 这些脚本也许会有些不同，
这取决于 FreeBSD 版本。 但是不管怎样，
需要考虑的一个重要方面是他们的启动配置文件要能被基本启动脚本识别捕获。

12.5.1. 扩展应用程序配置

现在 FreeBSD 提供了 rc.d， 这使得对应用软件的启动进行配置变得更加方便， 并提供了更多的其他功能。
例如， 使用在 rc.d 一节中所介绍的关键字， 应用程序就可以设置在某些其他服务， 例如 DNS 之后启动；
除此之外， 还可以通过 rc.conf 来指定一些额外的启动参数， 而不再需要将它们硬编码到启动脚本中。
基本的启动脚本如下所示：

```
#!/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 能够在 DAEMON 服务之后启动。 它同时也提供了设置和跟踪 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) 工具并提供更简单的通过 rc.conf 文件来配置的方法。

12.5.2. 用服务来启动服务

其他服务， 例如 POP3 服务器， IMAP， 等等， 也可以通过 inetd(8) 来启动。 这一过程包括从 Ports Collection 安装相应的应用程序， 并把配置加入到 /etc/inetd.conf 文件， 或去掉当前配置中的某些注释。 如何使用和配置 inetd 在 inetd 一节中进行了更为深入的阐述。

一些情况下， 通过 cron(8) 来启动系统服务也是一种可行的选择。 这种方法有很多好处， 因为 cron 会以 crontab 的文件属主身份执行那些进程。 这使得普通用户也能够执行他们的应用。

cron 工具提供了一个独有的功能， 以 @reboot 来指定时间。 这样的设置将在 cron(8) 启动时运行， 通常这也是系统初始化的时候。

12.6. 配置 cron

FreeBSD 最有用的软件包(utilities)中的一个是 cron(8)。 cron 软件在后台运行并且经常检查 /etc/crontab 文件。 cron 软件也检查 /var/cron/tabs 目录，搜索新的 crontab 文件。这些 crontab 文件存储一些 cron 在特定时间执行任务的信息。

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，第六个字段是用于执行命令的用户名。
这给予了系统 crontab 以任意用户身份执行命令的能力。 在用户 crontab 中，
第六个字段是要执行的命令， 所有的命令都会以这个用户自己的身份执行； 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， 通常并不需要给 root 建立单独的用户 crontab。

让我们来看一下 /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。 这样， 就可以编写注释来说明要执行什么操作，以及这样做的原因。 需要注意的是， 注释应该另起一行， 而不能跟命令放在同一行上，否则它们会被看成命令的一部分。 这个文件中的空行会被忽略。

② 首先应该定义环境变量。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 将会用默认值 sh。如果 PATH 变量被忽略，那么就没有默认值并且需要指定文件绝对位置。如果 HOME 被忽略，cron 将用用执行者的 home 目录。

③ 这一行定义了七个字段。它们是 minute、 hour、mday、 month、wday、 who 和 command。它们差不多已经说明了各自的用处。Minute 是命令要运行时的分钟，Hour 跟 minute 差不多，只是用小时来表示。Mday 是每个月的天。Month 跟 hour 还有 minute 都差不多，用月份来表示。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" 字段是特别的，并且只在 /etc/crontab 文件中存在。 这个字段指定了命令应该以哪个用户的身份来运行。当一个用户添加了他(她)的 crontab 文件的时候，他们就会没有这个字段选项。最后，是 command 字段。这是最后的一个字段，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说的值。注意这里我们有一个 /5 列表，紧跟着是一些 字符。* 字符代表"开始到最后"， 也可以被解释成 每次。所以，根据这行， 显然表明了无论在何时每隔 5 分钟以 root 身份来运行 atrun 命令。查看 atrun(8) 手册页以获得 atrun 的更多信息。命令可以有任意多个传递给它们的标志。无论怎样，扩展到多行的命令应该用反斜线("\")来续行。

这是每个 crontab 文件的基本设置， 虽然它们有一个不同。第六行我们指定的用户名只存在于系统 /etc/crontab 文件。这个字段在普通用户的 crontab 文件中应该被忽略。

12.6.1. 安装 Crontab

绝对不要用这种方法来编辑/安装系统 crontab。

> 您需要做的只是使用自己喜欢的编辑器： cron 程序会注意到文件发生了变化，并立即开始使用新的版本。参见 这个 FAQ 项目 以了解进一步的情况。

要安装刚写好的用户 crontab， 首先使用最习惯的编辑器来创建一个符合要求格式的文件，然后用 crontab 程序来完成。最常见的用法是：

```
% crontab crontab-file
```

在前面的例子中， crontab-file 是一个事先写好的 crontab。

还有一个选项用来列出安装的 crontab 文件： 只要传递 -l 选项给 crontab 然后看一下输出。

用户想不用模板(已经存在的文件)而直接安装他的 crontab 文件，用 crontab -e 选项也是可以的。它将会启动一个编辑器并且创建一个新文件，当这个文件被保存的时候， 它会自动的用 crontab 来安装这个文件。

如果您稍后想要彻底删除自己的用户 crontab 可以使用 crontab 的 -r 选项。

12.7. 在 FreeBSD 中使用 rc

在 2002 年， FreeBSD 整合了来自 NetBSD 的 rc.d 系统， 并通过它来完成系统的初始化工作。用户要注意在 /etc/rc.d 目录下的文件。 这里面的许多文件是用来管理基础服务的， 它们可以通过 start、stop， 以及 restart 选项来控制。 举例来说， sshd(8) 可以通过下面的命令来重启：

```
# /etc/rc.d/sshd restart
```

对其它服务的操作与此类似。 当然， 这些服务通常是在启动时根据 rc.conf(5) 自动启动的。 例如，要配置使系统启动时启动网络地址转换服务， 可以简单地通过在 /etc/rc.conf 中加入如下设置来完成：

```
natd_enable="YES"
```

如果 natd_enable="NO" 行已经存在， 只要简单的把 NO 改成 YES 即可。 rc 脚本在下次重新启动的时候会自动的装载所需要的服务， 像下面所描述的那样。

由于 rc.d 系统在系统启动/关闭时首先启动/停止服务，如果设置了适当的 /etc/rc.conf 变量，标准的 start、stop 和 restart 选项将会执行他们的动作。例如 sshd restart 命令只在 /etc/rc.conf 中的 sshd_enable 设置成 YES 的时候工作。不管是否在 /etc/rc.conf 中设置了，要 start、stop 或者 restart 一个服务，命令前可以加上一个"one"前缀。例如要不顾当前 /etc/rc.conf 的设置重新启动 sshd，执行下面的命令：

```
# /etc/rc.d/sshd onerestart
```

用选项 rcvar 可以简单来的检查 /etc/rc.conf 中用适当的 rc.d 脚本启动的服务是否被启用。从而管理员可以运行这样的程序来检查 sshd 是否真的在 /etc/rc.conf 中被启动了：

```
# /etc/rc.d/sshd rcvar
# sshd
$sshd_enable=YES
```

第二行 (# sshd) 是从 sshd 命令中输出的，而不是 root 控制台。

为了确定一个服务是否真的在运行，可以用 status 选项。例如验证 sshd 是否真的启动了：

```
# /etc/rc.d/sshd status
sshd is running as pid 433.
```

有些时候也可以 reload 服务。 这一操作实际上是向服务发送一个信号， 来强制其重新加载配置。
多数情况下， 发给服务的会是 SIGHUP 信号。 并非所有服务都支持这一功能。

rc.d 系统不仅用于网络服务， 它也为系统初始化中的多数过程提供支持。 比如 bgfsck 文件，
当它被执行时， 将会给出下述信息：

```
Starting background file system checks in 60 seconds.
```

这个文件用做后台文件系统检查，系统初始化的时候完成。

很多系统服务依赖其他服务提供的相应功能。例如，NIS 和其他基于 RPC 的服务启动可能在 rpcbind
服务启动之前失败。
要解决这个问题，依赖关系信息和其他头信息当作注释被包含在每个启动脚本文件的前面。
程序在系统初始化时分析这些注释以决定调用其他系统服务来满足依赖关系。

下面的字句必须被包含在所有的启动脚本文件里， （他们都是 rc.subr(8) 用来 "enable" 启动脚本必需的）：

- PROVIDE: 指定此文件所提供的服务的名字。

以下的字句可以被包含在启动文件的顶部。严格来说他们不是必需的， 但作为对于 rcorder(8)
有一定的提示作用：

- REQUIRE: 列出此服务启动之前所需要的其他服务。 此脚本提供的服务会在指定的那些服务 之后 启动。
- BEFORE: 列出依赖此服务的其他服务。 此脚本提供的服务将在指定的那些服务 之前 启动。

通过在启动脚本中仔细设定这些关键字， 系统管理员可以很有条理的控制脚本的启动顺序，
进而避免使用像其他 UNIX® 操作系统那样混乱的 "runlevels"。

更多关于 rc.d 系统的信息， 可以在 rc(8) 和 rc.subr(8) 联机手册中找到。 如果您有意撰写自己的 rc.d
脚本， 或对现有的脚本进行一些改进， 也可以参考 这篇文章。

12.8. 设置网卡

现在我们不可想象一台计算机没有网络连接的情况。 添加和配置一块网卡是任何 FreeBSD
系统管理员的一项基本任务。

12.8.1. 查找正确的驱动程序

在开始之前，您应该知道您的网卡类型，它用的芯片和它是 PCI 还是 ISA 网卡。FreeBSD 支持很多种 PCI 和
ISA 网卡。 可以查看您的版本硬件兼容性列表以确定您的网卡被支持。

确认系统能够支持您的网卡之后， 您还需要为它选择合适的驱动程序。 /usr/src/sys/conf/NOTES 和
/usr/src/sys/arch/conf/NOTES 将为您提供所支持的一些网卡和芯片组的信息。
如果您怀疑驱动程序是否使所要找的那一个， 请参考驱动程序的联机手册。
联机手册将提供关于所支持的硬件更详细的信息， 甚至还包括可能发生的问题。

如果您的网卡很常见的话， 大多数时候您不需要为驱动浪费精力。 常用的网卡在 GENERIC
内核中已经支持了， 所以您的网卡在启动时就会显示出来，像是：

```
dc0: <82c169 PNIC 10/100BaseTX> port 0xa000-0xa0ff mem 0xd3800000-0xd38
000ff irq 15 at device 11.0 on pci0
miibus0: <MII bus> on dc0
bmtphy0: <BCM5201 10/100baseTX PHY> PHY 1 on miibus0
bmtphy0:  10baseT, 10baseT-FDX, 100baseTX, 100baseTX-FDX, auto
dc0: Ethernet address: 00:a0:cc:da:da:da
dc0: [ITHREAD]
dc1: <82c169 PNIC 10/100BaseTX> port 0x9800-0x98ff mem 0xd3000000-0xd30
000ff irq 11 at device 12.0 on pci0
miibus1: <MII bus> on dc1
bmtphy1: <BCM5201 10/100baseTX PHY> PHY 1 on miibus1
bmtphy1:  10baseT, 10baseT-FDX, 100baseTX, 100baseTX-FDX, auto
dc1: Ethernet address: 00:a0:cc:da:da:db
dc1: [ITHREAD]
```

在这个例子中，我们看到有两块使用 dc(4) 驱动的网卡在系统中。

如果您的网卡没有出现在 GENERIC 中， 则需要手工加载合适的驱动程序。
要完成这项工作可以使用下面两种方法之一：

- 最简单的办法是用 kldload(8) 加载网卡对应的内核模块。 除此之外， 通过在 /boot/loader.conf 文件中加入适当的设置， 也可以让系统在引导时自动加载这些模块。 不过，
  并不是所有的网卡都能够通过这种方法提供支持； ISA 网卡是比较典型的例子。
- 另外， 您也可以将网卡的支持静态联编进内核。 察看 /usr/src/sys/conf/NOTES，
  /usr/src/sys/arch/conf/NOTES
  以及驱动程序的联机手册以了解需要在您的内核配置文件中加一些什么。
  要了解关于重新编译内核的进一步细节， 请参见 配置FreeBSD的内核。 如果您的卡在引导时可以被内核 (GENERIC) 识别， 您应该不需要编译新的内核。

12.8.1.1. 使用 Windows® NDIS 驱动程序

不幸的是， 许多厂商由于认为驱动程序会涉及许多敏感的商业机密，
至今仍不愿意将把驱动程序作为开放源代码形式发布列入他们的时间表。 因此， FreeBSD
和其他操作系统的开发者就只剩下了两种选择： 要么经历长时间的痛苦过程来对驱动进行逆向工程，
要么使用现存的为 Microsoft® Windows® 平台提供的预编译版本的驱动程序。 包括参与 FreeBSD
开发的绝大多数开发人员， 都选择了后一种方法。

得益于 Bill Paul (wpaul) 的工作， 已经可以 "直接地" 支持 网络驱动接口标准 (NDIS, Network Driver Interface Specification) 了。 FreeBSD NDISulator (也被称为 Project Evil) 可以支持二进制形式的 Windows® 驱动程序， 并让它相信正在运行的是 Windows®。 由于 ndis(4) 驱动使用的是用于 Windows® 的二进制形式的驱动， 因此它只能在 i386™ 和 amd64 系统上使用。

ndis(4) 驱动在设计时主要提供了 PCI、 CardBus 和 PCMCIA 设备的支持， 而 USB 设备目前则没有提供支持。

要使用 NDISulator， 您需要三件东西：

1. 内核的源代码
2. 二进制形式的 Windows® XP 驱动程序 (扩展名为 .SYS)
3. Windows® XP 驱动程序配置文件 (扩展名为 .INF)

您需要找到用于您的卡的这些文件。 一般而言， 这些文件可以在随卡附送的 CD 或制造商的网站上找到。

在下面的例子中， 我们用 W32DRIVER.SYS 和 W32DRIVER.INF 来表示这些文件。

不能在 FreeBSD/amd64 上使用 Windows®/i386 驱动程序。 必须使用 Windows®/amd64 驱动才能在其上正常工作。

接下来的步骤是将二进制形式的驱动程序组装成内核模块。 要完成这一任务， 需要以 root 用户的身份执行 ndisgen(8)：

```
# ndisgen /path/to/W32DRIVER.INF /path/to/W32DRIVER.SYS
```

ndisgen(8) 是一个交互式的程序， 它会提示您输入所需的一些其他的额外信息； 这些工作完成之后， 它会在当前目录生成一个内核模块文件， 这个文件可以通过下述命令来加载：

```
# kldload ./W32DRIVER_SYS.ko
```

除了刚刚生成的内核模块之外， 还必须加载 ndis.ko 和 if_ndis.ko 这两个内核模块， 在您加载需要 ndis(4) 的模块时， 通常系统会自动完成这一操作。 如果希望手工加载它们， 则可以使用下列命令：

```
# kldload ndis
# kldload if_ndis
```

第一个命令会加载 NDIS 袖珍端口驱动封装模块， 而第二条命令则加载实际的网络接口。

现在请查看 dmesg(8) 来了解是否发生了错误。 如果一切正常， 您会看到类似下面的输出：

```
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
```

这之后， 就可以像使用其它网络接口 (例如 dc0) 一样来使用 ndis0 设备了。

与任何其它模块一样， 您也可以配置系统， 令其在启动时自动加载 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. 配置网卡

现在正确的网卡驱动程序已经装载，那么就应该配置它了。 跟其他配置一样，网卡可以在安装时用 sysinstall 来配置。

要显示您系统上的网络接口的配置，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 0xffffff00 broadcast 192.168.1.255
        media: Ethernet autoselect (100baseTX <full-duplex>)
        status: active
dc1: flags=8802<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
        options=80008<VLAN_MTU,LINKSTATE>
        ether 00:a0:cc:da:da:db
        inet 10.0.0.1 netmask 0xffffff00 broadcast 10.0.0.255
        media: Ethernet 10baseT/UTP
        status: no carrier
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，显示出了下列设备：

- 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: 回环设备

FreeBSD 使用内核引导时检测到的网卡驱动顺序来命名网卡。例如 sis2 是系统中使用 sis(4) 驱动的第三块网卡。

在这个例子中，dc0 设备启用了。主要表现在：

1. UP 表示这块网卡已经配置完成准备工作。
2. 这块网卡有一个 Internet (inet) 地址 (这个例子中是 192.168.1.3)。
3. 它有一个有效的子网掩码 (netmask； 0xffffff00 等同于 255.255.255.0)。
4. 它有一个有效的广播地址 (这个例子中是 192.168.1.255)。
5. 网卡的 MAC (ether) 地址是 00:a0:cc:da:da:da
6. 物理传输媒介模式处于自动选择状态 (media: Ethernet autoselect (100baseTX <full-duplex>))。我们看到 dc1 被配置成运行在 10baseT/UTP 模式下。 要了解驱动媒介类型的更多信息，请查阅它们的使用手册。
7. 连接状态 (status)是 active，也就是说连接信号被检测到了。对于 dc1，我们看到 status: no carrier。这通常是网线没有插好。

如果 ifconfig(8) 的输出显示了类似于：

```
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，那么就是还没有配置网卡。

要配置网卡，您需要 root 权限。 网卡配置可以通过使用 ifconfig(8) 命令行方式来完成，
但是这样每次启动都要做一遍。放置网卡配置信息的文件是 /etc/rc.conf。

用您自己喜欢的编辑器打开 /etc/rc.conf。 并且您需要为每一块系统中存在的网卡添加一行，
在我们的例子中，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"
```

用自己正确的设备名和地址来替换例子中的 dc0，dc1 等内容。您应该应该查阅网卡驱动和 ifconfig(8)
的手册页来了解各选项，也要查看一下 rc.conf(5) 帮助页来了解 /etc/rc.conf 的语法。

如果在安装的时候配置了网络，关于网卡的一些行可能已经存在了。 所以在添加新行前仔细检查一下
/etc/rc.conf。

您也可能需要编辑 /etc/hosts 来添加局域网中不同的机器名称和 IP 地址， 如果它们不在那里的话。
请查看联机手册 hosts(5) 和 /usr/shared/examples/etc/hosts 以了解更多信息。

如果计划通过这台机器访问 Internet， 您还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. 测试和调试

对 /etc/rc.conf 做了必要的修改之后应该重启系统以应用对接口的修改，
并且确认系统重启后没有任何配置错误。 另外您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， 还需要运行下面的命令：

```
# /etc/rc.d/routing restart
```

网络系统重启之后， 应测试网络接口。

12.8.3.1. 测试以太网卡

为了确认网卡被正确的配置了，在这里我们要做两件事情。首先， ping 自己的网络接口，接着 ping
局域网内的其他机器。

首先测试本地接口：

```
% 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
```

现在我们应该 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
```

您如果您设置了 /etc/hosts 文件，也可以用机器名来替换 192.168.1.2。

12.8.3.2. 调试

调试硬件和软件配置一直是一件头痛的事情， 从最简单的开始可以减轻一些痛苦。
例如网线是否插好了？ 是否配置好了网络服务？ 防火墙配置正确吗？ 是否使用了被 FreeBSD 支持的网卡？
在发送错误报告之前您应该查看一下硬件说明， 升级 FreeBSD 到最新的 STABLE 版本，
看一下邮件列表或者在 Internet 上搜索一下。

如果网卡工作了， 但性能低下，应该好好阅读一下 tuning(7) 联机手册。 您也可以检查一下网络配置，
不正确的设置会导致慢速的网络连接。

一些用户可能会在一些网卡上经历一到两次 device timeouts， 这通常是正常现象。
如果经常这样甚至引起麻烦， 则应确定一下它跟其他设备没有冲突。 仔细检查网线连接，
或者换一块网卡。

有时用户会看到少量 watchdog timeout 错误。 这种情况要做的第一件事就是检查线缆连接。
一些网卡需要支持总线控制的 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)。
检查网卡和主板说明书来确定是不是这个问题。

No route to host 通常发生在如果系统不能发送一个路由到目的主机的包的时候。
这在没有指定默认路由或者网线没有插上时会发生。 检查 netstat -rn
的输出并确认有一个有效的路由能到达相应的主机。 如果没有，请查阅 高级网络应用。

ping: sendto: Permission denied 错误信息经常由防火墙的配置错误引起。 如果 ipfw
在内核中启用了但是没有定义规则， 那么默认的规则就是拒绝所有通讯，甚至 ping 请求！ 查阅 防火墙

以了解更多信息。

有时网卡性能低下或者低于平均水平， 这种情况最好把传输媒介模式从 autoselect 改变为正确的传输介质模式。 这通常对大多数硬件有用， 但可能不会解决所有人的问题。 接着，检查所有网络设置，并且阅读 tuning(7) 手册页。

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一样提供网络服务。 这通过在一个接口上绑定多个网络地址来实现。

一个特定的网络接口有一个"真实"的地址， 也可能有一些"别名"地址。这些别名通常用 /etc/rc.conf 中的记录来添加。

一个 fxp0 的别名记录类似于：

```
ifconfig_fxp0_alias0="inet xxx.xxx.xxx.xxx netmask xxx.xxx.xxx.xxx"
```

记住别名记录必须从 alias0 开始并且按顺序递增(例如 _alias1、 _alias2)。 配置程序将会停止在第一个缺少的数字的地方。

f计算别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。 任何在这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。

举例来说， 假设使用 fxp0 连接到两个网络， 分别是 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。 我们希望从 10.1.1.1 到 10.1.1.5 以及从 202.0.75.17 到 202.0.75.20 的地址能够互相访问。 如前所述， 只有两个网段中的第一个地址 (本例中， 10.0.1.1 和 202.0.75.17) 应使用真实的子网掩码； 其余的 (10.1.1.2 到 10.1.1.5 以及 202.0.75.18 到 202.0.75.20) 则必须配置为使用 255.255.255.255 作为子网掩码。

下面是根据上述描述所进行的 /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，这些包括：

/etc	一般的系统配置信息。这儿的数据是与特定系统相关的。
/etc/defaults	系统配置文件的默认版本。

/etc/mail	额外的 sendmail(8) 配置信息，其他 MTA 配置文件。
/etc/ppp	用于用户级和内核级 ppp 程序的配置。
/etc/namedb	named(8) 数据的默认位置。通常 named.conf 和区域文件存放在这里。
/usr/local/etc	被安装的应用程序配置文件。可以参考每个应用程序的子目录。
/usr/local/etc/rc.d	被安装程序的 启动/停止 脚本。
/var/db	特定系统自动产生的数据库文件，像 package 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 如何访问域名系统(DNS)。

resolv.conf 中最常见的记录是：

nameserver	按顺序要查询的名字服务器的 IP 地址，最多三个。
search	搜索机器名的列表。这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。

如果您在使用 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 早期使用的一个简单文本数据库。 它结合 DNS 和 NIS 提供名字到 IP 地址的映射。通过局域网连接的机器可以用这个简单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 名称的本地纪录以减轻需要从外部查询带来的负担。

```
# $FreeBSD$
#
#
# 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 用简单的格式：

```
[Internet address] [official hostname] [alias1] [alias2] ...
```

例如：

```
10.0.0.1 myRealHostname.example.com myRealHostname foobar1 foobar2
```

参考 hosts(5) 以获得更多信息。

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
哪种信息类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) 手册页以获得更多信息

12.10.3.2. newsyslog.conf

newsyslog.conf 是一个通常用 cron(8) 计划运行的 newsyslog(8) 程序的配置文件。 newsyslog(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 被移动到 logfile.0，logfile.0 被移动到 logfile.1 等等。另外，日志文件可以用 gzip(1) 来压缩，它们是这样的命名格式： logfile.0.gz，logfile.1.gz 等等。

newsyslog.conf 指出了哪个日志文件要被管理，要保留多少和它们什么时候被创建。
日志文件可以在它们达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) 手册页以获得更多信息。

12.10.4. sysctl.conf

sysctl.conf 和 rc.conf 这两个文件的风格很接近。 其中的配置均为 变量=值 这样的形式。
在这个文件中配置的值， 均会在系统进入多用户模式之后进行实际的修改操作。 需要注意的是，
并不是所有的变量都能够在多用户模式下修改。

如果希望关闭对收到致命的信号退出的进程进行记录， 并阻止普通用户看到其他用户的进程， 可以在
sysctl.conf 中进行下列配置：

```
# 不记录由于致命信号导致的进程退出 (例如信号 11，访问越界)
kern.logsigexit=0


# 阻止用户看到以其他用户 UID 身份执行的进程。
security.bsd.see_other_uids=0
```

12.11. 用 sysctl 进行调整

sysctl(8) 是一个允许您改变正在运行中的 FreeBSD 系统的接口。它包含一些 TCP/IP
堆栈和虚拟内存系统的高级选项， 这可以让有经验的管理员提高引人注目的系统性能。用 sysctl(8)
可以读取设置超过五百个系统变量。

基于这点，sysctl(8) 提供两个功能：读取和修改系统设置。

查看所有可读变量：

```
% sysctl -a
```

读一个指定的变量，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 这样的语法：

```
# 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')。

如果你想在每次机器启动时自动设置某些变量， 可将它们加入到文件 /etc/sysctl.conf
之中。更多信息，请参阅手册页 sysctl.conf(5) 及 sysctl.conf。

12.11.1. 只读的 sysctl(8)

有时可能会需要修改某些只读的 sysctl(8) 的值。 尽管有时不得不这样做，
但只有通过(重新)启动才能达到这样的目的。

例如一些膝上型电脑的 cardbus(4) 设备不会探测内存范围，并且产生看似于这样的错误：

```
cbb0: Could not map register memory
device_probe_and_attach: cbb0 attach returned 12
```

像上面的错误通常需要修改一些只读的 sysctl(8) 默认设置。要实现这点，用户可以在本地的
/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。

修复上面的问题用户需要在刚才所说的文件中设置 hw.pci.allow_unsupported_io_range=1。现在
cardbus(4) 就会正常的工作了。

12.12. 调整磁盘

12.12.1. Sysctl 变量

12.12.1.1. vfs.vmiodirenable

vfs.vmiodirenable sysctl 变量可以设置成0(关)或者1(开)；默认是1。
这个变量控制目录是否被系统缓存。大多数目录是小的，
在系统中只使用单个片断(典型的是1K)并且在缓存中使用的更小
(典型的是512字节)。当这个变量设置为关闭 (0) 时，
缓存器仅仅缓存固定数量的目录，即使您有很大的内存。 而将其开启 (设置为1) 时， 则允许缓存器用 VM
页面缓存来缓存这些目录，让所有可用内存来缓存目录。 不利的是最小的用来缓存目录的核心内存是大于
512 字节的物理页面大小(通常是 4k)。
我们建议如果您在运行任何操作大量文件的程序时保持这个选项打开的默认值。 这些服务包括 web
缓存，大容量邮件系统和新闻系统。 尽管可能会浪费一些内存，但打开这个选项通常不会降低性能。
但还是应该检验一下。

12.12.1.2. vfs.write_behind

vfs.write_behind sysctl 变量默认是 1 (打开)。 它告诉文件系统簇被收集满的时候把内容写进介质，
典型的是在写入大的连续的文件时。 主要的想法是， 如果可能对 I/O 性能会产生负面影响时，
应尽量避免让缓冲缓存被未同步缓冲区充满。
然而它可能降低处理速度并且在某些情况下您可能想要关闭它。

12.12.1.3. vfs.hirunningspace

vfs.hirunningspace sysctl 变量决定了在任何给定情况下， 有多少写 I/O
被排进队列以给系统的磁盘控制器。
默认值一般是足够的，但是对有很多磁盘的机器来说您可能需要把它设置成 4M 或
5M。注意这个设置成很高的值(超过缓存器的写极限)会导致坏的性能。
不要盲目的把它设置太高！高的数值会导致同时发生的读操作的迟延。

sysctl 中还有许多与 buffer cache 和 VM页面 cache 有关的值， 一般不推荐修改它们。
虚拟内存系统已经能够很好地进行自动调整了。

12.12.1.4. vm.swap_idle_enabled

vm.swap_idle_enabled sysctl
变量在有很多用户进入、离开系统和有很多空闲进程的大的多用户系统中很有用。
这些系统注重在空闲的内存中间产生连续压力的处理。通过 vm.swap_idle_threshold1 和
vm.swap_idle_threshold2 打开这个特性并且调整交换滞后
(在空闲时)允许您降低内存页中空闲进程的优先权，从而比正常的出页
(pageout)算法更快。这给出页守护进程带来了帮助。
除非您需要否则不要把这个选项打开，因为您所权衡的是更快地进入内存，
因而它会吃掉更多的交换和磁盘带宽。在小的系统上它会有决定性的效果，
但是在大的系统上它已经做了合适的页面调度这个选项允许 VM 系统容易的让全部的进程进出内存。

12.12.1.5. hw.ata.wc

FreeBSD 4.3 中默认将 IDE 的写缓存关掉了。 这会降低到 IDE 磁盘用于写入操作的带宽，
但我们认为这有助于避免硬盘厂商所引入的， 可能引致严重的数据不一致问题。 这类问题实际上是由于 IDE
硬盘就写操作完成这件事的不诚实导致的。 当启用了 IDE 写入缓存时， IDE
硬盘驱动器不但不会按顺序将数据写到盘上， 而且当磁盘承受重载时，
它甚至会自作主张地对推迟某些块的实际写操作。 这样一来， 在系统发生崩溃或掉电时，
就会导致严重的文件系统损坏。 基于这些考虑， 我们将 FreeBSD 的默认配置改成了更为安全的禁用 IDE
写入缓存。 然而不幸的是， 这样做导致了性能的大幅降低，
因此在后来的发行版中这个配置又改为默认启用了。 您可以通过观察 hw.ata.wc sysctl 变量，
来确认您的系统中所采用的默认值。 如果 IDE 写缓存被禁用， 您可以通过将内核变量设置为 1 来启用它。
这一操作必须在启动时通过 boot loader 来完成。 在内核启动之后尝试这么做是没有任何作用的。

要了解更多的信息，请查阅 ata(4)。

12.12.1.6. SCSI_DELAY (kern.cam.scsi_delay)

SCSI_DELAY 内核配置会缩短系统启动时间。 默认值在系统启动过程中有 15 秒的迟延时间，
这是一个足够多且可靠的值。把它减少到 5 通常也能工作(特别是现代的驱动器)。
您可以在系统引导时调整引导加载器变量 kern.cam.scsi_delay 来改变它。 需要注意的是，
此处使用的单位是 毫秒 而 不 是 秒 。

12.12.2. Soft Updates

tunefs(8) 程序能够用来很好的调整文件系统。 这个程序有很多不同的选项，但是现在只介绍 Soft Updates
的打开和关闭，这样做：

```
# tunefs -n enable /filesystem
# tunefs -n disable /filesystem
```

在文件系统被挂载之后不能用 tunefs(8) 来修改。打开 Soft Updates
的最佳时机是在单用户模式下任何分区被挂载前。

Soft Updates 极大地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。我们建议您在所有的文件系统上使用 Soft Updates。应该知道
Soft Updates 的两点：首先， Soft Updates 保证了崩溃后的文件系统完整性，但是很可能有几秒钟
(甚至一分钟！) 之前的数据没有写到物理磁盘。如果您的系统崩溃了您可能会丢失很多工作。

第二，SoftUpdates 推迟文件系统块的释放时间。如果在文件系统
(例如根文件系统)快满了的情况下对系统进行大规模的升级比如 make installworld，
可能会引起磁盘空间不足从而造成升级失败。

12.12.2.1. Soft Updates 的详细资料

有两种传统的方法来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
或者目录这些没有内容的数据)

从前，默认方法是同步更新这些元数据(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)在这之后以非同步形式写入。
这么做有利的一点是操作安全。如果更新时发生错误，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。 如果崩溃时找不到文件的数据块，fsck(8)
可以找到并且依靠把文件大小设置为 0 来修复文件系统。 另外，这么做既清楚又简单。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。 这包含它自己更新目录，inode
表和可能对文件分散的块的更新。 同样问题出现大的文件操作上(比如 tar -x)。

第二种方法是非同步元数据更新。这是 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。 也就是它们会混合在文件内容数据更新中。
这个方法的优点是不需要等待每个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。同样，
这个方法也是清楚且简单的，所以代码中的漏洞风险很小。
缺点是不能保证文件系统的状态一致性。如果更新大量元数据时失败
(例如掉电或者按了重启按钮)，文件系统会处在不可预知的状态。
系统再启动时没有机会检查文件系统的状态；inode
表更新的时候可能文件的数据块已经写入磁盘了但是相关联的目录没有，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损坏了，唯一的选择是使用 newfs(8)
并且从备份中恢复它。

这个问题通常的解决办法是使用 dirty region logging 或者 journaling
尽管它不是一贯的被使用并且有时候应用到其他的事务纪录中更好。
这种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。
过后他们将会被移动到正确的位置。因为纪录区很小，
磁盘上接近的区域磁头不需要移动很长的距离，所以这些比写同步快一些。
另外这个方法的复杂性有限，所以出现错误的机会也很少。缺点是元数据要写两次
(一次写到纪录区域，一次写到正确的区域)。正常情况下， 悲观的性能可能会发生。从另一方面来讲，
崩溃的时候所有未发生的元数据操作可以很快的在系统启动之后从记录中恢复过来。

Kirk McKusick，伯克利 FFS 的开发者，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")。这样的结果是，在繁重的元数据操作中，
如果先前的更新还在内存中没有被写进磁盘，后来的更新就会捕捉到。
所以所有的目录操作在写进磁盘的时候首先在内存中执行
(数据块按照它们的位置来排列，所以它们不会在元数据前被写入)。 如果系统崩溃了这将导致一个固定的
"日志回朔"： 所有不知如何写入磁盘的操作都像没有发生过一样。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标记例如块和 inodes。崩溃之后，
唯一的资源分配错误是一个实际是"空闲"的资源的资源被标记为"使用"。 fsck(8)
可以认出这种情况并且释放不再使用的资源。它对于忽略崩溃后用 mount -f
强制挂上的文件系统的错误状态是安全的。 为了释放可能没有使用的资源，fsck(8)
需要在过后的时间运行。一个主意是用 后台 fsck：系统启动的时候只有一个文件系统的 快照
被记录下来。fsck 可以在过后运行。所有文件系统可以在"有错误"的时候被挂接，
所以系统可以在多用户模式下启动。接着，后台 fsck
可以在所有文件系统需要的时候启动来释放可能没有使用的资源。 (尽管这样，不用 Soft Updates
的文件系统依然需要通常的 fsck。)

f它的优点是元数据操作几乎跟非同步一样快 (也就是比需要两次元数据写操作的 logging
更快)。缺点是代码的复杂性(意味着对于丢失用户敏感数据有更多的风险)
和高的内存使用量。另外它有些特点需要知道。崩溃之后， 文件系统状态会"落后"一些。同步的方法用 fsck
后在一些地方可能产生一些零字节的文件， 这些文件在用 Soft Updates 文件系统之后不会存在，
因为元数据和文件内容根本没有写进磁盘(可能发生在运行 rm
之后)。这可能在文件系统上安装大量数据时候引发问题， 没有足够的剩余空间来两次存储所有文件。

12.13. 调整内核限制

12.13.1. 文件/进程限制

12.13.1.1. kern.maxfiles

kern.maxfiles 可以根据系统的需要适当增减。 这个变量用于指定在系统中允许的文件描述符的最大数量。
当文件描述符表满的时候， file: table is full 会在系统消息缓冲区中反复出现， 您可以使用 dmesg
命令来观察这一现象。

每个打开的文件、 套接字和管道， 都会占用一个文件描述符。 在大型生产服务器上，
可能会轻易地用掉数千个文件描述符， 具体用量取决于服务的类型和并行启动的服务数量。

在早期版本的 FreeBSD 中， kern.maxfiles 的默认值， 是根据您内核配置文件中的 maxusers
选项计算的。 kern.maxfiles 这个数值， 会随 maxusers 成比例地增减。 当编译定制的内核时，
按照您系统的用途来修改这个值是个好主意。 这个数字同时还决定内核的许多预设的限制值。 有时，
尽管并不会真的有 256 个用户同时连接一台生产服务器， 但对于高负载的 web 服务器而言，
却可能需要与之类似的资源。

变量 kern.maxusers 会在系统启动时， 根据可用内存的尺寸进行计算， 在内核开始运行之后，
可以通过只读的 kern.maxusers sysctl 变量值来进行观察。 有些情况下，
可能会希望使用更大或更小一些的 kern.maxusers， 它可以以加载器变量的形式进行配置； 类似 64、 128
和 256 这样的值都并不罕见。 我们不推荐使用超过 256 的值， 除非您需要巨量的文件描述符； 根据
kern.maxusers 推算默认值的那些变量， 一般都可以在引导甚至运行时通过 /boot/loader.conf (请参见
loader.conf(5) 联机手册或 /boot/defaults/loader.conf 文件来获得相关的指导)
或这篇文档的其余部分所介绍的方式来调整。

在较早的版本中， 如果您明确地将 maxusers 设置为 0， 则系统会自动地根据硬件配置来确定这个值。。
在 FreeBSD 5.X 和更高版本中， maxusers 如果不指定的话， 就会取默认值 0。 如果希望自行管理
maxusers， 则应配置一个不低于 4 的值， 特别是使用 X Window System 或编译软件的时候。
这样做的原因是， maxusers 所决定的一个最为重要的表的尺寸会影响最大进程数， 这个数值将是 20 + 16
* maxusers。 因此如果将 maxusers 设置为 1， 您就只能同时运行 36 个进程， 这还包括了 18
个左右的系统引导时启动的进程， 以及 15 个左右的， 在您启动 X Window System 时所引发的进程。
即使是简单的任务， 如阅读联机手册， 也需要启动多至九个的进程， 用以过滤、 解压缩， 并显示它。 将
maxusers 设为 64 将允许您同时执行最多 1044 个进程， 这几乎足以满足任何需要了。 不过，
如果您看在启动其它程序， 或运行用以支持大量用户的服务 (例如 ftp.FreeBSD.org) 时， 看到令人担忧的
错误， 就应该提高这一数值， 并重新联编内核。

maxusers 并 不能 限制实际能够登录到您系统上来的用户的数量。
它的主要作用是根据您可能支持的用户数量来为一系列系统数据表设置合理的尺寸，
以便提供支持他们所需运行的进程资源。

12.13.1.2. kern.ipc.somaxconn

kern.ipc.somaxconn sysctl 变量 限制了接收新 TCP
连接侦听队列的大小。对于一个经常处理新连接的高负载 web服务环境来说，默认的 128 太小了。
大多数环境这个值建议增加到 1024 或者更多。 服务进程会自己限制侦听队列的大小(例如 sendmail(8) 或者
Apache)， 常常在它们的配置文件中有设置队列大小的选项。 大的侦听队列对防止拒绝服务
攻击也会有所帮助。

12.13.2. 网络限制

NMBCLUSTERS 内核配置选项指出了系统可用的网络Mbuf的数量。
一个高流量的服务器使用一个小数目的网络缓存会影响 FreeBSD 的性能。 每个 cluster
可能需要2K内存，所以一个1024的值需要在内核中给网络缓存保留2M内存。
可以用简单的方法计算出来需要多少网络缓存。 如果您有一个同时发生1000个以上连接的web服务器，
并且每个连接用掉16K接收和发送缓存， 就需要大概32M网络缓存来确保web服务器的工作。
一个好的简单计算方法是乘以2，所以2x32Mb/2Kb=64MB/2kb=32768。
我们建议在有大量内存的机器上把这个值设置在4096到32768之间。
没有必要把它设置成任意太高的值，它会在启动时引起崩溃。 netstat(1) 的 -m
选项可以用来观察网络cluster使用情况。

kern.ipc.nmbclusters 可以用来在启动时刻调节这个。 仅仅在旧版本的 FreeBSD 需要使用 NMBCLUSTERS config(8) 选项。

经常使用 sendfile(2) 系统调用的繁忙的服务器， 有必要通过 NSFBUFS 内核选项或者在 /boot/loader.conf (查看 loader(8) 以获得更多细节) 中设置它的值来调节 sendfile(2) 缓存数量。
这个参数需要调节的普通原因是在进程中看到 sfbufa 状态。sysctl kern.ipc.nsfbufs
变量在内核配置变量中是只读的。 这个参数是由 kern.maxusers 决定的，然而它可能有必要因此而调整。

即使一个套接字被标记成非阻塞，在这个非阻塞的套接字上呼叫 sendfile(2) 可能导致 sendfile(2) 呼叫阻塞直到有足够的 struct sf_buf 可用。

12.13.2.1. net.inet.ip.portrange.*

net.inet.ip.portrange.* sysctl 变量自动的控制绑定在 TCP 和 UDP 套接字上的端口范围。
这里有三个范围：一个低端范围，一个默认范围和一个高端范围。 大多数网络程序分别使用由 net.inet.ip.portrange.first 和 net.inet.ip.portrange.last 控制的从 1024 到 5000 的默认范围。端口范围用作对外连接，并且某些情况可能用完系统的端口， 这经常发生在运行一个高负荷 web 代理服务器的时候。 这个端口范围不是用来限制主要的例如 web 服务器进入连接或者有固定端口例如邮件传递对外连接的。 有时您可能用完了端口，那就建议适当的增加 net.inet.ip.portrange.last。 10000、20000 或者 30000 可能是适当的值。
更改端口范围的时候也要考虑到防火墙。 一些防火墙会阻止端口的大部分范围 (通常是低范围的端口)并且用高端口进行对外连接(-)。 基于这个问题建议不要把 net.inet.ip.portrange.first 设的太小。

12.13.2.2. TCP 带宽迟延(Bandwidth Delay Product)

限制 TCP 带宽延迟积和 NetBSD 的 TCP/Vegas 类似。 它可以通过将 sysctl 变量 net.inet.tcp.inflight.enable 设置成 1 来启用。 系统将尝试计算每一个连接的带宽延迟积，并将排队的数据量限制在恰好能保持最优吞吐量的水平上。

这一特性在您的服务器同时向使用普通调制解调器， 千兆以太网， 乃至更高速度的光与网络连接 (或其他带宽延迟积很大的连接) 的时候尤为重要， 特别是当您同时使用滑动窗缩放， 或使用了大的发送窗口的时候。 如果启用了这个选项， 您还应该把 net.inet.tcp.inflight.debug 设置为 0 (禁用调试)， 对于生产环境而言， 将 net.inet.tcp.inflight.min 设置成至少 6144 会很有好处。 然而， 需要注意的是， 这个值设置过大事实上相当于禁用了连接带宽延迟积限制功能。
这个限制特性减少了在路由和交换包队列的堵塞数据数量，
也减少了在本地主机接口队列阻塞的数据的数量。在少数的等候队列中、
交互式连接，尤其是通过慢速的调制解调器，也能用低的 往返时间操作。但是，注意这只影响到数据发送 (上载/服务端)。对数据接收(下载)没有效果。

调整 net.inet.tcp.inflight.stab 是 不 推荐的。 这个参数的默认值是 20， 表示把 2 个最大包加入到带宽延迟积窗口的计算中。 额外的窗口似的算法更为稳定，
并改善对于多变网络环境的相应能力， 但也会导致慢速连接下的 ping 时间增长 (尽管还是会比没有使用 inflight 算法低许多)。 对于这些情形， 您可能会希望把这个参数减少到 15， 10， 或 5；
并可能因此而不得不减少 net.inet.tcp.inflight.min (比如说， 3500) 来得到希望的效果。
减少这些参数的值， 只应作为最后不得已时的手段来使用。

12.13.3. 虚拟内存

12.13.3.1. kern.maxvnodes

vnode 是对文件或目录的一种内部表达。 因此， 增加可以被操作系统利用的 vnode 数量将降低磁盘的 I/O。 一般而言， 这是由操作系统自行完成的， 也不需要加以修改。 但在某些时候磁盘 I/O 会成为瓶颈，而系统的 vnode 不足， 则这一配置应被增加。 此时需要考虑是非活跃和空闲内存的数量。

要查看当前在用的 vnode 数量：

```
# sysctl vfs.numvnodes
vfs.numvnodes: 91349
```

要查看最大可用的 vnode 数量：

```
# sysctl kern.maxvnodes
kern.maxvnodes: 100000
```

如果当前的 vnode 用量接近最大值， 则将 kern.maxvnodes 值增大 1,000 可能是个好主意。 您应继续查看 vfs.numvnodes 的数值， 如果它再次攀升到接近最大值的程度， 仍需继续提高 kern.maxvnodes。 在 top(1) 中显示的内存用量应有显著变化， 更多内存会处于活跃 (active) 状态。

12.14. 添加交换空间

不管您计划得如何好，有时候系统并不像您所期待的那样运行。
如果您发现需要更多的交换空间，添加它很简单。
有三种方法增加交换空间：添加一块新的硬盘驱动器、通过 NFS
使用交换空间和在一个现有的分区上创建一个交换文件。

要了解关于如何加密交换区， 相关配置， 以及为什么要这样做， 请参阅手册的 对交换区进行加密。

12.14.1. 在新的硬盘驱动器上使用交换空间

这是添加交换空间最好的方法， 当然为了达到这个目的需要添加一块硬盘。
毕竟您总是可以使用另一块磁盘。如果能这么做， 重新阅读一下手册中关于交换空间的 初步配置
来了解如何最优地安排交换空间。

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时， 否则不推荐您使用 NFS 来作为交换空间使用。 NFS
交换会受到可用网络带宽限制并且增加 NFS 服务器的负担。

12.14.3. 交换文件

您可以创建一个指定大小的文件用来当作交换文件。 在我们的例子中我们将会使用叫做 /usr/swap0 的
64MB 大小的文件。当然您也可以使用任何您所希望的名字。

例 9. 在 FreeBSD 中创建交换文件

1. 确认您的内核配置包含虚拟磁盘(Memory disk)驱动 (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. 赋予它(/usr/swap0)一个适当的权限：

   ```
   # chmod 0600 /usr/swap0
   ```

4. 在 /etc/rc.conf 中启用交换文件：

```
swapfile="/usr/swap0"   # Set to name of swapfile if aux swapfile desired.
```

5. 通过重新启动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. 电源和资源管理

BIOS 接口管理，例如可插拔 BIOS (PNPBIOS)或者高级电源管理(APM)
等等。电源和资源管理是现代操作系统的关键组成部分。
例如您可能当系统温度过高的时候让您的操作系统能监视到 (并且可能提醒您)。

以有效的方式利用硬件资源是非常重要的。 在引入 ACPI 之前，
管理电源使用和系统散热对操作系统是很困难的。 硬件由 BIOS 进行管理，
因而用户对电源管理配置的控制和查看都比较困难。 一些系统通过 高级电源管理 (APM)
提供了有限的配置能力。 电源和资源管理是现代操作系统的一个关键组件。 例如，
您可能希望操作系统监视系统的一些限制， 例如系统的温度是否超出了预期的增长速度
(并在需要时发出警告)。

在 FreeBSD 使用手册的这一章节，我们将提供 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资源和电源管理接口 (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)，也就是说， 它给操作系统(OS)提供了更多的控制和弹性。 在引入 ACPI 之前，
现代操作系统使得目前即插即用接口的局限性更加 "凸现" 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 驱动作为中介来访问 APM 软件接口，
从而实现对电源使用的管理。 在 2000 年或更早的时期生产的计算机系统， 仍需要使用 APM。

APM 有四个主要的问题。 首先， 电源管理是通过 (制造商专属的) BIOS 实现的， 而 OS
则完全不了解其细节。 例如， 用户在 APM BIOS 中设置了硬盘驱动器的空闲等待数值，
当超过这一空闲时间的限制时， 它 (BIOS) 将会减慢硬盘驱动器的速度， 而不会征求 OS 的同意。 第二，
APM 逻辑是嵌入 BIOS 的， 因此它是在 OS 的控制之外运转的。 这意味着用户只能通过通过刷新他们 ROM
中的 APM BIOS 才能够解决某些问题； 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，
也就是说有很多第三方的 (重复的工作) 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。 最后但绝不是最小的问题， APM BIOS
没有为实现复杂的电源策略提供足够的余地， 也无法实现能够非常适合具体机器的策略。

即插即用 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 模拟才能够与 PNPBIOS 的方法 "接口"。

FreeBSD APM 驱动在 apm(4) 手册页中有描述。

12.15.3. 配置 ACPI

默认情况下， acpi.ko 驱动， 会在系统引导时由 loader(8) 加载， 而 不应 直接联编进内核。
这样做的原因是模块操作起来更方便， 例如， 无需重新联编内核就可以切换到另一个 acpi.ko 版本。
这样可以让测试变得更简单一些。 另一个原因是， 许多时候在启动已经启动之后再启动 ACPI
可能会有些问题。 如果您遇到了问题， 可以全面禁用 ACPI。 这个驱动不应， 目前也无法卸载，

因为系统总线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) 提示符处配置 hint.acpi.0.disabled="1" 来禁用。

ACPI 和 APM 不能共存， 相反， 它们应分开使用。
后加载的驱动如果发现系统中已经执行了其中的一个， 便会停止执行。

ACPI 可以用来让系统进入休眠模式， 方法是使用 acpiconf(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 则会让系统执行与下列命令效果类似的软关机：

```
# halt -p
```

除此之外， 还有一些通过 sysctl(8) 提供的选项。 请参见联机手册 acpi(4) 和 acpiconf(8) 以获得更多信息。

12.16. 使用和调试 FreeBSD ACPI

ACPI 是一种全新的发现设备、 管理电源使用、 以及提供过去由 BIOS 管理的访问不同硬件的标准化方法。
让 ACPI 在各种系统上都能正确使用的工作一直在进行， 但许多主板的 ACPI 机器语言 (AML) 字节代码中的 bug， FreeBSD 的内核中子系统设计的不完善， 以及 Intel® ACPI-CA 解释器中的 bug 仍然不时会出现。

这份文档期望能够帮助您协助 FreeBSD ACPI 的维护人员来找到您所观察到的问题的根源，
并通过调试找到其解决方法。 感谢您阅读这份文档， 我们也希望能够解决您的系统上的问题。

12.16.1. 提交调试信息

在提交问题之前， 请确认您已经在运行最新的 BIOS 版本， 此外，
也包括嵌入式控制器的固件版本。

如果您希望提交一个问题， 请确保将下述信息发到 freebsd-acpi@FreeBSD.org:

- 问题行为的描述， 包括系统类型、型号，以及任何触发问题的相关信息。 另外，
  请注意尽可能准确地描述这一问题是否对您是陌生的。
- 在 "boot -v" 之后得到的 dmesg(8) 输出， 以及任何在重现 bug 时出现的错误信息。
- 在禁用了 ACPI 之后的 "boot -v" 的 dmesg(8) 输出， 如果您发现禁用 ACPI 能够帮助消除问题。
- 来自 fsysctl hw.acpi的输出。 这也是找到您的系统所提供的功能的一种好办法。
- 能够得到您的 ACPI Source Language (ASL) 的 URL。 不要 把 ASL 直接发到邮件列表中，
  因为它们可能非常大。 为了得到 ASL 您可以运行这个命令：

  ```
  # acpidump -dt > name-system.asl
  ```

  (把 name 改为您的登录名， 并把 system 改为您的硬件制造商及其型号。 例如： njl-FooCo6000.asl)

许多开发者也会订阅 FreeBSD-CURRENT 邮件列表 但还是请发到 FreeBSD ACPI 邮件列表
这样它会被更多人看到。 请耐心等待， 因为我们都有全职的其他工作。 如果您的 bug 不是显而易见的，
我们可能会要求您通过 send-pr(1) 来提交一个 PR。 在输入 PR 时，请将同样的信息包含进去。
这将帮助我们来追踪和解决问题。 不要在给 FreeBSD ACPI 邮件列表 写信之前发送 PR
因为我们把它当作已知文体的备忘录而不是报告机制。 您的问题很可能已经被其他人报告过了。

12.16.2. 背景

ACPI 存在于采用 ia32 (x86)、 ia64 (安腾)、 以及 amd64 (AMD) 架构的所有现代计算机上。
完整的标准具有大量的各式功能， 包括 CPU 性能管理、 电源控制、 温度监控、 电池系统、
嵌入式控制器以及总线枚举。 绝大多数系统实现比完整标准的功能要少一些。 例如，
桌面系统通常只实现总线枚举部分， 而笔记本则通常支持降温和电源管理功能。
笔记本通常还提供休眠和唤醒支持， 并提供与此适应的复杂功能。

符合 ACPI 的系统中有许多组件。 BIOS 和芯片组制造商提供一些固定的表 (例如， FADT) 在存储器中，
以提供类似 APIC 映射 (用于 SMP)、 配置寄存器、 以及简单的配置值等等。 另外， 一个字节代码
(bytecode) 表 (系统区别描述表DSDT) 则提供了通过树状命名空间来指定设备及其功能的方法。

ACPI 驱动必须要处理固定表， 实现字节码解释器， 并修改驱动程序和内核， 以接受来自 ACPI
子系统的信息。 对于 FreeBSD， Intel® 提供了一个解释器 (ACPI-CA)， 它在 Linux 和 NetBSD
也可以使用。 ACPI-CA 源代码可以在 src/sys/contrib/dev/acpica 找到。 用于在 FreeBSD 中允许 ACPI-CA
正确运转的代码则在 src/sys/dev/acpica/Osd。 最后， 用于实现 ACPI 设备的驱动可以在
src/sys/dev/acpica 找到。

12.16.3. 常见问题

要让 ACPI 正常工作， 它的每一部分都必须工作正常。 下面是一些常见的问题， 按照出新的频繁程度排序，
并给出了一些绕过或修正它们的方法。

12.16.3.1. 鼠标问题

某些时候， 唤醒操作会导致鼠标不再正常工作。 已知的绕过这一问题的方法， 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，
请考虑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， 以及一个休眠到磁盘的状态 (STD)， 称作 S4。 S5 是
"软关机" 同时也是系统接好电源但没有开机时的正常状态。 S4 实际上可以用两种不同的方法来实现。
S4BIOS 是一种由 BIOS 辅助的挂起到磁盘方法， 而 S4OS 则是完全由操作系统实现的。

可以使用 sysctl hw.acpi 来查看与休眠有关的项目。 这里是我的 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 来测试 S3， S4OS， 以及 S5。 如果 s4bios 是一 (1)， 则可以使用 S4BIOS
来代替 S4OS。

当测试休眠/唤醒时， 从 S1 开始， 如果它被支持的话。 这个状态是最可能正常工作的状态，
因为它不需要太多的驱动支持。 没有人实现 S2 但如果您有它的支持， 则应该和 S1 类似。
下一件值得尝试的是 S3。 这是最深的 STR 状态，
并需要一系列驱动的支持才能够正常地重新初始化您的硬件。 如果您在唤醒系统时遇到问题，
请不要吝惜发邮件给 FreeBSD ACPI 邮件列表 邮件列表， 尽管不要指望问题一定会很快解决，
因为有许多驱动程序/硬件需要进行更多的测试和改进。

休眠和唤醒操作最常见的问题是某些设备驱动程序不会保存、 恢复或正确地重新初始化其固件、
寄存器或设备内存。 尝试调试这些问题时， 首先可以尝试：

```
# sysctl debug.bootverbose=1
# sysctl debug.acpi.suspend_bounce=1
# acpiconf -s 3
```

这个测试会模拟休眠和恢复过程而不真的进入 S3 状态。 有时， 您会用这种方式很容易地抓住问题 (例如，
丢失固件状态、 设备 watchdog 超时， 以及一直重试等)。 注意系统不会真的进入 S3 状态，
这意味着这些设备可能不会掉电， 而许多设备在完全不提供休眠和恢复方法时仍可正常工作，
而不像使用真的 S3 状态那样。

较难的情况则需要更多的硬件， 例如用于串口控制台的串口/线， 以及用于 dcons(4)
的火线口/线和内核调试技能。

为了帮助隔离问题， 请在内核中删去尽可能多的驱动。 如果这样做能够解决问题，
请尝试逐个加载驱动直到问题再次出现。 通常预编译的驱动程序如 nvidia.ko、 X11 显示驱动， 以及 USB
的问题最多， 而以太网卡的驱动则通常工作的很好。 如果您能够通过加载和卸载驱动使系统正常工作，
您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。
在这两个文件中有一个注释掉的卸载和加载驱动程序的例子供您参考。 另外您还可以将
hw.acpi.reset_video 设置为零 (0)， 如果您的显示在唤醒之后显得很混乱。
此外您还可以尝试更长或更短的 hw.acpi.sleep_delay 值看看是否有所助益。

另一件值得一试的事情是使用一个比较新的包含 ACPI 支持的 Linux 发行版来试试看他们的 休眠/唤醒
功能是否在同样的硬件上能够正常工作。 如果在 Linux 下正常， 则很可能是 FreeBSD 驱动程序的问题，
而隔离问题并找到存在问题的驱动有助于解决它。 需要注意的是 ACPI 的维护人员通常并不维护其他驱动
(例如 声音、 ATA， 等等) 因此如果最终发现是驱动的问题最好还是发到 FreeBSD-CURRENT 邮件列表
邮件列表并发给驱动程序的维护者。 如果您喜欢冒险， 则可以加一些 printf(3) 到有问题的驱动中，
以找到它的恢复功能发生问题的位置。

最后， 试试看禁用 ACPI 并代之以启用 APM。 如果 休眠/唤醒 能够在 APM 下正常工作， 使用 APM
可能会更好， 特别是对于较老的硬件 (2000年以前)。 硬件制造商需要一些时间来让老硬件的 ACPI
工作正常， 而 ACPI 的问题十之八九是 BIOS 中的毛病引发的。

12.16.3.3. 系统停止响应 (暂时或永久性地)

绝大多数系统停止响应是由于未能及时响应中断或发生了中断风暴导致的。 芯片组有很多问题最终会溯源到
BIOS 如何在引导系统之前配置中断， APIC (MADT) 表的正确性， 以及 系统控制中断 (SCI) 如何路由。

通过察看 vmstat -i 的输出中包括 acpi0 的那一行可以区分中断风暴和未能及时响应中断。
如果每秒计数器增长的速度多于一两个， 则您是遇到了中断风暴。 如果系统停止了响应，
您可以尝试停止内核并进入 DDB (在控制台上按 CTRL + ALT + ESC) 并输入 show interrupts。

处理中断问题的救命稻草是尝试禁用 APIC 支持， 这是通过在 loader.conf 中加入
hint.apic.0.disabled="1" 完成的。

12.16.3.4. 崩溃

崩溃对于 ACPI 是比较罕见的情况， 如果发现， 我们将会非常重视并很快修复它。
您要做的第一件事是设法隔离出能够重现崩溃 (如果可能的话) 的操作并获取一份调用堆栈。 请启用 options
DDB 并设置串行控制台 (参见 通过串口线进入DDB调试器) 或配置一个 dump(8) 分区。 您将在 DDB 中通过
tr 得到调用堆栈。 如果您只能用手抄的方法记录它， 一定要记下头五 (5) 行和最后五 (5) 行。

然后， 尝试通过在启动时禁用 ACPI 来隔离故障。 如果这样做能够正常工作， 请通过设置
debug.acpi.disable 的那组数值来隔离具体是哪个 ACPI 子系统的问题。 请参见 acpi(4)
联机手册中给出的那些例子。

12.16.3.5. 系统在休眠或关机之后又启动了

首先请尝试在 loader.conf(5) 中设置 hw.acpi.disable_on_poweroff="0"。 这将让 ACPI
不再在关机过程中禁用一些事件。 基于同样的原因， 一些系统需要把这个值设置为 "1" (这是默认值)。
这通常能够修复在休眠或关机时立即再次启动的问题。

12.16.3.6. 其他问题

如果您有 ACPI 的其他问题 (同 docking station 协同工作、 无法检测设备， 等等)，
请把描述发给邮件列表； 不过， 这些问题也有可能和 ACPI 中尚未完成的部分有关，
它们可能需要时间才能被实现。 请给点耐心， 并准备测试我们可能会发给您的补丁。

12.16.4. ASL、acpidump， 以及 IASL

最常见的问题是 BIOS 制造商提供的不正确 (甚至完全错误的!) 字节代码。
这通常会以类似下面这样的内核消息显示在控制台上：

```
ACPI-1287: *** Error: Method execution failed [\\_SB_.PCI0.LPC0.FIGD._STA] \\
```

```
(Node 0xc3f6d160), AE_NOT_FOUND
```

许多时候， 您可以通过将 BIOS 升级到最新版本来解决此类问题。 绝大多数控制台消息是无害的，但如果您有其他问题例如电池工作不正常， 则从 AML 开始查找问题将是一条捷径。 字节代码， 或常说的 AML， 是从一种叫做 ASL 的语言写成的源代码进行编译得到的结果。 AML 一般存放在 DSDT 表中。要得到您系统的 ASL， 需要使用 acpidump(8)。 需要同时指定 -t (显示固定标的内容) 和 -d (将 AML 反编译成 ASL) 两个选项。 请参见 如何提交调试信息 一节了解如何使用它。

最方便的初步检查是尝试重新编译 ASL 来看看是否有错误。 通常可以忽略这一过程中产生的警告，但错误一般就都是 bug， 它们通常就是导致 ACPI 无法正常工作的原因。 要重新编译您的 ASL，可以使用下面的命令：

```
# iasl your.asl
```

12.16.5. 修复 ASL

我们的长期目标是让每一个人都能够在不需要任何用户干预的情况下使用 ACPI。 然而，目前我们仍然在开发绕过 BIOS 制造商常见错误的方法。 Microsoft® 解释器 (acpi.sys 和 acpiec.sys) 并不会严格地检查是否遵守了标准， 因此许多只在 Windows® 中测试 ACPI 的 BIOS 制造商很可能永远不会修正他们的 ASL。 我们希望不断地找出并用文档说明 Microsoft® 的解释器到底允许那些不标准的行为， 并在 FreeBSD 进行对应的修改使它能够正常工作而不需要用户修正 ASL。 作为一项临时缓解问题的方法， 并帮助我们确认其行为， 您可以手工修正 ASL。如果这样能够解决问题， 请把新旧 ASL 的 diff(1) 发给我们， 这样我们就有可能绕过 ACPI-CA 中的错误行为， 从而不再需要您来手工修正。

下面是一些常见的错误信息， 它们的原因， 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。 您可以让 FreeBSD 声称自己是任意 OS 来看一看是否能够修正问题。 比较简单的办法是设置 hw.acpi.osname="Windows 2001" 到 /boot/loader.conf 中， 或使用您在 ASL 中找到的其他字符串。

12.16.5.2. Missing Return statements (缺少返回语句)

一些方法可能没按照标准要求的那样显式地返回值。 尽管 ACPI-CA 无法处理它， 但 FreeBSD 提供了一个绕过它并允许其暗含地返回值的方法。 您也可以增加一个显式的 Return 语句，如果您知道那里需要返回一个值的话。 要强制 iasl 编译 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， 您可以通过下面的命令编译它：

```
# iasl your.asl
```

可以使用 -f 标志来强制创建 AML， 即使在编译过程中发生了错误。 请注意某些错误 (例如， 缺少 Return 语句) 会自动被解释器忽略掉。

DSDT.aml 是 iasl 命令的默认输出文件名。 可以加载它来取代您 BIOS 中存在问题的副本 (它仍然存在于闪存中)， 其方法是按下面的说明编辑 /boot/loader.conf：

```
acpi_dsdt_load="YES"
acpi_dsdt_name="/boot/DSDT.aml"
```

一定要把您的 DSDT.aml 复制到 /boot 目录中。

12.16.6. 从 ACPI 中获取调试输出信息

ACPI 驱动程序提供了非常灵活的调试机制。 这允许您指定一组子系统， 以及所需要的详细信息。
需要调试的子系统可以按 "layers(层)" 来指定， 并分为 ACPI-CA 组件 (ACPI_ALL_COMPONENTS) 和 ACPI 硬件支持 (ACPI_ALL_DRIVERS)。 调试输出的详细程度可以通过 "level(详细度)" 来指定， 其范围是 ACPI_LV_ERROR (只报告错误) 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选项， 中间用空格分开。
实际使用中您应该考虑使用串行控制台来记录输出， 如果它太长以至于冲掉了控制台消息缓冲的话。
不同的层和输出详细度的完整列表可以在 acpi(4) 联机手册中找到。

调试输出默认并不开启。 要起用它， 您需要在内核设置中添加 options ACPI_DEBUG，
如果您的内核中编入了 ACPI 的话。 您还可以在 /etc/make.conf 中加入 ACPI_DEBUG=1 来在全局起用它。
如果它只是模块， 您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 并把所需的详细度和层在 loader.conf 中指定。
这个例子将启用所有 ACPI-CA 组件以及所有 ACPI 硬件驱动 (CPU、 LID， 等等) 的消息。 只输出错误信息，
也就是最低的详细度。

```
debug.acpi.layer="ACPI_ALL_COMPONENTS ACPI_ALL_DRIVERS"
debug.acpi.level="ACPI_LV_ERROR"
```

如果您需要的信息是由某个特定的事件触发的 (比如说， 休眠之后的唤醒)， 您可以不修改 loader.conf
而转而使用 sysctl 来在启动和为那个事件准备系统之后再指定层和详细度。 这些 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找到：

- The FreeBSD ACPI 邮件列表
- ACPI 邮件列表存档 http://lists.freebsd.org/pipermail/freebsd-acpi/
- 旧的 ACPI 邮件列表存档 http://home.jp.FreeBSD.org/mail-list/acpi-jp/
- The ACPI 标准
- FreeBSD 手册页: acpi(4), acpi_thermal(4), acpidump(8), iasl(8), acpidb(8)
- DSDT 调试资源. (使用 Compaq 作为例子但通常情况下都很有用。)

Chapter 13. FreeBSD 引导过程

13.1. 概述

启动电脑以及加载操作系统的过程被称为"引导过程"， 或者简称为"引导"。 FreeBSD
的引导过程给用户自定义启动提供了很大的伸缩性，
您可以选择启动不同的操作系统，或者是同一系统的不同版本及内核。

本章将详细介绍您能在 FreeBSD 引导过程中设置的配置选项。 这包括了引导内核、探测设备并启动 init(8)
等等之前所发生的所有事情。 这些事项一般发生在文本由白变灰时。

读完这章您将会知道：

- FreeBSD 引导系统里的各项组件， 以及它们之间的交互方式.
- 在 FreeBSD 引导时给各组件配置选项以控制引导过程。
- device.hints(5)的基本知识。

只适用于x86
本章只描述了运行于 Intel x86 体系之上的 FreeBSD 的引导过程。

13.2. 引导问题

启动电脑及启动和引导操作系统构成了一个有趣的两难境地。
按照定义在操作系统被启动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呢?

在 Munchausen男爵历险记 (The Adventures of Baron Munchausen)
这本书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， 然后靠着拉自己的靴襻 (bootstrap)
克服重重困难爬了出来。 在早期文献中， 多以术语 bootstrap 来指代操作系统的加载机制，
如今它逐渐被简写为 "booting"。

在 x86 硬件体系中，基本输入/输出系统 (BIOS) 负责加载操作系统， 为了做到这一点，BIOS
在磁盘上寻找主引导记录 (MBR)，而 MBR 必须在放置的磁盘的特定位置。BIOS 有足够的能力来读入和运行
MBR， 且假使地认为 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类。这一类引导器通常有更多代码位于磁盘第一 轨道或在操作系统的文件系统中。
(引导管理器有时也被称为boot loader， 但是FreeBSD对后面的引导阶段才使用这个术语。)
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。)

如果您只安装了一个操作系统，那么一个标准的 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， 然后运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。相关文件为 /boot/mbr。

如果您在磁盘上安装了多个操作系统那么您可以安装一个不同的
引导管理器，它能显示一张操作系统的列表，您能从中选择启动哪个。
这样的两种引导器将在下一小节中讨论。

启动系统的剩余部分被分为三个阶段。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。
第二阶段稍微干得多一些。第三阶段完成加载操作系统的任务。 工作被分为三个阶段是因为 PC
标准对第一第二阶段执行的程序的大小有所限制。 把这些任务连在一起使得 FreeBSD
可以提供更大伸缩性的加载器 (loader)。

然后内核启动，它开始探测设备并初始化它们。 一旦内核引导进程完成任务，内核将控制权交给用户进程
init(8)， 它确认磁盘是否处于可用状态。init(8) 然后开始用户级资源配置：

加载文件系统启动网卡，及粗略地启动所有 FreeBSD 系统加载时经常运行的进程。

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时被提为引导过程的 阶段0。这一小节便是前面提到引导器中的两种：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非常简单， 由于在中的程序只能有446字节长， 分区表和MBR末端的0x55AA标识也要挤占一些空间。) 如果你已经安装boot0 并且有多个操作系统在你的硬盘上，那么你如果您安装了 FreeBSD MBR 而且安装了多个操作系统， 则会在系统启动时看到类似下面的提示：

例 10. boot0 截屏

```
F1 DOS
F2 FreeBSD
F3 Linux
F4 ??
F5 Drive 1

Default: F2
```

目前已经知道一些其它操作系统，特别是 Windows® ， 会以自己的 MBR 覆盖现有 MBR。如果发生了这种事情， 或者您想用 FreeBSD 的 MBR 覆盖现有的 MBR，您可以使用以下的命令：

```
# fdisk -B -b /boot/boot0 device
```

device 是要写入 MBR 的设备名，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，如果你需要一个自行配置的MBR，请使用boot0cfg(8)。

The LILO Boot Manager: 要想安装这个引导管理器并也用来引导FreeBSD，
首先启动Linux，并将以下选项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的标示符指定了FreeBSD的主分区和驱动器， 将X替换为Linux驱动器字母，将Y替换为Linux主分区号。 如果您使用的是 SCSI 驱动器，您需要将 /dev/hd 改成 /dev/sd，这里再次使用了 XY 的语法。 如果您安装的两个系统在同一驱动器上，loader=/boot/chain.b 选项可以去掉。现在您可以执行 /sbin/lilo -v 使修改生效；应检查屏幕上的消息确认修改。

13.3.2. 第一阶段，/boot/boot1，和第二阶段， /boot/boot2

概念上，第一，第二阶段同属于一个程序，处于磁盘的相同区域。但由于空间限制，它们被分为两部分。可是您总是会一起安装它们。它们由安装器或 bsdlabel(见下文)复制自被组合而成的

/boot/boot。

它们位于文件系统外，引导分区的第一轨道，从第一扇区开始。在这里boot0，或者任何其它引导管理器，期望找到一个程序运行，继续引导进程。 所使用的扇区数可由/boot/boot的大小确定。

boot1 非常简单，因为它再多也只能有 512 字节， 只能识别储存着分区信息的 bsdlabel， 及寻找执行 boot2。

boot2 稍微有点加强，能够理解 FreeBSD 的文件系统以便于寻找里面的文件，
能提供选择内核和加载器的简单界面。

因为 loader 有着更强的功能， 提供了一套易于使用的引导配置，boot2 一般都执行 loader，
但以前它的任务是直接运行内核。

例 11. boot2 的屏幕输出

```
>> FreeBSD/i386 BOOT
Default: 0:ad(0,a)/boot/loader
boot:
```

如果您要更改已安装的 boot1 和 boot2，请使用命令 bsdlabel(8)。

```
# bsdlabel -B diskslice
```

diskslice 是用于引导的磁盘和分区， 比如 ad0s1 代表第一个 IDE 磁盘上的第一个分区。

dangerously dedicated
如果您在 bsdlabel(8) 命令中只使用了磁盘名，比如 ad0，就会破坏磁盘上的所有分区。
这当然不是您所希望的，所以在按下 回车 之前 一定要对命令进行多次确认。

13.3.3. 第三阶段，/boot/loader

加载器 (loader) 是三个阶段中的最后阶段， 且是放置在文件系统之中的，一般是文件 /boot/loader。

loader 被作为一种友好的配置方式，使用了一组内建且易用的命令集。
这些命令由一个强大的多的解释器支持构建，其本身带有复杂得多的命令集。

13.3.3.1. Loader 程序流程

初始时，loader 会探测控制台和磁盘，识别是从哪块盘引导的。 它会根据这些信息设置变量，
启动解释器以接受通过脚本或交互方式传来的用户命令。

loader 然后会读取并运行 /boot/loader.rc， 默认地读取 /boot/defaults/loader.conf
以设置可靠的默认变量，读取 /boot/loader.conf 对这些变量作本地修改。loader.rc
依据这些变量进行动作，加载任何被选择的模块和内核。

最后，默认地，loader 会停留 10 秒等待按键，
若没有发生中断，就开始引导内核。如果被中断，用户会得到一个命令行提示符，
在这里用户得更改变量、卸载所有模块、加载模块、最后引导 或重新引导。

13.3.3.2. Loader 内建的命令

这些是最常用的 loader 命令.对所有可用命令的解释请参见 loader(8)。

autobootseconds
在给定的时间内如果没有中断发生就引导内核。它显示一个倒数计时， 默认的时间范围是 10 秒。

boot [-options] [kernelname]
立即按指定的选项启动指定名字的内核 (如果有指定的话)。 只有首先执行过 unload 命令之后指定的内核名字才会生效， 否则， 启动的将是先前已经加载的内核。

boot-conf
基于变量对各种模块进行自动配置 (和引导内核时发生的一样)。 您只须记住要先使用 unload 命令，然后修改一些变量，比如 kernel。

help [topic]
显示从文件 /boot/loader.help 读取的帮助信息。如果给定的主题是 index，那么列出来的是所有可用的主题。

include filename …
通过给定的文件名处理文件。文件被读入，然后被一行一行地解释。 任何错误都会立即中止 include 命令。

load [-t type] filename
加载内核、内核模块，或者是给定类型的文件 (通过给定的文件名)。任何在文件名后面的参数都会被传给文件。

ls [-l] [path]
显示给定路径或者是根目录 (如果路径没有指定) 下面的文件列表。 如果指定了 -l 选项，文件大小也会显示。

lsdev [-v]
列出所有可以加载模块的设备。 如果指定了-v 选项，会显示出更多的细节。

lsmod [-v]
显示已被加载的模块。如果指明了 -v 选项， 会显示更多的细节。

more filename
显示指定的文件，每隔 LINES 停顿一次。

reboot
立即重启系统。

set variable
设置 loader 的环境变量。

unload
移除所有已被加载的模块。

13.3.3.3. Loader 示例

这里有一些实际中 loader 用法的示例

- 只是简单的引导默认内核，不同的是进入单用户模式：

```
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
```

您可以使用被称为通用内核的 kernel.GENERIC， 或者您以前安装的内核 kernel.old (当您升级或配置了您自己的内核等时候)。

使用以下命令加载常用的模块和另一个内核：

```
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. 启动时的 Splash 图像

在启动时出现的 splash 图像比起原本的启动信息更加可视话。
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 系统中有两个基本的环境。 第一个是默认传统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 桌面图形环境。 在安装了 X11 和一种图形 桌面环境，
比如 GNOME， KDE， 或者 XFce， X11 桌面可以用 startx 命令运行。

比起传统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 (还有其他 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，
它们展现给用户一个图形化的桌面。

在命令行环境， splash 图像将在显示登录提示符之前隐藏所有启动时的监测与任务启动的消息。 在 X11 环境， 用户将会获得一个视觉上更加清爽启动体验， 类似于某些像 (Microsoft® Windows® 或者非 UNIX® 类型的系统) 用户所希望体验到的。

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， 才够能在标准 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， 则需开启 FreeBSD 的 VESA 支持。
这可以通过在系统启动时加载 VESA 模块完成， 或者在内核配置文件中加入 VESA 选项并编译 (参阅 配置FreeBSD的内核)。 VESA 支持给予了用户显示覆盖整个显示器的启动画面能力。

在启动的时候 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关闭。

Splash 图像同样也会是 X11 之外默认的屏幕保护。 在一段时间的闲置后，屏幕便会转为周期性的变换显示 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= 选项控制。 saver= 选项有一些内置的屏幕保护可供选择， 完整的列表可以再 splash(4) 手册页中找到。
默认的屏幕保护被称为 "warp"。 请注意在 /etc/rc.conf 中所指定 saver= 选项仅限应用于虚拟控制台。
对于 X11 图形化的登录管理器无效。

一些有关启动引导器的信息， 包括启动选项菜单和一个定时倒数提示符都会在启动时显示， 即是开启了 splash 图像功能。

splash 图像文件样本可以从 http://artwork.freebsdgr.org 下载。 安装了 sysutils/bsd-splash-changer port 之后， 每次启动的时候便能从集合中随机选择 splash 图像。

13.3.3.4.2. 开启 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) 编辑 /boot/lodaer.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， 编辑 /boot/lodaer.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 为需要被使用的 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"
: 这将关闭显示启动选项菜单， 但是倒数记时仍然会出现。 即是在启动菜单选项被禁用的时候，
在倒数记时段键入相应的启动选项仍然有效。

loader_logo="beastie"
: 这将替换启动选项菜单右侧默认显示的 "FreeBSD" 为彩色的小魔鬼标志， 就像以往的发行版那样。

请参阅 splash(4)， loader.conf(5) 和 vga(4) 手册页获取更多详细信息。

13.4. 内核在引导时的交互

一旦内核被 loader (一般情况下) 或者 boot2 (越过 loader) 加载，
它将检查引导标志，如果有的话，就会进行必要的动作调整。

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，询问作为根加载的设备。

-C
: 从 CDROM 引导。

-c
运行 UserConfig (引导时的内核配置器)

-s
引导进入单用户模式

-v
在内核引导过程中显示更有的信息

还有更多的引导标志，阅读 boot(8) 以获取有关它们的信息。

13.5. Device Hints

在初始化系统启动时，loader(8) 会读取 device.hints(5)
文件。这个文件以变量的形式储存着内核引导信息， 有时被称为 "device hints"。 设备驱动程序用"device hints" 对设备进行配置。

Device hints 也可以在 第三阶段的boot loader 的命令行提示符中指定。变量可以用 set 命令添加，unset 命令删除， show 命令查看。在文件 /boot/device.hints 设置的变量亦可以在这里被覆盖。键入 boot loader 中的变量不是永久性的，在下次启动时就会被忘记。

一旦系统引导成功，kenv(1) 命令可以用来清楚所有的变量。

文件 /boot/device.hints 的语法是一行一个变量， 使用"#"作为注释标记。 每行是按照如下方式组织的：

```
hint.driver.unit.keyword="value"
```

第三阶段 boot loader 的语法是：

```
set hint.driver.unit.keyword=value
```

driver 是设备驱动程序名，unit 是设备驱动程序单位名，keyword 是 hint 关键字。
关键字可以由以下选项组成：

- at：指明设备所绑定的总线
- 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：指明设备占用的物理内存地址。
- flags：给设备设置各种标志位。
- disabled：如果设成 1， 设备被禁用。

设备驱动程序能够接受更多的 hints，推荐您参看它们的联机手册。参看 device.hints(5)、kenv(1)、loader.conf(5) 和 loader(8) 联机手册以获取更多的信息。

13.6. Init：进程控制及初始化

一旦内核完成引导，它就把控制权交给了用户进程 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. 自动重启过程

自动重启过程会确认系统中可用的文件系统处于健康的状态。 如果不是， 而且使用 fsck(8) 也无法修复这些问题， init(8) 会进入 单用户模式 以便系统管理员直接修正这些问题。

13.6.2. 单用户模式

此模式可以通过 自动重启过程 或者通过带有 -s 选项的用户引导或通过在 loader 里设置 boot_single 变量等多种方式来达到。

也可以在多用户模式下调动无重启 (-r) 选项和停机 (-h) 选项的 shutdown(8) 命令来进入单用户模式。

如果系统 控制台 在文件 /etc/ttys 中被设置为 不安全(insecure)， 在初始化单用户模式前会出现要求输入 root 密码的命令行提示符。

例 12. 在 /etc/ttys 文件中的不安全控制台

```
# name  getty                 type    status       comments
#
# If console is marked "insecure", then init will ask for the root password # when going
to single-user mode.
console none                  unknown off insecure
```

把控制台设置成 不安全 (insecure) 使只知道 root 密码的人才能进入单用户模式， 因为您认为控制台在物理上是不安全的。因此如果您考虑到安全性， 请选择 不安全 (insecure)，而非 安全 (secure)。

13.6.3. 多用户模式

如果 init(8) 发现您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，开始系统的资源配置。

13.6.3.1. 资源配置 (rc)

资源配置分别从文件 /etc/defaults/rc.conf、 /etc/rc.conf 中读取默认配置和细节配置， 然后加载在文件 /etc/fstab 中提及的文件系统、 启动网络服务、启动各种系统守护进程，最后启动本地安装包的启动脚本。

rc(8) 联机手册是关于资源配置的很好的参考。

13.7. 关机 (shutdown) 过程

由命令 shutdown(8) 的发起的关机过程中， init(8) 会试着运行 /etc/rc.shutdown 脚本， 给所有进程发送 TERM 信号， 最后给不按时停止的进程发送 KILL 信号。

在支持电源管理的平台上关闭 FreeBSD 系统的电源， 只要简单地使用命令 shutdown -p now 即可。 此外， 可以用命令 shutdown -r now 来重启 FreeBSD。 要执行 shutdown(8) 您必须是 root 用户或 operator 组的成员。 也可以使用 halt(8) 和 reboot(8) 命令来关闭系统， 请参看它们的联机手册以获得更多的信息。

电源管理需要支持， 这要求内核支持 acpi(4) 或以模块形式加载它。

Chapter 14. 用户和基本的帐户管理

14.1. 概述

FreeBSD允许多个用户同时使用计算机. 当然,这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们的工作.要使用系统,每个人都应该有一个帐户.

读完这章，您将了解到:

- 在一个FreeBSD系统上不同用户帐户之间的区别.
- 如何添加用户帐户.
- 如何删除用户帐户.
- 如何改变帐户细节，如用户的全名，或首选的shell.
- 如何在每个帐户基础上设置限制，来控制像内存，CPU时钟这样的资源.
- 如何使用组来使帐户管理更容易.

在阅读这章之前，您应当了解:

- 了解UNIX®和FreeBSD的基础知识 (UNIX 基础).

14.2. 介绍

所有访问系统的用户都是通过帐户完成的，所以用户和帐户管理是FreeBSD系统不可或缺的重要部分.

每个FreeBSD系统的帐户都有一些和它相对应的信息去验证它.

用户名
: 用户名在login: 提示符的后面键入。 用户名对于一台计算机来讲是唯一的;
您不可以使用两个相同的用户名来登录。 有很多用来创建正确用户名的规则， 具体请参考 passwd(5);
您使用的用户名通常需要8个或更少的小写字母。

口令
: 每个帐户都有一个口令与它对应。 口令可以是空的， 这样不需要口令就可以访问系统。
这通常不是一个好主意； 每个帐户都应该有口令。

用户 ID (UID)
: UID是系统用来识别用户的数字，传统上它的范围是0到65536之间，用以唯一地标识用户。
FreeBSD在内部使用UID来识别用户 - 在工作以前。 任何允许您指定一个用户名的 FreeBSD
命令都会把它转换成UID。 这意味着您可以用不同的用户名使用多个帐户， 但它们的UID是一样的。
FreeBSD 会把这些帐户认定是同一个用户。

组ID (GID)
: GID是用来识别用户所在的组的， 传统上范围在0到65536之间的数字。
组是一种基于用户GID而不是它们的UID的用来控制用户访问资源的机制。
这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。

登录类
: 登录类是对组机制的扩展,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 并不强制用户去改变他们的口令。
您可以以用户为单位强制要求一些或所有的用户定期改变他们的口令。

帐户的到期时间
: 默认情况下 FreeBSD 不会自动完成帐户过期操作。 如果您正在创建帐户，
您应该知道一个帐户的有效使用期限。 例如， 在学校里您会为每个学生建立一个帐户，

您可以指定它们何时过期。 帐户过期后， 虽然帐户的目录和文件仍然存在，
但帐户已经不能继续使用了。

用户的全名
: 用户名可以唯一地识别FreeBSD的帐户， 但它不会反映用户的全名。 这些信息可能与帐户是相关的。

主目录
: 主目录是用户用来启动的目录的完全路径。 一个通常的规则是把所有用户的主目录都放在
/home/username 下，或者 /usr/home/username 下。
用户将把他们的个人文件放在自己的主目录下， 他们可以在那里创建任何目录.

用户 shell
: Shell提供了用户用来操作系统的默认环境。 有很多不同的shell，
有经验的用户会根据他们的经验来选择自己喜好的shell。

有三种类型的帐户: 超级用户， 系统用户， 以及 普通用户。 超级用户帐户通常叫做 root，
可以没有限制地管理系统。 系统用户运行服务。 最后， 普通用户给那些登录系统以及阅读邮件的人使用。

14.3. 超级用户帐户

超级用户帐户， 通常叫做 root， 可以重新配置和管理系统， 在收发邮件，
系统检查或编程这样的日常工作中， 尽量不要使用root权限。

这是因为不象普通用户帐户， 超级用户能够无限制地操作系统，
超级用户帐户的滥用可能会引起无法想象的灾难。 普通的用户帐户不会由于出错而破坏系统，
所以要尽可能的使用普通帐户， 除非您需要额外的特权。

在使用超级用户命令时要再三检查， 因为一个额外的空格或缺少某个字符的命令都可能会引起数据丢失。

所以， 在阅读完这章后您第一件要做的事就是， 在平时使用的时候， 创建一个没有特权的用户帐户。
无论您使用的是单用户还是多用户系统这样的申请都是相同的。 在这章的后面，
我们将讨论如何创建一个额外的帐户和如何在普通用户和超级用户之间进行切换。

14.4. 系统帐户

系统用户是那些要使用诸如DNS、 邮件， web等服务的用户。 使用帐户的原因就是安全；
如果所有的用户都由超级用户来运行， 那它们就可以不受约束地做任何事情。

典型的系统帐户包括 daemon、 operator、 bind (供 域名服务 使用)、 news， 以及 www。

nobody 是普通的没有特权的系统用户。 然而， 大多数与用户联系很密切的服务是使用 nobody的，
记的这点非常重要， 这样可能使用户变的非常有特权。

14.5. 用户帐户

用户帐户是让真实的用户访问系统的主要方式， 这些帐户把用户和环境隔离，
能阻止用户损坏系统和其他用户， 在不影响其他用户的情况之下定制自己的环境。

任何人访问您的系统必须要有他们自己唯一的帐户。 这可以让您找到谁做了什么事，
并且阻止人们破坏其他用户的设置和阅读其他人的邮件等等。

每个用户能够设置他们自己的环境， 以利于他们通过改变shell， 编辑器，
键盘绑定和语言等适应并且更好的使用这个系统。

14.6. 修改帐户

在UNIX® 的处理用户帐户的环境中有很多不同的命令可用. 最普通的命令如下，
接下来是详细使用它们的例子。

命令	摘要
adduser(8)	在命令行添加新用户.
rmuser(8)	在命令行删除用户.
chpass(1)	一个灵活的用于修改用户数据库信息的工具.
passwd(1)	一个用于修改用户口令的简单的命令行工具.
pw(8)	一个强大灵活修改用户帐户的工具.

14.6.1. 添加用户

adduser(8) 是一个简单的添加新用户的命令. 它为用户创建 passwd 和 group 文件。
它也为新用户创建一个主目录， 之后， 它会复制一组默认的配置文件 ("dotfiles") 从 /usr/shared/skel 这个目录， 然后给新用户发送一封带欢迎信息的邮件。

例 13. 在 FreeBSD 中添加一个新用户

```
# 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!
#
```

您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。
请务必确保没有输错口令。

14.6.2. 删除用户

您可以使用rmuser(8) 从系统中完全删除一个用户. rmuser(8) 执行如下步骤:

1. 删除用户的 crontab(1) 记录 (如果有的话).
2. 删除属于用户的at(1) 工作.
3. 杀掉属于用户的所有进程.
4. 删除本地口令文件中的用户.
5. 删除用户的主目录 (如果他有自己的主目录).
6. 删除来自 /var/mail属于用户的邮件.
7. 删除所有诸如 /tmp的临时文件存储区中的文件.
8. 最后, 删除 /etc/group中所有属于组的该用户名.

如果一个组变成空，而组名和用户名一样,组将被删除.
adduser(8)命令建立每个用户唯一的组.

rmuser(8) 不能用来删除超级用户的帐户, 因为那样做是对系统极大的破坏.

默认情况下, 使用交互模式, 这样能够让您清楚的知道您在做什么.

例 14. 删除用户 交互模式下的帐户删除

```
# 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) 可以改变用户的口令, shells, 和包括个人信息在内的数据库信息.

只有系统管理员， 即超级用户， 才可以用 chpass(1) 改变其他用户口令和信息。

除了可选择的用户名， 不需要任何选项， chpass(1) 将显示一个包含用户信息的编辑器.
可以试图改变用户在数据库中的信息.

如果您不是超级用户的话， 在退出编辑状态之后， 系统会询问您口令。

例 15. 以超级用户交互执行 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:
```

普通用户只能改变他们自己很少的一部分信息.

例 16. 以普通用户交互执行 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) 的符号连接， 类似地， ypchpass(1), ypchfn(1) 以及 ypchsh(1) 也是这样。 NIS 是自动支持的， 不一定要在命令前指定 yp。
如果这让您有点不太明白， 不必担心， NIS 将在 介绍。

14.6.4. passwd命令

passwd(1) 是改变您自己作为一个普通用户口令或者作为超级用户口令常用的方法.

用户改变口令前必须键入原来的口令,
防止用户离开终端时非授权的用户进入改变合法用户的口令。

例 17. 改变您的口令

```
% passwd Changing local password for jru.
```

```
Old password:
New password:
Retype new password:
passwd: updating the database...
passwd: done
```

例 18. 改变其他用户的口令同超级用户的一样

```
# 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)的连接,
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。
它还有系统用户和组文件编辑器的功能。 pw(8) 有一个非常强大的命令行选项设置，
但新用户可能会觉得它比这里讲的其它命令要复杂很多。

14.7. 限制用户使用系统资源

如果您有一些用户， 并想要对他们所使用的系统资源加以限制， FreeBSD
提供了一些系统管理员限制用户访问系统资源的方法。 这些限制通常被分为两种: 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检查用户使用磁盘数量而不需要时刻计算的方法。
配额将在 文件系统配额讨论.

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。 这些是通过对登录进行分类完成的，
下面将做讨论。

登录的类由 /etc/login.conf 文件定义。 比较精确的描述超出了本章的范围， 但 login.conf(5)
联机手册会有比较详细的描述。 可以说每个用户都分配到一个登录类 (默认是 defalut)，
每个登录类都有一套和它相对应的功能。 登录功能是 名字=值 这样的一对值， 其中名字
是一个众所周知的标识符， 值 是一个根据名字经过处理得到的任意字符串。 设置登录类和功能相当简单，
在 login.conf(5) 联机手册会有比较详细的描述。

系统并不直接读取 /etc/login.conf 中的配置， 而是采用数据库文件 /etc/login.conf.db
以提供更快的查找能力。 要从 /etc/login.conf 文件生成 /etc/login.conf.db，
应使用下面的命令:

```
# cap_mkdb /etc/login.conf
```

资源限制与普通登录限制是有区别的。 首先， 对于每种限制， 有软限制 (比较常见) 和硬限制之分。
一个软限制可能被用户调整过， 但不会超过硬限制。 越往后可能越低， 但不会升高。 其次，

绝大多数资源限制会分配特定用户的每个进程， 而不是该用户的全部进程。 注意，
这些区别是资源限制的特殊操作所规定的， 不是登录功能框架的完成 (也就是说, 他们实际上
不是一个登录功能的特例)。

不再罗嗦了， 下面是绝大多数资源限制的例子 (您可以在 login.conf(5) 找到其它与登录功能相关的内容)。

coredumpsize
: 很明显， 由程序产生的核心文件大小的限制在磁盘使用上是属于其它限制的 (例如， 文件大小，
磁盘配额)。 不过， 由于用户自己无法产生核心文件， 而且通常并不删除它们，
设置这个可以尽量避免由于一个大型应用程序的崩溃所造成的大量磁盘空间的浪费。 (例如, emacs)
崩溃。

cputime
: 这是一个用户进程所能消耗的最长 CPU 时间。 违反限制的进程， 将被内核杀掉。

这是一个有关CPU消耗的时钟 限制, 不是top(1) 和 ps(1)
命令时屏幕上显示的CPU消耗的百分比。 在写此说明时，
后者的限制是不太可能和没有价值的： 编译器 - 编译一个可能是合法的工作 -
可以在某一时刻轻易的用掉 100% 的 CPU。

filesize
: 这是用户可以处理一个文件的最大值。 不象 磁盘配额， 这个限制是对单个文件强制执行的，
不是用户自己的所有文件。

maxproc
: 这是一个用户可以运行的最大进程数。 这包括前台和后台进程。 很明显， 这不可能比系统指定
kern.maxproc sysctl(8) 的限制要大。 同时也要注意， 设置的过小会妨碍用户的处理能力：
可能需要多次登录或执行多个管道。 一些任务， 例如编译一些大的程序， 也可能会产生很多进程
(例如， make(1)， cc(1) 以及其它一些预处理程序)。

memorylocked
: 这是一个进程允许锁到主存中的最大内存容量 (参见 mlock(2))。 大型程序， 例如像 amd(8)
在遇到问题时， 它们得到的巨大交换量无法传递给系统进行处理。

memoryuse
: 这是在给定时间内一个进程可能消耗的最大内存数量。 它包括核心内存和交换内存。
在限制内存消耗方面， 这不是一个完全的限制，但它是一个好的开始。

openfiles
: 这是一个进程可以打开的最大文件数。 在FreeBSD中， 文件可以被表现为套接字和IPC通道；
注意不要把这个数设置的太小。 系统级的限制是由 kern.maxfiles 定义的， 详情参见 sysctl(8)。

sbsize
: 这是网络内存数量的限制， 这主要是针对通过创建许多套接字的老式 DoS 攻击的，
但也可以用来限制网络通信。

stacksize
: 这是一个进程堆栈可能达到的最大值。 它不能单独的限制一个程序可能使用的内存数量； 所以，
需要与其它的限制手段配合使用。

在设置资源限制时， 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。

- 系统启动的进程/etc/rc会被指派给 守护程序 的登录类.
- 虽然 /etc/login.conf 文件是一个对绝大多数限制做合理配置的资源文件，
但只有您也就是系统管理员，才知道什么最适合您的系统。
设置的太高可能会因为过于开放而导致系统被滥用， 而设置过低， 则可能降低效率。
- 使用 X Window 的用户可能要比其他用户使用更多的资源。 因为X11本身就使用很多资源，
而且它鼓励用户同时运行更多的程序。

- 务必注意， 许多限制措施是针对单个进程来实施的， 它们并不限制某一用户所能用到的总量。 例如，将 openfiles 设置为 50 表示以该用户身份运行的进程最多只能打开 50 个文件。 因而，用户实际可以打开的文件总数就应该是 maxproc 和 openfiles 值的乘积。 对内存用量的限额与此类似。

有关资源限制,登录类的更深入信息可以查看相关联机手册: cap.mkdb(1), getrlimit(2), login.conf(5).

14.8. 组

组简单的讲就是一个用户列表. 组通过组名和GID (组 ID) 来识别。 在 FreeBSD (以及绝大多数其他 UNIX® 系统) 中， 内核用以决定一个进程是能够完成一项动作的两个因素是它所属的用户 ID 和组 ID。 与用户 ID 不同， 每个进程都有一个和它相关联的组的列表。 您可能听说过用户或进程的 "组 ID"； 大多数情况下，这表示列表中的第一个组。

与组ID对应的组名在/etc/group中。 这是一个由冒号来界定的文本文件。 第一部分是组名，第二部分是加密后的口令， 第三部分是组ID， 第四部分是以逗号相隔的成员列表。
它可以用手工方式进行编辑 (当然， 如果您能保证不出语法错误的话！)。 对于更完整的语法描述， 参见 group(5) 联机手册.

如果不想手工编辑 /etc/group， 也可以使用 pw(8) 添加和编辑组。 例如， 要添加一个叫 teamtwo 的组，确定它存在:

例 19. 使用pw(8)添加一个组

```
# pw groupadd teamtwo
# pw groupshow teamtwo
teamtwo:*:1100:
```

上面的数字 1100 是组 teamtwo 的组 ID。 目前， teamtwo 还没有成员， 因此也就没有多大用处。
接下来， 把 jru 加入到 teamtwo 组。

例 20. 使用 pw(8) 设置组的成员列表

```
# 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组中将要成为成员的用户列表。 前面我们已经知道， 口令文件中，每个用户已经指定了一个所属组。 之后用户被自动地添加到组列表里； 当我们使用 groupshow 命令时 pw(8) 用户列表不被显示出来。 但当通过 id(1) 或者类似工具查看时， 就会看到用户列表。 换言之， pw(8) 命令只能读取 /etc/group 文件； 它从不尝试从 /etc/passwd 文件读取更多信息。

例 21. 使用 pw(8) 为组添加新的成员

```
# pw groupmod teamtwo -m db
# pw groupshow teamtwo
teamtwo:*:1100:jru,db
```

-m 选项的参数是一个由逗号分隔的即将被添加进组的用户列表。 与先前那个例子的不同之处在于，这个列表中的用户将被添加进组而非取代组中的现有用户。

例 22. 使用id(1)来决定组成员

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

正如您所看到的， jru 是组 jru 和组 teamtwo的成员.

有关pw(8)的更多信息， 请参看其它联机手册。 更多的关于 /etc/group 文件格式的信息， 请参考 group(5) 联机手册。

Chapter 15. 安全

15.1. 概述

这一章将对系统安全的基本概念进行介绍， 除此之外， 还将介绍一些好的习惯， 以及 FreeBSD 下的一些更深入的话题。 这章的许多内容对于一般的系统和 Internet 安全也适用。 如今， Internet 已经不再像以前那样是一个人人都愿意与您作好邻居的 "友善" 的地方。 让系统更加安全， 将保护您的数据、 智力财产、 时间， 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证您的系统和网络的完整及安全。

读完这章，您将了解：

- 基本的 FreeBSD 系统安全概念。
- FreeBSD 中众多可用的密码学设施，例如 DES 和 MD5。
- 如何设置一次性口令验证机制。
- 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® 机器之间建构 VPN。
- 如何配置并使用 OpenSSH，以及 FreeBSD 的 SSH 执行方式。
- 系统 ACL 的概念，以及如何使用它们。
- 如何使用 Portaudit 工具来审核从 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。
- 对于进程记帐功能的感性认识， 并了解如何在 FreeBSD 中启用它。

在开始阅读这章之前，您需要：

- 理解基本的 FreeBSD 和 Internet 概念。

其他安全方面的话题， 则贯穿本书的始终。 例如， 强制性访问控制 (MAC) 在 强制访问控制 中进行了介绍， 而 Internet 防火墙则在 防火墙 中进行了讨论。

15.2. 介绍

安全是系统管理员自始至终的基本要求。 由于所有的 BSD UNIX® 多用户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， 确保用户的 "诚实" 可能也就是最需要系统管理员考虑的艰巨的工作了。 机器的安全性取决于您设置的安全设施， 而许多安全方面的考虑， 则会与人们使用计算机时的便利性相矛盾。 一般来说， UNIX® 系统能够胜任数目众多进程并发地处理各类任务， 这其中的许多进程是以服务身份运行的 - 这意味着， 外部实体能够与它们互联并产生会话交互。 如今的桌面系统， 已经能够达到许多昔日的小型机甚至主机的性能， 而随着这些计算机的联网和在更大范围内完成互联， 安全也成为了一个日益严峻的课题。

系统的安全也应能够应付各种形式的攻击， 这也包括那些使系统崩溃， 或阻止其正常运转， 并不仅限于试图窃取 root 帐号 ("破译 root") 的攻击形式。 安全问题大体可分为以下几类：

1. 拒绝服务攻击。
2. 窃取其他用户的帐户。
3. 通过可访问服务窃取root帐户。
4. 通过用户帐户窃取root帐户。
5. 建立后门。

拒绝式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)手段通过压倒性的流量来破坏服务器和网络栈， 以使机器崩溃或无法使用。 某些 DoS
攻击则利用在网络栈中的错误， 仅用一个简单的信息包就可以让机器崩溃，
这类情况通常只能通过给内核打补丁来修复。 在一些不利的条件下， 对服务器的攻击能够被修复，
只要适当地修改一下系统的选项来限制系统对服务器的负荷。 顽强的网络攻击是很难对付的。
例如，一个欺骗性信息包的攻击， 无法阻止入侵者切断您的系统与Internet的连接。
它不会使您的机器死掉，但它会把Internet连接占满。

窃取用户帐户要比D.o.S.攻击更加普遍。 许多系统管理员仍然在他们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。
结果是如果您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您系统的用户中，
一些人的口令会被人窃取。 仔细的系统管理员会从那些成功登录系统的远程访问日志中寻找可疑的源地址。

通常必须假定，如果一个入侵者已经访问到了一个用户的帐户， 那么它就可能使自己成为 root。 然而，
事实是在一个安全和维护做得很好的系统中， 访问用户的帐户不一定会让入侵者成为 root。
这个差别是很重要的，因为没有成为 root 则入侵者通常是无法隐藏它的轨迹的， 而且， 如果走运的话，
除了让用户的文件乱掉和系统崩溃之外， 它不能做什么别的事情。 窃取用户帐户是很普遍的事情，
因为用户往往不会对系统管理员的警告采取措施。

系统管理员必须牢牢记住，可能有许多潜在的方法会使他们机器上的 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帐户，
可以在自己的机器上运行一个 suid-root 程序来发现服务器的漏洞， 从而让他侵入到服务器并获取 root。
攻击者找到了入侵一台机器上 root 的途径之后， 他们就不再需要安装后门了。许多 root
漏洞被发现并修正之后， 入侵者会想尽办法去删除日志来消除自己的访问痕迹， 所以他们会安装后门。
后门能给入侵者提供一个简单的方法来重新获取访问系统的 root 权限，
但它也会给聪明的系统管理员一个检测入侵的简便方法。
让入侵者无法安装后门事实上对您的系统安全是有害的，
因为这样并不会修复那些侵入系统的入侵者所发现的新漏洞。

安全的管理方法应当使用像 "洋葱皮" 一样多层次的方法来实现， 这些措施可以按下面的方式进行分类：

1. 确保 root 和维护人员帐户的安全。
2. 确保 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. 确保用户帐户的安全。
4. 确保口令文件的安全。
5. 确保内核中核心组件、直接访问设备和文件系统的安全。
6. 快速检测系统中发生的不适当的变化。
7. 做个偏执狂。

这一章的下一节将比较深入地讲述上面提到的每一个条目。

15.3. 确保 FreeBSD 的安全

命令与协议

在这份文档中，我们使用 粗体 来表示应用程序， 并使用 单倍距 字体来表示命令。
这样的排版区分能够有效地区分类似 ssh 这样的概念，
因为它既可以表示命令，又可以表示协议。

接下来的几节中， 将介绍在这一章中 前一节 中所介绍的那些加强 FreeBSD 系统安全性的手段。

15.3.1. 确保 root 和维护人员帐户的安全

首先，如果您没有确保 root 帐户的安全，
就没必要先劳神确保用户帐户的安全了。绝大多数系统都会指派一个口令给 root 帐户。
我们的第一个假定是，口令 总是 不安全的。 这并不意味着您要把口令删掉。
口令通常对访问机器的控制台来说是必须的。 也就是说， 您应该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，确信您的 pty 终端在 /etc/ttys 文件中被指定为 insecure
(不安全)，这将使直接通过 telnet 或 rlogin 登录 root 会不被接受。 如果使用如 sshd

这样的其他登录服务， 也要确认直接登录 root 是关闭的。您可以通过编辑 /etc/ssh/sshd_config 文件来做到这一点，确信 PermitRootLogin 被设置成 no。 考虑到每一种访问方法 - 如FTP这样的服务，以免因为它们而导致安全性的损失。 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， 您应当获得 root身份， 因此， 我们开了一些后门来允许自己进入。但这些后门只有在经过了额外的口令确认之后才能使用。 一种让 root 可访问的方法是增加适当的用户帐户到 wheel 组 (在 /etc/group 中)。wheel 组中的用户成员可以使用 su 命令来成为 root。 绝对不应该通过在口令项中进行设置来赋予维护人员天然的 wheel 组成员身份。维护人员应被放置在 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进 wheel 组中。 当然，也可以通过 某种其它的验证手段，例如 Kerberos，可以通过 root 帐户中的 .k5login 文件来允许执行 ksu(1) 成为 root ，而不必把它们放进 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获得 root ，如果他拿到了口令文件，并能够进入职员的帐户。 尽管有 wheel 比什么都没有要强一些，但它并不是一种绝对安全的办法。

可以使用 pw(8) 命令来完全禁止某一个帐号：

```
# pw lock staff
```

这将阻止用户使用任何方法登录，包括 ssh(1)。

另一个阻止某个帐户访问的方法是使用一个 “*” 字符替换掉加密后的口令。这将不会与任何加密后的口令匹配，从而阻止了用户的访问。 举例说明：

```
foobar:R9DT/Fa1/LV9U:1000:1000::0:0:Foo Bar:/home/foobar:/usr/local/bin/tcsh
```

应被改为：

```
foobar:*:1000:1000::0:0:Foo Bar:/home/foobar:/usr/local/bin/tcsh
```

这会阻止用户 foobar 使用传统的方式登录。 但是对于使用了 Kerberos 或者配置了 ssh(1) 公钥/密钥对的情况下，用户依然可以访问。

这些安全机制同样假定， 从严格受限的机器向限制更宽松的机器上登录。 例如，如果您的服务器运行了所有的服务，那么，工作站应该什么都不运行。为了让工作站尽可能地安全，应该避免运行任何没有必要的服务， 甚至不运行任何服务。 另外，也应该考虑使用带口令保护功能的屏幕保护程序。 毋庸置疑， 如果攻击者能够物理地接触您的工作站，那么他就有能力破坏任何安全设施，这确实是我们需要考虑的一个问题，但同样地，真正能够物理接触您的工作站或服务器并实施攻击的人在现实生活中并不常见， 绝大多数攻击来自于网络，而攻击者往往无法物理地接触服务器或工作站。

使用类似 Kerberos 这样的工具，也为我们提供了使用一个工具来禁用某个用户， 或修改他们的口令，并在所有机器上立即生效的方法。 如果员工的帐号被窃取，能够在所有的其他机器上生效的口令变更将很有意义。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 还能够提供更多的限制，除了 Kerberos 令牌有很好的过期机制之外， 它还能够强制用户在某个特定的期限内修改口令(比如说，每月一次).

15.3.2. 确保以root用户权限运行的服务器和suid/sgid可执行程序的安全

谨慎的管理员只运行他们需要的服务， 不多， 不少。 要当心第三方的服务程序很可能有更多的问题。例如， 运行旧版的 imapd 或 popper 无异于将 root 令牌拱手送给全世界的攻击者。永远不要运行那些您没有仔细检查过的服务程序， 另外也要知道， 许多服务程序并不需要以 root 的身份运行。 例如， ntalk、 comsat， 以及 finger 这些服务， 都能够以一种被称作 沙盒 的特殊用户的身份运行。 除非您已经解决掉了许多麻烦的问题， 否则沙盒就不是完美的，但洋葱式安全规则仍然成立： 如果有人设法攻破了在沙盒中运行的程序， 那么在做更多坏事之前，他们还必须想办法攻破沙盒本身的限制。 攻击者需要攻破的层次越多， 他们成功的可能性就越小。 过去，

破解 root 的漏洞几乎在所有以 root 身份运行的服务上都发现过， 包括那些基本的系统服务。
如果您的机器只打算向外界提供 sshd 登录， 而用户不会使用 telnetd 或 rshd 甚至 rlogind 登录，
就应该毫不犹豫地关闭它们!

FreeBSD 现在默认在沙盒中运行 ntalkd, comsat, 以及 finger。此外， named(8) 也可以这样运行。
/etc/defaults/rc.conf 中包括了如何如此运行 named 的方法，只是这些内容被注释掉了。
如何升级或安装系统将决定这些沙盒所使用的特殊用户是否被自动安装。
谨慎的系统管理员将根据需要研究并实现沙盒。

此外，还有一些服务通常并不在沙盒中运行： sendmail, popper, imapd, ftpd,
以及一些其他的服务。当然，它们有一些替代品，但安装那些服务可能需要做更多额外的工作。 可能必须以
root 身份运行这些程序， 并通过其他机制来检测入侵。

系统中另一个比较大的 root 漏洞 是安装在其中的 suid-root 和 sgid 的可执行文件。 绝大多数这类程序，
例如 rlogin 会放在 /bin、 /sbin、 /usr/bin， 或 /usr/sbin 目录中。 尽管并没有 100%
的安全保证，但系统默认的 suid 和 sgid 可执行文件通常是相对安全的。
当然，偶尔也会发现一些存在于这些可执行文件中的 root 漏洞。1998年，Xlib 中发现了一处 root
漏洞，这使得 xterm (通常是做了suid的) 变得可以入侵。 做得安全些， 总比出现问题再后悔要强。
因此，谨慎的管理员通常会限制 suid 可执行文件，
并保证只有员工帐号能够执行它们，或只开放给特定的用户组，甚至彻底干掉 (chmod 000) 任何 suid
可执行文件， 以至于没有人能够执行它们。没有显示设备的服务器通常不会需要 xterm 可执行文件。 sgid
可执行文件通常同样地危险。 一旦入侵者攻克了sgid-kmem，那么他就能够读取 /dev/kmem
并进而读取经过加密的口令文件， 从而窃取任何包含口令的帐号。另外，攻破了 kmem
的入侵者能够监视通过 pty 传送的按键序列，即使用户使用的是安全的登录方式。 攻破了 tty
组的用户则能够向几乎所有用户的 tty
写入数据。如果用户正在运行一个终端程序，或包含了键盘模拟功能的终端仿真程序，
那么，入侵者能够以那个用户的身份执行任何命令。

15.3.3. 确保用户帐户的安全

用户帐号的安全通常是最难保证的。虽然您可以为您的员工设置严苛的登录限制， 并用 "星号"
替换掉他们的口令， 但您可能无法对普通的用户这么做。 如果有足够的决策权，
那么在保证用户帐号安全的斗争中或许会处于优势， 但如果不是这样，
您能做的只是警惕地监控这些帐号的异动。 让用户使用 ssh 或 Kerberos 可能会有更多的问题，
因为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， 这仍不失为一个好办法。

15.3.4. 确保口令文件的安全

能够确保起作用的唯一一种方法， 是将口令文件中尽可能多的口令用星号代替， 并通过 ssh 或 Kerberos
来使用这些账号。 即使只有 root 用户能够读取加密过的口令文件 (/etc/spwd.db)，
入侵者仍然可能设法读到它的内容， 即使他暂时还无法写入这个文件。

您的安全脚本应该经常检查并报告口令文件的异动 (参见后面的 检查文件完整性 一节)。

15.3.5. 确保内核中内核设备、直接访问设备和文件系统的安全

如果攻击者已经拿到了 root 那么他就有能力作任何事情， 当然， 有一些事情是他们比较喜欢干的。 例如，
绝大多数现代的内核都包括一个内建的听包设备。 在 FreeBSD 中，这个设备被称作 bpf
。攻击者通常会尝试在攻克的系统上运行它。 如果您不需要 bpf
设备提供的功能，那么，就不要把它编入内核。

但是， 即使您关闭了 bpf 设备， 仍需要关注 /dev/mem 和 /dev/kmem。 就事论事地说，
入侵者仍然能通过直接访问的方式写入磁盘设备。 另外， 还有一个称作模块加载器的内核机制，
kldload(8)。 有进取心的入侵者， 可以经由这一机制， 在正在运行的内核中通过 KLD 模块来安装自己的
bpf， 或其它听包设备。 为了避免这些问题， 您必须将内核的安全级别提高到至少 1。

内核的安全级别可以通过多种方式来设置。 最简单的设置正在运行的内核安全级的方法， 是使用 sysctl
来设置内核变量 kern.securelevel：

```
# sysctl kern.securelevel=1
```

默认情况下， FreeBSD 内核启动时的安全级别是 -1。 除非管理员或 init(8) 由于启动脚本加以改变，
安全级别会继续保持为 -1。 在系统启动过程中， 可以在 /etc/rc.conf 文件中， 将变量
kern_securelevel_enable 变量设置为 YES 并将 kern_securelevel 变量设置为希望的安全级别来提高它。

默认情况下， 在启动脚本执行完之后， FreeBSD 的安全级别设置是 -1。 这称作 "不安全模式"，
因为文件的不可修改标记 (immutable flag) 可以改为关闭， 而且全部设备可以直接进行读写， 等等。

一旦将安全级别设置为 1 或更高， 则只允许追加 (append-only) 和不可修改标记会被执行，
而且不可以关闭。 直接访问裸设备则会被拒绝。 更高的安全级别会施加进一步的访问限制。
关于安全级别的完整介绍， 请参阅联机手册 security(7) (对于 FreeBSD 7.0 之前的版本， 则是联机手册
init(8))。

将安全级别调整到 1 或更高可能会导致 X11 (访问 /dev/io 会被阻止)， 或从源代码联编
FreeBSD (这一过程中的 installworld
部分需要临时取消一些文件上的只允许追加和不可修改标记) 出现一些问题，
并导致一些其他小问题。 有些时候， 例如 X11 的情况，
可以通过在引导过程中较早的阶段启动 xdm(1) 来绕过， 因为这时安全级别还很低。
类似这样的方法， 对于某些安全级别或限制有可能不可用。
提前做好计划可能会是个好主意。 理解不同的安全级别所施加的限制非常重要，
因为一些限制可能让系统变得很难使用。 另外， 了解它们也有助于理性地配置默认设定。

如果内核的安全级别设为 1 或更高， 在重要的启动程序、 目录和脚本文件上设置 schg 标记
(也就是在系统启动到设置安全级别之前运行的程序和它们的配置) 就有意义了。 然而，
这样做也可能有些过火， 而由于系统运行于较高的安全级别， 升级系统也会变得困难的多。 作为妥协，
可以让系统以较高的安全级别运行， 但并不将所有的启动文件都配置 schg 标记。 另一种方法是将 / 和 /usr
以只读模式挂载。 请注意， 过分严苛的安全配置很可能限制您检测入侵的能力。

15.3.6. 检查文件完整性: 可执行文件，配置文件和其他文件

当实施严格的限制时，往往会在使用的方便性上付出代价。例如，使用 chflags 来把 schg 标记 应用到 / 和
/usr 中的绝大多数文件上可能会起到反作用， 因为尽管它能够保护那些文件，
但同时也使入侵检测无法进行。 层次化安全的最后一层可能也是最重要的 - 检测。
如果无法检测出潜在的入侵行为， 那么安全的其他部分可能相对来讲意义可能就不那么大了
(或者，更糟糕的事情是， 那些措施会给您安全的假象)。 层次化安全最重要的功能是减缓入侵者，
而不是彻底不让他们入侵， 这样才可能当场抓住入侵者。

检测入侵的一种好办法是查找那些被修改、 删除或添加的文件。 检测文件修改的最佳方法是与某个
(通常是中央的) 受限访问的系统上的文件进行比对。
在一台严格限制访问的系统上撰写您的安全脚本通常不能够被入侵者察觉，
因此，这非常重要。为了最大限度地发挥这一策略的优势，通常会使用只读的 NFS， 或者设置 ssh
钥匙对以便为其他机器提供访问。除了网络交互之外， NFS可能是一种很难被察觉的方法 -
它允许您监控每一台客户机上的文件系统，
而这种监控几乎是无法察觉的。如果一台严格受限的服务器和客户机是通过交换机连接的， 那么 NFS
将是一种非常好的方式。 不过，如果那台监控服务器和客户机之间通过集线器
(Hub)，或经过许多层的路由来连接，则这种方式就很不安全了， 此时，应考虑使用 ssh
，即使这可以在审计记录中查到。

一旦为这个受限的机器赋予了至少读取它应监控的客户系统的权限， 就应该为实际的监控撰写脚本。以 NFS
挂接为例，可以用类似 find(1) 和 md5(1) 这样的命令为基础来完成我们所需的工作。
最好能够每天对被控机的所有执行文件计算一遍 md5，同时，还应以更高的频率测试那些 /etc 和
/usr/local/etc 中的控制文件。一旦发现了不匹配的情形，监控机应立即通知系统管理员。
好的安全脚本也应该检查在系统分区，如 / 和 /usr 中是否有新增或删除的可执行文件，以及不适宜的 suid
。

如果打算使用 ssh 来代替 NFS，那么撰写安全脚本将变得困难许多。 本质上，需要在脚本中使用 scp
在客户端复制文件， 另一方面，用于检查的执行文件 (例如 find) 也需要使用 scp 传到客户端，因为 ssh
客户程序很可能已经被攻陷。 总之，在一条不够安全的链路上 ssh 可能是必须的，
但也必须应付它所带来的难题。

安全脚本还应该检查用户以及职员成员的权限设置文件： .rhosts、 .shosts、 .ssh/authorized_keys
等等。 这些文件可能并非通过 MD5 来进行检查。

如果您的用户磁盘空间很大， 检查这种分区上面的文件可能非常耗时。 这种情况下， 采用标志来禁止使用 suid 可执行文件将是一个好主意。 您可能会想看看 nosuid 选项 （参见 mount(8)）。 尽管如此，
这些扫描仍然应该至少每周进行一次， 这样做的意义并不是检测有效的攻击， 而是检查攻击企图。

进程记帐 (参见 accton(8)) 是一种相对成本较低的， 可以帮助您在被入侵后评估损失的机制。
对于找出入侵者是如何进入系统的这件事情来说，
它会非常的有所助益，特别是当入侵者什么文件都没有修改的情况下。

最后， 安全脚本应该处理日志文件， 而日志文件本身应该通过尽可能安全的方法生成 - 远程 syslog
可能非常有用。 入侵者会试图掩盖他们的踪迹，
而日志文件对于希望了解入侵发生时间的系统管理员来说则显得尤为重要。
保持日志文件的永久性记录的一种方法是在串口上运行系统控制台， 并在一台安全的机器上收集这些信息。

15.3.7. 偏执

带点偏执不会带来伤害。作为一种惯例，
系统管理员在不影响使用的便利的前提下可以启用任何安全特性，此外，
在经过深思熟虑之后，也可以增加一些 确实会 让使用变得不那么方便的安全特性。
更重要的是，有安全意识的管理员应该学会混合不同的安全策略 -
如果您逐字逐句地按照这份文档来配置您的机器，
那无异于向那些同样能得到这份文档的攻击者透露了更多的信息。

15.3.8. 拒绝服务攻击

这一节将介绍拒绝服务攻击。 DoS 攻击通常是基于数据包的攻击，
尽管几乎没有任何办法来阻止大量的伪造数据包耗尽网络资源，
但通常可以通过一些手段来限制这类攻击的损害，使它们无法击垮服务器：

1. 限制服务进程 fork。
2. 限制 springboard 攻击 (ICMP 响应攻击， ping 广播，等等)。
3. 使内核路由缓存过载。

一种比较常见的 DoS 攻击情形， 是通过攻击复制进程 (fork) 的服务， 使其产生大量子进程，
从而是其运行的机器耗尽内存、 文件描述符等资源， 直到服务器彻底死掉。 inetd (参见 inetd(8))
提供了许多选项来限制这类攻击。 需要注意的是， 尽管能够阻止一台机器彻底垮掉，
但通常无法防止服务本身被击垮。 请仔细阅读 inetd 的联机手册， 特别是它的 -c、 -C 以及 -R 这三个选项。
伪造 IP 攻击能够绕过 inetd 的 -C 选项， 因此， 这些选项需要配合使用。
某些独立的服务器也有类似的限制参数。

例如， Sendmail 就提供了自己的 -OMaxDaemonChildren 选项， 它通常比 Sendmail
的负载限制选项更为有效， 因为服务器负载的计算有滞后性。 您可以在启动 sendmail 时指定一个
MaxDaemonChildren 参数； 把它设的足够高以便承载您所需要的负荷， 当然， 不要高到足以让运行
Sendmail 的机器死掉。 此外， 以队列模式 (-ODeliveryMode=queued) 运行 Sendmail 并把服务程序
(sendmail -bd) 和队列执行程序分别执行 (sendmail -q15m) 也是一个好主意。
如果您希望保证队列的实时性， 可以考虑使用更短的间隔， 例如 -q1m，
但同时也需要指定一个合理的子进程数， 也就是通过 MaxDaemonChildren 选项以免 那个Sendmail
造成重叠的故障。

Syslogd 可以被直接地攻击，因此， 强烈建议只要可行，就在启动它的时候加上 -s 参数，
其他情况下，则至少应该加上 -a。

对于基于连接的服务，例如 TCP Wrapper 的 reverse-identd， 都应该格外的小心，
因为它们都可能直接遭受攻击。 一般情况下， 基于安全考虑， 不应使用 TCP Wrapper 所提供的 reverse-ident 这样的功能。

此外， 将内部服务保护起来， 阻止来自其他主机的访问也十分重要，
这些工作可以通过设置边界路由器来完成。 主要的想法， 是阻止来自您的 LAN 以外的访问， 这有助于避免
root 受到攻击。 尽可能配置排他式的防火墙， 例如， "用防火墙阻止所有的网络流量 除了 端口 A、B、
C、D，以及 M-Z"。 通过采用这种方法， 您可以很容易地将低端口的访问阻止在外，
而又不难配置使防火墙放过那些明确需要开放的服务， 例如 named
(如果您的机器准备作为域的主要解析服务器)， ntalkd， sendmail，以及其他可以从 Internet

访问的服务。 如果您尝试以其他方式配置防火墙 - 采用比较宽松的策略， 那么您将很有可能忘记 "关掉"
一两个服务， 或者在增加了一些服务之后忘记更新防火墙策略。 尽管如此，
仍然可以考虑允许让数据进入编号较高的那一部分端口， 这将保证那些需要这样特性的服务能够正常工作，
而又不影响低端口服务的安全性。 此外， 还应注意到 FreeBSD 允许您来控制动态绑定的端口的范围，
即一系列 net.inet.ip.portrange 变量，通过 sysctl 来完成设置。 (sysctl -a | fgrep portrange)。
这使得您完成较复杂的防火墙策略变得易如反掌。 例如， 您可能希望普通的高段端口的起止范围是 4000 到
5000， 而更高范围则是 49152 到 65535， 随后在防火墙中阻止低于 4000 的所有端口 (当然，
除了那些特地为 Internet 访问而开设的端口)。

另一种常被称作 springboard 的攻击也是非常常见的 DoS 攻击 -
它通过使服务器产生其无法处理的响应来达到目的。 最常见的攻击就是 ICMP ping 广播攻击。
攻击者通过伪造 ping 包， 将其源 IP 设置为希望攻击的机器的 IP。 如果您的边界路由器没有进行禁止 ping
广播地址的设置， 则您的网络将最终陷于响应伪造的 ping 包之中，
特别是当攻击者同时使用了多个不同的网络时。 广播攻击能够产生超过 120 兆位的瞬时流量。
另一种常见的针对 ICMP 错误报告系统的 springboard 攻击， 通过建立可以生成 ICMP 出错响应的包，
攻击者能够攻击服务器的网络下行资源， 并导致其上行资源耗尽。
这种类型的攻击也可以通过耗尽内存来使得使得被攻击的服务器崩溃，
特别是当这些服务器无法足够快地完成 ICMP 响应的时候。 较新的内核可以通过调整 sysctl 变量
net.inet.icmp.icmplim 来限制这种攻击。 最后一类主要的 springboard 是针对某些 inetd 的内部服务，
例如 udp echo 服务进行的。 攻击者简单地伪造一个来自服务器 A 的 echo 口的 UDP 包，
然后将这个包发到 B 的 echo 口。 于是， 两台服务器将不停地将包弹给对方。
攻击者能够将两台服务器的这种服务都耗竭， 并且通过这种方式， 只需要很少的包就可以让 LAN 超载。
类似的问题对 chargen 口也是存在的。 好的系统管理员应该关闭这些 inetd 的测试服务。

伪造的包攻击也可以用来使内核的路由缓存过载。 请参考 net.inet.ip.rtexpire， rtminexpire， 以及
rtmaxcache sysctl 参数。 伪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临时的缓存项， 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。 如果内核发现路由表变得太大， 它会动态地降低 rtexpire 但以 rtminexpire 为限。
这引发了两个问题：

1. 在访问量不大的服务器上， 内核对于突然袭击的反应不够快。
2. rtminexpire 的值没有低到让内核在此类攻击时活下去的程度。

如果您的服务器通过 T3 或更快的线路接入 Internet， 那么通过 sysctl(8) 来手动地降低 rtexpire 和
rtminexpire 就非常必要。 当然，绝不要把它们设置为零 (除非您想让机器崩溃) 将这两个参数设置为 2
通常已经足以抵御这类攻击了。

15.3.9. Kerberos 和 SSH 的访问问题

如果您打算使用， 那么 Kerberos 和 ssh 都有一些需要解决的问题。 Kerberos 5 是一个很棒的验证协议，
但使用了它的 telnet 和 rlogin 应用程序有一些 bug， 使得它们不适合处理二进制流。 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 在默认时加密所有的会话内容。

除了默认转发加密密钥之外， ssh 在所有的其他方面都做得很好。
这意味着如果您持有供您访问系统其他部分密钥的工作站作了很好的安全防护，
而您连到了一台不安全的机器上， 则您的密钥可能被别人获得。 尽管实际的密钥并没有被泄漏， 但由于 ssh
会在您登录的过程中启用一个转发端口， 如果攻击者拿到那台不安全的机器上的 root
那么他将能够利用那个端口来使用您的密钥， 从而访问您能够访问的那些机器。

我们建议您在使用 ssh 时配合 Kerberos 来完成工作人员的登录过程。 Ssh 在编译时可以加入 Kerberos
支持。 在减少了潜在地暴露 ssh 密钥的机会的同时， 它还能够通过 Kerberos 来保护口令。 Ssh
密钥只有在做过安全防护的机器上执行自动操作时才应使用 (这是 Kerberos 不适合的情形)。
此外，我们还建议您要么在 ssh 配置中关闭密钥转发， 要么在 authorized_keys 中增加 from=IP/DOMAIN
选项， 来限制这些密钥能够登录的来源机器。

15.4. DES、 Blowfish、 MD5， 以及 Crypt

UNIX® 系统上的每个用户都有一个与其帐户关联的口令。 很显然，
密码只需要被这个用户和操作系统知道。 为了保证口令的私密性， 采用了一种称为 "单向散列"
的方法来处理口令， 简单地说， 很容易从口令推算出散列值， 反之却很难。 其实，
刚才那句话可能并不十分确切： 因为操作系统本身并不 真的 知道您的口令。 它只知道口令

经过加密的形式。 获取口令对应 "明文" 的唯一办法是采用暴力在口令可能的区间内穷举。

不幸的是，当 UNIX® 刚刚出现时，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)。 于是这给那些非美国居民带来了问题， 因为 DES 的源代码在当时不能被出口到美国以外的地方， FreeBSD 必须找到符合美国法律，但又要与其他那些使用 DES 的 UNIX® 版本兼容的办法。

解决方案是把加密函数库分割为两个， 于是美国的用户可以安装并使用 DES 函数库，而国际用户则使用另外一套库提供的一种可以出口的加密算法。 这就是 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. 识别您采用的加密算法

现在这个库支持 DES、 MD5 和 Blowfish 散列函数。默认情况下， FreeBSD 使用 MD5 来加密口令。

可以很容易地识别 FreeBSD 使用哪种加密方法。 检查 /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 散列得到的长一些， 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鉴别的特征， 但他们要比 MD5 短， 并且以不包括 $ 在内的 64 个可显示字符来表示， 因此相对比较短的、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。 请参考 login.conf(5) 联机帮助以获得更进一步的详情。

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。 第一种是您常用的 UNIX® 风格或 Kerberos 口令；我们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， 并为 opiepasswd(1) 以及登录提示所接受的一次性口令，我们称其为 "一次性口令"。 最后一类口令是您输入给 opiekey 程序 (有些时候是 opiepasswd 程序) 用以产生一次性口令的秘密口令，我们称其为 "秘密口令" 或通俗地简称为 "口令"。

秘密口令和您的 UNIX® 口令毫无关系， 尽管可以设置为相同的， 但不推荐这么做。 OPIE 秘密口令并不像旧式的 UNIX® 口令那样只能限于8位以内。 您想要用多长的口令都可以。 有六、七个词的短句是很常见的选择。 在绝大多数时候， 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 还有两组至关重要的数据。 其一被称作 "种子" 或 "key"，它包括两个字符和五个数字。 另一个被称作 "迭代轮数"， 这是一个 1 到 100 之间的数字。 OPIE 通过将种子加到秘密口令后面， 并执行迭代轮数那么多次的 MD4/MD5 散列运算来得到结果，并将结果表示为 6 个短的英文单词。 这 6 个英文单词就是您的一次性口令。 验证系统 (主要是 PAM) 会记录上次使用的一次性口令， 如果用户提供的口令的散列值与上次一致， 则可以通过身份验证。 由于使用了单向的散列函数， 因此即使截获了上次使用的口令， 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轮数递减， 这样用户和登录程序将保持同步。 每当迭代轮数减少到 1 时，都必须重新初始化 OPIE。

接下来将讨论和每个系统有关的三个程序。 opiekey 程序能够接收带迭代计数， 种子和秘密口令，并生成一个一次性口令， 或一张包含连续的一组一次性口令的表格。 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 程序可以用于检查相应的验证数据文件 (/etc/opiekeys) 并显示执行命令的用户当前的迭代轮数和种子。

我们将介绍四种不同的操作。 在安全的连接上通过 opiepasswd 来第一次设置一次性口令，或修改口令及种子。 第二类操作是在不安全的连接上使用 opiepasswd 辅以在安全连接上执行的 opiekey 来完成同样的工作。 第三类操作是在不安全的连接上使用 opiekey 来登录。 最后一类操作是采用 opiekey 来生成大批的密码， 以便抄下来或打印出来，在没有安全连接的地方使用。

15.5.1. 安全连接的初始化

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， 应输入一个密码或口令字。
请留意， 这并不是您用于登录的口令， 它用于生成一次性的登录密钥。 "ID" 这一行给出了所需的参数：
您的登录名， 迭代轮数， 以及种子。 登录系统时， 它能够记住这些参数并呈现给您， 因此无需记忆它们。
最后一行给出了与您的秘密口令对应的、用于登录的一个一次性口令； 如果您立即重新登录，
则它将是您需要使用的那个口令。

15.5.2. 不安全连接初始化

如果您需要通过一个不安全的连接来初始化， 则应首先在安全连接上执行过一次 opiekey；
您可能希望在可信的机器的 shell 提示符下完成。 此外还需要指定一个迭代轮数 (100
也许是一个较好的选择) 也可以选择一个自己的种子， 或让计算机随机生成一个。 在不安全的连接上
(当然是连到您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
```

为了接受默认的种子， 按下 Return （回车）。 在输入访问口令之前， 到一个有安全连接的机器上，
并给它同样的参数：

```
% 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
```

现在回到不安全的连接， 并将生成的一次性口令粘贴到相应的应用程序中。

15.5.3. 生成一个一次性密码

一旦初始化过 OPIE， 当您登录时将看到类似这样的提示：

```
% telnet example.com
Trying 10.0.0.1...
Connected to example.com
Escape character is '^]'.

FreeBSD/i386 (example.com) (ttypa)

login: <username>
otp-md5 498 gr4269 ext
Password:
```

另外， OPIE 提示有一个很有用的特性 (这里没有表现出来)： 如果您在口令提示处按下 `Return` (回车) 系统将回显刚键入的口令， 您可以藉此看到自己所键入的内容。 如果试图手工键入一个一次性密码，这会非常有用。

此时您需要生成一个一次性密码来回答这一提示。 这项工作必须在一个可信的系统上执行 opiekey 来完成。(也可以找到 DOS、 Windows® 以及 Mac OS® 等操作系统上运行的版本)。
这个程序需要将迭代轮数和种子提供给它。 您可以从登录提示那里复制和粘贴它们。

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
```

现在就可以用刚刚获得的一次性口令登录了。

15.5.4. 产生多个一次性口令

有时，会需要到不能访问可信的机器或安全连接的地方。 这种情形下， 可以使用 opiekey 命令来一次生成许多一次性口令。 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 按顺序请求 5 个口令， 30 则指定了最后一个迭代轮数应该是多少。
注意这些口令将按与使用顺序相反的顺序来显示。 如果您比较偏执， 可以手工写下这些结果；
一般来说把它粘贴到 lpr 就可以了。 注意，每一行都显示迭代轮数及其对应的一次性的密码；
一些人建议用完一个就划掉一个。

15.5.5. 限制使用 UNIX® 口令

OPIE 可以对 UNIX® 口令的使用进行基于 IP 的登录限制。 对应的文件是 /etc/opieaccess，
这个文件默认情况下就是存在的。 请参阅 opieaccess(5) 以了解关于这个文件进一步的情况，
以及安全方面需要进行的一些考虑。

下面是一个示范的 opieaccess 文件：

```
permit 192.168.0.0 255.255.0.0
```

这行允许指定 IP 地址的用户 (再次强调这种地址容易被伪造) 在任何时候使用 UNIX® 口令登录。

如果 opieaccess 中没有匹配的规则， 则将默认拒绝任何非 OPIE 登录。

15.6. TCP Wrappers

每一个熟悉 inetd(8) 都应该听说过 TCP Wrappers， 但几乎没有人对它在网络环境中的作用有全面的理解。
几乎每个人都会安装防火墙来处理网络连接， 然而虽然防火墙有非常广泛的用途， 它却不是万能的，
例如它无法处理类似向连接发起者发送一些文本这样的任务。 而 TCP Wrappers
软件能够完成它以及更多的其他事情。 接下来的几段中将讨论许多 TCP Wrappers 提供的功能， 并且，
还给出了一些配置实例。

TCP Wrappers 软件扩展了 inetd 为受其控制的服务程序实施控制的能力。 通过使用这种方法，
它能够提供日志支持、 返回消息给联入的连接、 使得服务程序只接受内部连接， 等等。
尽管防火墙也能够完成其中的某些功能， 但这不仅增加了一层额外的保护，
也提供了防火墙无法提供的功能。

然而， 由 TCP Wrappers 提供的一些额外的安全功能， 不应被视为好的防火墙的替代品。 TCP Wrappers
应结合防火墙或其他安全加强设施一并使用， 为系统多提供一层安全防护。

由于这些配置是对于 inetd 的扩展， 因此， 读者应首先阅读 配置 inetd 这节。

尽管由 inetd(8) 运行的程序并不是真正的 "服务程序"，
但传统上也把它们称为服务程序。 下面仍将使用这一术语。

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 中启动时包含了 -Ww 选项；
这是默认的设置。 当然， 还需要对 /etc/hosts.allow 进行适当的配置， 但 syslogd(8)
在配置不当时会在系统日志中记录相关消息。

与其它的 TCP Wrappers 实现不同， 使用 hosts.deny
在这里被认为是不推荐和过时的做法。 所有的配置选项应放到 /etc/hosts.allow 中。

在最简单的配置中， 服务程序的连接策略是根据 /etc/hosts.allow 允许或阻止。 FreeBSD 中的默认配置是允许一切发到由 inetd 所启动的服务的连接请求。 在基本配置之后将讨论更复杂的情况。

基本配置的形式通常是 服务 : 地址 : 动作。 这里 服务 是从 inetd 启动的服务程序的名字。 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， 分别用于允许和禁止相应的访问。 在配置时您需要注意所有的配置都是按照第一个匹配的规则运转的， 这表示配置文件将按照顺序查找匹配规则， 而一旦找到匹配， 则搜索也就停止了。

另外也有许多其他选项， 这些将在后面介绍。 简单的配置行从上面这些描述之中可以很容易得出。 例如， 允许 POP3 连接通过 mail/qpopper 服务， 应把下面的行添加到 hosts.allow：

```
# This line is required for POP3 connections:
qpopper : ALL : allow
```

增加这样之后， 需要重新启动 inetd。 可以通过使用 kill(1) 命令来完成这项工作， 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选项； 它们能够用来对如何处理连接实施更多的控制。 一些时候， 返回一个说明到特定的主机或请求服务的连接可能是更好的办法。 其他情况下， 记录日志或者发送邮件给管理员可能更为适合。 另外， 一些服务可能只希望为本机提供。 这些需求都可以通过使用 通配符， 扩展字符以及外部命令来实现。 接下来的两节将介绍这些。

15.6.2.1. 外部命令

假设由于发生了某种状况， 而导致连接应该被拒绝掉， 而将其原因发送给发起连接的人。 如何完成这样的任务呢？ 这样的动作可以通过使用 twist 选项来实现。 当发起了连接请求时， twist 将调用一个命令或脚本。 在 hosts.allow 文件中已经给出了一个例子：

```
# The rest of the daemons are protected.
ALL : ALL \
      : severity auth.info \
      : twist /bin/echo "You are not welcome to use %d from %h."
```

这个例子将把消息 "You are not allowed to use daemon from hostname." 返回给访问先前没有配置过允许访问的服务客户。 对于希望把消息反馈给连接发起者， 然后立即切断这样的需求来说， 这样的配置非常有用。 请注意所有反馈信息 必须 被引号 " 包围； 这一规则是没有例外的。

如果攻击者向服务程序发送大量的连接请求， 则可能发动一次成功的拒绝服务攻击。

另一种可能是针对这种情况使用 spawn。 类似 twist， spawn 选项也暗含拒绝连接， 并可以用来执行外部命令或服务。 与 twist 不同的是， spawn 不会向连接发起者发送回应。 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
```

这将拒绝来自 *.example.com 域的所有连接； 同时还将记录主机名， IP 地址，
以及对方所尝试连接的服务名字到 /var/log/connections.log 文件中。

除了前面已经介绍过的转义字符， 例如 %a 之外， 还有一些其它的转义符。 参考 hosts_access(5)
联机手册可以获得完整的列表。

15.6.2.2. 通配符选项

前面的例子都使用了 ALL。 其它选项能够将功能扩展到更远。 例如， ALL 可以被用来匹配每一个服务、
域，或 IP 地址。 另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。
下面的例子将给您更多的感性认识：

```
# 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。 在设置之前， 管理员应该谨慎地考虑。

要了解关于通配符和他们的功能， 请参考 hosts_access(5) 联机手册。

为了使设置能够生效， 应该首先把 hosts.allow 的第一行配置注释掉。 这节的开始部分已经说明了这一点。

15.7. Kerberos5

Kerberos 是一组附加的网络系统/协议， 用以让用户通过一台安全服务器提供的服务来验证身份。
包括远程登录、远程复制、在系统间安全地复制文件， 以及其它高危险性的操作，
由于其存在而显著地提高了安全型并且更加可控。

Kerberos 可以理解为一种身份验证代理系统。 它也被描述为一种以受信第三方为主导的身份验证系统。
Kerberos 只提供一种功能 - 在网络上安全地完成用户的身份验证。 它并不提供授权功能
(也就是说用户能够做什么操作) 或审计功能 (记录用户作了什么操作)。 一旦客户和服务器都使用了
Kerberos 来证明各自的身份之后， 他们还可以加密全部的通讯以保证业务数据的私密性和完整性。

因此， 强烈建议将 Kerberos 同其它提供授权和审计服务的安全手段联用。

接下来的说明可以用来指导如何安装 FreeBSD 所附带的 Kerberos。 不过，
您仍然需要参考相应的联机手册以获得完整的描述。

为了展示 Kerberos 的安装过程， 我们约定：

- DNS 域 ("zone") 为 example.org。
- Kerberos 领域是 EXAMPLE.ORG。

在安装 Kerberos 时请使用实际的域名即使您只是想在内部网上用一用。 这可以避免 DNS
问题并保证了同其它 Kerberos 之间的互操作性。

15.7.1. 历史

Kerberos 最早由 MIT 作为解决网络安全问题的一个方案提出。 Kerberos 协议采用了强加密，
因此客户能够在不安全的网络上向服务器 (以及相反地) 验证自己的身份。

Kerberos 是网络验证协议名字， 同时也是用以表达实现了它的程序的形容词。 (例如 Kerberos telnet)。
目前最新的协议版本是 5，在 RFC 1510 中有所描述。

该协议有许多免费的实现， 这些实现涵盖了许多种不同的操作系统。 最初研制 Kerberos 的麻省理工学院 (MIT) 也仍然在继续开发他们的 Kerberos 软件包。 在 US 它被作为一种加密产品使用， 因而历史上曾经受到 US 出口管制。 MITKerberos 可以通过 port (security/krb5) 来安装和使用。 Heimdal Kerberos 是另一种第 5 版实现， 并且明确地在 US 之外的地区开发， 以避免出口管制 (因此在许多非商业的类 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。

为使尽可能多的读者从中受益， 这份说明以 FreeBSD 附带的 Heimdal 软件包为准。

15.7.2. 配置 Heimdal KDC

密钥分发中心 (KDC) 是 Kerberos 提供的集中式验证服务 - 它是签发 Kerberos tickets 的那台计算机。 KDC 在 Kerberos 领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。

需要说明 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， 首先请确认您的 /etc/rc.conf 文件包含了作为一个 KDC 所需的设置 (您可能需要适当地调整路径以适应自己系统的情况)：

```
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
```

请注意这个 /etc/krb5.conf 文件假定您的 KDC 有一个完整的主机名， 即 kerberos.example.org。 如果您的 KDC 主机名与它不同， 则应添加一条 CNAME (别名) 项到 zone 中去。

对于有正确地配置过的 BINDDNS 服务器的大型网络， 上述例子可以精简为：

```
[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
```

要让客户机能够找到 Kerberos 服务， 就 必须 首先配置完整或最小配置的 /etc/krb5.conf并且 正确地配置 DNS 服务器。

接下来需要创建 Kerberos 数据库。 这个数据库包括了使用主密码加密的所有实体的密钥。
您并不需要记住这个密码， 它会保存在一个文件 (/var/heimdal/m-key) 中。 要创建主密钥， 需要执行 kstash 并输入一个口令。

主密钥一旦建立， 您就可以用 kadmin 程序的 -l 参数 (表示 "local") 来初始化数据库了。 这个选项让 kadmin 直接地修改数据库文件而不是通过 kadmind 的网络服务。
这解决了在数据库创建之前连接它的鸡生蛋的问题。 进入 kadmin 提示符之后， 用 init 命令来创建领域的初始数据库。

最后， 仍然在 kadmin 中， 使用 add 命令来创建第一个 principal。 暂时使用全部的默认设置，
随后可以在任何时候使用 modify 命令来修改这些设置。 另外， 也可以用 ? 命令来了解可用的选项。

典型的数据库创建过程如下：

```
# 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
```

现在是启动 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。 尽管此时还没有任何正在运行的 Kerberos 服务，
但您仍然可以通过获取并列出您刚刚创建的那个 principal (用户) 的 ticket 来验证 KDC 确实在正常工作，
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， 可以撤消这一 ticket：

```
% kdestroy
```

15.7.3. 为 Kerberos 启用 Heimdal 服务

首先我们需要一份 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， 或通过软盘) 复制 KDC 上的版本， 并覆盖掉客户机上的对应文件就可以了。

接下来需要一个 /etc/krb5.keytab 文件。 这是提供 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 得以验证它们的身份。 此文件必须以安全的方式传到服务器上， 因为如果密钥被公之于众， 则安全也就毁于一旦。 也就是说， 通过明文的通道， 例如 FTP 是非常糟糕的想法。

一般来说， 您会希望使用 kadmin 程序来把 keytab 传到服务器上。 由于也需要使用 kadmin 来为主机建立 principal (KDC 一端的 krb5.keytab)， 因此这并不复杂。

注意您必须已经获得了一个 ticket 而且这个 ticket 必须许可使用 kadmind.acl 中的 kadmin 接口。 请参考 Heimdal info 中的 "Remote administration(远程管理)" 一节 (info heimdal) 以了解如何设计访问控制表。 如果不希望启用远程的 kadmin 操作， 则可以简单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 在本地执行管理操作。

安装了 /etc/krb5.conf 文件之后， 您就可以使用 Kerberos 上的 kadmin 了。 add --random-key 命令可以用于添加主机 principal， 而 ext 命令则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 (这是 "extract" 的简写) 默认会把导出的密钥放到 /etc/krb5.keytab 中。

如果您由于没有在 KDC 上运行 kadmind (例如基于安全理由) 因而无法远程地使用 kadmin 您可以直接在 KDC 上添加主机 principal (host/myserver.EXAMPLE.ORG) 随后将其导出到一个临时文件中 (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 或软盘)。 一定要指定一个不同于默认的 keytab 名字以免覆盖 KDC 上的 keytab。

到现在您的服务器已经可以同 KDC 通讯了 (因为已经配置了 krb5.conf 文件)， 而且它还能够证明自己的身份 (由于配置了 krb5.keytab 文件)。 现在可以启用一些 Kerberos 服务。 在这个例子中， 我们将在 /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) 服务来使设置生效：

```
telnet   stream  tcp    nowait  root   /usr/libexec/telnetd  telnetd -a user
```

关键的部分是 -a (表示验证) 类型设置为用户 (user)。 请参考 telnetd(8) 联机手册以了解细节。

15.7.4. 使用 Heimdal 来启用客户端 Kerberos

设置客户机是非常简单的。 在正确配置了 Kerberos 的网络中， 只需要将位于 /etc/krb5.conf 的配置文件进行一下设置就可以了。 这一步骤可以简单地通过安全的方式将文件从 KDC 复制到客户机上来完成。

尝试在客户机上执行 kinit、 klist， 以及 kdestroy 来测试获取、 显示并删除 刚刚为 principal 建立的 ticket 是否能够正常进行， 如果能， 则用其它的 Kerberos 应用程序来连接启用了 Kerberos 的服务。 如果应用程序不能正常工作而获取 ticket 正常， 则通常是服务本身， 而非客户机或 KDC 有问题。

在测试类似 telnet 的应用程序时， 应考虑使用抓包程序 (例如 tcpdump(1)) 来确认您的口令没有以明文方式传输。 尝试使用 telnet 的 -x 参数， 它将加密整个数据流 (类似 ssh)。

许多非核心的 Kerberos 客户应用程序也是默认安装的。 在 Hemidal 的 "最小" 安装理念下， telnet 是唯一一个采用了 Kerberos 的服务。

Heimdal port 则提供了一些默认不安装的客户应用程序， 例如启用了 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 客户应用程序。

15.7.5. 用户配置文件： .k5login 和 .k5users

在某个领域中的用户往往都有自己的 Kerberos principal (例如 tillman@EXAMPLE.ORG) 并映射到本机用户帐户 (例如本机上名为 tillman 的帐户)。 客户端应用程序， 如 telnet 通常并不需要用户名或 principal。

不过, 有时您可能会需要赋予某些没有匹配 Kerberos principal 的人使用本地用户帐户的权限。 例如 tillman@EXAMPLE.ORG 可能需要访问本地的 webdevelopers 用户帐号。 其它 principal 可能也会需要访问这个本地帐号。

用户 home 目录中的 .k5login 和 .k5users 这两个文件可以配合 .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， 则列出的两个 principals 都可以使用那个帐号， 而无须共享口令。

建议您在开始实施之前首先阅读这些命令的联机帮助。 特别地， ksu 的联机手册包括了 .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， 需要确认 PATH 环境变量把 Kerberos 客户应用列在系统自带的版本之前。
- 同一领域内的所有计算机的时间设置是否同步？ 如果不是的话， 则身份验证可能会失败。 通过 NTP 进行时钟同步 描述了如何使用 NTP 来同步时钟。
- MIT 和 Heimdal 能够很好地互操作。 一个例外是 kadmin， 因为这个协议没有被标准化。
- 如果您改变了主机名， 您还需要修改您的 host/ principal 并更新 keytab。 这一规律也适用于类似 Apache 的 www/mod_auth_kerb 所使用的 www/ principal 这样的特殊 keytab 项。
- 您的领域中的每一台主机必须在 DNS (或至少在 /etc/hosts 中) 可以解析 (同时包括正向和反向)。

CNAME 能够正常使用， 但必须有正确的对应 A 和 PTR 记录。 此时给出的错误信息可能很让人困惑：Kerberos5 refuses authentication because Read req failed: Key table entry not found。

- 某些作为客户使用您的 KDC 的操作系统可能没有将 ksu 设置为 setuid root 的权限。 这意味着 ksu 将不能够正常工作， 从安全角度说这是一个不错的主意， 但可能令人烦恼。 这类问题并不是 KDC 的错误。
- 使用 MITKerberos 时， 如果希望允许一个 principal 拥有超过默认的十小时有效期的 ticket 则必须使用 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长有效期的 ticket。 *

如果在 KDC 上运行了听包程序， 并在工作站上执行 kinit， 您可能会注意到 TGT 是在 kinit 一开始执行的时候就发出了的 - 甚至在您输入口令之前！ 关于这个现象的解释是 Kerberos 服务器可以无限制地收发 TGT (Ticket Granting Ticket) 给任何未经授权的请求； 但是， 每一个 TGT 都是使用用户的口令派生出来的密钥进行加密的。 因此， 当用户输入口令时它并不会发送给 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时间戳的有效 ticket， 则说明用户的 Kerberos 凭据有效。 这些凭据包含了一个会话密钥用以在随后建立 Kerberos 服务器的加密通讯， 传递由服务器自己的私钥加密的实际的 ticket-granting ticket。 这个第二层加密对于用户来说是看不到的， 但它使得 Kerberos 服务器能够验证每一个 TGT 的真实性。

- 如果需要有效期更长的 ticket (例如一周) 而且您使用 OpenSSH 连接保存您的 ticket 的机器， 请确认 sshd_config 中的 Kerberos TicketCleanup 被设置为 no 否则在注销时会自动删除所有的 ticket。
- 切记主机的 principals 的 ticket 有效期一定要比用户的长。 如果您的用户 principal 的有效期是一周， 而所连接的主机的有效期是九个小时， 则缓存的主机 principal 将先行过期， 结果是 ticket 缓存无法正常工作。
- 当配置 krb5.dict 文件来防止使用特定的简单口令 (kadmind 的联机手册中简要介绍了它)， 请切记只有指定了口令策略的 principals 才会使用它们。 krb5.dict 文件的格式很简单：每个串占一行。 创建一个到 /usr/shared/dict/words 的符号连接会很有用。

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) 的命令和协议。 如果您的 KDC 是 MIT 的， 则其影响是不能使用 Heimdal 的 kadmin 程序来远程管理 KDC (或相反)。

完成同样工作的命令可能会有些许的不同。 推荐按照 MITKerberos 的网站 (http://web.mit.edu/Kerberos/www/) 上的说明来操作。 请小心关于路径的问题， MIT port 会默认安装到 /usr/local/， 您因此可能会执行 "普通的" 系统应用程序而非 MIT, 如果您的 PATH 环境变量把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， 如果您想知道为什么通过 telnetd 和 klogind 登录时会出现一些诡异的现象的话。 最重要地， "incorrect permissions on cache file(缓存文件权限不正确)" 行为需要使用 login.krb5 来进行验证， 才能够正确地修改转发凭据的属主。

除此之外， 还应修改 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. 缓解 Kerberos 的限制

15.7.8.1. Kerberos 是一种 all-or-nothing 方式

在网络上启用的每个服务都必须进行修改以便让其能够配合 Kerberos 工作
(否则就只能使用其它方法来保护它们不受网络攻击的侵害)， 如果不是这样，
则用户的凭据就有可能被窃取并再次使用。 一个例子是对所有的远程 shell (例如通过 rsh 和 telnet) 启用了 Kerberos 但没有将使用明文验证的 POP3 邮件服务器 Kerberos化。

15.7.8.2. Kerberos 是为单用户工作站设计的

在多用户环境中 Kerberos 的安全性会被削弱。 这是因为它把 ticket 保存到 /tmp 目录中，
而这个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)，
则这个用户的 ticket 就可能被其它用户窃取 (复制)。

可以通过使用 -c 文件名 这样的命令行选项， 或者(推荐的)改变 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 目录并简单地设置权限就能够缓解这个问题。

15.7.8.3. KDC 会成为单点崩溃故障点

根据设计， KDC 必须是安全的， 因为主密码数据库保存在它上面。 决不应该在 KDC上面运行其它服务，
而且还应确保它的物理安全。 由于 Kerberos 使用同一个密钥 (传说中的那个 "主" 密钥)
来加密所有的密码， 而将这个文件保存在 KDC， 因此其安全尤为重要

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
数据库以及产生随机数发生器的种子。 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)
则网络服务将由于验证服务无法进行而不能使用， 从而导致更大范围的拒绝服务攻击。 通过部署多个 KDC
(一个主服务器， 配合一个或多个从服务器) 并采用经过仔细设计和实现的备用验证方式可以避免这种问题
(PAM 是一个不错的选择)。

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、主机或服务验证
KDC。 这意味着种过木马的程序，例如 kinit 有可能记录用户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检查工具来避免此类情况的发生。

15.7.9. 相关资源和其它资料

- 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讯层基础上的加密传输层； 这些功能为许多网络应用和服务程序所广泛使用。

对 OpenSSL 的一些常见用法包括加密邮件客户的身份验证过程， 基于 Web 的交易如信用卡等等。 许多 ports 如 www/apache13-ssl， 以及 mail/claws-mail 等等都提供了编译进 OpenSSL 支持的方法。

绝大多数情况下 Ports Collection 会试图使用 security/openssl 除非明确地将 WITH_OPENSSL_BASE make 变量设置为 "yes"。

FreeBSD 中附带的 OpenSSL 版本能够支持 安全套接字层 v2/v3 (SSLv2/SSLv3) 和 安全传输层 v1 (TLSv1)

三种网络协议， 并可作为通用的密码学函数库使用。

尽管 OpenSSL 支持 IDEA 算法， 但由于美国专利， 它在默认情况下是不编译的。 如果想使用它， 请查阅相应的授权， 如果认为授权可以接受， 则可以在 make.conf 中设置 MAKE_IDEA。

为应用软件提供证书是 OpenSSL 最为常用的功能之一。
证书是一种能够确保公司或个人有效身份不被伪造的凭据。 如果证书没有被众多 "权威发证机构"， 或 CA 中的某一个确认， 则会产生一个警告。 权威发证机构通常是一家公司， 例如 VeriSign，
它能够通过签署来证明个人或公司证书的有效性。 这个过程是需要付费的， 当然，
这不是使用证书的必要条件； 然而， 这样做会让那些比较偏执的用户感到轻松。

15.8.1. 生成证书

为了生成证书， 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
```

请注意， 在 "Common Name" 提示后面我们输入的是一个域名。 这个提示要求输入服务器的名字，
这个名字今后将用于完成验证过程； 如果在这里输入域名以外的内容， 那么证书也就失去其意义了。
您还可以指定一些其他的选项， 比如证书的有效期， 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) 联机手册中找到。

在您执行前述命令的目录中将生成两个文件。 证书申请， 即 req.pem， 可以发给一家发证机构，

它将验证您输入的凭据的真实性， 并对申请进行签名， 再把证书返还给您。 第二个文件的名字将是 cert.pem， 它包含了证书的私钥， 应被全力保护； 如果它落入别人手中， 则可以被用来伪造您 (或您的服务器)。

如果不需要来自 CA 的签名， 也可以创建自行签名的证书。 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
```

然后用这个密钥来创建证书：

```
# openssl req -new -x509 -days 365 -key myca.key -out new.crt
```

上述步骤将在当前目录中生成两个新文件： 一个是权威发证机构的签名文件， myca.key； 另一个是证书本身， new.crt。 这些文件应该放到同一个目录中， 一般而言， 推荐放到 /etc， 并且只允许 root 读取。 建议把权限设置为 0700， 这可以通过 chmod 工具来完成。

15.8.2. 使用证书的一个例子

那么有了这些文件可以做些什么呢？ 一个比较典型的用法是用来加密 SendmailMTA 的通讯连接。 这可以解决用户通过本地 MTA 发送邮件时使用明文进行身份验证的问题。

这个用法可能并不完美， 因为某些 MUA 会由于没有在本地安装证书而向用户发出警告。 请参考那些软件的说明了解关于安装证书的信息。

下面的设置应添加到本地的 .mc 文件

```
dnl SSL Options
define(`confCACERT_PATH',`/etc/certs')dnl
define(`confCACERT',`/etc/certs/new.crt')dnl
define(`confSERVER_CERT',`/etc/certs/new.crt')dnl
define(`confSERVER_KEY',`/etc/certs/myca.key')dnl
define(`confTLS_SRV_OPTIONS', `V')dnl
```

这里， /etc/certs/ 是准备用来在本地保存证书和密钥的位置。 最后， 需要重新生成本地的 .cf 文件。 这一工作可以简单地通过在 目录中执行 makeinstall 来完成。 接下来， 可以使用 makerestart 来重新启动 Sendmail 服务程序。

如果一切正常的话， 在 /var/log/maillog 中就不会出现错误提示， Sendmail 也应该出现在进程列表中。

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" 则说明一切正常。

15.9. IPsec 上的 VPN

使用 FreeBSD 网关在两个被 Internet 分开的网络之间架设 VPN。

15.9.1. 理解 IPsec

这一节将指导您完成架设 IPsec。为了配置 IPsec， 您应当熟悉如何编译一个定制的内核的一些概念 (参见 配置FreeBSD的内核)。

IPsec 是一种建立在 Internet 协议 (IP) 层之上的协议。 它能够让两个或更多主机以安全的方式来通讯 (并因此而得名)。 FreeBSD IPsec "网络协议栈" 基于 KAME 的实现， 它支持两种协议族， 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， 通过使用对称加密算法 (例如 Blowfish、 3DES)。
- Authentication Header (AH), 保护 IP 包头不被第三方介入和伪造， 通过计算校验和以及对 IP 包头的字段进行安全散列来实现。 随后是一个包含了散列值的附加头， 以便能够验证包。

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讯 (也就是 传输模式)； 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讯 (也就是 隧道模式)。 后一种更多的被称为是 虚拟专用网 (VPN)。 ipsec(4) 联机手册提供了关于 FreeBSD 中 IPsec 子系统的详细信息。

要把 IPsec 支持放进内核， 应该在配置文件中加入下面的选项：

```
options   IPSEC     IP security
device    crypto
```

如果需要 IPsec 的调试支持， 还应增加：

```
options   IPSEC_DEBUG  debug for IP security
```

15.9.2. 问题

由于对如何建立 VPN 并不存在标准， 因此 VPN 可以采用许多种不同的技术来实现，
每种技术都有其强项和弱点。 这篇文章将展现一个具体的应用情景， 并为它设计了适合的 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，并且通过这条 VPN 就像在同一个网络一样。

现有条件如下：

- 至少有两个不同的站点
- 每个站点都使用内部的 IP
- 两个站点都通过运行 FreeBSD 的网关接入 Internet。
- 每个网络上的网关至少有一个公网的 IP 地址。
- 网络的内部地址可以是公网或私有的 IP 地址， 这并不是问题。它们并不冲突，比如它们不同时使用 192.168.1.x 这样的地址。

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够帮助配置的应用程序。

下一步是创建两个 gif(4) 伪设备用来在两个网络间传输数据包的 "隧道"。 使用 root 身份运行以下命令，
并用真实的内部外部网关替换命令中的 internal 和 external 项：

```
# ifconfig gif0 create
```

```
# ifconfig gif0 internal1 internal2
```

```
# ifconfig gif0 tunnel external1 external2
```

比如，公司 LAN 对外的 IP 地址是 172.16.5.4， 内部的 IP 地址为 10.246.38.1。 家庭 LAN 对外的 IP 地址是 192.168.1.12， 内部的 IP 地址为 10.0.0.5。

这看起来可能有些混乱，所以我们通过 ifconfig(8) 命令输出再回顾一下：

```
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 0xffffff00

Gateway 2:
```

```
gif0: flags=8051 mtu 1280
tunnel inet 192.168.1.12 --> 172.16.5.4
inet 10.0.0.5 --> 10.246.38.1 netmask 0xffffff00
inet6 fe80::250:bfff:fe3a:c1f%gif0 prefixlen 64 scopeid 0x4
```

一旦完成以后，两个私有的 IP 地址都应该能像下面 ping(8) 命令输出那样互相访问。

```
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
```

正如预期的那样，两边都有从私有地址发送和接受 ICMP 数据包的能力。下面，
两个网关都必须配置路由规则以正确传输两边的网络流量。 下面的命令可以实现这个：

```
# 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，不论从网关还是网关后的机器都能访问内部的网络。 这很容易通过以下的例子确认：

```
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。 配置一条安全链接则是个更加深入的过程。 下面的配置是使用 pre-shared（PSK） RSA 密钥。除了 IP 地址外，两边的 /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;
}
```

解释所有可用的选项， 连同这些例子里列出的都超越了这份文档的范围。 在 racoon

配置手册页中有着丰富的相关信息。

SPD 策略也需要配置一下， 这样 FreeBSD 和 racoon 就能够加密和解密主机间的网络流量了。

这可以通过在公司的网关上运行一个类似下面简单的 shell 脚本实现。保存到
/usr/local/etc/racoon/setkey.conf， 这个文件会被在系统初始化的时候用到。

```
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两边的网关上都启动 racoon：

```
# /usr/local/sbin/racoon -F -f /usr/local/etc/racoon/racoon.conf -l /var/log/racoon.log
```

输出将会类似这样的：

```
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)
```

确认一下 "隧道" 能正常工作， 切换到另外一个控制台用如下的 tcpdump(1)
命令查看网络流量。根据需要替换掉下面的 em0 网卡界面。

```
# 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，调试的话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应该有防火墙的保护。 要使得这两个网络能互相访问，就需要添加一些进出包的规则。 就 ipfw(8) 来说，加入下面的几行进配置文件：

```
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
```

规则号可能需要根据现有机器上的配置做相应的修改。

对于 pf(4) 或者 ipf(8) 的用户， 下面的几行规则应该可行：

```
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 是一组用于安全地访问远程计算机的连接工具。 它可以作为 rlogin、 rsh rcp 以及 telnet 的直接替代品使用。 更进一步， 其他任何 TCP/IP 连接都可以通过 SSH 安全地进行隧道/转发。 OpenSSH 对所有的传输进行加密， 从而有效地阻止了窃听、 连接劫持， 以及其他网络级的攻击。

OpenSSH 由 OpenBSD project 维护， 它基于 SSH v1.2.12 并包含了最新的错误修复和更新。 它同时兼容 SSH 协议的 1 和 2 两个版本。

15.10.1. 使用 OpenSSH 的好处

一般说来， 在使用 telnet(1) 或 rlogin(1) 时， 数据是以未经加密的明文的形式发送的。 这样一来，
在客户机和服务器之间的网络上运行的听包程序， 便可以在会话中窃取到传输的用户名/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 的启用是作为 FreeBSD 安装中 Standard 安装过程中的一步来进行的。 要查看 sshd 是否已被启用，
请检查 rc.conf 文件中的：

```
sshd_enable="YES"
```

这表示在下次系统启动时加载 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

ssh(1) 的工作方式和 rlogin(1) 非常类似。

```
# 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 建立的会话非常类似。 在连接时， SSH
会利用一个密钥指纹系统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。
之后的连接将会验证预先保存下来的密钥指纹。 如果保存的指纹与登录时接收到的不符， 则将会给出警告。
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， 较新版本的 OpenSSH 只接受 SSH v2 连接。 如果能用版本 2 则客户程序会自动使用，
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 来相应地强制使用版本 1 或 2。
保持客户端的版本 1 能力是为了考虑较早版本的兼容性。

15.10.4. 安全复制

scp(1) 命令和 rcp(1); 的用法类似， 它用于将文件复制到远程的机器上， 或复制过来， 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经保存了指纹， 使用 scp(1) 时会自动地加以验证。

scp(1) 使用的参数同 cp(1) 类似。 第一个参数是一个或一组文件， 然后是复制的目标。 由于文件是通过

SSH 在网上传递的， 因此某些文件的名字需要写成 用户名@主机名:<远程文件路径>。

15.10.5. 配置

针对 OpenSSH 服务程序和客户端的系统级配置文件在 /etc/ssh 目录中。

ssh_config 用于配置客户端的设定， 而 sshd_config 则用于配置服务器端。

另外 sshd_program (默认是 /usr/sbin/sshd)， 以及 sshd_flags 这两个 rc.conf 选项提供了更多的配置选择。

15.10.6. ssh-keygen

用于取代口令的一种方法是使用 ssh-keygen(1) 来生成 DSA 或 RSA 密钥对用于验证用户的身份：

```
% 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钥对用于验证身份。 私钥将保存到 ~/.ssh/id_dsa 或 ~/.ssh/id_rsa，而公钥则被存放到 ~/.ssh/id_dsa.pub 或 ~/.ssh/id_rsa.pub， 文件名取决于您选择的 DSA 和 RSA 密钥类型。 RSA 或者 DSA 公钥必须被存放到远程机器上的 ~/.ssh/authorized_keys 才能够使系统正确运转。

这将允许从远程连接时以基于 SSH 密钥的验证来代替口令验证。

如果在 ssh-keygen(1) 中使用了通行字， 则每次使用私钥时都需要输入它。 ssh-agent(1) 能够缓解多次输入长通行字的压力， 并将在接下来的 ssh-agent 和 ssh-add 予以详述。

选项和配置文件可能随 OpenSSH 的版本不同而不同； 为了避免出现问题， 您应参考 ssh-keygen(1) 联机手册。

这将使到远程机器的连接基于 SSH 密钥而不是口令。

如果在运行 ssh-keygen(1) 时使用了通行字， 每次使用私钥的时候用户都将被要求输入通行字。 ssh-agent(1) 能够减缓重复输入较长通行字的负担， 有关更详细的探究在 ssh-agent 和 ssh-add 下一节 .

随着你系统上的 OpenSSH 版本的不同，各种选项和配置文件也会不同；为了避免此类问题， 你需要参阅 ssh-keygen(1) 联机手册。

15.10.7. ssh-agent 和 ssh-add

ssh-agent(1) 和 ssh-add(1) 这两个工具， 提供了一种将 SSH 秘钥加载到内存中以便使用，而不必每次都输入通行字的方法。

ssh-agent(1) 工具能够使用加载到其中的私钥来处理验证过程。 ssh-agent(1) 应被用于启动另一个应用程序。 最基本的用法是， 使用它来启动 shell，

而高级一些的用法则是用它来启动窗口管理器。

要在 shell 中使用 ssh-agent(1)， 首先应把 shell 作为参数来启动它。 随后， 应通过 ssh-add(1) 并输入通行字， 来向它提供身份验证信息。 一旦这些步骤都做完了， 用户就应该能够 ssh(1) 到任何一个安装了对应公钥的机器了。 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)， 调用 ssh-agent(1) 的过程应置于 ~/.xinitrc 之中。 这将把 ssh-agent(1) 服务提供给所有在 X11 中运行的程序。 下面是一个 ~/.xinitrc 文件的实例：

```
exec ssh-agent startxfce4
```

这将启动 ssh-agent(1)， 而后者将在每次 X11 启动时运行 XFCE。 作完这些之后就可以重启 X11 以便使修改生效。 随后您就可以运行 ssh-add(1) 来加载全部 SSH 密钥了。

15.10.8. SSH 隧道

OpenSSH 能够创建隧道以便用加密的会话来封装其他协议。

下面的命令告诉 ssh(1) 为 telnet 创建一个隧道：

```
% 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
: 强制 ssh 使用第2版的协议 (如果需要和较老的 SSH 一同工作请不要使用这个选项)。

-N
: 表示不使用命令行， 或者说只使用隧道。 如果省略， ssh 将同时初始化会话。

-f
: 强制 ssh 在后台执行。

-L
: 表示产生一条 本地端口:远程主机:远程端口 形式的隧道。

user@foo.example.com
: 远程 SSH 服务器。

SSH 隧道通过监听 localhost 上面指定端口来完成工作。 它将把本机主机/端口上接收到的连接通过 SSH 连接转发到远程主机/端口。

本例中， 位于 localhost 的 5023 端口 被用于转发 localhost 的连接到远程主机的 23 端口。 由于 23 是 telnet 使用的， 因此它将通过 SSH 隧道完成 telnet 会话。

这可以用来封装任意不安全的 TCP 协议， 例如 SMTP、 POP3、 FTP等等。

例 23. 使用 SSH 为 SMTP 创建安全隧道

```
% 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 ESMTP
```

这可以与 ssh-keygen(1) 以及额外的用户帐号配合来建立一个更透明的 SSH 隧道环境。
密钥可以被用在需要输入口令的地方， 而且可以为不同的用户配置不同的隧道。

15.10.8.1. 实用的 SSH 通道例子

15.10.8.1.1. 加强 POP3 服务的安全

工作时， 有一个允许外来连接的 SSH 服务器。 同一个办公网络中有一个邮件服务器提供 POP3 服务。
这个网络， 或从您家到办公室的网络可能不， 或不完全可信。 基于这样的原因，
您需要以安全的方式来查看邮件。 解决方法是创建一个到办公室 SSH 服务器的连接，
并通过这个连接来访问 POP3 服务：

```
% ssh -2 -N -f -L 2110:mail.example.com:110 user@ssh-server.example.com
user@ssh-server.example.com's password: ******
```

当这个通道连上时， 您可以把 POP3 请求发到 localhost 端口 2110。 这个连接将通过通道安全地转发到 mail.example.com。

15.10.8.1.2. 绕过严厉的防火墙

一些大脑长包的网络管理员会使用一些极端的防火墙策略， 不仅过滤进入的连接， 而且也过滤连出的连接。
一些时候您可能只能连接远程机器 22 端口，以及 80 端口用来进行 SSH 和网页浏览。

您可能希望访问一些其它的 (也许与工作无关的) 服务， 例如提供音乐的 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乐， 则您将无法访问它。

解决方法是建立一个到您的网络的防火墙之外的网络上的 SSH 服务器， 并通过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: *******
```

现在您可以把客户程序指定到 localhost 的 8888 端口， 它将把请求转发给 music.example.com 的 8000 端口， 从而绕过防火墙。

15.10.9. 允许用户登录 AllowUsers 选项

通常限制哪些用户能够登录， 以及从何处登录会是好主意。 采用 AllowUsers
选项能够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 文件中加入下述设置：

```
AllowUsers root@192.168.1.32
```

要允许用户 admin 从任何地方登录， 则只需列出用户名：

```
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； 否则他们将被锁在外面。

在完成对 /etc/ssh/sshd_config 的修改之后您必须告诉 sshd(8) 重新加载其配置文件， 方法是执行：

```
# /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. 文件系统访问控制表

与文件系统在其他方面的加强， 如快照等一道， FreeBSD 提供了通过文件系统访问控制表 (ACL) 实现的安全机制。

访问控制表以高度兼容 (POSIX®.1e) 的方式扩展了标准的 UNIX® 权限模型。 这一特性使得管理员能够利用其优势设计更为复杂的安全模型。

如果想为 UFS 文件系统启用 ACL 支持， 则需要添加下列选项：

```
options UFS_ACL
```

并重新编译内核。 如果没有将这个选项编译进内核， 则在挂接支持 ACL 的文件系统时将会收到警告。 这个选项在 GENERIC 内核中已经包含了。 ACL 依赖于在文件系统上启用扩展属性。 在新一代的 UNIX® 文件系统， UFS2 中内建了这种支持。

在 UFS1 上配置扩展属性需要比 UFS2 更多的管理开销。 而且， 在 UFS2 上的扩展属性的性能也有极大的提高。 因此， 如果想要使用访问控制表， 推荐使用 UFS2 而不是 UFS1。

ACL 可以在挂接时通过选项 acls 来启动， 它可以加入 /etc/fstab。 另外， 也可以通过使用 tunefs(8) 修改超级块中的 ACL 标记来持久性地设置自动的挂接属性。 一般而言， 后一种方法是推荐的做法， 其原因是：

- 挂接时的 ACL 标记无法被重挂接 (mount(8) -u) 改变， 只有完整地 umount(8) 并做一次新的 mount(8) 才能改变它。 这意味着 ACL 状态在系统启动之后就不可能在 root 文件系统上发生变化了。

另外也没有办法改变正在使用的文件系统的这个状态。

- 在超级块中的设置将使得文件系统总被以启用 ACL 的方式挂接，即使在 fstab 中的对应项目没有作设置，或设备顺序发生变化时也是如此。这避免了不慎将文件系统以没有启用 ACL 的状态挂接，从而避免没有强制 ACL 这样的安全问题。

可以修改 ACL 行为，以允许在没有执行一次全新的 mount(8) 的情况下启用它，但我们认为，不鼓励在未启用 ACL 时这么做是有必要的，因为如果启用了 ACL，然后关掉它，然后在没有刷新扩展属性的情况下重新启用它是很容易造成问题的。一般而言，一旦启用了文件系统的 ACL 就不应该再关掉它，因为此时的文件系统的保护措施可能和用户所期待的样子不再兼容，而重新启用 ACL 将重新把先前的 ACL 附着到文件上，而由于它们的权限发生了变化，就很可能造成无法预期的行为。

在查看目录时，启用了 ACL 的文件将在通常的属性后面显示 + (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
```

这里我们看到了 directory1、directory2，以及 directory3 目录使用了 ACL。而 public_html 则没有。

15.11.1. 使用 ACL

文件系统 ACL 可以使用 getfacl(1) 工具来查看。例如，如果想查看 test 的 ACL 设置，所用的命令是：

```
% getfacl test
  #file:
  #owner:1001
  #group:1001
  user::rw-
  group::r--
  other::r--
```

要修改这个文件上的 ACL 设置，则需要使用 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。一般来说应该使用 -b 因为它会保持让 ACL 正常工作的那些项不变。

```
% setfacl -m u:trhodes:rwx,group:web:r--,o::--- test
```

在前面的命令中，-m 选项被用来修改默认的 ACL 项。由于已经被先前的命令删除，因此没有预先定义的项，于是默认的选项被恢复，并附加上指定的选项。请小心地检查，如果您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 的错误提示。

15.12. 监视第三方安全问题

过去几年中， 安全领域在如何处理漏洞的评估方面取得了长足的进步。
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，
然而为每一个第三方工具都发布安全公告则超出了 FreeBSD Project 的能力。 在这一前提下，
一种减轻第三方漏洞的威胁， 并警告管理员存在已知的安全问题的方法也就应运而生。 名为 Portaudit 的 FreeBSD 附加工具能够帮助您达成这一目的。

ports-mgmt/portaudit port 会下载一个数据库， 这一数据库是由 FreeBSD Security Team 和 ports 开发人员维护的， 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， 以便让 Portaudit 能够在每天的安全审计过程中运行。
一定要保证发到 root 帐号的每日安全审计邮件确实有人在读。 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库，
并查看目前安装的软件包中所存在的已知安全漏洞：

```
# portaudit -Fda
```

由于每天执行 periodic(8) 时都会自动更新数据库， 因此， 运行这条命令是可选的。
在这里只是作为例子给出。

在任何时候， 如果希望对通过 Ports Collection 安装的第三方软件工具进行审计，
管理员都可以使用下面的命令：

```
# portaudit -a
```

针对存在漏洞的软件包， Portaudit 将生成类似下面的输出：

```
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.
```

通过访问上面给出的 URL， 管理员能够了解关于那个漏洞的进一步信息。 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。

简而言之， Portaudit 是一个强大的工具， 并能够配合 Portupgrade port 来非常有效地工作。

15.13. FreeBSD 安全公告

像其它具有产品级品质的操作系统一样， FreeBSD 会发布 "安全公告"。
通常这类公告会只有在相应的发行版本已经正确地打过补丁之后发到安全邮件列表并在勘误中说明。
本节将介绍什么是安全公告， 如何理解它， 以及为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 安全公告的样式类似下面的范例， 这一例子来自 FreeBSD 安全问题通知邮件列表 邮件列表。

```
=============================================================================
=
FreeBSD-SA-XX:XX.UTIL                          Security Advisory
                                    The FreeBSD Project

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(标题) 一栏说明了问题到底是什么。 它基本上是对所发现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， 这一栏可能是 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 的软件， 例如 sendmail。 最后是 ports， 它表示该弱点影响了 Ports Collection 中的某个第三方软件。

③ Module(模块) 一栏给出了组件的具体位置， 例如 sys。 在这个例子中， 可以看到 sys 模块是存在问题的； 因此， 这个漏洞会影响某个在内核中的组件。

④ Announced(发布时间) 一栏反映了与安全公告有关的数据是什么时候公之于众的。 这说明安全团队已经证实问题确实存在， 而补丁已经写入了 FreeBSD 的代码库。

⑤ Credits(作者) 一栏给出了注意到问题存在并报告它的个人或团体。

⑥ The Affects(影响范围) 一栏给出了 FreeBSD 的哪些版本存在这个漏洞。 对于内核来说， 检视受影响的文件上执行的 ident 输出可以帮助确认文件版本。 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 代码库同步并每日构建， 它很可能是有问题的。

⑦ Corrected(修正时间) 一栏给出了发行版本中修正问题的具体日期、时间和时差。 在公共漏洞数据库 (Common Vulnerabilities Database) 系统中预留的， 用于查看漏洞的标识信息。

⑧ Background(技术背景) 一栏提供了受影响的组件的作用。 多数时候这一部分会说明为什么 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) 一栏深入阐述安全漏洞的技术细节。 这部分有时会包括有问题的代码相关的详细情况， 甚至是这个部件如何能够被恶意利用并打开漏洞的细节。

⑩ Impact(影响) 一栏描述了问题能够造成的影响类型。 例如， 可能导致拒绝服务攻击， 权限提升， 甚至导致得到超级用户的权限。

⑪ Workaround(应急方案) 一栏给出了系统管理员在暂时无法升级系统时可以采取的临时性对策。 这些原因可能包括时间限制， 网络资源的限制， 或其它因素。 不过无论如何， 安全不能够被轻视， 有问题的系统要么应该打补丁， 要么应该实施这种应急方案。

⑫ Solution(解决方案) 一栏提供了如何给有问题的系统打补丁的方法。 这是经过逐步测试和验证过的给系统打补丁并让其安全地工作的方法。

⑬ =Correction Details(修正细节) 一栏展示了针对 CVS 分支或某个发行版的修正特征。 同时也提供了每个分支上相关文件的版本号。

⑭ References(文献) 一栏通常会给出其它信息的来源。 这可能包括 URL， 书籍、 邮件列表以及新闻组。

15.14. 进程记帐

进程记帐是一种管理员可以使用的跟踪系统资源使用情况的手段， 包括它们分配给了哪些用户、 提供系统监视手段， 并且可以精细到用户执行的每一个命令。

当然， 这种做法是兼有利弊的。 它的好处是， 查找入侵时可以迅速把范围缩小到攻击者进入的时刻； 而这样做的缺点， 则是记帐会产生大量的日志， 因而需要很多磁盘空间来存储它们。 这一节将带领管理员一步一步地配置基本的进程记帐。

15.14.1. 启用并利用进程记帐

在使用进程记帐之前， 必须先启用它。 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
```

一旦启用之后， 记帐就会开始跟踪 CPU 统计数据、 命令， 等等。
所有的记帐日志不是以可读的方式记录的， 要查看它们， 需要使用 sa(8) 这个工具。
如果没有给出其他参数， 则 sa 将按用户， 以分钟为单位显示他们所使用的时间、 总共的 CPU
和用户时间， 以及平均的 I/O 操作数目， 等等。

要显示关于刚刚发出的命令的相关信息， 则应使用 lastcomm(1) 工具。 lastcomm
命令可以用来显示在某一 ttys(5) 上的用户信息， 例如：

```
# lastcomm ls
  trhodes ttyp1
```

将会显示出所有已知的 trhodes 在 ttyp1 终端上执行 ls 的情况。

更多的可用选项在联机手册 lastcomm(1)、 acct(5) 和 sa(8) 中有所介绍。

Chapter 16. Jails

16.1. 概述

这一章将为您介绍 FreeBSD jail 是什么， 以及如何使用它们。 Jail， 有时也被认为是对 chroot 环境的一种增强型替代品， 对于管理员而言是非常强大的工具， 同时， 它的一些基本用法， 对高级用户而言也相当有用。

读完这章， 您将了解：

- jail 是什么， 以及它在您安装的 FreeBSD 中所能发挥的作用。
- 如何联编、 启动和停止 jail。
- 如何从 jail 内部或主机上进行管理的一些基础知识。

其他一些能够为您提供关于 jail 的有用信息的地方还有：

- jail(8) 联机手册。 这是关于 jail - 用于在 FreeBSD 中启动、 停止和控制 FreeBSD jails - 工具的完整说明书。
- 邮件列表及其存档。 由 FreeBSD 邮件列表服务器 提供的 FreeBSD 一般问题邮件列表 和其他邮件列表的存档， 已经包含了一系列关于 jails 的有价值的信息。 通常搜索存档或询问 freebsd-questions 邮件列表能够给您带来很多有用的信息。

16.2. 与 Jail 相关的一些术语

为了帮助您更好地理解与 jail 有关的 FreeBSD 系统知识， 以及它们如何与 FreeBSD 的其它部分相互作用， 您应理解下列术语：

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) (环境)
: 在 "chroot" 中运行的进程环境。 这包括类似文件系统中的可见部分、 可用的用户及用户组 ID、 网络接口以及其他 IPC 机制等资源。

jail(8) (命令)
: 用以在 jail 环境中运行进程的系统管理工具。

宿主 (系统、 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， 并能够控制 jail 环境内外的进程。 宿主系统与 jail 的一项重要区别是， 在宿主系统中的超级用户进程， 并不像在 jail 中那样受到一系列限制。

hosted (系统、 进程、 用户等等)
: 可访问资源受 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费解的任务， 因此人们开发了一系列强大的工具， 来让管理员的工作变得更加简单。 这些改进通常是让系统能够以更简单的方式安装、 配置， 并毫无问题地持续运转。 这其中， 许多管理员希望能够为系统正确地进行安全方面的配置， 使其能够用于真正的用途， 而阻止安全方面的风险。

FreeBSD 系统提供的一项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进， 使得它成为了一个强大而灵活的系统。 目前仍然在对其进行持续的开发， 以提高其可用性、 性能和安全性。

16.3.1. Jail 是什么

BSD-类的操作系统从 4.2BSD 开始即提供了 chroot(8)。 chroot(2) 工具能够改变一组进程的根目录的位置， 从而建立一个与系统中其他部分相隔离的安全环境： 在 chroot 环境中的进程， 将无法访问其外的文件或其他资源。 正是由于这种能力， 即使攻击者攻破了某一个运行于 chroot 环境的服务， 也不能攻破整个系统。 chroot(8) 对于那些不需要很多灵活性或复杂的高级功能的简单应用而言相当好用。 另外， 在引入 chroot 概念的过程中， 曾经发现过许多跳出 chroot 环境的方法， 尽管这些问题在较新的 FreeBSD 版本中已经修正， 但很明显地， chroot(8) 并不是一项用于加固服务安全的理想解决方案。 因此， 必须实现一个新的子系统来解决这些问题。

这就是为什么要开发 jail 最主要的原因。

Jail 以多种方式改进了传统的 chroot(2) 环境概念。 在传统的 chroot(2) 环境中， 只限制了进程能够访问文件系统的哪些部分。 其他部分的系统资源 (例如系统用户、 正在运行的进程， 以及网络子系统) 是由 chroot 进程与宿主系统中的其他进程共享的。 jail 扩展了这个模型， 它不仅将文件系统的访问虚拟化， 而且还将用户、 FreeBSD 的网络子系统， 以及一些其他系统资源虚拟化。 关于这些精细控制以及调整 jail 环境访问能力的更具体的介绍， 可参见 微调和管理。

jail 具有以下四项特点：

- 目录子树 - 进入 jail 的起点。 一旦进入了 jail， 进程就不再被允许访问这棵子树以外的对象。 传统上影响到最初 chroot(2) 设计的安全问题不会影响 FreeBSD jail。
- 主机名 - 将用于 jail 的主机名。 jail 主要用于存放网络服务， 因此在每个 mail 上能够标注一个有意义的主机名， 能够在很大程度上简化系统管理员的工作。
- IP 地址 - 这个地址是指定给 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别名地址， 但这并不是必需的。
- 命令 - 准备在 jail 中执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 也可以拥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 用户不能执行除了其对应 jail(8) 环境之外的系统中的一些关键操作。 关于 root 用户的能力和限制， 在后面的 微调和管理 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 分为两类： "完整的" jail， 通常包含真正的 FreeBSD 系统， 以及 "服务" jail， 专用于执行一个可能使用特权的应用或服务。 这只是一种概念上的区分， 并不影响如何建立 jail 的过程。 在联机手册 jail(8) 中对如何创建 jail 进行了清晰的阐述：

```
# 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 第一步就是为 jail 选择一个位置。 这个路径是在宿主系统中 jail 的物理位置。 一种常用的选择是 /usr/jail/jailname， 此处 jailname 是 jail 的主机名。 /usr/ 文件系统通常会有足够的空间来保存 jail 文件系统， 对于 "完整" 的 jail 而言， 它通常包含了 FreeBSD 默认安装的基本系统中每个文件的副本。

② 如果你已经通过使用 make world 或者 make buildworld 重新编译过了你的 userland， 则可以跳过这一步骤并把现有的 userland 安装进新的 jail。

③ 这个命令将在 jail 目录中安装所需的可执行文件、 函数库以及联机手册等。

④ distribution 这个 make target 将安装全部配置文件， 或者换句话说， 就是将 /usr/src/etc/ 复制到 jail 环境中的 /etc： $D/etc/。

⑤ 在 jail 中不是必须要挂接 devfs(8) 文件系统。 而另一方面，
几乎所有的应用程序都会需要访问至少一个设备， 这主要取决于应用程序的性质和目的。 控制 jail 中能够访问的设备非常重要， 因为不正确的配置， 很可能允许攻击者在 jail 中进行一些恶意的操作。 通过 devfs(8) 实施的控制， 可以通过由联机手册 devfs(8) 和 devfs.conf(5) 介绍的规则集配置来实现。

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， 这些参数在 Jail 是什么 中进行了介绍。 除了这四个参数之外， 您还可以指定一些其他参数， 例如， 以特定用户身份来在 jail 中运行程序等等。 这里， command 参数取决于您希望建立的 jail 的类型； 对于 虚拟系统， 可以选择 /etc/rc， 因为它会完成真正的 FreeBSD 系统启动所需的操作。 对于 服务 jail， 执行的命令取决于将在 jail 中运行的应用程序。

Jail 通常应在系统启动时启动， 因此， FreeBSD rc 机制提供了一些很方便的机制来简化这些工作。

1. 在引导时需要启动的 jail 列表应写入 rc.conf(5) 文件：

```
jail_enable="YES"  # 如果设为 NO 则表示不自动启动 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， 而不应使用其他字符。

2. 对于 jail_list 中列出的 jail， 还应指定一系列对应的 rc.conf(5) 设置， 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 中应用的devfs 规则集
```

默认情况下， 在 rc.conf(5) 中配置启动的 jail 会执行其中的 /etc/rc 脚本， 也就是说，
默认情况下将 jail 作为虚拟系统方式来启动。 对于服务 jail， 您应另外指定启动命令，
方法是设置对应的 jailjailnameexec_start 配置。

如欲了解全部可用的选项， 请参阅联机手册 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关闭 jail(8)。 这是因为通常用于正常关闭系统的命令， 目前尚不能在 jail 中使用。 目前， 关闭 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
```

更进一步的详细说明， 请参见联机手册 jail(8)。

16.5. 微调和管理

您可以为 jail 设置许多不同的选项，并让 FreeBSD 宿主系统以不同的方式与 jail 交互，以支持更高级别的应用。这一节将介绍：

- 一些用于微调 jail 行为和安全限制的选项。
- 一些可以通过 FreeBSD Ports 套件安装的高级 jail 管理应用程序，这些程序可以用于实现一般的基于 jail 的解决方案。

16.5.1. FreeBSD 提供的用于微调 jail 的系统工具

对于 jail 的配置微调，基本上都是通过设置 sysctl(8) 变量来完成的。系统提供了一个特殊的 sysctl 子树，全部相关的选项均在这棵子树中；这就是 FreeBSD 内核的 security.jail.* 选项子树。下面是与 jail 有关的主要 sysctl，以及这些变量的默认值。这些名字都比较容易理解，如欲了解进一步的详情，请参阅联机手册 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

系统管理员可以在 宿主系统 中，透过设置这些变量的值来默认为 root 用户增加或取消限制。需要注意的是，某些限制是不能够取消的。在 jail(8) 中的 root 用户，无法挂载或卸下文件系统，此外在 jail 中的 root 用户也不能加载或卸载 devfs(8) 规则集、配置防火墙规则，或执行其他需要修改内核数据的管理操作，例如设置内核的 securelevel 等等。

FreeBSD 的基本系统包含一系列用于查看目前在使用的 jail 信息，以及接入 jail 并执行管理命令所需的基本工具。jls(8) 和 jexec(8) 命令都是 FreeBSD 基本系统的一部分，并可用于执行简单的任务：

- 列出在用的 jail 以及对应的 jail 标识 (JID)、IP 地址、主机名和路径。
- 从宿主系统中接入正在运行的 jail，并在其中执行命令，以完成一系列 jail 管理任务。这在 root 希望干净地关闭 jail 时非常有用。jexec(8) 工具也可以用于在 jail 中启动 shell 以便对其进行管理；例如：

```
# jexec 1 tcsh
```

16.5.2. 由 FreeBSD Ports 套件提供的高级管理工具

在众多第三方 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。请参见其网站以了解进一步的详情。

16.6. Jail 的应用

16.6.1. 服务 Jail

这一节主要基于 Simon L. B. Nielsen <simon@FreeBSD.org> 的 http://simon.nitro.dk/service-jails.html 中的思路，以及由 Ken Tom locals@gmail.com 更新的文档。这一节中描述了如何配置 FreeBSD 系统的 jail(8) 功能为其增加一个安全层次。这部分假定您运行 RELENG_6_0 或更新版本，并理解本章之前部分的内容。

16.6.1.1. 设计

jail 的一个主要问题是如何对它们进行升级和管理。 由于每个 jail 都是从头联编的， 对于单个 jail 而言升级也许还不是个很严重的问题， 因为升级不会太过麻烦， 而对于多个 jail 而言， 升级不仅会耗费大量时间， 并且是十分乏味的过程。

这个配置过程需要您对 FreeBSD 有较多的配置和使用经验。 如果这些过程显得太过复杂， 您应考虑使用较简单的系统， 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) 挂接， 这会让升级简单许多， 从而使为每个服务建立不同的 jail 这种方案变得更加可行。 另外， 它也为增加、删除以及升级 jail 提供了更为便捷的方法。

在这里服务的常见例子包括： HTTP 服务、 DNS 服务、 SMTP 服务等等， 诸如此类。

这节介绍的配置的目的包括：

- 建立简单并易于理解的 jail 结构。 也就是说 不必 为每个 jail 执行完整的 installworld 操作。
- 使增删 jail 更容易。
- 使更新或升级 jail 更容易。
- 使运行自订的 FreeBSD 分支成为可能。
- 对安全的更偏执的追求， 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， 这个设计极大程度上依赖于将一份只读的主模板 (known as nullfs) 挂接到每一个 jail 中， 并为每个 jail 配置一个可读写的设备。 这种设备可以是物理磁盘、 分区， 或以 vnode 为后端的 md(4) 设备。 在这个例子中， 我们将使用可读写的 nullfs 挂接。

下面的表中描述了文件系统格局：

- 每个 jail 挂接到 /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 有一个对应的空目录。
- 每个 jail 中都有一个 /s 目录， 这个目录将连接到系统中的可读写部分。
- 每个 jail 应基于 /home/j/skel 建立其可读写空间。
- 每个 jailspace (jail 中的可读写部分) 应创建到 /home/js。

这假定所有的 jail 都放置于 /home 分区中。 当然， 您可以根据需要将这个配置改为需要的任何样子， 但在接下来的例子中， 也应相应地加以变动。

16.6.1.2. 建立模板

这一节将介绍创建 jail 所需的只读主模板所需的步骤。

一般来说， 您应将系统升级到最新的 FreeBSD -RELEASE 分支， 具体做法请参见本手册的相关 章节。 当更新不可行时， 则需要完成 buildworld 过程， 另外， 您还需要 sysutils/cpdup 软件包。 我们将使用 portsnap(8) 工具来下载 FreeBSD Ports 套件。 在使用手册的 Portsnap 章节 中， 提供了针对初学者的介绍。

1. 首先， 需要为将要存放只读的 FreeBSD 执行文件的文件系统建立一个目录， 接着进入 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， 准备一份 FreeBSD Ports 套件， 以及用于执行 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. 创建系统中可读写部分的骨架：

```
# 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 创建的多余目录：

```
# 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， 将可读写文件系统连接到只读文件系统中。 请确保您在 s/ 目录中建立了适当的符号连接。 如果没有建立目录或建立的位置不正确， 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， 创建一个默认的包含下列配置的 /home/j/skel/etc/make.conf：

```
WRKDIRPREFIX?= /s/portbuild
```

配置 WRKDIRPREFIX 使得在每个 jail 中分别编译 FreeBSD 成为可能。 请注意 ports 目录是只读系统的一部分。 而自订的 WRKDIRPREFIX 则使得联编过程得以在 jail 中的可读写部分完成。

16.6.1.3. 建立 Jail

现在我们已经有了完整的 FreeBSD jail 模板， 可以在 /etc/rc.conf 中安装并配置它们了。 这个例子中演示了建立 3 个 jail： "NS"、 "MAIL" 和 "WWW"。

1. 在 /etc/fstab 文件中加入下列配置， 以便让系统自动挂接 jail 的只读模板和读写空间：

```
/home/j/mroot   /home/j/ns    nullfs  ro  0   0
/home/j/mroot   /home/j/mail  nullfs  ro  0   0
/home/j/mroot   /home/j/www   nullfs  ro  0   0
/home/js/ns     /home/j/ns/s  nullfs  rw  0   0
/home/js/mail   /home/j/mail/s nullfs  rw  0   0
/home/js/www    /home/j/www/s  nullfs  rw  0   0
```

扫描批次号 (pass number) 为 0 的分区不会在启动时使用 fsck(8) 进行检查， 而转存批次号 (dump number) 为 0 的分区则不会在 dump(8) 时备份。 我们不希望 fsck 检查 nullfs 挂接， 或让 dump 备份 jail 中的只读 nullfs 挂接。 这就是为什么在每个 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"
```

应把 jailnamerootdir 变量设置成 /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 的一个符号连接。 而

jailnamerootdir 变量必须是一个 不 包含符号连接的路径， 否则 jail 将拒绝启动。 可以使用 realpath(1) 工具来决定这一变量应被赋予一个什么样的值。 更详细的信息请参阅安全公告 FreeBSD-SA-07:01.jail

3. 为每个 jail 创建所需的只读文件系统挂接点：

```
# mkdir /home/j/ns /home/j/mail /home/j/www
```

4. 在 jail 中安装可读写的模板。 注意您需要使用 sysutils/cpdup，
它能够帮助您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. 这样， 就完成了 jail 的制作， 可以运行了。 首先为 jail 挂接文件系统， 然后使用 /etc/rc.d/jail 脚本来启动它们：

```
# mount -a
# /etc/rc.d/jail start
```

现在 jail 应该就启动起来了。 要检查它们是否运行正常， 可以使用 jls(8) 命令。 它的输出应该类似这样：

```
# 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
```

这时， 就可以登入 jail 并增加用户和配置服务了。 JID 列给出了正在运行的 jail 的标识编号。
您可以使用下面的命令来在 JID 编号为 3 的 jail 中执行管理任务：

```
# jexec 3 tcsh
```

16.6.1.4. 升级

有时， 由于安全问题， 或新增功能有用， 会希望将系统升级到一个新版本的 FreeBSD。
这种安装方式的设计使得升级现有 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关闭。 另外， 它也提供了简单的回退到先前版本的方法。

1. 第一步是按通常的方法升级主机的系统。 接着， 在 /home/j/mroot2 中建立一个新的临时模板：

```
# mkdir /home/j/mroot2
```

```
# cd /usr/src
# make installworld DESTDIR=/home/j/mroot2
# cd /home/j/mroot2
# cpdup /usr/src usr/src
# mkdir s
```

在运行 installworld 时会创建一些不需要的目录， 应将它们删除：

```
# chflags -R 0 var
# rm -R etc var root usr/local tmp
```

2. 重建到主系统中的可读写符号连接：

```
# 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. 现在是时候关闭 jail 了：

```
# /etc/rc.d/jail stop
```

4. 卸下原先的文件系统：

```
# 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
```

可读写的文件系统 (/s) 会在只读系统之后挂接， 因此应首先卸载。

5. 将先前的只读文件系统挪走， 换成新的系统。 这样做也同时保留了先前系统的备份，
从而可以在出现问题时从中恢复。 这里我们根据新系统的创建时间来命名。 此外我们把先前的
FreeBSD Ports 套件直接移动到新的文件系统中， 以节省磁盘空间和 inode：

```
# cd /home/j
# mv mroot mroot.20060601
# mv mroot2 mroot
```

```
# mv mroot.20060601/usr/ports mroot/usr
```

6. 现在新的只读模板就可以用了， 剩下的事情是重新挂接文件系统并启动 jails：

```
# mount -a
# /etc/rc.d/jail start
```

最后用 jls(8) 检查 jail 启动是否正常。 不要忘记在 jail 中运行 mergemaster。 配置文件和 rc.d 脚本在升级时应进行更新。

Chapter 17. 强制访问控制

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两个， 是文件系统访问控制列表 (ACL) 和强制访问控制 (MAC) 机制。
强制访问控制允许加载新的访问控制模块， 并借此实施新的安全策略，
其中一部分为一个很小的系统子集提供保护并加强特定的服务，
其他的则对所有的主体和客体提供全面的标签式安全保护。 定义中有关强制的部分源于如下事实，
控制的实现由管理员和系统作出， 而不像自主访问控制 (DAC, FreeBSD 中的标准文件以及 System V IPC 权限) 那样是按照用户意愿进行的。

本章将集中讲述强制访问控制框架 (MAC 框架) 以及一套用以实施多种安全策略的插件式的安全策略模块。

阅读本章之后， 您将了解：

- 目前 FreeBSD 中具有哪些 MAC 安全策略模块， 以及与之相关的机制。
- MAC 安全策略模块将实施何种策略， 以及标签式与非标签式策略之间的差异。
- 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块。
- 如何用 MAC 框架构建更为安全的环境， 并举例说明。
- 如何测试 MAC 配置以确保正确构建了框架。

阅读本章之前， 您应该：

- 了解 UNIX® 和 FreeBSD 的基础 (UNIX 基础)。
- 熟悉内核配置/编译 (配置FreeBSD的内核) 的基础。
- 对安全及其如何与 FreeBSD 相配合有些了解； (安全)。

对本章信息的不当使用可能导致丧失系统访问权， 激怒用户， 或者无法访问 X11 提供的特性。 更重要的是， MAC 不能用于彻底保护一个系统。 MAC 框架仅用于增强现有安全策略； 如果没有健全的安全条例以及定期的安全检查，
系统将永远不会绝对安全。

此外还需要注意的是， 本章中所包含的例子仅仅是例子。
我们并不建议在一个生产用系统上进行这些特别的设置。
实施各种安全策略模块需要谨慎的考虑与测试，
因为那些并不完全理解所有机制如何工作的人，
可能会发现需要对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 框架有关的诸多方面的安全问题； 而新的 MAC 安全策略模块的开发成果则不会涉及。
MAC 框架中所包含的一部分安全策略模块， 具有一些用于测试及新模块开发的特定属性， 其中包括 mac_test(4)、 mac_stub(4) 以及 mac_none(4)。
关于这些安全策略模块及其提供的众多机制的详细信息，请参阅联机手册中的内容。

17.2. 本章出现的重要术语

在阅读本章之前， 有些关键术语需要解释， 希望能藉此扫清可能出现的疑惑， 并避免在文中对新术语、新信息进行生硬的介绍。

- 区间(compartment)： (译注： 区间 这一术语， 在一些文献中也称做类别 (category)。 此外，
  在其它一些翻译文献中， 该术语也翻译为 "象限"。) 指一组被划分或隔离的程序和数据， 其中，
  用户被明确地赋予了访问特定系统组件的权限。 同时， 区间也能够表达分组， 例如工作组、 部门、
  项目， 或话题。 可以通过使用区间来实施 need-to-know 安全策略。

- 高水位线(high water mark)： 高水位线策略是一种允许提高安全级别，
以期访问更高级别的信息的安全策略。 在多数情况下， 当进程结束时， 又会回到原先的安全级别。
目前， FreeBSD MAC 框架尚未提供这样的策略， 在这里介绍其定义主要是希望给您一个完整的概念。
- 完整性(integrity)： 作为一个关键概念， 完整性是数据可信性的一种程度。 若数据的完整性提高，
则数据的可信性相应提高。
- 标签(label)： 标签是一种可应用于文件、 目录或系统其他客体的安全属性，
它也可以被认为是一种机密性印鉴。 当一个文件被施以标签时， 其标签会描述这一文件的安全参数，
并只允许拥有相似安全性设置的文件、 用户、 资源等访问该文件。
标签值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对某一客体的完整性和保密性的表述，
而其它一些策略则会用标签保存访问规则。
- 程度(level)： 对某种安全属性加强或削弱的设定。 若程度增加， 其安全性也相应增加。
- 低水位线(low water mark)： 低水位线策略允许降低安全级别， 以访问安全性较差的信息。
多数情况下， 在进程结束时， 又会回到原先的安全级别。 目前在 FreeBSD
中唯一实现这一安全策略的是 mac_lomac(4)。
- 多重标签(multilabel)： multilabel 属性是一个文件系统选项。 该选项可在单用户模式下通过 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。
该选项将允许管理员对不同客体施以不同的 MAC 标签。 该选项仅适用于支持标签的安全策略模块。
- 客体(object)： 客体或系统客体是一种实体， 信息随 主体 的导向在客体内部流动。 客体包括目录、
文件、 区段、 显示器、 键盘、 存储器、 磁存储器、 打印机及其它数据存储/转移设备。 基本上，
客体就是指数据容器或系统资源。 对 客体 的访问实际上意味着对数据的访问。
- 策略(policy)： 一套用以规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规定其访问者的规则，就是其中一例。
- 敏感性(sensitivity)： 通常在讨论 MLS 时使用。
敏感性程度曾被用来描述数据应该有何等的重要或机密。 若敏感性程度增加，
则保密的重要性或数据的机密性相应增强。
- 单一标签(single label)： 整个文件系统使用一个标签对数据流实施访问控制， 叫做单一标签。
当文件系统使用此设置时， 即无论何时当 多重标签 选项未被设定时，
所有文件都将遵守相同标签设定。
- 主体(subject)： 主体就是引起信息在两个 客体 间流动的任意活动实体， 比如用户，
用户进程(译注：原文为 processor)， 系统进程等。 在 FreeBSD 中，
主体几乎总是代表用户活跃在某一进程中的一个线程。

17.3. 关于 MAC 的说明

在掌握了所有新术语之后， 我们从整体上来考虑 MAC 是如何加强系统安全性的。 MAC
框架提供的众多安全策略模块可以用来保护网络及文件系统， 也可以禁止用户访问某些特定的端口、
套接字及其它客体。 将策略模块组合在一起以构建一个拥有多层次安全性的环境，
也许是其最佳的使用方式， 这可以通过一次性加载多个安全策略模块来实现。 在多层次安全环境中，
多重策略模块可以有效地控制安全性， 这一点与强化型 (hardening) 策略，
即那种通常只强化系统中用于特定目的的元素的策略是不同的。 相比之下，
多重策略的唯一不足是需要系统管理员先期设置好参数， 如多重文件系统安全标志、
每一位用户的网络访问权限等等。

与采用框架方式实现的长期效果相比， 这些不足之处是微不足道的。 例如，
让系统具有为特定配置挑选必需的策略的能力， 有助于降低性能开销。 而减少对无用策略的支持，
不仅可以提高系统的整体性能， 而且提供了更灵活的选择空间。
好的实施方案中应该考虑到整体的安全性要求， 并有效地利用框架所提供的众多安全策略模块。

这样一个使用 MAC 特性的系统， 至少要保证不允许用户任意更改安全属性； 所有的用户实用工具、
程序以及脚本， 必须在所选安全策略模块提供的访问规则的约束下工作； 并且系统管理员应掌握 MAC
访问规则的一切控制权。

细心选择正确的安全策略模块是系统管理员专有的职责。 某些环境也许需要限制网络的访问控制权，
在这种情况下， 使用 mac_portacl(4)、 mac_ifoff(4) 乃至 mac_biba(4) 安全策略模块都会是不错的开始；
在其他情况下， 系统客体也许需要严格的机密性， 像 mac_bsdextended(4) 和 mac_mls(4)

这样的安全策略模块就是为此而设。

对安全策略模块的决定可依据网络配置进行，也许只有特定的用户才应该被允许使用由 ssh(1) 提供的程序以访问网络或互联网，mac_portacl(4) 安全策略模块应该成为这种情况下的选择。但对文件系统又该作些什么呢？是由特定的用户或群组来确定某些目录的访问权限，抑或是将特定客体设为保密以限制用户或组件访问特定文件？

在文件系统的例子中，也许访问客体的权限对某些用户是保密的，但对其他则不是。比如，一个庞大的开发团队，也许会被分成许多由几人组成的小组，A 项目中的开发人员可能不被允许访问 B 项目开发人员创作的客体，但同时他们还需要访问由 C 项目开发人员创作的客体，这正符合上述情形。使用由 MAC 框架提供的不同策略，用户就可以被分成这种小组，然后被赋予适当区域的访问权，由此，我们就不用担心信息泄漏的问题了。

因此，每一种安全策略模块都有其处理系统整体安全问题的独特方法。对安全策略模块的选择应在对安全策略深思熟虑的基础之上进行。很多情况下，整体安全策略需要重新修正并在系统上实施。理解 MAC 框架提供的不同安全策略模块会帮助管理员就其面临的情形选择最佳的策略模块。

FreeBSD 的默认内核并不包含 MAC 框架选项，因此，在尝试使用本章中的例子或信息之前，您应该添加以下内核选项：

```
options MAC
```

此外，内核还需要重新编译并且重新安装。

尽管有关 MAC 的许多联机手册中都声明它们可以被编译到内核中，但对这些策略模块的使用仍可能导致锁死系统的网络及其他功能。使用 MAC 就像使用防火墙一样，因此必须要小心防止将系统完全锁死。在使用 MAC 时，应该考虑是否能够回退到之前的配置，在远程进行配置更应加倍小心。

17.4. 理解 MAC 标签

MAC 标签是一种安全属性，它可以被应用于整个系统中的主体和客体。

配置标签时，用户必须能够确切理解其所进行的操作。客体所具有的属性取决于被加载的策略模块，不同策略模块解释其属性的方式也差别很大。由于缺乏理解或无法了解其间联系而导致的配置不当，会引起意想不到的，也许是不愿看到的系统异常。

客体上的安全标签是由安全策略模块决定的安全访问控制的一部分。在某些策略模块中，标签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块中，标签则可能被作为一个庞大规则体系的一部分进行处理。

举例来说，在文件上设定 biba/low 标签，意味着此标签隶属 Biba 策略模块，其值为 "low"。

某些在 FreeBSD 中支持标签特性的策略会提供三个预定义的标签，分别是 low、high 及 equal 标签。尽管这些标签在不同安全策略模块中会对访问控制采取不同措施，但有一点是可以肯定的，那就是 low 标签表示最低限度的设定，equal 标签会将主体或客体设定为被禁用的或不受影响的，high 标签则会应用 Biba 及 MLS 安全策略模块中允许的最高级别的设定。

在单一标签文件系统的环境中，同一客体上只会应用一个标签，于是，一套访问权限将被应用于整个系统，这也是很多环境所全部需要的。另一些应用场景中，我们需要将多重标签应用于文件系统的客体或主体，如此一来，就需要使用 tunefs(8) 的 multilabel 选项。

在使用 Biba 和 MLS 时可以配置数值标签，以标示分级控制中的层级程度。数值的程度可以用来划分或将信息按组分类，从而只允许同程度或更高程度的组对其进行访问。

多数情况下，管理员将仅对整个文件系统设定单一标签。

等一下， 这看起来很像 DAC！ 但我认为 MAC 确实只将控制权赋予了管理员。 此句话依然是正确的。
在某种程度上， root 是实施控制的用户， 他配置安全策略模块以使用户们被分配到适当的类别/访问 levels 中。 唉， 很多安全策略模块同样可以限制 root 用户。 对于客体的基本控制可能会下放给群组， 但 root 用户随时可以废除或更改这些设定。 这就是如 Biba 及 MLS 这样一些安全策略模块所包含的 hierarchal/clearance 模型。

17.4.1. 配置标签

实际上， 有关标签式安全策略模块配置的各种问题都是用基础系统组件实现的。
这些命令为客体和主体配置以及配置的实施和验证提供了一个简便的接口。

所有的配置都应该通过 setfmac(8) 及 setpmac(8) 组件实施。 setfmac 命令是用来对系统客体设置 MAC 标签的， 而 setpmac 则是用来对系统主体设置标签的。 例如：

```
# setfmac biba/high test
```

若以上命令不发生错误则会直接返回命令提示符， 只有当发生错误时， 这些命令才会给出提示， 这和 chmod(1) 和 chown(8) 命令类似。 某些情况下， 以上命令产生的错误可能是 Permission denied，
一般在受限客体上设置或修改设置时会产生此错误。 系统管理员可使用以下命令解决此问题：

```
# setfmac biba/high test
Permission denied
# setpmac biba/low setfmac biba/high test
# getfmac test
test: biba/high
```

如上所示， 通过 setpmac 对被调用的进程赋予不同的标签， 以覆盖安全策略模块的设置。 getpmac 组件通常用于当前运行的进程， 如 sendmail： 尽管其使用进程编号来替代命令， 其逻辑是相同的。
如果用户试图对其无法访问的文件进行操作， 根据所加载的安全策略模块的规则， 函数 mac_set_link 将会给出 Operation not permitted 的错误提示。

17.4.1.1. 一般标签类型

mac_biba(4)、 mac_mls(4) 及 mac_lomac(4) 策略模块提供了设定简单标签的功能， 其值应该是 high、equal 及 low 之一。 以下是对这些标签功能的简单描述：

- low 标签被认为是主体或客体所具有的最低层次的标签设定。 对主体或客体采用此设定，
  将阻止其访问标签为 high 的客体或主体。
- equal 标签只能被用于不希望受策略控制的客体上。
- high 标签对客体或主体采用可能的最高设定。

至于每个策略模块， 每种设定都会产生不同的信息流指令。
阅读联机手册中相关的章节将进一步阐明这些一般标签配置的特点。

17.4.1.1.1. 标签高级配置

如下所示， 用于 比较方式:区间+区间 (comparison:compartment+compartment) 的标签等级数：

```
biba/10:2+3+6(5:2+3-20:2+3+4+5+6)
```

其含义为：

"Biba 策略标签"/"等级 10"： "区间 2、 3及6"： ("等级5 …")

本例中， 第一个等级将被认为是 "有效区间" 的 "有效等级"， 第二个等级是低级等级，
最后一个则是高级等级。 大多数配置中并不使用这些设置， 实际上， 它们是为更高级的配置准备的。

当把它们应用在系统客体上时， 则只有当前的等级/区间，
因为它们反映可以实施访问控制的系统中可用的范围， 以及网络接口。

等级和区间， 可以用来在一对主体和客体之间建立一种称为 "支配 (dominance)" 的关系，
这种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"
这种情况会在两个标签相等时发生。 由于 Biba 的信息流特性， 您可以设置一系列区间， "need to know"， 这可能发生于项目之间， 而客体也由其对应的区间。 用户可以使用 su 和 setpmac
来将他们的权限进一步细分， 以便在没有限制的区间里访问客体。

17.4.1.2. 用户和标签设置

用户本身也需要设置标签， 以使其文件和进程能够正确地与系统上定义的安全策略互动，
这是通过使用登录分级在文件 login.conf 中配置的。 每个使用标签的策略模块都会进行用户分级设定。

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 选项用以设定用户分级默认标签， 该标签将由 MAC 执行。 用户绝不会被允许更改该值，
因此其从用户的观点看不是可选的。 当然， 在真实情况的配置中， 管理员不会希望启用所有策略模块。
我们建议您在实施以上配置之前阅读本章的其余部分。

用户也许会在首次登录后更改其标签， 尽管如此， 这仅仅是策略的主观局限性。
上面的例子告诉 Biba 策略， 进程的最小完整性是为5， 最大完整性为15，
默认且有效的标签为10。 进程将以10的完整性运行直至其决定更改标签，

> 这可能是由于用户使用了 setpmac 命令 (该操作将在登录时被 Biba
> 限制在一定用户范围之内)。

在所有情况下， 修改 login.conf 之后， 都必须使用 cap_mkdb 重编译登录分级 capability 数据库，
这在接下来的例子和讨论中就会有所体现。

很多站点可能拥有数目可观的用户需要不同的用户分级， 注意到这点是大有裨益的。
深入来说就是需要事先做好计划， 因为管理起来可能十分困难。

在 FreeBSD 以后的版本中， 将包含一种将用户映射到标签的新方式， 尽管如此， 这也要到 FreeBSD 5.3
之后的某个时间才能实现。

17.4.1.3. 网络接口和标签设定

也可以在网络接口上配置标签， 以控制进出网络的数据流。 在所有情况下，
策略都会以适应客体的方式运作。 例如， 在 biba 中设置为高的用户， 就不能访问标记为低的网络接口。

maclabel 可以作为 ifconfig 的参数用于设置网络接口的 MAC 标签。 例如：

```
# ifconfig bge0 maclabel biba/equal
```

将在 bge(4) 接口上设置 biba/equal 的 MAC 标签。 当使用类似 biba/high(low-high) 这样的标签时，
整个标签应使用引号括起来； 否则将发生错误。

每一个支持标签的策略模块都提供了用于在网络接口上禁用该 MAC 标签的系统控制变量。 将标签设置为
equal 的效果与此类似。 请参见 sysctl 的输出、 策略模块的联机手册， 或本章接下来的内容，
以了解更进一步的详情。

17.4.2. 用单一标签还是多重标签?

默认情况下， 系统采用的是 singlelabel 选项。 但这对管理员意味着什么呢?
两种策略之间存在很多的不同之处， 它们在系统安全模型的灵活性方面， 提供了不同的选择。

singlelabel 只允许在每个主体或客体上使用一个标签， 如 biba/high。 这降低了管理的开销，
但也同时降低了支持标签的策略的灵活性。 许多管理员可能更希望在安全策略中使用 multilabel。

multilabel 选项允许每一个主体或客体拥有各自独立的 MAC 标签， 起作用与标准的、
只允许整个分区上使用一个的 singlelabel 选项类似。 multilabel 和 single
标签选项只有对实现了标签功能的那些策略， 如 Biba、 Lomac、 MLS 以及 SEBSD 才有意义。

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 以及许多混合策略的 FreeBSD web-服务器。
- 这台机器上的整个系统中只需要一个标签， 即 biba/high。 此处的文件系统并不需要 multilabel 选项，
  因为有效的 label 只有一个。
- 因为这台机器将作为 Web 服务器使用， 因此应该以 biba/low 运行 Web 服务， 以杜绝向上写。 Biba
  策略以及它如何运作将在稍后予以讨论， 因此， 如果您感觉前面的说明难以理解的话，
  请继续阅读下面的内容， 再回来阅读这些内容就会有较为清晰的认识了。 服务器可以使用设置为
  biba/low 的单独的分区， 用于保持其运行环境的状态。 这个例子中还省略了许多内容， 例如，
  如何为数据配置访问限制、 参数配置和用户的设置； 它只是为前述的内容提供一个简单的例子。

如果打算使用非标签式策略， 就不需要 multilabel 选项了。 这些策略包括 seeotheruids、 portacl 和
partition。

另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开销，
因为文件系统中的所有客体都需要指定标签。 这包括对目录、文件， 甚至设备节点。

接下来的命令将在需要使用多个标签的文件系统上设置 multilabel。 这一操作只能在单用户模式下完成：

```
# tunefs -l enable /
```

交换区不需要如此配置。

某些用户可能会在根分区上配置 multilabel 标志时遇到困难。 如果发生这样的情况，
请复查本章的 MAC 框架的故障排除。

17.5. 规划安全配置

在实施新技术时， 首先进行规划都是非常好的习惯。 在这段时间， 管理员一般都应 "进行全面的考察"，
这至少应包括下列因素：

- 方案实施的必要条件；
- 方案实施的目标；

就实施 MAC 而言， 这包括：

- 如何在目标系统上对信息和资源进行分类。
- 需要限制哪类信息或资源的访问， 以及应采用何种限制。
- 需要使用哪些 MAC 模块来完成这些目标。

尽管重新配置并修改系统资源和安全配置是可行的，
但查找整个系统并修复暨存的文件和用户帐号并不是一件轻而易举的事情。
规划有助于完成无问题且有效的可信系统实施。 事先 对采用 MAC 的可信系统，
以及其配置做试运行十分有益， 因为这对实施的成败至关重要。 草率散漫地配置 MAC
通常是导致失败的祸根。

不同的环境可能会有不同的需求。 建立多层次而完备的安全配置，
可以减少系统正式运转之后所需要的微调。 同样地，
接下来的章节将介绍管理员能够使用的各种不同的模块； 描述它们的使用和配置；
除此之外还有一些关于它们最适合的情景的介绍。 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时， mac_partition(4)
则是不错的选择。

17.6. 模块配置

在 MAC 框架中的每个模块， 都可以像前述那样连编入内核， 或作为运行时内核模块加载。 推荐的用法，
是通过在 /boot/loader.conf 加入适当的设置， 以便在系统启动时的初始化操作过程中加载这些模块。

接下来的一些小节， 将讨论许多 MAC 模块， 并简单介绍它们的功能。 此外，
这一章还将介绍一些具体环境中的用例。 某些模块支持一种称为标签 (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" 的标签来实现访问控制。
标签配置文件可以控制允许的文件访问方式、 网络通讯， 以及许多其他权限。 在前一节中，
我们已经展示了文件系统中如何通过 multilabel 标志来启用基于文件或分区的访问控制的方法。

单标签配置在整个系统中只强制一个标签的限制， 这也是 tunefs 选项为什么是 multilabel 的原因。

17.7. MAC seeotheruids 模块

模块名： mac_seeotheruids.ko

对应的内核配置： options MAC_SEEOTHERUIDS

引导选项： mac_seeotheruids_load="YES"

mac_seeotheruids(4) 模块模仿并扩展了 security.bsd.see_other_uids 和 security.bsd.see_other_gids

sysctl 变量。 这一模块并不需要预先配置标签， 它能够透明地与其他模块协同工作。

加载模块之后， 下列 sysctl 变量可以用来控制其功能：

- security.mac.seeotheruids.enabled 将启用模块的功能， 并使用默认的配置。
  这些默认设置将阻止用户看到其他用户的进程和 socket。
- security.mac.seeotheruids.specificgid_enabled 将允许特定的组从这一策略中和面。
  要将某些组排除在这一策略之外， 可以用 security.mac.seeotheruids.specificgid=XXX sysctl 变量。
  前述例子中， XXX 应替换为希望不受限的组 ID 的数值形式。
- security.mac.seeotheruids.primarygroup_enabled 可以用来将特定的主要组排除在策略之外。
  使用这一变量时， 不能同时设置 security.mac.seeotheruids.specificgid_enabled。

17.8. MAC bsdextended 模块

模块名： mac_bsdextended.ko

对应的内核配置： options MAC_BSDEXTENDED

引导选项： mac_bsdextended_load="YES"

mac_bsdextended(4) 模块能够强制文件系统防火墙策略。
这一模块的策略提供了标准文件系统权限模型的一种扩展， 使得管理员能够建立一种类似防火墙的规则集，
以文件系统层次结构中的保护文件、 实用程序，以及目录。 在尝试访问文件系统客体时， 会遍历规则表，
直至找到匹配的规则， 或到达表尾。 这一行为可以通过修改 sysctl(8) 参数，
security.mac.bsdextended.firstmatch_enabled 来进行设置。 与 FreeBSD 中的其他防火墙设置类似，
也可以建一个文件来配置访问控制策略， 并通过 rc.conf(5) 变量的配置在系统引导时加载它。

规则表可以通过工具 ugidfw(8) 工具来输入， 其语法类似 ipfw(8)。 此外还可以通过使用 libugidfw(3)
库来开发其他的工具。

当使用这一模块模块时应极其小心； 不正确的使用将导致文件系统的某些部分无法访问。

17.8.1. 例子

在加载了 mac_bsdextended(4) 模块之后， 下列命令可以用来列出当前的规则配置：

```
# ugidfw list
0 slots, 0 rules
```

如希望的那样， 目前还没有定义任何规则。 这意味着一切都还可以访问。 要创建一个阻止所有用户，
而保持 root 不受影响的规则， 只需运行下面的命令：

```
# ugidfw add subject not uid root new object not uid root mode n
```

这本身可能是一个很糟糕的主意， 因为它会阻止所有用户执行哪怕最简单的命令， 例如 ls。
更富于爱心的规则可能是：

```
# ugidfw set 2 subject uid user1 object uid user2 mode n
# ugidfw set 3 subject uid user1 object gid user2 mode n
```

这将阻止任何 user1 对 user2 的主目录的全部访问， 包括目录列表。

user1 可以用 not uid user2 代替。 这将同样的强制访问控制实施在所有用户， 而不是单个用户上。

root 用户不会受到这些变动的影响。

我们已经给出了 mac_bsdextended(4) 模块如何帮助加强文件系统的大致介绍。 要了解更进一步的信息，请参见 mac_bsdextended(4) 和 ugidfw(8) 联机手册。

17.9. MAC ifoff 模块

模块名： mac_ifoff.ko

对应的内核配置： 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， 以及阻止在系统初启时启用网络接口而设计的。
它不需要再系统中配置任何标签， 也不依赖于其他 MAC 模块。

绝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 环回接口 (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。

最为常用的 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监视网络。
另一个建议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发现新的或修改过的文件时切断网络。

17.10. MAC portacl 模块

模块名： mac_portacl.ko

对应的内核配置： 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 变量来限制绑定本地的 TCP 和 UDP 端口。 本质上 mac_portacl(4) 使得 非-root 用户能够绑定到它所指定的特权端口， 也就是那些编号小于 1024 的端口。

在加载之后， 这个模块将在所有的 socket 上启用 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 将设置为 mac_portacl(4) 所保护的最高端口号。
- security.mac.portacl.suser_exempt 如果设置为非零值， 表示将 root 用户排除在策略之外。
- security.mac.portacl.rules 将指定实际的 mac_portacl 策略； 请参见下文。

实际的 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,…] 其中可以给出任意多个规则。 每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， 分别表示将 id 参数解释为用户 id 或组 id。 protocol 参数可以用来确定希望应用到 TCP 或 UDP 协议上， 方法是把这一参数设置为 tcp 或 udp。 最后的 port 参数则给出了所指定的用户或组能够绑定的端口号。

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户 ID、 组 ID，
以及端口等参数。 换言之， 您不能使用用户、 组， 或端口服务的名字来指定它们。

默认情况下， 在 类-UNIX® 系统中， 编号小于 1024 的端口只能为特权进程使用或绑定， 也就是那些以 root 身份运行的进程。 为了让 mac_portacl(4) 能够允许非特权进程绑定低于 1024 的端口，
就必须首先禁用标准的 UNIX® 限制。 这可以通过把 sysctl(8) 变量 net.inet.ip.portrange.reservedlow 和 net.inet.ip.portrange.reservedhigh 设置为 0 来实现。

请参见下面的例子， 或 mac_portacl(4) 联机手册中的说明， 以了解进一步的信息。

17.10.1. 例子

下面的例子更好地展示了前面讨论的内容：

```
# sysctl security.mac.portacl.port_high=1023
# sysctl net.inet.ip.portrange.reservedlow=0 net.inet.ip.portrange.reservedhigh=0
```

首先我们需要设置使 mac_portacl(4) 管理标准的特权端口， 并禁用普通的 UNIX® 绑定限制。

```
# sysctl security.mac.portacl.suser_exempt=1
```

您的 root 用户不应因此策略而失去特权， 因此请把 security.mac.portacl.suser_exempt 设置为一个非零的值。 现在您已经成功地配置了 mac_portacl(4) 模块， 并使其默认与 类-UNIX® 系统一样运行了。

```
# sysctl security.mac.portacl.rules=uid:80:tcp:80
```

允许 UID 为 80 的用户 (正常情况下， 应该是 www 用户) 绑定到 80 端口。 这样 www 用户就能够运行 web 服务器， 而不需要使用 root 权限了。

```
# sysctl security.mac.portacl.rules=uid:1001:tcp:110,uid:1001:tcp:995
```

允许 UID 为 1001 的用户绑定 TCP 端口 110 ("pop3") 和 995 ("pop3s")。 这样用户就能够启动接受来发到 110 和 995 的连接请求的服务了。

17.11. MAC partition (分区) 模块

模块名： mac_partition.ko

对应的内核配置： options MAC_PARTITION

引导选项： mac_partition_load="YES"

mac_partition(4) 策略将把进程基于其 MAC 标签放到特定的 "partitions" (分区) 中。 这是一种特殊类型的 jail(8)， 但对两者进行比较意义不大。

这个模块应加到 loader.conf(5) 文件中， 以便在启动过程中启用这些规则。

绝大多数这一策略的配置是通过 setpmac(8) 工具来完成的， 它将在后面介绍。 这个策略可以使用下面的 sysctl：

- security.mac.partition.enabled 将启用强制的 MAC 进程 partitions。

当启用了这个规则时， 用户将只能看到他们自己的， 以及其他与他们同处一个 partition 的进程， 而不能使用能够越过 partition 的工具。 例如， insecure class 中的用户， 就无法使用 top 命令， 以及其他需要产生新进程的工具。

要设置或删除 partition 标签中的工具， 需要使用 setpmac：

```
# setpmac partition/13 top
```

这将把 top 命令加入到 insecure class 中的用户的标签集。 注意， 所有由 insecure class 中的用户产生的进程， 仍然会留在 partition/13 标签中。

17.11.1. 例子

下面的命令将显示 partition 标签以及进程列表：

```
# ps Zax
```

接下来的这个命令将允许察看其他用户的进程 partition 标签， 以及那个用户正在运行的进程：

```
# ps -ZU trhodes
```

除非加载了 mac_seeotheruids(4) 策略， 否则用户就看不到 root 的标签。

非常手工化的实现， 可能会在 /etc/rc.conf 中禁用所有的服务， 并用脚本来按不同的标签来启动它们。

下面的几个策略支持基于所给出的三种标签的完整性设定。 这些选项， 连同它们的限制， 在模块的联机手册中进行了进一步介绍。

17.12. MAC 多级 (Multi-Level) 安全模块

模块名： mac_mls.ko

对应的内核配置： options MAC_MLS

引导选项： mac_mls_load="YES"

mac_mls(4) 策略， 通过严格控制信息流向来控制系统中主体和客体的访问。

在 MLS 环境中， "许可 (clearance)" 级别会在每一个主体或客体标签上进行设置， 连同对应的区间。 由于这些透明度或敏感度可以有六千多个层次， 因此为每一个主体或客体进行配置将是一件让任何系统管理员都感到头疼的任务。 所幸的是， 这个策略中已经包含了三个 "立即可用的" 标签。

这些标签是 mls/low、 mls/equal 以及 mls/high。 由于这些标签已经在联机手册中进行了介绍， 这里只给出简要的说明：

- mls/low 标签包含了最低配置， 从而允许其他客体支配它。 任何标记为 mls/low 的客体将是地透明度的， 从而不允许访问更高级别的信息。 此外， 这个标签也阻止拥有较高透明度的客体向其写入或传递信息。
- mls/equal 标签应放到不希望使用这一策略的客体上。
- mls/high 标签是允许的最高级别透明度。 指定了这个标签的客体将支配系统中的其他客体； 但是， 它们将不允许向较低级别的客体泄露信息。

MLS 提供了：

- 提供了一些非层次分类的层次安全模型；
- 固定规则： 不允许向上读， 不允许向下写 (主体可以读取同级或较低级别的客体， 但不能读取高级别的。 类似地， 主体可以向同级或较高级写， 而不能向下写)；

- 保密 (防止不适当的数据透露)；
- 系统设计的基础要点， 是在多个敏感级别之间并行地处理数据 (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) 设备标记为 mls/equal。
- security.mac.mls.revocation_enabled 可以用来在标签转为较低 grade 时撤销客体访问权。
- security.mac.mls.max_compartments 可以用来设置客体的最大区间层次； 基本上，
  这也就是系统中所允许的最大区间数。

要管理 MLS 标签， 可以使用 setfmac(8) 命令。 要在客体上指定标签， 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 标签：

```
# getfmac test
```

以上是对于 MLS 策略提供功能的概要。 另一种做法是在 /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. 规划托管敏感性

通过使用多级安全策略模块， 管理员可以规划如何控制敏感信息的流向。 默认情况下，
由于其默认的禁止向上读以及向下写的性质， 系统会默认将所有客体置于较低的状态。 这样，
所有的客体都可以访问， 而管理员则可以在配置阶段慢慢地进行提高信息的敏感度这样的修改。

除了前面介绍的三种基本标签选项之外， 管理员还可以根据需要将用户和用户组进行分组，
以阻止它们之间的信息流。 一些人们比较熟悉的信息限界词汇， 如 机密、 秘密， 以及 绝密
可以方便您理解这一概念。 管理员也可以简单地根据项目级别建不同的分组。 无论采用何种分类方法，
在实施限制性的策略之前， 都必须首先想好如何进行规划。

这个安全策略模块最典型的用例是电子商务的 web 服务器，
其上的文件服务保存公司的重要信息以及金融机构的情况。 对于只有两三个用户的个人工作站而言，
则可能不甚适用。

17.13. MAC Biba 模块

模块名： mac_biba.ko

对应的内核配置： options MAC_BIBA

引导选项： mac_biba_load="YES"

mac_biba(4) 模块将加载 MAC Biba 策略。 这个策略与 MLS 策略非常类似，
只是信息流的规则有些相反的地方。 通俗地说， 这就是防止敏感信息向下传播， 而 MLS
策略则是防止敏感信息的向上传播； 因而， 这一节的许多内容都可以同时应用于两种策略。

在 Biba 环境中， "integrity" (完整性) 标签， 将设置在每一个主体或客体上。
这些标签是按照层次级别建立的。 如果客体或主体的级别被提升， 其完整性也随之提升。

被支持的标签是 biba/low， biba/equal 以及 biba/high； 解释如下：

- biba/low 标签是客体或主体所能拥有的最低完整性级别。 在客体或主体上设置它，
  将阻止其在更高级别客体或主体对其进行的写操作， 虽然读仍被允许。

- biba/equal 标签只应在那些希望排除在策略之外的客体上设置。
- biba/high 允许向较低标签的客体上写， 但不允许读那些客体。
  推荐在那些可能影响整个系统完整性的客体上设置这个标签。

Biba 提供了：

- 层次式的完整性级别， 并提供了一组非层次式的完整性分类；
- 固定规则： 不允许向上写， 不允许向下读 (与 MLS 相反)。 主体可以在它自己和较低的级别写，
  但不能向更高级别实施写操作。 类似地， 主体也可以读在其自己的， 或更高级别的客体，
  但不能读取较低级别的客体；
- 完整性 (防止对数据进行不正确的修改)；
- 完整性级别 (而不是 MLS 的敏感度级别)。

下列 sysctl 可以用于维护 Biba 策略。

- security.mac.biba.enabled 可以用来在机器上启用/禁用是否实施 Biba 策略。
- security.mac.biba.ptys_equal 可以用来在 pty(4) 设备上禁用 Biba 策略。
- security.mac.biba.revocation_enabled 将在支配主体发生变化时强制撤销对客体的访问权。

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. 规划托管完整性

与敏感性不同， 完整性是要确保不受信方不能对信息进行篡改。 这包括了在主体和客体之间传递的信息。
这能够确保用户只能修改甚至访问需要他们的信息。

mac_biba(4) 安全策略模块允许管理员指定用户能够看到和执行的文件和程序，
并确保这些文件能够为系统及用户或用户组所信任， 而免受其他威胁。

在最初的规划阶段， 管理员必须做好将用户分成不同的等级、 级别和区域的准备。 在启动前后，
包括数据以及程序和使用工具在内的客体， 用户都会无法访问。 一旦启用了这个策略模块，
系统将默认使用高级别的标签， 而划分用户级别和等级的工作则交由管理员来进行配置。
与前面介绍的级别限界不同， 好的规划方法可能还包括 topic。 例如， 只允许开发人员修改代码库、
使用源代码编译器， 以及其他开发工具， 而其他用户则分入其他类别， 如测试人员、 设计人员，
以及普通用户， 这些用户可能只拥有读这些资料的权限。

通过其自然的安全控制， 完整性级别较低的主体， 就会无法向完整性级别高的主体进行写操作；
而完整性级别较高的主体， 也不能观察或读较低完整性级别的客体。 通过将客体的标签设为最低级，
可以阻止所有主体对其进行的访问操作。 这一安全策略模块预期的应用场合包括受限的 web 服务器、
开发和测试机， 以及源代码库。 而对于个人终端、 作为路由器的计算机， 以及网络防火墙而言，
它的用处就不大了。

17.14. MAC LOMAC 模块

模块名： mac_lomac.ko

对应的内核配置： options MAC_LOMAC

引导选项： 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级别之后，
才允许在不破坏完整性规则的前提下访问较低完整性级别的客体。

MAC 版本的 Low-watermark 完整性策略不应与较早的 lomac(4) 实现相混淆，
除了使用浮动的标签来支持主体通过辅助级别区间降级之外， 其工作方式与 Biba 大体相似。
这一次要的区间以 [auxgrade] 的形式出现。 当指定包含辅助级别的 lomac 策略时， 其形式应类似于：
lomac/10[2] 这里数字二 (2) 就是辅助级别。

MAC LOMAC 策略依赖于系统客体上存在普适的标签， 这样就允许主体从较低完整性级别的客体读取，
并对主体的标签降级， 以防止其在之后写高完整性级别的客体。 这就是前面讨论的 [auxgrade] 选项，
因此这个策略能够提供更大的兼容性， 而所需要的初始配置也要比 Biba 少。

17.14.1. 例子

与 Biba 和 MLS 策略类似； setfmac 和 setpmac 工具可以用来在系统客体上放置标签：

```
# setfmac /usr/home/trhodes lomac/high[low]
# getfmac /usr/home/trhodes
lomac/high[low]
```

注意， 这里的辅助级别是 low， 这一特性只由 MAC LOMAC 策略提供。

17.15. MAC Jail 中的 Nagios

下面给出了通过多种 MAC 模块， 并正确地配置策略来实现安全环境的例子。 这只是一个测试，
因此不应被看作四海一家的解决之道。 仅仅实现一个策略， 而忽略它不能解决任何问题，
并可能在生产环境中产生灾难性的后果。

在开始这些操作之前， 必须在每一个文件系统上设置 multilabel 选项，
这些操作在这一章开始的部分进行了介绍。 不完成这些操作， 将导致错误的结果。 首先，
请确认已经安装了 net-mngt/nagios-plugins、 net-mngt/nagios， 和 www/apache13 这些 ports，
并对其进行了配置， 且运转正常。

17.15.1. 创建一个 insecure (不安全) 用户 Class

首先是在 /etc/login.conf 文件中加入一个新的用户 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 用户 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

现在暂时还不要重新启动， 我们还需要在 /boot/loader.conf 中增加下面几行，
以便让模块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 设为属于默认的 class：

```
# pw usermod root -L default
```

所有非 root 或系统的用户， 现在需要一个登录 class。 登录 class 是必须的，
否则这些用户将被禁止使用类似 vi(1) 这样的命令。 下面的 sh 脚本应能完成这个工作：

```
# for x in `awk -F: '($3 >= 1001) && ($3 != 65534) { print $1 }' \

# /etc/passwd`; do pw usermod $x -L default; done;
```

将 nagios 和 www 这两个用户归入不安全 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； 您可以把下面的实例放到 /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读入系统：

```
# 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. 启用网络

在 /boot/loader.conf 中增加下列内容：

```
security.mac.biba.trust_all_interfaces=1
```

将下述内容加入 rc.conf 中的网络接口配置。 如果主 Internet 配置是通过 DHCP 完成的，
则需要在每次系统启动之后手工执行类似的配置：

```
maclabel biba/equal
```

17.15.6. 测试配置

首先要确认 web 服务以及 Nagios 不会随系统的初始化和重启过程而自动启动。 在此之前， 请在此确认
root 用户不能访问 Nagios 配置目录中的任何文件 如果 root 能够在 /var/spool/nagios 中运行 ls(1)，
则表示配置有误。 如果配置正确的话， 您会收到一条 "permission denied" 错误信息。

如果一切正常， Nagios、 Apache， 以及 Sendmail 就可以按照适应安全策略的方式启动了。
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。 如果不是的话， 请检查日志文件和错误信息。 此外， 还可以用 sysctl(8)
来临时禁用 mac_biba(4) 安全策略模块的强制措施， 并象之前那样进行配置和启动服务。

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。
下面的命令可以对安全策略进行降级， 并启动一个新的 shell：

```
# setpmac biba/10 csh
```

要阻止这种情况发生， 就需要配置 login.conf(5) 中许可的命令范围了。 如果
setpmac(8) 尝试执行超越许可范围的命令， 则会返回一个错误， 而不是执行命令。

在这个例子中， 可以把 root 设为 biba/high(high-high)。

17.16. User Lock Down

这个例子针对的是一个相对较小的存储系统， 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， 还可以访问一些其他资源。

在这个场景中， mac_bsdextended(4) 可以与 mac_seeotheruids(4) 并存， 以达到禁止访问非授权资源，
同时隐藏其他用户的进程的目的。

首先， 在 /boot/loader.conf 中加入：

```
mac_seeotheruids_load="YES"
```

随后， 可以通过下述 rc.conf 变量来启用 mac_bsdextended(4) 安全策略模块：

```
ugidfw_enable="YES"
```

默认规则保存在 /etc/rc.bsdextended 中， 并在系统初始化时加载； 但是，
其中的默认项可能需要进行一些改动。 因为这台机器只为获得了授权的用户提供服务，
因此除了最后两项之外， 其它内容都应保持注释的状态。 这两项规则将默认强制加载属于用户的系统客体。

在这台机器上添加需要的用户并重新启动。 出于测试的目的，
请在两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进程。
此外， 在其他用户的主目录中运行 ls(1) 命令， 如果配置正确， 则这个命令会失败。

不要尝试以 root 用户的身份进行测试， 除非您已经修改了特定的 sysctl 来阻止超级用户的访问。

在添加新用户时， 他们的 mac_bsdextended(4) 规则不会自动出现在规则集表中。
要迅速更新规则集， 只需简单地使用 kldunload(8) 和 kldload(8)
工具来卸载并重新加载安全策略模块。

17.17. MAC 框架的故障排除

在开发过程中， 有一些用户报告了正常配置下出现的问题。 其中的一些问题如下所示：

17.17.1. 无法在 / 上启用 multilabel 选项

multilabel 标志在根 (/) 分区上没有保持启用状态！

看起来每五十个用户中就有一个遇到这样的问题， 当然， 在我们的初始配置过程中也出现过这样的问题。
更进一步的观察使得我相信这个所谓的 "bug" 是由于文档中不确切的描述， 或对其产生的误解造成的。
无论它是因为什么引发的， 下面的步骤应该能够解决此问题：

1. 编辑 /etc/fstab 并将根分区设置为 ro， 表示只读。
2. 重新启动并进入单用户模式。
3. 在 / 上运行 tunefs -l enable
4. 重新启动并进入正常的模式。
5. 运行 mount -urw/ 并把 /etc/fstab 中的 ro 改回 rw， 然后再次重新启动。
6. 再次检查来自 mount 的输出， 已确认根文件系统上正确地设置了 multilabel。

17.17.2. 在 MAC 之后无法启动 X11 了

在使用 MAC 建立安全的环境之后， 就无法启动 X 了!

这可能是由于 MAC partition 策略， 或者对某个 MAC 标签策略进行了错误的配置导致的。
要调试这个问题， 请尝试:

1. 检查错误信息； 如果用户是在 insecure class 中， 则 partition 策略就可能导致问题。
   尝试将用户的 class 重新改为 default class， 并使用 cap_mkdb 命令重建数据库。
   如果这无法解决问题， 则进入第二步。
2. 仔细检查标签策略。 确认针对有问题的用户的策略是正确的， 特别是 X11 应用， 以及 /dev 项。
3. 如果这些都无法解决问题， 将出错消息和对您的环境的描述， 发送到 TrustedBSD 网站上的 TrustedBSD 讨论邮件列表， 或者 FreeBSD 一般问题邮件列表 邮件列表。

17.17.3. Error: _secure_path(3) cannot stat .login_conf

当我试图从 root 用户切换到其同中的其他用户时， 出现了错误提示 _secure_path: unable to state .login_conf。

这个提示通常在用户拥有高于它将要成为的那个用户的 标签设定时出现。 例如， 如果系统上的一个用户 joe 拥有默认的 biba/low 标签， 而 root 用户拥有 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 查看在较低完整性级别中的客体。

17.17.4. root 用户名被破坏了!

在普通模式， 甚至是单用户模式中， root 不被识别。 whoami 命令返回了 0 (零) 而 su 则提示 who are you?。 到底发生了什么?

标签策略被禁用可能会导致这样的问题， 无论是通过 sysctl(8) 或是卸载了策略模块。 如果打算禁用策略， 或者临时禁用它， 则登录性能数据库需要重新配置， 在其中删除 label 选项。 仔细检查 login.conf 以确保所有的 label 选项都已经删除， 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访问 master.passwd 文件或对应的那个数据库时发生。
这主要是由于管理员修改受某一 label 限制的文件， 而与系统级的通用策略发生了冲突。 这时，
用户信息将由系统直接读取， 而在文件继承了新的 label 之后则会拒绝访问。 此时， 只需使用 sysctl(8) 禁用这一策略， 一切就会恢复正常了。

Chapter 18. 安全事件审计

18.1. 概述

FreeBSD 中包含了对于细粒度安全事件审计的支持。 事件审计能够支持可靠的、 细粒度且可配置的，
对于各类与安全有关的系统事件， 包括登录、 配置变更， 以及文件和网络访问等的日志记录。
这些日志记录对于在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 实现了 Sun 所发布的 BSM API 和文件格式， 并且与 Sun™ 的 Solaris™ 和 Apple® 的 Mac OS® X 审计实现兼容。

这一章的重点是安装和配置事件审计。 它介绍了事件策略， 并提供了一个审计的配置例子。

读完这章， 您将了解：

- 事件审计是什么， 以及它如何工作。
- 如何在 FreeBSD 上为用户和进程配置事件审计。
- 如何使用审计记录摘要和复审工具来对审计记录进行复审。

阅读这章之前， 您应该：

- 理解 UNIX® 和 FreeBSD 的基础知识 (UNIX 基础)。
- 熟悉关于内核配置和编译的基本方法 (配置FreeBSD的内核)。
- 熟悉安全知识以及如何在 FreeBSD 运用它们 (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，
另外某些登录机制， 例如基于 X11 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确配置审计。

安全审计机制能够对系统活动生成非常详细的记录信息： 在繁忙的系统中，
记帐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。
管理员应考虑审计配置中的导致磁盘空间需求的这些问题。 例如， 可能需要为 /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时影响其它文件系统。

18.2. 本章中的一些关键术语

在开始阅读这章之前， 我们需要解释一下与审计有关的一些关键的术语：

- 事件 (event)： 可审计事件是指能够被审计子系统记录的任何事件。 举例说来，
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。
任何事件必要么是 "有主 (attributable)" 的， 即可以最终归于某一已通过验证的用户的身份， 反之，
则称该事件是 "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，
例如尝试一次无效密码等。
- 类 (class)： 事件类是指相关事件的一个命名集合， 通常在筛选表达式中使用。 常用的事件类包括 "创建文件" (fc)、 "执行" (ex) 和 "登入和注销" (lo)。
- 记录 (record)： 记录是指描述一个安全事件的日志项。 记录包括记录事件类型、 执行操作的主体 (用户) 信息、 日期和事件信息， 以及与之相关的对象或参数信息， 最后是操作成功或失败。
- 账目 (trail)： 审计账目， 或日志文件， 包含了一系列描述安全事件的审计记录。 典型情况下，
审计账目基本上是以事件发生的时间顺序记录的。 只有获得授权的进程，
才能够向审计账目中提交记录。
- 筛选表达式 (selection expression)： 筛选表达式是包含一系列前缀和审计事件类名字，
用以匹配事件的字符串。
- 预选 (preselection)： 系统通过这一过程来识别事件是否是管理员所感兴趣的，
从而避免为他们不感兴趣的事件生成记录。 预选配置使用一系列选择表达式， 用以识别事件类别、
要审计的用户， 以及适用于验证过用户身份， 以及未验证用户身份的进程的全局配置。

- 浓缩 (reduction)： 从现有的审计记帐中筛选出用于保留、 打印或分析的过程。 除此之外，
  它也表示从审计记帐中删去不需要的审计记录的过程。 通过使用浓缩操作，
  管理员可以实现预留审计数据的策略。 例如， 详细的审计记帐信息， 可能会保留一个月之久，
  但在这之后， 则对这些记帐信息执行浓缩操作， 只保留登录信息用于存档。

18.3. 安装审计支持

对于事件审计的支持， 已经随标准的 installworld 过程完成。 管理员可以通过查看 /etc/security
的内容来确认这一点。 您应能看到一些名字以 audit 开头的文件， 例如 audit_event。

对于审计功能的用户态支持目前是作为 FreeBSD 基本系统的一部分来安装的。
默认内核中也包含了对于事件审计的内核支持， 但如果您使用的是定制内核，
就必须在内核配置文件中明确指定希望添加这一支持：

```
options AUDIT
```

接下来， 您应按照 配置FreeBSD的内核 中所介绍的步骤来完成一次内核的编译和安装。

在编译好并安装了内核， 并重新启动了系统之后， 就可以在 rc.conf(5)
中增加下列配置来启用审计服务了：

在编译、安装了开启审计功能的内核，并重新启动计算机之后， 就可以在 rc.conf(5)
中增加下列配置来启用审计服务了：

```
auditd_enable="YES"
```

此后， 必须重新启动系统， 或通过下面的命令手工启动审计服务来启动审计支持：

```
/etc/rc.d/auditd start
```

18.4. 对审计进行配置

所有用于安全审计的配置文件， 都可以在 /etc/security 找到。 要启动审计服务， 下面这些文件必须存在：

- audit_class - 包含对于审计类的定义。
- audit_control - 控制审计子系统的特性， 例如默认审计类、 在审计日志所在的卷上保留的最小空间、
  审计日志的最大尺寸， 等等。
- audit_event - 文字化的系统审计事件名称和描述， 以及每个事件属于哪个类别。
- audit_user - 针对特定用户的审计需求， 这些配置在登录时会与全局的默认值合并。
- audit_warn - 由 auditd 调用的一个可定制的 shell 脚本， 用于在意外情况，
  如用于审计日志的空间过少， 或审计日志文件被翻转时， 生成警告信息。

在编辑和维护审计配置文件时一定要小心，
因为配置文件中的错误会导致记录事件不正确。

18.4.1. 事件筛选表达式

在审计配置文件中的许多地方会用到筛选表达式来确定哪些事件是需要审计的。
表达式中需要指定要匹配的事件类型， 并使用前缀指定是否应接受或忽略匹配的事件， 此外，
还可以指定一个可选项指明匹配成功或失败的操作。 选择表达式是按从左到右的顺序计算的，
而对于两个表达式的情形， 则是通过将后一个追加到前一个之后来实现的。

下面列出了在 audit_class 中的默认事件类型：

- all - all (全部) - 表示匹配全部事件类。
- ad - administrative (管理) - 所有在系统上所进行的管理性操作。
- ap - application (应用) - 应用程序定义的动作。
- cl - file close (文件关闭) - 审计对 close 系统调用的操作。
- ex - exec (执行) - 审计程序的执行。 对于命令行参数和环境变量的审计是通过在 audit_control(5) 中 policy 的 argv 和 envv 参数来控制的。
- fa - file attribute access (造访文件属性) - 审计访问对象属性， 例如 stat(1)、 pathconf(2) 以及类似事件。
- fc - file create (创建文件) - 审计创建了文件的事件。
- fd - file delete (删除文件) - 审计所发生的文件删除事件。
- fm - file attribute modify (修改文件属性) - 审计文件属性发生变化的事件， 例如 chown(8)、 chflags(1)、 flock(2)， 等等。
- fr - file read (读文件数据) - 审计读取数据、 文件以读方式打开等事件。
- fw - file write (写文件数据) - 审计写入数据、 文件以写方式打开等事件。
- io - ioctl - 审计对 ioctl(2) 系统调用的使用。
- ip - ipc - 审计各种形式的进程间通信 (IPC)， 包括 POSIX 管道和 System V IPC 操作。
- lo - login_logout - 审计系统中发生的 login(1) 和 logout(1) 事件。
- na - non attributable (无主) - 审计无法归类的事件。
- no - invalid class (无效类) - 表示不匹配任何事件。
- nt - network (网络) - 与网络操作有关的事件， 例如 connect(2) 和 accept(2)。
- ot - other (其它) - 审计各类杂项事件。
- pc - process (进程) - 审计进程操作， 例如 exec(3) 和 exit(3)。

这些审计事件， 可以通过修改 audit_class 和 audit_event 这两个配置文件来进行定制。

这个列表中， 每个审计类均包含一个表示匹配成功/失败操作的前缀，
以及这一项是否是增加或删去对事件类或类型的匹配。

- (none) 审计事件的成功和失败实例。
- + 审计这一类的成功事件。
- - 审计这一类的失败事件。
- ^ 不审计本类中的成功或失败事件。
- ^+ 不审计本类中的成功事件。
- ^- 不审计本类中的失败事件。

下面例子中的筛选字符串表示筛选成功和失败的登录/注销事件， 而对执行事件， 则只审计成功的：

```
lo,+ex
```

18.4.2. 配置文件

多数情况下， 在配置审计系统时， 管理员只需修改两个文件： audit_control 和 audit_user。
前者控制系统级的审计属性和策略， 而后者则用于针对具体的用户来微调。

18.4.2.1. audit_control 文件

audit_control 文件指定了一系列用于审计子系统的默认设置。 通过查看这个文件，
我们可以看到下面的内容：

```
dir:/var/audit
flags:lo
minfree:20
naflags:lo
policy:cnt
filesz:0
```

这里的 dir 选项可以用来设置用于保存审计日志的一个或多个目录。 如果指定了多个目录，
则将在填满一个之后换用下一个。 一般而言， 审计通常都会配置为保存在一个专用的文件系统之下，
以避免审计系统与其它子系统在文件系统满的时候所产生的冲突。

flags 字段用于为有主事件配置系统级的预选条件。 在前面的例子中，
所有用户成功和失败的登录和注销都会被审计。

minfree 参数用于定义保存审计日志的文件系统上剩余空间的最小百分比。 当超过这一阈值时，
将产生一个警告。 前面的例子中， 最小剩余空间比例设置成了两成。

naflags 选项表示审计类审计无主事件， 例如作为登录进程和系统服务的那些进程的事件。

policy 选项用于指定一个以逗号分隔的策略标志表， 以控制一系列审计行为。 默认的 cnt
标志表示系统应在审计失败时继续运行 (强烈建议使用这个标志)。 另一个常用的标志是 argv，
它表示在审计命令执行操作时， 同时审计传给 execve(2) 系统调用的命令行参数。

filesz 选项指明了审计日志在自动停止记录和翻转之前允许的最大尺寸。 默认值 0 表示禁用自动日志翻转。
如果配置的值不是零， 但小于最小值 512k， 则这个配置会被忽略， 并在日志中记录这一消息。

18.4.2.2. audit_user 文件

audit_user 文件允许管理员为特定用户指定进一步的审计需求。 每一行使用两个字段来配置用户的审计：
第一个是 alwaysaudit 字段， 它指明了一组对该用户总会进行审计的事件； 而第二个则是 neveraudit
字段， 它指明了一系列对该用户不审计的事件。

在下述 audit_user 示例文件中， 审计了 root 用户的 登录/注销 事件， 以及成功的命令执行事件， 此外，
还审计了 www 用户的文件创建和成功的命令执行事件。 如果与前面的示范 audit_control 文件配合使用，
则 root 的 lo 项就是多余的， 而对 www 用户而言， 其登录/注销事件也会被审计：

```
root:lo,+ex:no
www:fc,+ex:no
```

18.5. 管理审计子系统

18.5.1. 查看审计日志

审计记帐是以 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) 命令能够将记帐文件转换为简单的文本格式； 而 auditreduce(1) 命令则可以为分析、
存档或打印目的来浓缩审计日志文件。 auditreduce 支持一系列筛选参数， 包括事件类型、 事件类、
用户、 事件的日期和时间， 以及文件路径或操作对象。

例如， praudit 工具会将指定的审计记帐转存为简单文本格式的审计日志：

```
# praudit /var/audit/AUDITFILE
```

此处 AUDITFILE 是要转存的审计日志文件。

审计记帐中包括一系列审计记录， 这些记录由一系列短语 (token) 组成， 而 praudit 能把它们顺序显示为一行。 每个短语都属于某个特定的类型， 例如 header 表示审计记录头， 而 path 则表示在一次名字查找中的文件路径。 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
```

这个审计记录表示一次成功的 execve 调用， 执行了 finger doug。 在参数短语中是由 shell 提交给内核的命令行。 path 短语包含了由内核查找得到的可执行文件路径。 attribute 短语中包含了对可执行文件的描述， 特别地， 它包括了文件的权限模式， 用以确定应用程序是否是 setuid 的。 subject(主体) 短语描述了主体进程， 并顺序记录了审计用户 ID、 生效用户 ID 和组 ID、 实际用户 ID 和组 ID、 进程 ID、 会话 ID、 端口 ID， 以及登录地址。 注意审计用户 ID 和实际用户 ID 是不同的： 用户 robert 在执行这个命令之前已经切换为 root 帐户， 但它会以最初进行身份验证的用户身份进行审计。 最后， return 短语表示执行成功， 而 trailer 表示终结这一记录。

praudit 可以选择使用 -x 参数来支持 XML 格式的输出。

18.5.2. 浓缩审计记帐

由于审计日志可能会很大， 管理员可能会希望选择记录的一个子集来使用， 例如与特定用户相关的记录：

```
# auditreduce -u trhodes /var/audit/AUDITFILE | praudit
```

这将选择保存在 AUDITFILE 中的所有由 trhodes 产生的审计日志。

18.5.3. 委派审计复审权限

在 audit 组中的用户， 拥有读取 /var/audit 下的审计记帐的权限； 默认情况下， 这个组是空的， 因此只有 root 用户可以读取审计记帐。 如果希望给某个用户指定审计复审权， 则可以将其加入 audit。 由于查看审计日志的内容可以提供关于用户和进程行为的大量深度信息， 在您委派这些权力时， 请务必谨慎行事。

18.5.4. 通过审计管道来实时监控

审计管道是位于设备文件系统中的自动复制 (cloning) 的虚拟设备， 用于让应用程序控制正在运行的审计记录流， 这主要是为了满足入侵检测和系统监控软件作者的需要。 不过， 对管理员而言， 审计管道设备也提供了一种无需冒审计记帐文件属主出现问题的麻烦， 或由于日志翻转而打断事件流的麻烦， 而实现实时监控的方便途径。 要跟踪实时事件流， 使用下面的命令行：

```
# praudit /dev/auditpipe
```

默认情况下， 审计管道设备节点只有 root 用户才能访问。 如果希望 audit 组的成员能够访问它， 应在 devfs.rules 中加入下述 devfs 规则：

```
add path 'auditpipe*' mode 0440 group audit
```

请参见 devfs.rules(5) 以了解关于配置 devfs 文件系统的进一步信息。

很容易配置出审计事件反馈循环， 也就是查看事件的操作本身会产生更多的事件。 例如，
如果所有的网络 I/O 均被审计， 又在 SSH 会话中执行 praudit(1)，
就会以很高的速率产生持续的审计事件流， 因为每显示一个事件都会产生新的事件。
建议您在需要在审计管道设备上执行 praudit 时， 选择一个没有进行细粒度 I/O
审计的会话来运行。

18.5.5. 审计记帐文件的轮转

审计计账只由内核写入， 且只能由 auditd 管理。 管理员不应尝试使用 newsyslog.conf(5)
或其它工具来完成审计日志的轮转工作。 您可以使用 audit 管理工具来关闭审计、 重新配置审计系统，
并完成日志轮转。 下面的命令将让审计服务创建新的审计日志， 并发信号给内核要求其使用新的日志。
旧日志将终止并被改名， 此时， 管理员就可以操作它了。

```
# audit -n
```

如果 auditd 服务程序没有在运行， 则这个命令将失败并给出错误提示。

在 /etc/crontab 加入如下设置， 将使 cron(8) 每十二小时将日志轮转一次。

```
0   */12    *    *    *    root    /usr/sbin/audit -n
```

这些修改会在您保存 /etc/crontab 后生效。

对于审计记帐文件基于尺寸的自动翻转， 可以通过 audit_control(5) 中的 filesz 选项来配置，
这个选项在这一章的配置文件一节中已经介绍过。

18.5.6. 压缩审计记帐

由于审计记帐文件会变得很大， 通常会希望在审计服务关闭它时， 对其进行压缩或归档。 audit_warn
脚本可以用来在一系列与审计有关的事件发生时， 执行一些用户定义的操作，
这也包括在审计记帐翻转时进行清理操作。 举例而言， 可以在 audit_warn
脚本中加入下列内容来在审计记帐关闭时压缩它：

```
#
# Compress audit trail files on close.
#
if [ "$1" = closefile ]; then
        gzip -9 $2
fi
```

其它存档操作也包括将审计记帐复制到一个中央的服务器， 删除旧的记帐文件，
或浓缩审计记帐并删除不需要的记录等。 这个脚本会在审计记帐文件正常关闭时执行一次，
因此在非正常关闭系统时， 就不会执行它了。

Chapter 19. 存储

19.1. 概述

这章介绍了 FreeBSD 中磁盘的使用方法。包括内存盘， 网络附属磁盘和标准的 SCSI/IDE 存储设备，以及使用 USB 的设备。

读完这章，您将了解到：

- FreeBSD 中用来描述硬盘上数据组织的术语 (partitions and slices)。
- 如何在您的系统上增加硬盘。
- 如何配置 FreeBSD 来使用 USB 存储设备。
- 如何设置虚拟文件系统，例如内存磁盘。
- 如何使用配额来限制磁盘空间的使用。
- 如何增加磁盘安全来预防功击。
- 如何刻录 CD 和 DVD 。
- 用于备份的多种存储媒介。
- 如何在 FreeBSD 上使用备份程序。
- 如何备份到软磁盘。
- 文件系统快照是什么， 以及如何有效地使用它们。

在读这章之前，您应该：

- 知道怎样去配置和安装新的 FreeBSD 内核 (配置FreeBSD的内核).

19.2. 设备命名

下面是在 FreeBSD 上被支持的物理存储设备和它们被分配的设备名。

表 7. 物理磁盘命名规则

驱动器类型	驱动设备命名
IDE 硬盘驱动器	ad
IDE CDROM 驱动器	acd
SCSI 硬盘以及 USB 大容量存储设备	da
SCSI CDROM 驱动器	cd
各类非标准 CDROM 驱动器	用于 Mitsumi CD-ROM 的 mcd 以及用于 Sony CD-ROM 驱动器的 scd
Floppy drives	fd
SCSI tape drives	sa
IDE tape drives	ast
Flash drives	fla for DiskOnChip® Flash device
RAID drives	aacd for Adaptec® AdvancedRAID, mlxd and mlyd for Mylex®, amrd for AMI MegaRAID®, idad for Compaq Smart RAID, twed for 3ware® RAID.

19.3. 添加磁盘

下面这节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，然后按操作规程来安装驱动器，控制器和驱动程序。由于

各厂家生产的产品各不相同，具体的安装细节不在此文档介绍之内。

以 root 用户登录。安装完驱动后，检查一下 /var/run/dmesg.boot 有没有找到新的磁盘。在我们的例子中新增加的磁盘就是 da1，我们从 /1 挂上它。 (如果您正添加 IDE 驱动器， 则设备名应该是 ad1)。

因为 FreeBSD 运行在 IBM-PC 兼容机上，它必须遵循 PC BIOS 分区规范。 这与传统的 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 您可以使用 dedicated 模式。另外， FreeBSD 必须安装在 PC BIOS 支持的分区内。FreeBSD 把分区叫作 slices 这可能会把人搞糊涂。您也可以在只安装 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。这不会影响其它操作系统的 fdisk 分区工具。

在 slice 方式表示下，驱动器被添加到 /dev/da1s1e。 可以读作：SCSI 磁盘，编号为 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，也可以简化为 /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现力限于每个磁盘 2^32-1 个扇区， 或 2TB。 fdisk(8) 格式允许的起始扇区号不能高于 2^32-1， 而分区尺寸也不能超过 2^32-1，这样一来通常情况下分区尺寸不能超过 2TB， 而磁盘尺寸则不能超过 4TB。 sunlabel(8) 格式的限制是每个分区 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。对于更大的磁盘， 可以使用 gpart(8) 来创建 GPT 分区。 GPT 除了支持大磁盘之外， 还不受 4 个 slice 的限制。

19.3.1. 使用 sysinstall(8)

1. 使用 Sysinstall

   您可以使用 sysinstall 命令的菜单来分区和标记一个新的磁盘。 这一操作需要有 root 权限，您可以直接使用 root 账户登录或者使用 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，上下滚动， 选择 Fdisk 条目。

2. fdisk 分区编辑器

   进入 fdisk 分区编辑器后，选择 A ，FreeBSD 将使用全部的磁盘。当被告知 "remain cooperative with any future possible operating systems"时，回答 YES。使用 W 保存刚才的修改。现在使用 Q 退出 FDISK 编辑器。下面会看到有关 "主引导区" 的信息。
   现在您已经在运行的系统上添加了一个磁盘， 因此应该选择 None。

3. Disk Label 编辑器

   接下来，您应该退出 sysinstall 并且再次启动它，并按照上面的步骤直接进入 Label 选项。进入磁盘标签编辑器。 这就是您要创建的 BSD 分区。一个磁盘最多可以有 8 个分区，标记为 a-h。有几个分区标签有特殊的用途。 a 分区被用来作为根分区(/)。
   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。其它分区作为一般分区来使用。

   sysinstall 的标签编辑器用 e 表示非 root 和非 swap 分区。在标签编辑器中，可以使用键入 C 创建一个文件系统。当提示这是否是一个 FS（文件系统）或 swap 时，选择 FS，然后给出一个加载点（如： /mnt）。 当在 post-install 模式时添加一个磁盘， sysinstall 不会在 /etc/fstab 中创建记录，所以是否指定加载点并不重要。

   现在已经准备把新标签写到磁盘上，然后创建一个文件系统，可以按下 W。出现任何错误都会不能创建新的分区。可以退出标签编辑 器然后重新执行 sysinstall 。

4. 完成

   下面一步就是编辑 /etc/fstab，为您的磁盘添加一个新 记录。

19.3.2. 使用命令行工具

19.3.2.1. 使用 Slices

这步安装将允许磁盘与可能安装在您计算机上的其它操作系统一起
正确工作，而不会搞乱其它操作系统的分区。推荐使用这种方法来安装 新磁盘，除非您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 #加上标签
# bsdlabel -e da1s1 # 现在编辑您刚才创建的磁盘分区
# mkdir -p /1
# newfs /dev/da1s1e # 为您创建的每个分区重复这个操作
# mount /dev/da1s1e /1 # 挂上分区
# vi /etc/fstab # 完成之后，添加合适的记录到您的 /etc/fstab文件。
```

如果有一个 IDE 磁盘，记得要用 ad 替换前面的 da。

19.3.2.2. 专用模式

如果您并没有安装其它的操作系统，可以使用 dedicated 模式。记住这种模式可能会弄乱 Microsoft 的操作系统，但不会对它进行破坏。 它不识别找到的 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           # 创建 `e' 分区
# newfs /dev/da1e
# mkdir -p /1
# vi /etc/fstab         # 为 /dev/da1e添加一个记录
# 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           # 为 /dev/da1e添加一个记录
# mount /1
```

19.4. RAID

19.4.1. 软件 RAID

19.4.1.1. 连接磁盘驱动器配置 (CCD)

选择一个大容量存储比较好的解决方案，最重要的因素是产品的速度、 性能和成本。
通常这三者不可能都满足;要获得比较快和可靠的大容量存储设备，
就比较昂贵。但如果将成本降下来，那它的速度或可靠性就会打折扣。

在设计下面描述的系统时， 价格被选为最重要的因素， 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 驱动器上的所有数据都被备份到了 CD-R 盘， 可以很容易地对数据进行恢复。

在选择一个大容量的存储解决方案时，第一步是要设计您自己的需求。
如果您的需求更偏重于速度和性能，那么您的解决方案将就不同于上面的设计。

19.4.1.1.1. 安装硬件

除了 IDE 系统磁盘外，还有三个 Western Digital 30GB、5400 RPM 的 IDE 磁盘构成了大约 90G 的连接磁盘驱动存储空间。 理想情况是每个 IDE 硬盘都独占 IDE 控制器和数据线，
但为了尽可能降低成本， 通常并不会安装更多的控制器， 而是通过配置跳线，使每个 IDE 控制器都管理一个主盘和一个从盘。

重启动后，系统 BIOS 被配置成自动检测硬盘。FreeBSD 检测到它们：

```
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 没有检测到它们，请确定它们的跳线是否设置 正确。大多数 IDE 磁盘有一个 "Cable Select" 跳线。这个 不是 设置 master/slave 硬盘的跳线。查阅文档信息来确定正确的跳线设置。

接下来考虑的是，如何创建文件系统。应该好好研究一下 vinum(4) (Vinum 卷管理程序)和 ccd(4) 两种方式，在这里我们选择 ccd(4)

19.4.1.1.2. 安装 CCD

ccd(4) 允许用户将几个相同的的磁盘通过一个逻辑文件系统 连接起来。要使用 ccd(4)，您需要在内核中配置 ccd(4) 支持选项。把这行加入到内核配置文件中，然后重建内核：

```
device   ccd
```

对 ccd(4) 的支持也可以内核模块的形式载入。

要安装 ccd(4), 首先需要使用 bsdlabel(8) 来编辑硬盘：

```
bsdlabel -w ad1 auto
bsdlabel -w ad2 auto
bsdlabel -w ad3 auto
```

此处将整个硬盘创建为 ad1c, ad2c 和 ad3c。

下一步是改变 disklable 的类型。也可以使用 bsdlabel(8) 来编辑：

```
bsdlabel -e ad1
bsdlabel -e ad2
bsdlabel -e ad3
```

这儿在每个已经设置了 EDITOR 环境变量的磁盘上打开了 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 分区给 ccd(4) 用。这可以是 c 分区的一个副本， 但 fstype 必须 是 4.2BSD。做完之后，您会看到一面这些：

```
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

现在已给每个磁盘都加上了标签，下面需要建立 ccd(4)。要这样做， 需要使用 ccdconfig(8) 工具，同时要提供类似下面的选项：

```
ccdconfig ccd0 32 0 /dev/ad1e /dev/ad2e /dev/ad3e
```

每个选项的意义和用法如下所示： 配置设备的第一个参数，在这是 /dev/ccd0c。 /dev/ 部分是任选项。 下一个参数是文件系统的插入页(interleave)。插入页定义了一个 磁盘块中一个分段或条带(stripe)的大小，通常是 512 个字节。所以一个为 32 的插入页将是 16,384 字节。 插入页为 ccdconfig(8) 附带了标记。如果您要启用驱动器镜像， 需要在这儿指定它。在这个配置中没有做 ccd(4) 的镜像，所以把它 设为 0 (zero)。 ccdconfig(8) 的最后配置是设备的排列问题。使用完整的设备路径名。

运行 ccdconfig(8) 后 ccd(4) 就配置好了。现在要创建文件 系统了，参考 newfs(8) 选项，执行下同的命令：

```
newfs /dev/ccd0c
```

19.4.1.1.4. 自动创建

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
```

当重新启动系统时，如果 /etc/ccd.conf 存在， 脚本 /etc/rc 就运行 ccdconfig -C。 这样就能自动配置 ccd(4) 以到它能被挂上。

如果启动进入了单用户模式，在 mount(8) 上 ccd(4) 之前，需要执行下面的命令来配置队列：

```
ccdconfig -C
```

要自动挂接 ccd(4),需要为 ccd(4) 在 /etc/fstab 中配置一个记录，以便在启动时它能被挂上。 如下所示：

```
/dev/ccd0c      /media    ufs   rw    2    2
```

19.4.1.2. Vinum 卷管理

Vinum 卷管理是一个实现虚拟磁盘的块驱动设备工具。它使磁盘从块设备的接口和数据映射中独立出来。与传统的存储设备相比，增加了 灵活性、性能和可靠性。 vinum(4) 实现了 RAID-0、RAID-1 和 RAID-5 三种模式，它们既可以独立使用，也可组合使用。

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。 这些硬件不需要 FreeBSD 指定软件来管理 RAID 系统。

使用 BIOS 支持的硬件，一般情况下这些硬件可以自行操作。 下面是一个简明的描述设置一个 Promise IDERAID 控制器。
当硬件设备装好且系统重启后，屏幕上显示一个询问信息。接着进入硬件设置屏幕。在这里，您可以把所有的磁盘联合在一起使用。这样 FreeBSD 将磁盘看作一个驱动器。其它 级别的 RAID 也可以相应的进行设置。

19.4.3. 重建 ATA RAID1 阵列

FreeBSD 允许您热插拔阵列中损坏的磁盘。 在您重新启动系统之前请注意这一点。

您可能会在 /var/log/messages 或者在 dmesg(8) 的输出中看到类似下面这些的内容：

```
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)，查看更多的信息：

```
# atacontrol 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

# atacontrol status ar0
ar0: ATA RAID1 subdisks: ad4 ad6 status: DEGRADED
```

1. 首先您应将包含故障盘的 ata 通道卸下， 以便安全地将其拆除：

```
# atacontrol detach ata3
```

2. 换上磁盘
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. 将新盘作为热备盘加入阵列：

```
# atacontrol addspare ar0 ad6
```

5. 重建阵列：

```
# atacontrol rebuild ar0
```

6. 可以通过下面的命令来查看进度：

```
# 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 存储设备

到目前为止，有许多外部外部存储解决方案， 例如：通用串行总线 (USB)：硬盘、USB thumbdrives、CD-R burners 等等。 FreeBSD 为这些设备提供了支持。

19.5.1. 配置

USB 大容量存储设备驱动，在 umass(4), 中提供了对 USB 存储设备的支持。如果您使用 GENERIC 内核，您不必要改变配置文件里的任何内容。
如果您使用了定制的内核，就要确定下面的行出现在您的内核配置文件里：

```
device scbus
device da
device pass
device uhci
device ohci
device ehci
device usb
device umass
```

umass(4) 驱动程序使用 SCSI 子系统来访问 USB 存储设备， 您的 USB 设备将被系统看成为一个 SCSI 设备。依靠您主板上的 USB 芯片， 您只须选择 device uhci 或用于 USB 1.X 支持的 device ohci 二者之一即可， 但是两者都加入内核配置文件当中也是无害的。 对于 USB 2.X 控制器的支持由 ehci(4) 提供 (device ehci 这一行)。 不要忘了如果您加入了上面的几行要重新编译和安装内核。

如果您的 USB 设备是一个 CD-R 或 DVD 刻录机， SCSI CD-ROM 驱动程序， cd(4), 就必须加入内核中通过下面这行：

```
device cd
```

由于刻录机被视为 SCSI 设备， 因此， 不应该在内核配置文件中使用 atapicam(4) 驱动程序。

19.5.2. 测试配置

配置好后准备进行测试：插入您的 USB 设备， 在系统信息中 (dmesg(8)), 应该会出现像下面的设备：

```
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)
```

当然啦，商标，设备标识 (da0) 和其它的细节信息会根据您的配置不同 而有所不同。

因为 USB 设备被看作 SCSI 设备中的一个， camcontrol 命令也能够用来列出 USB 存储设备和系统的关联：

```
# camcontrol devlist
<Generic Traveling Disk 1.11>     at scbus0 target 0 lun 0 (da0,pass0)
```

如果设备上已经包含了文件系统， 现在应该就可以挂接它了。 如果需要， 请参阅 添加磁盘 来了解如何在 USB驱动器上格式化和创建分区。

允许非可信用户挂载任意介质， 例如通过使用前面介绍的 vfs.usermount 来启用的功能， 从安全角度来看是很不保险的。 FreeBSD 中的绝大多数文件系统并不提供针对恶意设备的内建防护能力。

如果希望设备能够被普通用户挂接， 还需要做一些其它操作。 首先， 在 USB 存储设备连接到计算机上时，系统自动生成的设备文件， 必须是该用户能够读写的。 一种做法是让所有属于 operator 组的用户都可以访问该设备。 要完成这项工作， 首先需要用 pw(8) 来给用户指定组。 其次，在生成设备文件时， operator 组应能读写它们。 这可以通过在 /etc/devfs.rules 中增加一些相应的设置来实现：

```
[localrules=5]
add path 'da*' mode 0660 group operator
```

如果系统中已经有其它 SCSI 磁盘， 则上述操作必须做一些变化。 例如，如果系统中已经存在了设备名为 da0 到 da2 的磁盘， 则第二行应改为：

```
add path 'da[3-9]*' mode 0660 group operator
```

这会将系统中已经存在的磁盘， 排除在属于 operator 组的设备之外。

另外， 您还需要在 /etc/rc.conf 文件中， 启用 devfs.rules(5) 规则集：

```
devfs_system_ruleset="localrules"
```

接下来， 需要配置内核， 令普通用户能够挂接文件系统。 最简单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时才会生效。 另外， 您也可以使用 sysctl(8) 来设置这个变量。

最后一步是创建将要挂接文件系统的目录。 这个目录必须是属于将要挂接文件系统的用户的。 以 root 身份为用户建立属于该用户的 /mnt/username (此处 username 应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
```

假设已经插入了一个 USB 读卡设备， 并且系统将其识别为 /dev/da0s1， 由于这些设备通常是 FAT

文件系统， 用户可以这样挂接它们：

```
% mount -t msdosfs -o -m=644,-M=755 /dev/da0s1 /mnt/username
```

如果拔出设备 (必须首先将其对应的磁盘卷卸下)， 则您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. 深入阅读

除了 Adding Disks 和 Mounting and Unmounting File Systems 章之外， 阅读各种手册页也是有益的：umass(4), camcontrol(8), 和 FreeBSD 8.X 的 usbconfig(8) 或者对于更早期 FreeBSD 版本的 usbdevs(8)。

19.6. 创建和使用光学介质(CD)

19.6.1. 介绍

CD 与普通的磁盘相比有很多不同的特性。最初它们是不能被用户写入的。
由于没有磁头和磁道移动时的延迟，所以它们可以连续的进行读取。
方便的在两个系统之间进行数据的传输，比起相同大小的存储介质来说。

CD 有磁道，这关系到数据读取时的连续性而不是物理磁盘的性能。 要在 FreeBSD 中制作一个 CD，您要准备好要写到 CD 上的数据文件， 然后根据每个 tracks 写入到 CD。

ISO 9660 文件系统被设计用来处理这些差异。 但令人遗憾的是， 它也有一些其他文件系统所没有的限制，不过幸运的是， 它提供了一项扩展机制， 使得正确写入的 CD 能够超越这些限制，而又能在不支持这些扩展的系统上正常使用。

sysutils/ port 包括了 mkisofs(8)， 这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。他也提供了对于一些扩展的支持选项，下面将详细介绍。

使用哪个工具来刻录 CD 取决于您的 CD 刻录机是 ATAPI 的， 还是其他类型的。 对于 ATAPI CD 刻录机，可以使用基本系统附带的 burncd 程序。 SCSI 和 USB CD 刻录机， 则需要配合 cdrecord 程序使用，它可以通过 sysutils/cdrtools port 安装。 除此之外， 在 ATAPI 接口的刻录机上， 也可以配合 ATAPI/CAM 模块 来使用 cdrecord 以及其它为 SCSI 刻录机撰写的工具。

如果您想使用带图形界面的 CD 刻录软件， 可以考虑一下 X-CD-Roast 或 K3b。这些工具可以通过使用预编译安装包， 或通过 sysutils/xcdroast 和 sysutils/k3b ports 来安装。 X-CD-Roast 和 K3b 需要 ATAPI/CAM 模块 配合 ATAPI 硬件。

19.6.2. mkisofs

mkisofs(8) 程序作为 sysutils/cdrtools port 的一部分， 将生成 ISO 9660 文件系统，其中包含 UNIX® 命名空间中的文件名。 最简单的用法是：

```
# mkisofs -o imagefile.iso /path/to/tree
```

这个命令将创建一个包含 ISO9660 文件系统的 imagefile.iso 文件，它是目录树 /path/to/tree

的一个副本。 在处理过程中， 它将文件名称映射为标准的 ISO9660
文件系统的文件名，将排除那些不典型的 ISO 文件系统的文件。

有很多选项能够用来克服那些限制。特别的，-R 选项能够启用 Rock Ridge 扩展一般的 UNIX® 系统，-J
选项能启用用于 Microsoft 系统的 Joliet 扩展，-hfs 选项能用来创建用于 Mac OS® 系统的 HFS 文件系统。

对于那些即将要在 FreeBSD 系统中使用 CD 的人来说，-U 选项能用来消除所有文件名的限制。当使用 -R
选项时，它会产生一个 文件系统映像，它与您从那儿启动 FreeBSD
树是一样的，虽然它在许多方面也违反了 ISO 9660 的标准。

最后一个常用的选项是 -b。 它用来指定启动映像的位置， 用以生成 "El Torito" 启动 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) 会以常说的
"软盘模拟" 方式来创建 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 发行版磁盘， 并不使用模拟模式； 这种情况下， 需要使用 -no-emul
-boot 选项。 因此， 如果 /tmp/myboot 是一个包含了启动映像文件 /tmp/myboot/boot/cdboot
的可引导的 FreeBSD 系统， 您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
```

完成这些工作之后， 如果您的内核中配置了 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
```

可以发现 /mnt 和 /tmp/myboot 是一样的。

还可以使用 mkisofs(8) 的其它选项来调整它的行为。特别是修改 ISO 9660 的划分格式，创建 Joliet 和 HFS
格式的磁盘。查看 mkisofs(8) 联机手册得到更多的帮助。

19.6.3. burncd

如果用的是 ATAPI 的 CD 刻录机，可以使用 burncd 命令来刻录您的 CD ISO 映像文件。 burncd
命令是基本 系统的一部分，中以使用 /usr/sbin/burncd 来安装。 用法如下：

```
# burncd -f cddevice data imagefile.iso fixate
```

在 cddevice 上刻录一份 imagefile.iso 的副本。 默认的设备是 /dev/acd0。 请参考 burncd(8)
以了解设置写入速度的参数，如何在刻录完成之后自动弹出CD，以及刻录音频数据。

19.6.4. cdrecord

如果没有一个 ATAPI CD 刻录机，必须使用 cdrecord 来刻录您的 CD 。 cdrecord
不是基本系统的一部分;必须 从 sysutils/cdrtools 或适当的 package
安装它。基本系统的变化可能会引起这个程序的错误。可能是由 "coaster"
引起的。当升级系统时，同时需要升级 port， 或者如果您 使用 -STABLE， 那么在升级到新版本时也要升级
port。

cdrecord 有许多选项，基本用法与 burncd 相似。刻录一个 ISO 9660 映像文件只需这样做：

```
# cdrecord dev=device imagefile.iso
```

使用 cdrecord 的比较巧妙的方法是找到使用的 dev 。要找到正确的设置，可以使用 cdrecord 的 -scanbus
标记，这会产生这样的结果：

```
# 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) *
```

这个列表列出了设备的的适当的 dev 值。找到您的 CD burner ,使用三个用逗号分隔的数值来表示 dev.在这个例子中，CRW 是 dev=1,5,0，所以正确的输入应是 dev=1,5,0
。有一个很容易的方法可以指定这个值;看看 cdrecord(1) 的介绍了解有关音轨，控制速度和其他的东西。

19.6.5. 复制音频 CD

您可以这样复制 CD，把 CD 上面的音频数据解压缩出一系列的文件， 再把这些文件写到一张空白 CD 上。这个过程对于 ATAPI 和 SCSI 驱动器来说有些微的不同。

Procedure: SCSI 驱动器

1. 使用 cdda2wav 来解压缩音频。

   ```
   % cdda2wav -vall -D2,0 -B -Owav
   ```

2. 使用 cdrecord 来写 .wav 文件。

   ```
   % cdrecord -v dev=2,0 -dao -useinfo  *.wav
   ```

   确保 2,0 被适当地设置了， 具体方法在 cdrecord 中有所描述。

Procedure: ATAPI 驱动器

借助于 ATAPI/CAM 模块， cdda2wav 同样也能在 ATAPI 设备上使用。
此工具比起下面推荐的方法通常是个更好的选择(抖动修正， 字节序问题， 等等)。

1. ATAPI CD 驱动用 /dev/acddtnn表示每个轨道， 这里 d 是驱动器号， nn 是轨道号，由两位小数位组成，省略前缀零。 所以第一个盘片上的第一个轨道就是 /dev/acd0t01，第二个就是 /dev/acd0t02，第三个就是 /dev/acd0t03，等等。

   请务必确认在 /dev 中出现了对应的文件。 如果您发现有某些项目缺失，
   则应强制系统重新识别介质：

   ```
   # dd if=/dev/acd0 of=/dev/null count=1
   ```

2. 使用 dd(1) 解压缩每个轨道。当解压缩文件的时候您也必须使用 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 把解压缩的文件刻录到光盘上。您必须指定 这些文件是音频文件，这样 burncd 会在刻录完成时 结束光盘。

   ```
   # burncd -f /dev/acd0 audio track1.cdr track2.cdr ... fixate
   ```

19.6.6. 复制数据 CD

您可以把数据 CD 复制成一个与之等价的映像文件， 可以使用 mkisofs(8) 创建这种文件，
或使用它来复制任何数据 CD。 这里给出的例子假定您的 CDROM 设备是 acd0，
您应将其替换为您实际使用的 CDROM 设备。

```
# dd if=/dev/acd0 of=file.iso bs=2048
```

现在您有一个映像文件了，您可以像上面描述的那样把它刻录成 CD。

19.6.7. 使用数据 CD

现在您已经创建了一张标准的数据 CDROM，您或许想要 挂载来读取上面的设备。 默认情况下，mount(8) 假定文件系统是 ufs 类型的。如果您尝试下面的命令：

```
# mount /dev/cd0 /mnt
```

您会得到一条 Incorrect super block 的错误信息，没有挂载成功。CDROM 不是 UFS 文件系统，所以试图这样挂载它是 是不行的。您需要告诉 mount(8) 文件系统是 ISO9660 类型的，这样就可以了。只需要指定 mount(8) 的 -t cd9660 选项。例如， 如果您想要挂载 CDROM 设备， /dev/cd0 到 /mnt 目录，您需要执行：

```
# mount -t cd9660 /dev/cd0 /mnt
```

注意您的设备名 (在这个例子中是 /dev/cd0)可能 有所不同，取决于您的 CDROM 使用的接口。另外， -t

cd9660 选项等同于执行 mount_cd9660(8)。上面的例子可以缩短 为：

```
# mount_cd9660 /dev/cd0 /mnt
```

用这种方法您基本可以使用任何买到的数据 CDROM。 然而某些有 ISO 9660 扩展的光盘可能会行为古怪。 例如，joliet 光盘用两个字节的 unicode 字符存储所有的文件名。 FreeBSD 内核并不使用 Unicode， 但 FreeBSD CD9660 驱动可以将 Unicode 字符自动转换为内核可以识别的形式。
如果您发现有些非英文字符显示为问号， 就绪要使用 -C 选项来指定字符集了。 欲了解进一步的详情， 请参见联机手册 mount_cd9660(8)。

如果希望通过 -C 选项来进行字符集转换， 则内核会需要加载 cd9660_iconv.ko 模块。 这项工作可以通过在 loader.conf 中加入下列配置：

```
cd9660_iconv_load="YES"
```

并重新启动计算机来完成， 除此之外， 也可以通过 kldload(8) 来手动加载。

有时候，当您试图挂载 CDROM 的时候，会得到一条 Device not configured 的错误信息。这通常 表明 CDROM 驱动认为托盘里没有光盘， 或者驱动器在总线上不可见。 需要几秒钟时间等待 CDROM 驱动器辨别已经接到反馈的信息， 请耐心等待。

有时候，SCSI CDROM 可能会找不到，因为没有足够的 时间来应答总线的 reset 信号。如果您有一个 SCSI CDROM 请将下面的选项添加到您的内核 配置文件并重建您的内核。

```
options SCSI_DELAY=15000
```

这个告诉您的 SCSI 总线启动时暂停 15 秒钟， 给您的 CDROM 驱动器足够的机会来应答 总线 reset 信号。

19.6.8. 刻录原始数据 CD

您可以选择把一个文件目录刻录到 CD 上而不用 创建 ISO 9660 文件系统。有些人这么做是为了备份的 目的。这个运行的比刻录一个标准 CD 速度要快得多：

```
# burncd -f /dev/acd1 -s 12 data archive.tar.gz fixate
```

要重新找回这样刻录到 CD 上的数据， 您必须从原始设备节点读取数据：

```
# tar xzvf /dev/acd1
```

您不能像挂载一个通常的 CDROM 一样挂载这张光盘。 这样的 CDROM 也不能在除了 FreeBSD 之外的任何操作系统上读出。 如果您想要可以挂载 CD，或者 和另一种操作系统共享数据，您必须像上面描述的那样使用 mkisofs(8)。

19.6.9. 使用 ATAPI/CAM 驱动

这个驱动允许 ATAPI 设备(CD-ROM, CD-RW, DVD 驱动器等…)通过 SCSI 子系统访问， 这样允许使用像 sysutils/cdrdao 或者 cdrecord(1) 这样的程序。

要使用这个驱动， 您需要把下面这行添加到 /boot/loader.conf 文件中：

```
atapicam_load="YES"
```

接下来， 重新启动计算机。

如果您希望将 atapicam(4) 以静态联编的形式加入内核，
则需要在内核配置文件中加入这行：

```
device atapicam
```

此外还需要在内核配置文件中加入：

```
device ata
device scbus
device cd
device pass
```

这些应该已经有了。 然后， 重新联编并安装新内核， 并重新启动计算机。

在引导过程中， 您的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 PIO4
cd0 at ata1 bus 0 target 0 lun 0
cd0: <MATSHITA CDRW/DVD UJDA740 1.00> Removable CD-ROM SCSI-0 device
cd0: 16.000MB/s transfers
cd0: Attempt to query device size failed: NOT READY, Medium not present - tray closed
```

驱动器现在可以通过 /dev/cd0 设备名访问了，例如要 挂载 CD-ROM 到 /mnt，只需要键入下面的 命令：

```
# mount -t cd9660 /dev/cd0 /mnt
```

作为 root，您可以运行下面的 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)
```

这样 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 系统的更多信息， 可以参阅 atapicam(4) 和 cam(4) 手册 页。

19.7. 创建和使用光学介质(DVD)

19.7.1. 介绍

和 CD 相比，DVD 是下一代光学存储介质技术。 DVD 可以容纳比任何 CD
更多的数据，已经成为现今视频出版业的标准。

我们称作可记录 DVD 的有五种物理记录格式：

- DVD-R：这是第一种可用的 DVD 可记录格式。 DVD-R 标准由 DVD Forum 定义。 这种格式是一次可写的。
- DVD-RW：这是 DVD-R 标准的可覆写版本。 一张 DVD-RW 可以被覆写大约 1000 次。
- DVD-RAM：这也是一种被 DVD Forum 所支持的可覆写格式。 DVD-RAM 可以被看作一种可移动硬盘。 然而，这种介质和大部分 DVD-ROM 驱动器以及 DVD-Video 播放器不兼容； 只有少数 DVD 刻录机支持 DVD-RAM。 请参阅 使用 DVD-RAM 以了解关于如何使用 DVD-RAM 的进一步详情。
- DVD+RW：这是一种由 DVD+RW Alliance 定义的可覆写格式。一张 DVD+RW 可以被覆写大约 1000 次。
- DVD+R：这种格式是 DVD+RW 格式的一次可写变种。

一张单层的可记录 DVD 可以存储 4,700,000,000 字节，相当于 4.38 GB 或者说 4485 MB (1 千字节等于 1024 字节)。

必须说明一下物理介质与应用程序的分歧。 例如 DVD-Video 是一种特殊的文件系统，可以被覆写到任何可记录的 DVD 物理介质上： DVD-R、DVD+R、DVD-RW 等等。在选择介质类型之前， 您一定要确认刻录机和 DVD-Video 播放器 (一种单独的播放器或者计算机上的 DVD-ROM 驱动器) 是和这种介质兼容的。

19.7.2. 配置

growisofs(1) 将被用来实施 DVD 刻录。 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 介质类型。

这些工具将使用 SCSI 子系统来访问设备，因此 ATAPI/CAM 支持 必须加入内核。 如果您的刻录机采用 USB 接口则不需要这么做，请参考 USB 存储设备 来了解 USB 设备配置的进一步详情。

此外，还需要启用 ATAPI 设备的 DMA 支持。 这一工作可以通过在 /boot/loader.conf 文件中加入下面的行来完成：

```
hw.ata.atapi_dma="1"
```

试图使用 dvd+rw-tools 之前您应该参考 dvd+rw-tools 硬件兼容性列表 是否有与您的 DVD 刻录机有关的信息。

如果您想要一个图形化的用户界面，您应该看一看 K3b (sysutils/k3b)，它提供了 growisofs(1) 的一个友好界面和许多其他刻录工具。

19.7.3. 刻录数据 DVD

growisofs(1) 命令是 mkisofs 的前端，它会调用 mkisofs(8) 来创建文件系统布局，完成到 DVD 上的刻录。 这意味着您不需要在刻录之前创建数据映像。

要把 /path/to/data 目录的数据刻录到 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 选项传递给 mkisofs(8) 用于文件系统创建 (这表示创建带有带有 joliet 和 Rock Ridge 扩展的 ISO 9660 文件系统)， 参考 mkisofs(8) 联机手册了解更多细节。

选项 -Z 用来在任何情况下初始刻录会话： 不管多会话与否。 DVD 设备，/dev/cd0，必须依照您的配置做出改变。 -dvd-compat 参数会结束光盘， 光盘成为不可附加的。这会提供更多的和 DVD-ROM 驱动器的介质兼容性。

也可以刻录成一个 pre-mastered 映像, 例如记录一个映像文件 imagefile.iso, 我们可以运行:

```
# growisofs -dvd-compat -Z /dev/cd0=imagefile.iso
```

刻录的速度可以被检测到并自动进行调整， 根据介质和驱动器的使用情况。如果您想强制改变速度，
可以使用 -speed= 参数。更多的信息，请看 growisofs(1) 联机手册。

如果需要在刻录的编录中添加超过 4.38GB 的单个文件， 就必须使用 mkisofs(8)
或其他相关工具 (例如 growisofs(1)) 的 -udf -iso-level 3 参数来创建 UDF/ISO-9660
混合文件系统。 只有在创建 ISO 映像文件或直接在盘上写数据时才需要这样做。
以这种方式创建的光盘必须通过 mount_udf(8) 工具以 UDF 文件系统挂载，
因此只有操作系统支持 UDF 时才可以这样做，
否则盘上的文件数据可能会无法正确读出。

要创建这样的 ISO 文件:

```
% mkisofs -R -J -udf -iso-level 3 -o imagefile.iso /path/to/data
```

直接将文件刻录到光盘上:

```
# 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 映像文件时， 就不需要在运行 growisofs(1)
来将映像文件刻录成光盘时指定任何额外的选项了。

另外， 在映像文件中增加或直接刻录巨型文件时， 还需要注意使用最新的
sysutils/cdrtools (包含了 mkisofs(8))， 因为旧版并不提供巨型文件支持。
如果您遇到问题， 也可以尝试一下开发版本的软件包， 例如 sysutils/cdrtools-devel
并参阅 mkisofs(8) 联机手册。

19.7.4. 刻录 DVD-Video

DVD-Video 是一种特殊的基于 ISO 9660 和 micro-UDF (M-UDF) 规范的文件系统。 DVD-Video
也呈现了一个特殊的数据格式， 这就是为什么您需要一个特殊的程序像 multimedia/dvdauthor 来制作
DVD 的原因。

如果您已经有了 DVD-Video 文件系统的映像，
就可以以同样的方式制作另一个映像，可以参看前面章节的例子。 如果您想制作 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录 DVD-Video:

```
# growisofs -Z /dev/cd0 -dvd-video /path/to/video
```

-dvd-video 选项将传递给 mkisofs(8) 并指示它创建一个 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 在每一次使用前必须先格式化。 growisofs(1)
程序将会适时的自动对其进行适当的处理， 这是 recommended 的方式。您也可以使用 dvd+rw-format
来对 DVD+RW 进行格式化:

```
# dvd+rw-format /dev/cd0
```

您只需要执行这样的操作一次，牢记只有空白的 DVD+RW
介质才需要格式化。您可以以前面章节同样的方式来刻录 DVD+RW。

如果您想刻录新的数据 (刻录一个新的完整的文件系统 而不仅仅是追加一些数据) 到
DVD+RW，您不必再将其格式化成空白盘， 您只须要直接覆盖掉以前的记录即可。
(执行一个新的初始化对话), 像这样：

```
# growisofs -Z /dev/cd0 -J -R /path/to/newdata
```

DVD+RW 格式化程序为简单的向以前的记录追加数据提供了可能性。
这个操作有一个新的会话和一个已经存在的会话合并而成。 它不需要多个写会话过程， growisofs(1)
将在介质上 增加 ISO 9660 文件系统。

例如，我们想追加一些数据到到我们以前的 DVD+RW 上，我们可以使用下面的命令：

```
# growisofs -M /dev/cd0 -J -R /path/to/nextdata
```

在以后的写操作时， 应使用与最初的刻录会话时相同的 mkisofs(8) 选项。

如果您想获得与 DVD-ROM 驱动更好的兼容性，可以使用 -dvd-compat 选项。 在
DVD+RW 这种情况下， 这样做并不妨碍您添加数据。

如果出于某种原因您真的想要空白介质盘， 可以执行下面的命令：

```
# growisofs -Z /dev/cd0=/dev/zero
```

19.7.6. 使用 DVD-RW

DVD-RW 接受两种光盘格式：增补顺序写入和受限式覆写。默认的 DVD-RW 盘是顺序写入格式。

空白的 DVD-RW 能够直接进行刻录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。

要格式化一张 DVD-RW 为顺序写入模式，运行：

```
# 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 选项来进行，如果 DVD-RW 要以 Disk-At-Once
(DAO) 模式刻录的话。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) 试图最低限度的检测
(快速格式化) 介质并进行 DAO 写入。

事实上对于任何 DVD-RW 都应该使用受限式覆写模式，
这种格式比默认的增补顺序写入更加灵活。

在一张顺序 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
```

如果您想在您以前的刻录上附加数据，您必须使用 growisofs(1) 的 -M 选项。然而，
如果您在一张增补顺序写入模式的 DVD-RW 上附加数据，
将会在盘上创建一个新的区段，结果就是一张多区段光盘。

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， 您只是要用 -Z 选项覆写光盘，这和
DVD+RW 的情形是相似的。也可以用和 DVD+RW 同样方式的 -M 选项把现存的 ISO 9660
文件系统写入光盘。 结果会是一张单区段 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

几乎没有哪个 DVD-ROM 驱动器支持多区段 DVD，它们大多数时候都只读取第一个区段。 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 (未关闭) 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
```

对 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。
结果就是一张单区段光盘。 这是在这些介质上用于在最初的写操作之后添加数据的方式。

介质上的一些空间用于区段之间区段的开始与结束。
因此，应该用大量的数据添加区段来优化介质空间。 对于 DVD+R 来说区段的数量限制为
154， 对于 DVD-R 来说大约是 2000，对于双层 DVD+R 来说是 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) 联机手册找到，在 dvd+rw-tools web site 和
cdwrite mailing list 联接中也可找到。

dvd+rw-mediainfo 命令的输出结果记录， 以及介质的问题会被用来做问题报告。
如果没有这些输出， 就很难帮您解决问题。

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 两种接口之一。 对于 ATAPI 设备， DMA 传输模式必须手工启用。 这一工作可以通过在 /boot/loader.conf 文件中增加下述配置来完成：

```
hw.ata.atapi_dma="1"
```

19.7.9.2. 初始化介质

如本章前面的介绍所言， DVD-RAM 可以视为一移动硬盘。 与任何其它型号的移动硬盘类似，首次使用它之前， 应首先 "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
```

您应根据实际情况将 acd0 改为您所使用的设备名。

19.7.9.3. 使用介质

一旦您在 DVD-RAM 上完成了前面的操作， 就可以像普通的硬盘一样挂接它了：

```
# mount /dev/acd0 /mnt
```

然后就可以正常地对 DVD-RAM 进行读写了。

19.8. 创建和使用软盘

把数据存储在软盘上有时也是十分有用的。 例如， 在没有其它可靠的存储介质，或只需将少量数据传到其他计算机时。

这一章将介绍怎样在 FreeBSD 上使用软盘。 在使用 DOS 3.5 英寸软盘时首要要涉及的就是格式化，但其概念与其它的软盘格式化极为类似。

19.8.1. 格式化软盘

19.8.1.1. 设备

软盘的访问像其它设备一样是通过在 /dev 中的条目来实现的。 直接访问软盘时， 只需简单地使用 /dev/fdN 来表示。

19.8.1.2. 格式化

一张软盘在使用这前必须先被低级格式化。
通常卖主已经做过了，但格式化是检测介质完整性的一种好方法。
尽管这有可能会强取大量（或少量）的硬盘大小，但 大部分磁盘都能被格式化设计为 1440kB 。

低级格式化软盘你需要使用 fdformat(1) 命令。这个程序需要设备名作为参数。

要留意一切错误信息，这些信息能够帮助你确定 磁盘的好与坏。

19.8.1.2.1. 软盘的格式化

使用 /dev/fdN 设备来格式化软盘。插入一张新的 3.5 英寸的软盘在你的设备中：

```
# /usr/sbin/fdformat -f 1440 /dev/fd0
```

19.8.2. 磁盘标签

经过低级格式化后， 你需要给它分配一个标签。 这个磁盘标签以后会被删去，
但系统需要使用它来确定磁盘的尺寸。

新的磁盘标签将会接管整个磁盘，会包括所有合适的关于软盘的 geometry 信息。 磁盘标签的 geometry 值列在 /etc/disktab中。

现在可以用下面的方法来使用 bsdlabel(8) 了：

```
# /sbin/bsdlabel -B -w /dev/fd0 fd1440
```

19.8.3. 文件系统

现在对软盘进行高级格式化。 这会在它上面安置一个新的文件系统，可使 FreeBSD 来对它进行读写。
在创建完新的文件系统后，磁盘标签将被消毁，所以如果你想重新格式化磁盘， 你必须重新创建磁盘标签。

软盘的文件系统可以选择 UFS 或 FAT 。 FAT 是通常情况下软盘比较好的选择。

要制作新的文件系统在软盘上，可以使用下面的命令：

```
# /sbin/newfs_msdos /dev/fd0
```

现在磁盘已经可以进行读取和使用。

19.8.4. 使用软盘

要使用软盘，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/mtools 程序。

19.9. 用磁带机备份

主流的磁带机有 4mm, 8mm, QIC, mini-cartridge 和 DLT。

19.9.1. 4mm (DDS: Digital Data Storage)

4mm 磁带机正在逐步取代 QIC 成为工作站备份数据的首选设备。 在 Conner 收购了 QIC
磁带机领域领先的制造商 Archive 之后不久， 即不再生产这种磁带机， 这使得这一趋势变得愈加明显。
4mm 的驱动器更加小和安静，但对于数据保存的可靠性仍不及 8mm 驱动器。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 的一样，读写关的寿命都不长，因为它们同样使用螺旋式的方式来读写。

这些设备的数据传输的速度约在 ~150 kB/s 到 ~500 kB/s 之间， 存储空间从 1.3 GB 到 2.0 GB
之间，硬件压缩可使空间加倍。磁带库 单元可以有 6 台磁带机，120
个磁带匣，以自动切换的方式使用同一个磁带柜， 磁带库的容量可达 240 GB 。

DDS-3 标准现在支持的磁带机容量最高可达到 12 GB (或压缩的 24 GB)。

4mm 和 8mm 同样都使用螺旋式读写的方式，所有螺旋式读写的优点及缺点， 都可以在 4mm 和 8mm

磁带机上看到。

磁带在经过 2,000 次的使用或 100 次的全部备份后，就该退休了。

19.9.2. 8mm (Exabyte)

8mm 磁带机是最常见的 SCSI 磁带机，也是磁带交换的最佳选择。几乎每个 工作站都有一台 2 GB 8mm 磁带机。8mm 磁带机可信度高、方便、安静。 卡匣小 (4.8 x 3.3 x 0.6 inches; 122 x 84 x 15 mm)而且不贵。8mm 磁带机
的下边是一个短短的读写头，而读写头的寿命取决于磁带经过读写头时，相对高 速运动情况。

数据传输速度约在 250 kB/s 到 500 kB/s 之间，可存储的空间从 300 MB 到 7 GB，硬件压缩可使空间加倍。磁带库单元可以有 6 台磁 带机，120 个磁带匣，以自动切换的方式使用同一个磁带柜，磁带库的容量可达 840+ GB。

Exabyte "Mammoth" 模型支持 12 GB 的容量在一个磁带 上(压缩后可达 24 GB)相当于普通磁带的二倍。

数据是使用螺旋式读写的方式记录在磁带上的，读写头和磁带约相差 6 度， 磁带以 270 度缠绕着轴，并抵住读写头，轴适时地旋转，使得磁带具有高密度，
从一端到另一端并可使磁道紧密地分布。

19.9.3. QIC

QIC-150 磁带和磁带机可能是最常见的磁带机和介质了。 QIC 磁带机是最便宜的 "正规" 备份设备。
它的缺点在于介质的价格较高。 QIC 磁带要比 8mm 或 4mm 磁带贵， 每 GB 的数据存储价格可能最高高出 5 倍。 但是， 如果您的需求能够为半打磁带所满足的话， 那么 QIC 可能是明智之选。 QIC 是 最常见的磁带机。 每个站点都会有某种密度的 QIC。 这有时是一种麻烦， QIC 有很多在外观上相似（有时一样），但是密度不同的磁带。 QIC 磁带机噪音很大。
它们在寻址以及读写时都会发出声音。 QIC 磁带的规格是 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。较新的 QIC 磁带机具有硬件压缩的功能。 QIC 的使用率愈来愈低，渐渐被 DAT 所取代。

数据以磁道的方式记录在磁带上，磁道数及磁道的宽度会根据容量而有所不同。
通常新的磁带机具有的向后兼容的读取功能（通常也具备写入的功能）。对于数据 的安全性，QIC 具有不错的评价。

磁带机在经过 5,000 次的使用后，就该退休了。

19.9.4. DLT

在这一章列出的磁带机中 DLT 具有最快的数据传输率。 1/2" (12.5mm) 的 磁带包含在单轴的磁带匣 (4 x 4 x 1 inches; 100 x 100 x 25 mm)中。磁带匣
的一边是一个旋转匣道，通过匣道的开合，可以让磁带卷动。磁带匣内只有一个 轴，而本章中所提到的其他磁带匣都是有两个轴的（9磁道磁带机例外）。

数据传输的速度约 1.5 MB/s，是 4mm, 8mm, 或 QIC 磁带机的三倍。 可存储的空间从 10 GB 到 20 GB，具有磁带机数据库。磁带机数据库 单元可以有 1 to 20 台磁带机，5 到 900 个磁带匣，磁带机数据库的容量可达 50 GB 到 9 TB 。

如果要压缩的话，DLT 型 IV 格式的磁带机最高可支持 70 GB 的存储 容量。

数据存储在平行于磁带运行方向的磁道上（就像 QIC 磁带），一次写入两个 磁道。读写头的寿命相当长，每当磁带停止前进，磁带与读写头之间没有相对运动。

19.9.5. AIT

AIT 是 Sony 开发的一种新格式，每个磁带最高可以存储 50 GB。磁带 机使用内存芯片来保存磁带上的索引内容。这个索引能够被磁带机驱动器快速阅读 来搜索磁带机上文件所处的位置，而不像其他的磁带机需要花几分钟的时间才能找 到文件。像 SAMS:Alexandria 这样的软件：能够操 作四十或者更多的 AIT

磁带库，直接使用内存芯片来进行通信把内容显示在屏幕上，
以决定把什么文件备份到哪个磁带上，加载和恢复数据。

像这样的库成本大概在 $20,000 美元左右，零售市场可能还要贵一点。

19.9.6. 第一次使用新的磁带机

当在一块完全空白的磁带上尝试定入数据时，会得到类似下面这样的错误信息：

```
sa0(ncr1:4:0): NOT READY asc:4,1
sa0(ncr1:4:0):  Logical unit is in process of becoming ready
```

信息指出这块磁带没有块编号 (block 编号为 0)。在 QIC-525 之后的所有 QIC 磁带，都采用 QIC-525 标准，必须写入一个 Identifier Block 。对于这种问题， 有以下两种解决的办法：

- 用mt fsf 1 可以让磁带机对磁带写入 Identifier Block 。
- 使用面板上的按钮磁带。

  再插入一次，并存储 dump 数据到磁带上。

  这时dump 将传回 DUMP: End of tape detected ，然后您会得到这样的错误信息： HARDWARE FAILURE info:280 asc:80,96。

  这时用 mt rewind 来倒转磁带。

  磁带操作的后续操作就完成了。

19.10. 用软盘备份

19.10.1. 能够使用软盘来备份数据吗

软磁盘通常是用来备份的设备中不太合适的设备：

- 这种设备不太可靠，特别是长期使用。
- 备份和恢复都很慢
- 它们只有非常有限的存储容量。

然而，如果没有其它的备份数据的方法，那软盘备份总比没有备份要好。

如果必须使用软盘的话，必须确保盘片的质量。软盘在办公室中使用已经有许多
年了。最好使用一些名牌厂商的产品以确保质量。

19.10.2. 如何备份数据到软盘

最好的备份数据到软盘的方法是使用 tar(1) 程序加上 -M 选项， 它可以允许数据备份到多张软盘上。

要备份当前目录中所有的文件可以使用这个命令 (需要有 root权限)：

```
# tar Mcvf /dev/fd0 *
```

当第一张盘满的时候， tar(1) 会指示您插入下一张盘，插入第二张盘之后就按回车。

```
Prepare volume 2 for /dev/fd0 and hit return:
```

这个步骤可能需要重复很多次，直到这些文件备份完成为止。

19.10.3. 可以压缩备份吗

不幸的是，tar(1) 在为多卷文件作备份时是不允许使用 -z 选项的。当然，可以用 gzip(1) 压缩所有的文件，把它们打包到磁盘，以后在用 gunzip(1) 解开。

19.10.4. 如何恢复备份

要恢复所有文件：

```
# tar Mxvf /dev/fd0
```

有两种方法来恢复软盘中的个别文件。首先，就要用第一张软盘启动：

```
# tar Mxvf /dev/fd0 filename
```

tar(1) 程序会提示您插入后面的软盘，直到它找到所需要的文件。

如果您知道哪个文件在哪个盘上，您就可以插入那张盘，然后使用上同同样的命令。
如果软盘上的第一个文件与前面的文件是连续的，那 tar(1) 命令会警告您它无法恢复，即使您不要求它这样做。

19.11. 备份策略

设计备份计划的第一要务是确认以下问题皆已考虑到：

- 磁盘故障
- 文件的意外删除
- 随机的文件损毁
- 机器完全损毁 (例如火灾)， 包括破坏全部在线备份。

针对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够同时兼顾前面所有的需要。

可以采用的技术包括：

- 对整个系统的数据进行存档， 备份到永久性的离线介质上。
  这种方法实际上能够提供针对前面所有问题的保护， 但这样做通常很慢， 而且恢复时会比较麻烦。
  您可以将备份置于近线或在线的状态， 然而恢复文件仍然是一个难题，
  特别是对没有特权的那些用户而言。
- 文件系统快照。 这种技术实际上只对无意中删除文件这一种情况有用， 但在这种情况下它会提供 非常大的帮助， 而且访问迅速， 操作容易。
- 直接复制整个文件系统和/或磁盘 (例如周期性地对整个机器做 rsync(1))。
  通常这对于在网络上的单一需求最为适用。 要为磁盘故障提供更为通用的保护， 通常这种方法要逊于 RAID。 对于恢复无意中删除的文件来说， 这种方法基本上与 UFS 快照属于同一层次，
  使用哪一个取决于您的喜好。
- RAID。 它能够最大限度地减少磁盘故障导致的停机时间。 其代价是需要处理更为频繁的磁盘故障 (因为磁盘的数量增加了)， 尽管这类故障不再需要作为非常紧急的事项来处理。
- 检查文件的指纹。 mtree(8) 工具对于这种操作非常有用。 尽管这并不是一种备份的技术，
  但它能够确保您有机会注意到那些您需要求助于离线备份的事情。 这对于离线备份非常重要，
  而且应有计划地加以检查。

很容易列举更多的技术， 它们中有许多实际上是前面所列出的方法的变种。
特别的需求通常会需要采用特别的技术 (例如， 备份在线运行的数据库，
往往需要数据库软件提供某种方法来完成中间步骤) 来满足。 最重要的事情是，
一定要了解需要将数据保护起来免受何种风险， 以及发生问题时应该如何处理。

19.12. 备份程序

有三个主要的备份程序 dump(8)、tar(1) 和 cpio(1)。

19.12.1. Dump 和 Restore

dump 和 restore 是 UNIX® 传统的备份程序。 它以 block 而不是以文件为单位来备份数据、链接或目录。
dump 备份的是设备上的整个文件系统，
不能只备份一个文件系统的部分或是用到两个以上文件系统的目录树。 与其他备份软件不同的是， dump
不会写文件和目录到磁带机， 而是写入包含文件 和目录的原始数据块。 当需要恢复数据的时候，restore
默认在 /tmp/ 下保存临时数据 - 如果你正在操作的恢复盘只有比较小的 /tmp 的话， 你可能需要把环境变量
TMPDIR 设置到一个有更多空间的目录， 使得此过程更容易成功。

如果在您的 root 目录使用 dump， 将不需要备份 /home、/usr 或其他目录，
因为这些是典型的其他文件系统或符号连接到那些文件系统的加载点。

dump 是最早出现于 AT&T UNIX 的 Version 6 (约 1975)。 默认的参数适用于 9-track 磁带(6250 bpi)，
所以如果要用高密度的磁带（最高可达 62,182 ftpi）， 就不能用默认的参数， 而要另外指定参数。
这些默认值必须在命令行被修改以更好地利用当前磁带机的功能。

rdump 和 rrestore 可以通过网络在另一台计算机的磁带机上备份数据。 这两个程序都是依靠 rcmd(3) 和
ruserok(3) 来访问远程的磁带机。 因此，运行备份的用户必须要有远程主机的 .rhosts 访问权。 rdump 和
rrestore 的参数必须适用于远程主机 例如，当您从 FreeBSD 连到一台 SUN 工作站 knomodo
去使用磁带机时，使用：

```
# /sbin/rdump 0dsbfu 54000 13000 126 komodo:/dev/nsa8 /dev/da0a 2>&1
```

要注意的是：必须检查您在使用 .rhosts 时的安全情况。

也可以通过使用 ssh 用一个更安全的方式来使用 dump 和 restore 。

例 24. 通过 ssh 使用 dump

```
# /sbin/dump -0uan -f - /usr | gzip -2 | ssh -c blowfish \
    targetuser@targetmachine.example.com dd of=/mybigfiles/dump-usr-l0.gz
```

或使用 dump 的 built-in 方法， 设置环境变量 RSH：

例 25. 通过设置 ssh 环境变量 RSH 使用 dump

```
# RSH=/usr/bin/ssh /sbin/dump -0uan -f
targetuser@targetmachine.example.com:/dev/sa0 /usr
```

19.12.2. tar

tar(1) 也同样是在第 6 版 AT&T UNIX (大约是 1975 前后) 出现的。 tar 对文件系统直接操作；

其作用是把文件和目录写入磁带。 tar 并不支持 cpio(1) 所提供的全部功能， 但也不需要 cpio 所需要使用的诡异的命令行管道。

要 tar 到连接在名为 komodo 的 Sun 机器上的 Exabyte 磁带机， 可以使用：

```
# tar cf - . | rsh komodo dd of=tape-device obs=20b
```

如果您担心通过网络备份会有安全问题，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带机程序。它有执行字节交换的选项，可以用几种不同的格式写入，并且可以将数据用管道传给其他程序。 cpio 没办法自动查找目录树内的文件列表，必须通过标准 输入 stdin 来指定。

cpio 不支持通过网络的备份方式。可以使用 pipeline 和 rsh 来传送数据给远程的磁带机。

```
# for f in directory_list; do
find $f >> backup.list
done
# cpio -v -o --format=newc < backup.list | ssh user@host "cat > backup_device"
```

这里的 directory_list 是要备份的目录列表， user@host 结合了将要执行备份的用户名和主机名，backup_device 是写 入备份的设备（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 间有些不兼容。为了防止这种情况，并使其标准化，POSIX® 出了这套新的工具程序。 pax 尝试可以读写各种 cpio 和 tar 的格式，并可以自己增加新的格式。它的命令 集比 tar 更接近 cpio。

19.12.5. Amanda

Amanda (Advanced Maryland Network Disk Archiver) 并非单一的程序，而是一个客户机/服务器模式的备份系统 。一台 Amanda 服务器可以备份任意数量执行 Amanda 的客户机或是将连上 Amanda 服务器的计算机上的数据备份到一台磁带机上。一个常见的问题是，数据写入磁带机的时间将超过取行数据的时间，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" 的一组磁带，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 产生的网络传输。 Amanda 可以使用上述任何一个设备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。不需要预算，不需要备份的计划表，全部都不用。如果您的数据发生了什么问题， 忽略它！

如果您的时间和数据不值得您做这些事，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，您可能在几个月 内，就发现您已经收集了不少对您来说相当具有价值的文件和程序。

"Do nothing" 对于像 /usr/obj 和其他可由您的计算机产生的文件来说，是最好的方法。例如这本手册包含有 HTML 或 PostScript® 格式的文件。这些文档格式是从 SGML 输入文件创建的。创建 HTML 或 PostScript® 格式的文件的备份就没有必要了。只要经常备份 SGML 文件就够了。

19.12.7. 哪个备份程序最好?

在dump(8) 时期 Elizabeth D. Zwicky 测试了所有以上列出的备份程序。在各种各样怪异的文件系统中，dump 是您明智的选择。Elizabeth 建立起各种各样、
奇怪或常见的文件系统，并用各种备份程序，测试在各种文件系统上备份
及恢复数据。这些怪异之处包括：具有 holes 和一个 nulls block 的文件，
文件名具有有趣字符，无法读写的文件及设备，在备份时改变文件大小，在
备份时建立或删除的文件。她将结果写在： LISA V in Oct. 1991. 参阅 torture-testing Backup and Archive Programs.

19.12.8. 应急恢复程序

19.12.8.1. 在出现灾难前

在遇到灾难前，只需要执行以下四个步骤：

第一，打出您的每个磁盘驱动器的磁盘标签 (例如： bsdlabel da0 | lpr)，文件系统表， (/etc/fstab)
，以及所有启动信息， 并将其复制两份。

第二， 刻录一张 "livefs" CDROM。 这个 CDROM 包含了用于引导进入 FreeBSD "livefs" 修复模式的支持，
这种模式允许用户执行许多任务， 例如执行 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。记得将 磁盘写保护。

第四， 测试在第二步所建立的 "livefs" CDROM 及备份的磁带。 写下笔记， 并和这张 CDROM、
打印副本以及磁带放在一起。 您在需要恢复数据时可能正心慌意乱，
而这些记录可能会帮助您避免毁掉备份磁带 （怎么会发生这种情况呢？ 举例来说， 本应执行 tar xvf /dev/sa0 命令时， 您可能会不小心输入 tar cvf /dev/sa0， 从而覆盖备份磁带）。

保险起见， 您可以制作两份 "livefs" CDROM 和备份磁带。 其中一份应放到其它地方，
这里说的其他地方当然不是指同一栋办公楼的地下室， 世贸中心的一大批公司已经学到了血的教训。
保存这份备份的位置应该与您的计算机和磁盘驱动器越远越好。

19.12.8.2. 出现灾难后

关键问题是： 您的硬件是否幸免于难？ 由于已经做好了定期的备份工作， 因此并不需要担心软件的问题。

如果硬件已经损坏， 这些部分应该在尝试使用计算机之前换掉。

如果硬件还能用， 将 "livefs" CDROM 插入 CDROM 驱动器并引导系统。 您将看到最初安装系统时的菜单。
选择正确的国家之后， 选择 Fixit — Repair mode with CDROM/DVD/floppy or start a shell 选项，
然后再选择 CDROM/DVD — Use the live filesystem CDROM/DVD 这项。 您可以使用 restore
以及其他位于 /mnt2/rescue 的工具。

分别恢复每一个文件系统

试着 mount （例如： mount /dev/da0a /mnt） 第一个磁盘上的 root 分区。 如果 bsdlabel 已经毁坏，
则需要使用 bsdlabel 根据您先前打印存档的记录来重新分区并分配磁盘标签。 接着使用 newfs
重建文件系统。 以读写方式重新挂载磁盘的根分区 (mount -u -o rw /mnt)。
使用您的备份程序以及备份磁带恢复文件系统数据 （例如 restore vrf /dev/sa0）。 最后卸下文件系统
（例如 umount /mnt）。 对于毁掉的其他文件系统， 重复执行前面这些操作。

当您的系统正常启动后， 将您的数据备份到新的磁带。 任何造成数据丢失的的灾难都可能再次发生。
现在花一些时间， 也许可以在下次发生灾难时救您一把。

19.13. 网络、内存和 和以及映像文件为介质的虚拟文件系统

除了插在您计算机上的物理磁盘： 软盘、 CD、 硬盘驱动器， 等等之外， FreeBSD
还能识别一些其他的磁盘形式 - 虚拟磁盘。

这还包括， 如 网络文件系统 (Network File System) 和 Coda一类的网络文件系统、
内存以及映像文件为介质的虚拟文件系统。

随运行的 FreeBSD 版本不同，
用来创建和使用以映像文件介质文件系统和内存文件系统的工具也不尽相同。

系统会使用 devfs(5) 来创建设备节点， 这对用户来说是透明的。

19.13.1. 以映像文件为介质的文件系统

在 FreeBSD 系统中， 可以用 mdconfig(8) 程序来配置和启用内存磁盘， md(4)。 要使用 mdconfig(8)，
就需要在内核配置文件中添加 md(4) 模块来支持它：

```
device md
```

mdconfig(8) 命令支持三种类型的虚拟文件系统： 使用
malloc(9)，来分配内存文件系统，内存文件系统作为文件或作为
备用的交换分区。一种使用方式是在文件中来挂载一个软盘和 CD 镜像。

将一个暨存的映像文件作为文件系统挂载：

例 26. 使用 mdconfig 挂载已经存在的映像文件

```
# mdconfig -a -t vnode -f diskimage -u 0
# mount /dev/md0 /mnt
```

使用 mdconfig(8) 来创建新的映像文件:

例 27. 使用 mdconfig 将映像文件作为文件系统挂载

```
# 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
```

如果没有通过 -u 选项指定一个标识号 mdconfig(8) 将使用 md(4) 为它自动选择一个未用的设备标识号。
分配给它的标识名将被输出到标准输出设备， 其形式是与 md4 类似。 如果希望了解更多相关信息，
请参见联机手册 mdconfig(8)。

mdconfig(8) 功能很强大， 但在将映像文件作为文件系统挂载时， 仍需使用许多行的命令。 为此 FreeBSD
也提供了一个名为 mdmfs(8) 的工具， 该程序使用 mdconfig(8) 来配置 md(4) 设备， 并用 newfs(8)
在其上创建 UFS 文件系统， 然后用 mount(8) 来完成挂载操作。 例如，
如果想创建和挂接像上面那样的文件系统映像， 只需简单地执行下面的步骤：

例 28. 使用 mdmfs 命令配置和挂载一个映像文件为文件系统

```
# 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
```

如果你使用没有加标识号的 md 选项， mdmfs(8) 将使用 md(4) 的自动标示号特性来自动为其
选择一个未使用的设备。更详细的 mdmfs(8)，请参考联机手册。

19.13.2. 以内存为介质的文件系统

一般来说， 在建立以内存为介质的文件系统时， 应使用 "交换区作为介质 (swap backing)"。
使用交换区作为介质， 并不意味着内存盘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)
创建以内存作为介质的文件系统。 不过在内存不足时， 这种方式可能引致系统崩溃。

例 29. 用 mdconfig 创建新的内存盘设备

```
# 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. 从系统中移除内存盘设备

当不再使用内存盘设备时， 应将其资源释放回系统。 第一步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离， 以释放资源。

例如， 要分离并释放所有 /dev/md4 使用的资源， 应使用命令：

```
# mdconfig -d -u 4
```

mdconfig -l 命令可以列出关于配置 md(4) 设备的信息。

19.14. 文件系统快照

FreeBSD 提供了一个和 Soft Updates 关联的新功能: 文件系统快照

快照允许用户创建指定文件系统的映像，并把它们当做一个文件来对待。
快照文件必须在文件系统正在使用时创建，一个用户对每个文件系统创建的
快照不能大于20个。活动的快照文件被记录在超级块中，所以它们可以在系统
启动的时候一块进行挂接后摘掉。当一个快照不再需要时，可以使用标准的 rm(1)
使用来使其删除。快照可以以任何顺序进行移除，但所有使用
的快照不可能同时进行移除，因为其它的快照将有可能互相引用一些块。

不可改的 snapshot 文件标志， 是由 mksnap_ffs(8) 在完成创建快照文件时设置的。 unlink(1) 命令是一个特例， 以允许删除快照文件。

快照可以通过 mount(8) 命令创建。 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
```

作为选择，你也可以使用 mksnap_ffs(8) 来创建一个快照：

```
# mksnap_ffs /var /var/snapshot/snap
```

可以查找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
```

当快照文件被创建好后，可以用于下面一些目的：

- 有些管理员用文件快照来进行备份， 因为快照可以被转移到 CD 或磁带上。

- 文件系统一致性检查程序 fsck(8) 可以用来检查快照文件。 如果文件系统在挂接前是一致的，则检查结果也一定是一致的 (也就是不会做任何修改)。 实际上这也正是后台 fsck(8) 的操作过程。
- 在快照上运行 dump(8) 程序。 dump 将返回包含文件系统和快照的时间戳。dump(8) 也能够抓取快照，使用 -L 标志可以首先创建快照， 完成 dump 映像之后再自动删除它。
- 用 mount(8) 来挂接快照作为文件系统的一个冻结的镜像。 要 mount(8) 快照 /var/snapshot/snap 运行：

```
# mdconfig -a -t vnode -f /var/snapshot/snap -u 4
# mount -r /dev/md4 /mnt
```

现在你就可以看到挂接在 /mnt 目录下的 /var 文件系统的快照。
每一样东西都保存的像它创建时的状态一样。 唯一例外的是更早的快照文件将表现为长度为 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
```

想了解更多关于 softupdates 和 文件系统快照的信息， 包括技术说明， 可以访问 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的功能， 它允许管理员以文件系统为单元，
限制分派给用户或组成员所使用的磁盘空间大小或是使用的总文件数量。
这经常被用于那些分时操作的系统上， 对于这些系统而言，
通常希望限制分派到每一个用户或组的资源总量， 从而可以防止某个用户占用所有可用的磁盘空间。

19.15.1. 配置系统来启用磁盘配额

在决定使用磁盘配额前，确信磁盘配额已经在内核中配置好了。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 这章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 通过添加下面这行来完成：

```
enable_quotas="YES"
```

为了更好的控制配额时的启动，还有另外一个可配置的变量。通常
启动时，集成在每个文件系统上的配额会被配额检查程序 quotacheck(8)
自动检查。配额检查功能能够确保在配额数据库中 的数据正确地反映了文件系统的数
据情况。这是一个很耗时间的处理进程，它会影响系统的启动时间。如果 想跳过这一步，可以在文件
/etc/rc.conf 加入 下面这一行来达到目的：

```
check_quotas="NO"
```

最后，要编辑 /etc/fstab 文件，以在每一个

文件系统基础上启用磁盘配额。这是启用用户和组配额，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每个用户的配额，可以在 /etc/fstab 里添加 userquota
选项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 选项来代替 userquota
选项。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。可以查看 fstab(5) 联机手册了解更多信息。 尽管联机手册 fstab(5) 提到， 可以为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，可以用新内核重新启动系统。 /etc/rc 将自动
运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长度的配额 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，然而可能需要阅读与他们的操作 相似的联机手册。

19.15.2. 设置配额限制

一旦您配置好了启用配额的系统，可以检查一下它们是真的有用。 可以这样做：

```
# quota -v
```

您应该能够看到一行当前正在使用的每个文件系统启用的磁盘配额 使用情况的摘要信息。

现在可以使用 edquota(8) 命令准备启用配额限制。

有几个有关如何强制限制用户或组可以分配到的磁盘空间大小的选项。 您可以限制磁盘存储块的配额，
或文件的数量， 甚至同时限制两者。 这些限制最终可分为两类： 硬限制和软限制。

硬性限制是一种不能越过的限制。 一旦用户达到了系统指定的硬性限制，
他就无法在对应的文件系统分配到更多的资源。 例如， 如果文件系统上分给用户的硬性限制是 500 KB，
而现在已经用掉了 490 KB， 那么这个用户最多还能再分配 10 KB 的空间。 换言之， 如果这时试图再分配
11 KB， 则会失败。

而与此相反， 软性限制在一段时间内是允许越过的。 这段时间也称为宽限期， 其默认值是一周。
如果一个用户延缓时间太长的话，软限制将会变成硬限制， 而继续分配磁盘空间的操作将被拒绝。
当用户占用的空间回到软性限制值以下时， 宽限期将重新开始计算。

下面是一个运行 edquota(8) 时看到的例子。当 edquota(8) 命令被调用时，会被转移进 EDITOR
环境变量指派的编辑 器中，允许编辑配额限制。如果环境变量没有设置，默认在 vi 编辑器上进行。

```
# 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)
```

在每一个启用了磁盘配额的文件系统上，通常会看到两行。一行是 block 限制，另一行是 inode 限制。简单地改变要修改的配额限制的值。 例如，提高这个用户软限制的数值到 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)
```

当离开编辑器的时候，新的配额限制设置将会被保存。

有时，在 UIDs 的范围上设置配额限制是非常必要的。这可以通过在 edquota(8) 命令后面加上 -p 选项来完成。首先， 给用户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 用户 test 已经有了所需要的配额限制，下面的命令 可以被用来复制那些 UIDs 为10,000 到 19,999 的配额限制：

```
# edquota -p test 10000-19999
```

更多细节请参考 edquota(8) 联机手册。

19.15.3. 检查配额限制和磁盘使用

既可以使用 quota(1) 也可以使用 repquota(8) 命令来检查 配额限制和磁盘使用情况。 quota(1) 命令能够检查单个用户和组的配置 使用情况。只有超级用户才可以检查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 作为例子。 此用户目前已经比软限制 50 KB 超出了 15 KB， 还剩下 5 天的宽限期。 请注意，星号 * 说明用户已经超出了其配额限制。

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，即使已经为那个用户指定了配额。 而使用 -v 选项则会显示它们， 例如前面例子中的 /usr/var。

19.15.4. 通过 NFS 使用磁盘配额

配额能够在 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们的 quota 统计信息。

可以这样在 /etc/inetd.conf 中启用 rpc.rquotad：

```
rquotad/1    dgram rpc/udp wait root /usr/libexec/rpc.rquotad rpc.rquotad
```

现在重启 inetd：

```
# /etc/rc.d/inetd restart
```

19.16. 加密磁盘分区

FreeBSD 提供了极好的数据保护措施，防止未受权的数据访问。 文件权限和强制访问控制(MAC)(看强制访问控制) 可以帮助预防在操作系统处于运行状态和计算机加电时未受权的第三方访问数据。
但是，和操作系统强制受权不相关的是，如果黑客有物理上访问计算机的可能，
那他就可以简单的把计算机的硬件安装到另一个系统上复制出敏感的数据。

无论攻击者如何取得停机后的硬件或硬盘驱动器本身， FreeBSD GEOM Based Disk Encryption (基于 GEOM 的磁盘加密， gbde) 和 geli 加密子系统都能够保护计算机上的文件系统数据，
使它们免受哪怕是训练有素的攻击者获得有用的资源。 与那些只能加密单个文件的笨重的加密方法不同，
gbde 和 geli 能够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对 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 所进行的介绍重新编译并安装内核。

   重新引导进入新的内核。

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. 准备加密盘

下面这个例子假设您添加了一个新的硬盘在您的系统并将拥有一个单独的加密分区。 这个分区将挂接在 /private目录下。 gbde 也可以用来加密 /home 和 /var/mail， 但是这需要更多的复杂命令来执行。

1. 添加新的硬盘

   添加新的硬盘到系统中可以查看在 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经添加到系统中如： /dev/ad4s1c。在例子中

/dev/ad0s1* 设备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. 创建一个目录来保存 gbde Lock 文件

```
# mkdir /etc/gbde
```

gbde lock 文件包含了 gbde 需要访问的加密分区的信息。 没有 lock 文件， gbde 将不能解密包含在加密分区上的数据。 每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， 这个初始化过程只需要执行一次：

```
# gbde init /dev/ad4s1c -i -L /etc/gbde/ad4s1c.lock
```

gbde(8) 将打开您的编辑器， 提示您去设置在一个模板文件中的配置变量。 使用 UFS1 或 UFS2，设置扇区大小为 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) 将让您输入两次用来加密数据的密钥短语。 两次输入的密钥必须相同。 gbde 保护您数据的能力依靠您选择输入的密钥的质量。

gbde init 命令为您的 gbde 分区创建了一个 lock 文件， 在这个例子中存储在 /etc/gbde/ad4s1c.lock中。 gbde lock 文件必须使用 ".lock" 扩展名才能够被 /etc/rc.d/gbde 启动脚本正确识别。

gbde lock 文件 必须 和加密分区上的内容同时备份。 如果发生只有 lock 文件遭到删除的情况时， 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， 不经过大量细致的工作也无法访问加密分区上的数据， 而这是在设计 gbde(8) 时完全没有考虑过的。

4. 把加密分区和内核进行关联

```
# gbde attach /dev/ad4s1c -l /etc/gbde/ad4s1c.lock
```

在加密分区的初始化过程中您将被要求提供一个密码短语。 新的加密设备将在 /dev 中显示为 /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. 在加密设备上创建文件系统

   当加密设备和内核进行关联后， 您就可以使用 newfs(8) 在此设备上创建文件系统， 使用 newfs(8) 来初始化一个 UFS2 文件系统比初始化一个 UFS1 文件系统还要快，推荐使用 -O2 选项。

   ```
   # newfs -U -O2 /dev/ad4s1c.bde
   ```

   newfs(8) 命令必须在一个 gbde 分区上执行， 这个分区通过一个存在的 *.bde 设备名进行标识。

6. 挂接加密分区

   为加密文件系统创建一个挂接点。

   ```
   # mkdir /private
   ```

   挂接加密文件系统。

   ```
   # mount /dev/ad4s1c.bde /private
   ```

7. 校验加密文件系统是否有效

   加密的文件系统现在对于 df(1) 应该可见并可以使用。

   ```
   % 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

每次系统启动后， 在使用加密文件系统前必须和内核重新进行关联，
校验错误和再次挂接。使用的命令必须由 root用户来执行。

1. 关联 gbde 分区到内核

   ```
   # gbde attach /dev/ad4s1c -l /etc/gbde/ad4s1c.lock
   ```

   接下来系统将提示您输入在初始化加密的 gbde 分区时所用的密码短语。

2. 校验文件系统错误

   加密文件系统不能列在 /etc/fstab 文件中进行自动加载， 在加载前必须手动运行 fsck(8)
   命令对文件系统进行错误检测。

   ```
   # fsck -p -t ffs /dev/ad4s1c.bde
   ```

3. 挂接加密文件系统

   ```
   # mount /dev/ad4s1c.bde /private
   ```

   加密后的文件系统现在可以有效使用。

19.16.1.2.1. 自动挂接加密分区

可以创建脚本来自动地附加、 检测， 并挂接加密分区， 然而， 处于安全考虑， 这个脚本不应包含 gbde(8)
密码。 因而， 我们建议这类脚本在控制台或通过 ssh(1) 执行并要求用户输入口令。

除此之外， 系统还提供了一个 rc.d 脚本。 这个脚本的参数可以通过 rc.conf(5) 来指定， 例如：

```
gbde_autoattach_all="YES"
gbde_devices="ad4s1c"
gbde_lockdir="/etc/gbde"
```

在启动时将要求输入 gbde 的口令。 在输入正确的口令之后， gbde 加密分区将被自动挂接。 对于将 gbde
用在笔记本电脑上时， 这就很有用了。

19.16.1.3. gbde 提供的密码学保护

gbde(8) 采用 CBC 模式的 128-位 AES 来加密扇区数据。 磁盘上的每个扇区都采用不同的 AES 密钥来加密。
要了解关于 gbde 的密码学设计， 包括扇区密钥如何从用户提供的口令字中生成等细节， 请参考 gbde(4)。

19.16.1.4. 兼容性问题

sysinstall(8) 是和 gbde 加密设备不兼容的。 在启动 sysinstall(8) 时必须将 *.bde
设备和内核进行分离，否则在初始化探测设备时将引起冲突。
与加密设备进行分离在我们的例子中使用如下的命令：

```
# gbde detach /dev/ad4s1c
```

还需要注意的是， 由于 vinum(4) 没有使用 geom(4) 子系统， 因此不能同时使用 gbde 与 vinum 卷。

19.16.2. 使用 geli 对磁盘进行加密

还有另一个可用于加密的 GEOM class - geli。 它目前由 Paweł Jakub Dawidek <pjd@FreeBSD.org> 开发。 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进行密码学运算。

geli(8) 最重要的功能包括：

- 使用了 crypto(9) 框架 - 如果系统中有加解密硬件加速设备， 则 geli 会自动加以利用。
- 支持多种加密算法 (目前支持 AES、 Blowfish， 以及 3DES)。
- 允许对根分区进行加密。 在系统启动时， 将要求输入用于加密根分区的口令。
- 允许使用两个不同的密钥 (例如， 一个 "个人密钥" 和一个 "公司密钥")。
- geli 速度很快 - 它只进行简单的扇区到扇区的加密。
- 允许备份和恢复主密钥。 当用户必须销毁其密钥时， 仍然可以通过从备份中恢复密钥来存取数据。
- 允许使用随机的一次性密钥来挂接磁盘 - 这对于交换区和临时文件系统非常有用。

更多 geli 功能介绍可以在 geli(8) 联机手册中找到。

下面的步骤介绍了如何启用 FreeBSD 内核中的 geli 支持， 并解释了如何创建新和使用 geli 加密 provider。

由于需要修改内核， 您需要拥有超级用户权限。

1. 在内核中加入 geli 支持

   在内核配置文件中加入下面两行：

   ```
   options GEOM_ELI
   device crypto
   ```

   按照 配置FreeBSD的内核 介绍的步骤重新编译并安装内核。

   另外， geli 也可以在系统引导时加载。 这是通过在 /boot/loader.conf 中增加下面的配置来实现的：

   ```
   geom_eli_load="YES"
   ```

   geli(8) 现在应该已经为内核所支持了。

2. 生成主密钥

   下面的例子讲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。 这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 的扇区尺寸为 4kB。 此外， 这里的讨论将介绍如何挂载 geli provider， 在其上创建文件系统， 如何挂接并在其上工作， 最后将其卸下。

   建议您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则将是 /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须的； 您也可以只使用其中的一种来加密主密钥。

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 与所生成的密钥关联

```
# geli attach -k /root/da2.key /dev/da2
Enter passphrase:
```

新的明文设备将被命名为 /dev/da2.eli。

```
# ls /dev/da2*
/dev/da2  /dev/da2.eli
```

4. 创建新的文件系统

```
# dd if=/dev/random of=/dev/da2.eli bs=1m
# newfs /dev/da2.eli
# mount /dev/da2.eli /private
```

现在加密的文件系统应该已经可以被 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. 卸下卷并断开 provider

一旦在加密分区上的工作完成， 并且不再需要 /private 分区， 就应考虑将其卸下并将 geli 加密分区从内核上断开。

```
# umount /private
# geli detach da2.eli
```

关于如何使用 geli(8) 的更多信息， 可以在其联机手册中找到。

19.16.2.1. 使用 geli rc.d 脚本

geli 提供了一个 rc.d 脚本， 它可以用于简化 geli 的使用。 通过 rc.conf(5) 配置 geli 的方法如下：

```
geli_devices="da2"
geli_da2_flags="-p -k /root/da2.key"
```

这将把 /dev/da2 配置为一个 geli provider， 其主密钥文件位于 /root/da2.key， 而 geli 在连接 provider 时将不使用口令字 (注意只有在 geli init 阶段使用了 -P 才可以这样做)。 系统将在关闭之前将 geli provider 断开。

关于如何配置 rc.d 的详细信息可以在使用手册的 rc.d 一节中找到。

19.17. 对交换区进行加密

FreeBSD 提供了易于配置的交换区加密机制。 随所用的 FreeBSD 版本， 可用的配置选项会有所不同， 而配置方法也会有一些差异。 可以使用 gbde(8) 和 geli(8) 两种加密系统来进行交换区的加密操作。 前面所说的这两种加密系统， 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简单的讨论。

19.17.1. 为什么需要对交换区进行加密?

与加密磁盘分区类似， 加密交换区有助于保护敏感信息。 为此，
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， 则一切相安无事。 然而，
如果操作系统开始将内存页换出到交换区， 以便为其他应用程序腾出内存时，
这些口令就可能以未加密的形式写到磁盘上， 并为攻击者所轻易获得。
加密交换区能够有效地解决这类问题。

19.17.2. 准备

在本节余下的部分中， 我们约定使用 ad0s1b 作为交换区。

到目前为止， 交换区仍是未加密的。 很可能其中已经存有明文形式的口令或其他敏感数据。
要纠正这一问题， 首先应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 中与交换区对应的行中， 设备名应追加 .bde 后缀：

```
# 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

另一种方法是使用 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 中交换区对应的行中， 设备名应追加 .eli 后缀：

```
# 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为 256-位的 AES 加密算法。

当然， 这些默认值是可以通过 /etc/rc.conf 中的 geli_swap_flags 选项来修改的。 下面的配置表示让 rc.d 脚本 encswap 创建一个 geli(8) 交换区， 在其上使用密钥长度为 128-位 的 Blowfish 加密算法， 4 kilobytes 的扇区尺寸， 并采用 "最后一次关闭时卸下" 的策略：

```
geli_swap_flags="-e blowfish -l 128 -s 4096 -d"
```

请参见 geli(8) 联机手册中关于 onetime 命令的说明， 以了解其他可用的选项。

19.17.5. 验证所作的配置能够发挥作用

在重启系统之后， 就可以使用 swapinfo 命令来验证加密交换区是否已经在正常运转了。

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

高可用性是担负关键业务的应用的一项主要需求， 而高可用存储则是这类环境中的一项关键组件。
高可用存储 Highly Available STorage， 或 HAST， 是由 Paweł Jakub Dawidek <pjd@FreeBSD.org> 开发的一种用于提供在两台物理上隔离的系统之间以透明的方式， 通过 TCP/IP 网络传输数据的高可用性框架。 HAST 可以看作通过网络进行的 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 可以用来构建可以抗御硬件故障的高可用存储集群。

读完这节， 您将了解：

- 何为 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。

在阅读这节之前， 您应：

- 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 (设置和调整)。
- 理解 FreeBSD 的网络功能 (网络通讯)。
- 使用 FreeBSD 8.1-RELEASE 或更新版本。

HAST 项目是由 FreeBSD 基金会资助完成的， 并得到了来自 OMCnet Internet Service GmbH 和 TransIP BV 的支持。

19.18.2. HAST 的功能

HAST 系统提供的功能主要包括：

- 可以掩盖本地硬盘的 I/O 错误。
- 文件系统无关， 因而可以配合 FreeBSD 支持的任何文件系统使用。
- 高效率的快速重新同步机制， 令系统只同步在另一节点停机时修改过的块。
- 可以在已经部署好的环境中添加冗余。
- 配合 CARP、 Heartbeat 或其他类似的工具， 可以实现健壮的可靠存储系统。

19.18.3. HAST 的运行机制

由于 HAST 本质上是在多个机器间同步地进行块级复制， 因此它需要至少两个节点 (物理的机器) - 其一作为 主 (也称作 master) 节点， 另一个作为 从 (slave) 节点。 这两台机器会共同构成一个集群。

目前 HAST 只能使用最多两个集群节点。

由于 HAST 是配置成以主从节点的方式运行， 在任何时刻都只能有唯一的一个节点是主节点。 主 节点， 也称作 活跃 节点， 负责处理由 HAST 管理的设备的全部 I/O 请求。 而 从 节点则会自动从 主 节点同步数据的变更操作。

在 HAST 系统中的物理设备包括：

- 本地磁盘 (在主节点上)
- 远程磁盘 (在从节点上)

HAST 在块的级别上同步运行， 这使其对文件系统和应用程序透明。 HAST 在 /dev/hast/ 目录中提供标准的 GEOM 设备供其他工具或应用程序使用， 因此， 在使用上， 对应用程序或文件系统而言， HAST 提供的设备与普通的裸盘或分区等没有任何区别。

发到本地磁盘的每次写、 删除或缓存刷写操作， 都会同时通过 TCP/IP 发到远程磁盘上。 读操作是由本地磁盘完成， 除非本地磁盘上的数据不是最新的， 或发生了 I/O 错误。 在这种情况下， 读操作会在从节点上完成。

19.18.3.1. 同步及复制模式

HAST 希望提供快速的故障恢复能力。 基于这一考量， 减少在某个节点停机后需要的同步时间就十分重要。 为了提供快速的同步能力， HAST 会维护一份保存在磁盘上的脏区段位映射表 (bitmap of dirty extents)， 在普通的同步模式中， 它只同步这些部分的数据 (初始的同步除外)。

处理同步有多种不同的方式， HAST 计划实现以下几种同步方式：

- memsync： 当本地的写操作已经完成， 并且远程节点汇报已经收到数据时， 便认为数据的写操作已经完成， 而不是等待远程节点完成数据的写操作。 远程节点在发出回应之后， 会立即开始执行写操作。 这种模式的目标是减少响应时间， 但在同时仍然保持很好的可靠性。 目前 memsync 复制模式尚未实现。
- fullsync： 只有在本地写操作完成， 并且远程的写操作也已经完成的情况下， 才认为数据的写操作已经完成。 这种模式是最保险， 同时也是最慢的一种复制模式。 这是目前系统预设的复制模式。

- async： 在本地写操作完成时， 即认为数据已经写完。 这是最快， 同时也是风险最大的复制模式， 一般而言只有在另一节点的延迟较大时才应考虑使用。 目前 async 复制模式尚未实现。

目前， 只支持 fullsync 复制模式。

19.18.4. HAST 的配置

HAST 需要 GEOM_GATE 支持才能正常工作。 系统自带的预设 GENERIC 内核 并不 包含 GEOM_GATE， 但默认的 FreeBSD 安装包含了 geom_gate.ko 内核模块。 如果对系统进行了裁剪， 则应确认这个模块是否可用。 此外， GEOM_GATE 也可以静态联编进内核， 方法是在内核的编译配置中添加下面的设置：

```
options GEOM_GATE
```

从操作系统的角度， HAST 框架包含了下面这些部件：

- 负责进行数据同步的 hastd(8) 服务程序，
- 用于执行管理操作的 hastctl(8) 用户态管理工具，
- 配置文件 hast.conf(5)。

下面的例子将介绍使用 HAST 在两个节点之间以 主-从 模式复制数据的方法。 两个节点的名字分别是 hasta 其 IP， 地址为 172.16.0.1， 以及 hastb， 其 IP 地址为 172.16.0.2。 这两台机器都使用尺寸相同的磁盘 /dev/ad6 来专用于 HAST 的运行。 HAST 存储池 (有时也称为资源， 例如位于 /dev/hast/ 的设备文件) 将命名为 test。

HAST 的配置文件是 /etc/hast.conf。 在两个节点上， 这个文件的内容应该是完全一样的。 最简配置如下：

```
resource test {
  on hasta {
    local /dev/ad6
    remote 172.16.0.2
  }
  on hastb {
    local /dev/ad6
    remote 172.16.0.1
  }
}
```

如果需要更高级的配置， 请参阅联机手册 hast.conf(5)。

在 remote 语句中也可以使用主机名。 这种情况下需要确保这些主机名是可以解析的， 例如在 /etc/hosts 文件中， 或在本地 DNS 中进行了定义。

现在在两个节点上都有同样的配置了， 接下来我们需要创建 HAST 存储池。 在两个节点上分别运行下面的命令来初始化本地此怕， 并启动 hastd(8) 服务：

```
# hastctl create test
# /etc/rc.d/hastd onestart
```

没有 办法使用已经包含文件系统的 GEOM 设备来创建存储池 (换言之，
已经存在的文件系统无法转换为 HAST 管理的存储池)，
这是因为创建存储池的过程需要保存一些元数据，
而已经写入文件系统的设备不再能提供保存这些元数据所需的空间。

HAST 并不负责选择节点的角色 (主 或 从)。 节点的角色是由管理员手工， 或由类似 Heartbeat
这样的软件通过 hastctl(8) 来完成配置的。 在希望成为主节点的系统 (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讯， 但又都配置为主节点这样的情况； 这种称作
脑分裂 的状态是十分危险的。 在 从脑分裂状态恢复
中介绍了如何从这种状态中恢复的方法。

接下来， 可以在两个节点上分别用 hastctl(8) 工具来验证节点身份是否正确：

```
# hastctl status test
```

这其中比较重要的是 status(状态) 这行， 在两个节点上， 其输出均应为 complete(完好)。
如果系统给出的输出是 degraded (降级)， 则表示出现了问题。 正常情况下， 节点间的同步已经开始。 当
hastctl status 命令报告的 dirty 数据块数量为 0 字节时， 表示两个节点的数据已经完全同步。

最后一步是在 GEOM 设备 /dev/hast/test 上创建文件系统。 这项工作必须在 主 节点上进行 (因为
/dev/hast/test 只在 主 节点上出现)， 随硬盘尺寸的不同， 这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， 最后一步就是确保 HAST 在系统引导过程中会自动启动了。
为了达到这个目的， 应在 /etc/rc.conf 文件中添加这行配置：

```
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的存储系统， 令其能够抵御在任何一个节点上发生的故障。
这其中的关键任务是对集群中的 主 节点发生故障的情形进行及时的补救处理。 当发生这种情况时， 从
节点可以无缝地接手主节点的工作， 对文件系统进行检查并挂接， 从而继续运行， 而不损失任何数据。

为了达成这一任务， 需要使用 FreeBSD 提供的另一项功能 - CARP 所提供的 IP 层自动故障转移能力。
CARP 是共用地址冗余协议 Common Address Redundancy Protocol 的缩写，
它允许多个同网段的主机共享同一 IP 地址。 请根据 Common Address Redundancy Protocol (CARP，
共用地址冗余协议 的介绍在两个节点上都配置 CARP。 完成这些配置之后， 两个节点都会有自己的 carp0
网络接口， 共用 IP 地址 172.16.0.254。 显然， 集群中的 HAST 主节点也必须是 CARP 主节点。

前面一节中创建的 HAST 存储池现在可以提供给网络上的其他主机使用了。 其上的文件系统可以通过 NFS、Samba 等等， 以共用 IP 地址 172.16.0.254 来访问。 现在余下的唯一问题是自动化对主节点故障的处理。

当 CARP 网络接口的链路状态发生变化时， FreeBSD 操作系统会产生一个 devd(8) 消息， 这样就可以监视 CARP 网络接口的状态了。 CARP 接口的状态变化表示节点发生故障， 或重新回到了网络中。
这些情况下需要运行特定的脚本来完成对应的处理。

为了截获 CARP 网络接口的状态变化， 需要在两个节点的 /etc/devd.conf 文件中添加如下的设置：

```
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";
};
```

为使编辑的配置生效， 需要在两个节点上执行下面的命令：

```
# /etc/rc.d/devd restart
```

当网络接口 carp0 的状态发生变化时， 系统会产生一个通知消息， 这允许 devd(8)
子系统运行管理员指定的任意脚本， 在这个例子中是 /usr/local/sbin/carp-hast-switch。
这个脚本的作用是自动化故障转移。 关于前面 devd(8) 配置的具体含义， 请参阅联机手册 devd.conf(5)。

下面是一个这种脚本的示例：

```
#!/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
```

简而言之， 在节点成为网络的 master / primary 节点时， 脚本会进行下面的操作：

- 在本节点升格为 HAST 存储池的主节点。
- 检查 HAST 存储池上的文件系统。
- 挂接存储池中的文件系统到适当的位置。

当节点成为 backup / secondary 节点时：

- 卸下 HAST 存储池。
- 将本节点降格为 HAST 存储池的从节点。

务必注意， 上面的脚本只是概念性的介绍。 它并不能处理所有可能发生的情况，
因此应根据实际情况进行修改， 例如启动/停止必要的服务， 等等。

在前面的例子中， 出于示范的目的我们使用的是标准的 UFS 文件系统。
为了减少恢复所需的时间， 可以使用带日志的 UFS 文件系统， 或者使用 ZFS 文件系统。

更具体的信息和例子请参阅 HAST Wiki 页面。

19.18.5. 故障排除

19.18.5.1. 一般故障排除提示

HAST 通常都能够无故障地运行， 不过， 和任何其他软件产品一样，
有时它也可能会无法以希望的方式运转。 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时间是同步的。

当尝试排除 HAST 故障时， 应提高 hastd(8) 的调试级别。 这可以通过在启动 hastd(8) 服务时指定 -d 参数来实现。 需要说明的是， 可以多次指定这一参数来进一步提高调试级别。 此外， 还可以考虑使用 -F 参数来启动服务， 它会令 hastd(8) 服务在前台运行。

19.18.5.2. 从脑分裂状态恢复

当集群中的两个节点之间无法相互通讯时， 两个节点都会认为自己是主节点， 从而导致 脑分裂 的状态。
这种情形十分危险， 因为两个节点会产生互相无法合并的数据。 这种情形需要系统管理员实施手工干预。

从这种状态中恢复时， 管理员必须决定哪一个节点包含最重要的数据变动 (或者手工合并这些改动) 并让 HAST 进行一次完整的同步操作， 覆盖有问题的那个节点的数据。
要完成这个工作，在有问题的节点上执行下面的命令：

```
# hastctl role init <resource>
# hastctl create <resource>
# hastctl role secondary <resource>
```

Chapter 20. GEOM: 模块化磁盘变换框架

20.1. 概述

本章将介绍以 FreeBSD GEOM 框架来使用磁盘。 这包括了使用这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 如何处理或控制 I/O、 其下层的子系统或代码。 您可以从 geom(4) 联机手册及其众多 SEE ALSO 参考文献中得到这些信息。 这一章也不是对 RAID 配置的权威介绍， 它只介绍由 支持GEOM 的 RAID 级别。

读完这章， 您将了解：

- 通过 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 级别。
- 如何通过 GEOM 使用镜像、 条带、 加密和挂接在远程的磁盘设备。
- 如何排除挂接在 GEOM 框架上的磁盘设备的问题。

阅读这章之前， 您应：

- 理解 FreeBSD 如何处理磁盘设备 (存储)。
- 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

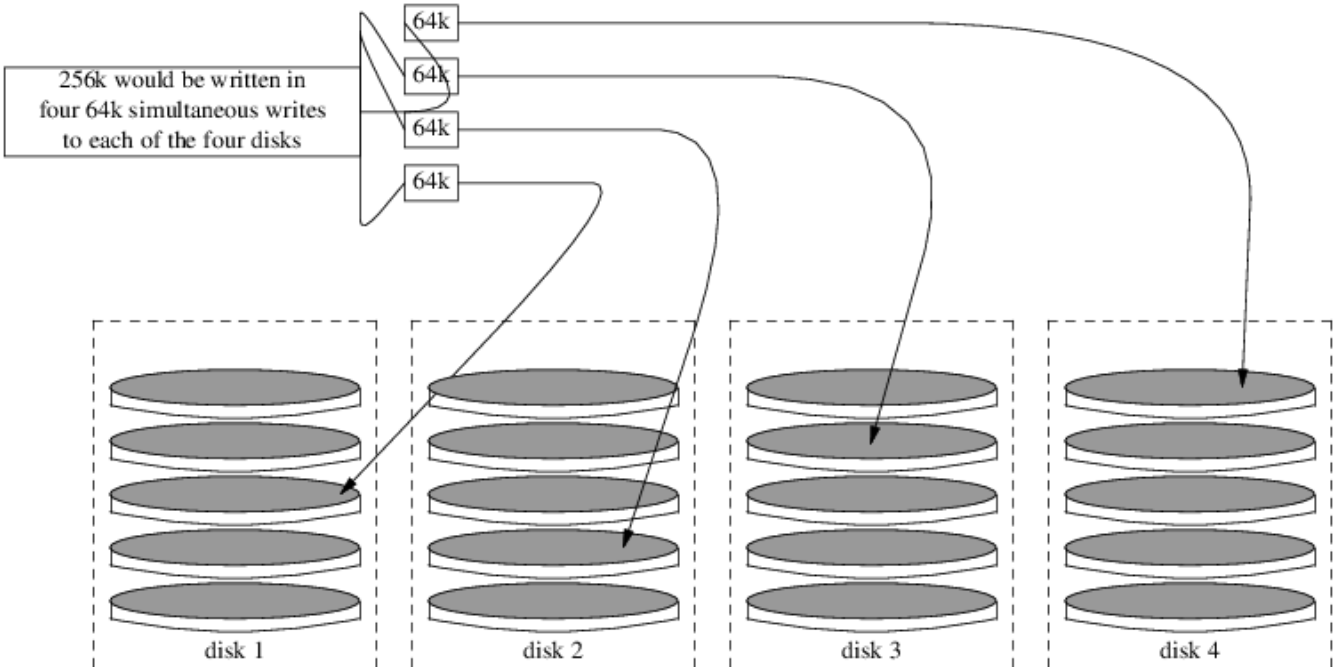
GEOM 允许访问和控制类 (classes) - 主引导记录、 BSD 标签 (label)， 等等 - 通过使用 provider， 或在 /dev 中的特殊文件。 它支持许多软件 RAID 配置， GEOM 能够向操作系统， 以及在其上运行的工具提供透明的访问方式。

20.3. RAID0 - 条带

条带是一种将多个磁盘驱动器合并为一个卷的方法。 许多情况下， 这是通过硬件控制器来完成的。 GEOM 磁盘子系统提供了 RAID0 的软件支持， 它也成为磁盘条带。

在 RAID0 系统中， 数据被分为多个块， 这些块将分别写入阵列的所有磁盘。 与先前需要等待系统将 256k 数据写到一块磁盘上不同， RAID0 系统， 能够同时分别将打碎的 64k 写到四块磁盘上， 从而提供更好的 I/O 性能。 这一性能提升还能够通过使用多个磁盘控制器来进一步改进。

在 RAID0 条带中的每一个盘的尺寸必须一样， 因为 I/O 请求是分散到多个盘上的， 以便让这些盘上的读写并行完成。



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. 确信存在合适的挂接点 (mount point)。 如果这个卷将成为根分区，
那么暂时把它挂接到其他位置i， 如 /mnt：

```
# mkdir /mnt
```

3. 确定将被做成条带卷的磁盘的设备名， 并创建新的条带设备。 举例而言， 要将两个未用的、
尚未分区的 ATA 磁盘 /dev/ad2 和 /dev/ad3 做成一个条带设备：

```
# gstripe label -v st0 /dev/ad2 /dev/ad3
Metadata value stored on /dev/ad2.
Metadata value stored on /dev/ad3.
Done.
```

4. 接着需要写标准的 label， 也就是通常所说的分区表到新卷上， 并安装标准的引导代码：

```
# bsdlabel -wB /dev/stripe/st0
```

5. 上述过程将在 /dev/stripe 目录中的 st0 设备基础上建立两个新设备。 这包括 st0a 和 st0c。
这时， 就可以在 st0a 设备上用下述 newfs 命令来建立文件系统了：

```
# newfs -U /dev/stripe/st0a
```

在屏幕上将滚过一些数字， 整个操作应该能在数秒内完成。 现在可以挂接刚刚做好的卷了。

要挂接刚创建的条带盘：

```
# mount /dev/stripe/st0a /mnt
```

要在启动过程中自动挂接这个条带上的文件系统， 需要把关于卷的信息放到 /etc/fstab 文件中。为达到此目的， 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 模块也必须通过在 /boot/loader.conf 中增加下述设置，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须中断的备份技术。 简单地说， 镜像的概念就是 磁盘B 是同步复制 (replicate) 的 磁盘A 的副本， 或者 磁盘C+D 是 diskA+B 的同步复制副本， 等等。 无论磁盘配置如何， 这种技术的共同特点都是一块磁盘或分区的内容会同步复制到另外的地方。 这样，除了能够很容易地恢复信息之外， 还能够在无须中断服务或访问的情况下进行备份，甚至直接将副本送到数据保安公司异地储存。

在开始做这件事之前， 首先请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. 对主磁盘进行镜像

假定您现有系统中的 FreeBSD 安装到了第一个， 也就是 da0 盘上， 则应告诉 gmirror(8) 将主要数据保存在这里。

在开始构建镜像卷之前， 可以启用更多的调试信息， 并应开放对设备的完全访问。 这可以通过将 sysctl(8) 变量 kern.geom.debugflags 设置为下面的值来实现：

```
# sysctl kern.geom.debugflags=17
```

接下来需要创建镜像。 这个过程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创建 /dev/mirror/gm 设备：

在引导用的设备基础上新建镜像时，有可能会导致保存在磁盘上最后一个扇区的数据丢失。 在新安装 FreeBSD 之后立即创建镜像可以减低此风险。 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， 这会导致数据丢失，并有可能导致系统无法引导。

```
# gmirror label -vb round-robin gm0 /dev/da0
```

系统应给出下面的回应：

```
Metadata value stored on /dev/da0.
Done.
```

初始化 GEOM， 这步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 目录中创建设备节点 gm0。

配置在系统初始化过程中自动加载 geom_mirror.ko：

```
# echo 'geom_mirror_load="YES"' >> /boot/loader.conf
```

编辑 /etc/fstab 文件， 将其中先前的 da0 改为新的镜像设备 gm0。

如果 vi(1) 是你喜欢的编辑器， 以下则是完成此项任务的一个简便方法：

```
# vi /etc/fstab
```

在 vi(1) 中备份现有的 fstab 内容， 具体操作是 :w /etc/fstab.bak。 接着， 把所有旧的 da0 替换成 gm0， 也就是输入命令 :%s/da/mirror\/gm/g。

修改完后的 fstab 文件应该是下面的样子。 磁盘驱动器是 SCSI 或 ATA 甚至 RAID 都没有关系，最终的结果都是 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
```

重启系统：

```
# shutdown -r now
```

在系统初始化过程中， 新建的 gm0 会代替 da0 设备工作。 系统完成初始化之后， 可以通过检查 mount 命令的输出来查看效果：

```
# 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
```

这个输出是正常的。 最后， 使用下面的命令将 da1 磁盘加到镜像卷中， 以开始同步过程：

```
# gmirror insert gm0 /dev/da1
```

在构建镜像卷的过程中， 可以用下面的命令查看状态：

```
# gmirror status
```

一旦镜像卷的构建操作完成， 这个命令的输出就会变成这样：

```
   Name   Status  Components
mirror/gm0  COMPLETE  da0
            da1
```

如果有问题或者构建仍在进行， 输出中的 COMPLETE 就会是 DEGRADED。

20.4.2. 故障排除

20.4.2.1. 系统拒绝引导

如果系统引导时出现类似下面的提示：

```
ffs_mountroot: can't find rootvp
Root mount failed: 6
mountroot>
```

这种情况应使用电源或复位按钮重启机器。 在引导菜单中， 选择第六 (6) 个选项。 这将让系统进入 loader(8) 提示符。 在此处手工加载内核模块：

```
OK? load geom_mirror
OK? boot
```

如果这样做能解决问题， 则说明由于某种原因模块没有被正确加载。 检查 /boot/loader.conf 中相关条目是否正确。 如果问题仍然存在，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时， 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 出现了故障并需要替换， 要替换它， 首先确定哪个磁盘出现了故障，
并关闭系统。 此时， 可以用换上新的磁盘， 并重新启动系统。
这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 命令来监看进度。 就是这样简单。

20.5. RAID3 - 使用专用校验设备的字节级条带

RAID3 是一种将多个磁盘组成一个卷的技术， 在这个配置中包含一个专用于校验的盘。 在 RAID3 系统中，
数据会以字节为单位拆分并写入除校验盘之外的全部驱动器中。 这意味着从 RAID3
中读取数据时将会访问所有的驱动器。 采用多个磁盘控制器可以进一步改善性能。 RAID3
阵列最多可以容忍其中的 1 个驱动器出现故障， 它可以提供全部驱动器总容量的 1 - 1/n， 此处 n
是阵列中的磁盘数量。 这类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 阵列时， 至少需要 3 块磁盘。 所有的盘的尺寸必须一致， 因为 I/O
请求会并发分派到不同的盘上。 另外， 由于 RAID3 本身的设计， 盘的数量必须恰好是 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 实现的。 在 FreeBSD 中建立专用的 RAID3
阵列需要下述步骤。

虽然理论上从 RAID3 阵列启动 FreeBSD 是可行的， 但这并不常见， 也不推荐您这样做。

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点目录：

   ```
   # mkdir /multimedia/
   ```

3. 确定将要加入阵列的磁盘设备名， 并创建新的 RAID3 设备。 最终， 这个设备将代表整个阵列。
   下面的例子使用三个未经分区的 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. 为新建的 gr0 设备分区， 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
```

屏幕上会滚过许多数字， 这个过程需要一段时间才能完成。 此后，
您就完成了创建卷的全部操作， 可以挂载它了。

5. 最后一步是挂载文件系统：

```
# mount /dev/raid3/gr0p1 /multimedia/
```

现在可以使用 RAID3 阵列了。

为了让上述配置在系统重启后继续可用， 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， 可以让系统在引导过程中自动加载这个模块：

```
geom_raid3_load="YES"
```

2. 您需要在 /etc/fstab 文件中加入下列配置， 以便让系统引导时自动挂载阵列上的文件系统：

```
/dev/raid3/gr0p1   /multimedia ufs rw  2   2
```

20.6. GEOM Gate 网络设备

通过 gate 工具， GEOM 支持以远程方式使用设备， 例如磁盘、 CD-ROM、 文件等等。 这和 NFS 类似。

在开始工作之前， 首先要创建一个导出文件。 这个文件的作用是指定谁可以访问导出的资源，
以及提供何种级别的访问授权。 例如， 要把第一块 SCSI 盘的第四个 slice 导出， 对应的 /etc/gg.exports
会是类似下面的样子：

```
192.168.1.0/24 RW /dev/da0s4d
```

这表示允许同属私有子网的所有机器访问 da0s4d 分区上的文件系统。

要导出这个设备， 首先请确认它没有被挂接， 然后是启动 ggated(8) 服务：

```
# ggated
```

现在我们将在客户机上 mount 该设备， 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， 设备应该已经可以通过挂接点 /mnt 访问了。

请注意， 如果设备已经被服务器或网络上的任何其他机器挂接， 则前述操作将会失败。

如果不再需要使用这个设备， 就可以使用 umount(8) 命令来安全地将其卸下了，
这一点和其他磁盘设备类似。

20.7. 为磁盘设备添加卷标

在系统初始化的过程中， FreeBSD 内核会为检测到的设备创建设备节点。 这种检测方式存在一些问题，
例如， 在通过 USB 添加设备时应如何处理？ 很可能有闪存盘设备最初被识别为 da0 而在这之后， 则由 da0
变成了 da1。 而这则会在挂接 /etc/fstab 中的文件系统时造成问题， 这些问题，
还可能在系统引导时导致无法正常启动。

解决这个问题的一个方法是以连接拓扑方式链式地进行 SCSI 设备命名， 这样， 当在 SCSI
卡上增加新设备时， 这些设备将使用一个未用的编号。 但如果 USB 设备取代了主 SCSI 磁盘的位置呢？
由于 USB 通常会在 SCSI 卡之前检测到， 因此很可能出现这种现象。 当然，
可以通过在系统引导之后再插入这些设备来绕过这个问题。 另一种绕过这个问题的方法， 则是只使用 ATA
驱动器， 并避免在 /etc/fstab 中列出 SCSI 设备。

还有一种更好的解决方法。 通过使用 glabel 工具， 管理员或用户可以为磁盘设备打上标签， 并在
/etc/fstab 中使用这些标签。 由于 glabel 会将标签保存在对应 provider 的最后一个扇区，
在系统重启之后， 它仍会持续存在。 因此， 通过将具体的设备替换为使用标签表示，
无论设备节点变成什么， 文件系统都能够顺利地完成挂接。

这并不是说标签一定是永久性的。 glabel 工具既可以创建永久性标签，
也可以创建临时性标签。 在重启时， 只有永久性标签会保持。 请参见联机手册 glabel(8)
以了解两者之间的差异。

20.7.1. 标签类型和使用示范

有两种类型的标签， 一种是普通标签， 另一种是文件系统标签。
标签可以是永久性的或暂时性的。永久性的标签可以通过 tunefs(8) 或 newfs(8)
命令创键。根据文件系统的类型， 它们将在 /dev 下的一个子目录中被创建。例如， UFS2
文件系统的标签会创建到 /dev/ufs 目录中。永久性的标签还可以使用 glabel label
创建。它们不再是文件系统特定的，而是会在 /dev/label 目录中被创建。

暂时性的标签在系统下次重启时会消失， 这些标签会创建到 /dev/label 目录中，
很适合测试之用。可以使用 glabel create 创建暂时性的标签。请参阅 glabel(8)
手册页以获取更多详细信息。

要为一个 UFS2 文件系统创建永久性标签， 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， 这可能会导致数据损坏； 不过， 如果文件系统快满了，
此时应首先删除一些无用的文件， 而不是增加标签。

现在， 您应可以在 /dev/ufs 目录中看到标签， 并将其加入 /etc/fstab：

```
/dev/ufs/home     /home       ufs    rw        2    2
```

当运行 tunefs 时， 应首先卸下文件系统。

现在可以像平时一样挂接文件系统了：

```
# mount /home
```

现在， 只要在系统引导时通过 /boot/loader.conf 配置加载了内核模块 geom_label.ko，
或在联编内核时指定了 GEOM_LABEL 选项， 设备节点由于增删设备而顺序发生变化时，
就不会影响文件系统的挂接了。

通过使用 newfs 命令的 -L 参数， 可以在创建文件系统时为其添加默认的标签。 请参见联机手册 newfs(8)
以了解进一步的详情。

下列命令可以清除标签：

```
# glabel destroy home
```

以下的例子展示了如何为一个启动磁盘打上标签。

例 31. 为启动磁盘打上标签

为启动磁盘打上永久性标签， 系统应该能够正常启动，
即使磁盘被移动到了另外一个控制器或者转移到了一个不同的系统上。 此例中我们假设使用了一个
ATA 磁盘， 当前这个设备被系统识别为 ad0。 还假设使用了标准的 FreeBSD 分区划分方案， /, /var,
/usr 和 /tmp 文件系统， 还有一个 swap 分区。

重启系统，在 loader(8) 提示符下键入 4 启动到单用户模式。 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
```

系统加继续启动进入多用户模式。 在启动完毕后， 编辑 /etc/fstab
用各自的标签替换下常规的设备名。 最终 /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， 系统可以正常启动并且 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 新增了一种用于 UFS 文件系统唯一标识符， ufsid 的标签支持。 这些标签可以在 /dev/ufsid 目录中找到， 它们会在系统引导时自动创建。 在 /etc/fstab 机制中， 也可以使用 ufsid 标签。 您可以使用 glabel status 命令来获得与文件系统对应的 ufsid 标签列表：

```
% 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 则代表了 /usr 文件系统。 您可以使用这些 ufsid 值来挂载它们， 在 /etc/fstab 中配置类似这样：

```
/dev/ufsid/486b6fc38d330916     /var     ufs     rw     2    2
/dev/ufsid/486b6fc16926168e     /usr     ufs     rw     2    2
```

所有包含了 ufsid 的标签都可以用这种方式挂载， 从而消除了需要手工创建永久性标签的麻烦， 而又能够提供提供与设备名无关的挂载方式的便利。

20.8. 通过 GEOM 实现 UFS 日志

随着 FreeBSD 7.0 的发布， 提供了长期为人们所期待的日志功能的实现。 这个实现采用了 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， 换言之， 完成一次完整的磁盘写入操作所需的变动， 这些记录会在元数据以及文件数据写盘之前， 写入到磁盘中。 这种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问题。

这种方法是另一种阻止文件系统丢失数据并发生不一致的方法。 与 Soft Updates 追踪并确保元数据更新顺序这种方法不同， 它会实际地将日志保存到指定为此项任务保留的磁盘空间上， 在某些情况下可全部存放到另外一块磁盘上。

与其他文件系统的日志实现不同， gjournal 采用的是基于块， 而不是作为文件系统的一部分的方式 - 它只是作为一种 GEOM 扩展实现。

如果希望启用 gjournal， FreeBSD 内核需要下列选项 - 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， 还需加载 geom_journal.ko 内核模块， 将以下这行加入 /boot/loader.conf：

```
geom_journal_load="YES"
```

这个功能也可被编译进一个定制的内核， 需在内核配置文件中加入以下这行：

```
options GEOM_JOURNAL
```

现在， 可以为空闲的文件系统创建日志了。 对于新增的 SCSI 磁盘 da4， 具体的操作步骤为：

```
# gjournal load
# gjournal label /dev/da4
```

这样， 就会出现一个与 /dev/da4 设备节点对应的 /dev/da4.journal 设备节点。 接下来， 可以在这个设备上建立文件系统：

```
# newfs -O 2 -J /dev/da4.journal
```

这个命令将建立一个包含日志设备的 UFS2 文件系统。

然后就可以用 mount 命令来挂接设备了：

```
# 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 这两个 slice， 则 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。 对于这类情形， 应该在启用日志的设备后面，给出日志提供者或存储设备。 在暨存的文件系统上， 可以用 tunefs 来启用日志； 不过， 在尝试修改文件系统之前， 您应对其进行备份。 多数情况下， 如果无法创建实际的日志， 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 系统的启动磁盘使用日志也是可能的。 请参阅 Implementing UFS Journaling on a Desktop PC 以获得更多详细信息。

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。
它们允许用户上载和存储文件，提供对数据的访问，当然，
是使硬盘能具有实际的用途。不同的操作系统通常都有一个共同的主要方面，
那就是它们原生的文件系统。在 FreeBSD 上这个文件系统通常被称为快速文件系统或者 FFS，
这是基于原来的 Unix™ 文件系统，通常也被称为 UFS。 这是 FreeBSD
用于在磁盘上访问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访问数据的支持，
那些就是指存放在本地挂载的 USB 存储设备，闪存设备和硬盘上的数据。还支持一些非原生的文件系统。
这些文件系统是在其他的操作系统上开发的，像 Linux® 的扩展文件系统 （EXT），和 Sun™ 的 Z 文件系统
（ZFS）。

FreeBSD 上对于各种文件系统的支持分成不同的层次。
一些要求加载内核模块，另外的可能要求安装一系列的工具。 这一章节旨在帮助 FreeBSD
用户在他们的系统上访问其他的文件系统， 由 Sun™ 的 Z 文件系统开始。

在阅读了这一章节之后，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。
- 如何起用，配置，访问和使用非原生的文件系统。

在阅读这章以前，你应该：

- 了解 UNIX® 和 FreeBSD 基本知识 (UNIX 基础)。
- 熟悉基本的内核配置/编译方法 (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，存储和设备名 (存储)。

21.2. Z 文件系统 (ZFS)

Z 文件系统是由 Sun™ 开发使用存储池方法的新技术。
这就是说只有在需要存储数据的时候空间才会被使用。
它也为保护数据最大完整性而设计的，支持数据快照， 多份拷贝和数据校验。增加了被称为 RAID-Z
的新的数据复制类型。RAID-Z 是种类似于 RAID5类型, 但被设计成防止写入漏洞。

21.2.1. 调整 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校来为日常应用提供最大化的效能。 作为
FreeBSD 的一项试验性的特性，这可能在不久的将来有所变化；
无论如何，下面的这些步骤是我们推荐的：

21.2.1.1. 内存

总共的系统内存至少应有 1GB，推荐 2GB 或者更多。 在此处所有的例子中，我们使用了 1GB
内存的系统并配合了一些恰当的调校。

有些人在少于 1GB 内存的环境有幸正常使用，
但是在这样有限的物理内存的条件下，当系统的负载很高时， FreeBSD 极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驱动和选项从内核配置文件中去除。

既然大部份的驱动都有以模块的形式存在，它们就可以很容易的通过 /boot/loader.conf 加载。

i386™ 构架的用户应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 能够超越 1 GB 的限制(PAE为 2 GB)。 为了找出这个选项最合适的值， 把以兆(MB)为单位所需的地址空间除以 4 得到。 在这个例子中，512 则为 2 GB。

21.2.1.3. Loader 可调参数

所有构架上 FreeBSD 都应该加大 kmem 地址空间。在有 1GB 物理内存的测试系统上，在 /boot/loader.conf 中加入如下的参数并且重启后通过了测试。

```
vm.kmem_size="330M"
vm.kmem_size_max="330M"
vfs.zfs.arc_max="40M"
vfs.zfs.vdev.cache.size="5M"
```

更多 ZFS 相关推荐调校的细节请参阅 http://wiki.freebsd.org/ZFSTuningGuide.

21.2.2. 使用 ZFS

FreeBSD 有一种启动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
```

这份文档剩余的部分假定系统中有 3 块 SCSI 磁盘可用， 它们的设备名分别为 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

在单个磁盘上创建一个简单， 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
```

这份输出清楚的表明了 example 存储池不仅创建成功而且被 挂载 了。

我们能像访问普通的文件系统那样访问它，就像以下例子中演示的那样，用户能够在上面创建文件并浏览：

```
# cd /example
# ls
# touch testfile
# ls -al
total 4
drwxr-xr-x  2 root  wheel    3 Aug 29 23:15 .
drwxr-xr-x 21 root  wheel  512 Aug 29 23:12 ..
-rw-r--r--  1 root  wheel    0 Aug 29 23:15 testfile
```

遗憾的是这个存储池并没有利用到 ZFS 的任何特性。 在这个存储池上创建一个文件系统，并启用压缩：

```
# zfs create example/compressed
# zfs set compression=gzip example/compressed
```

现在 example/compressed 是一个启用了压缩的 ZFS 文件系统了。 可以尝试复制一些大的文件到 /example/compressed。

使用这个命令可以禁用压缩：

```
# zfs set compression=off example/compressed
```

使用如下的命令卸载这个文件系统，并用 df 工具确认：

```
# 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访问，并用 df 确认：

```
# 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
```

存储池与文件系统也可通过 mount 的输出查看：

```
# 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，
还有很多其他的特性是可用的。在下面的例子中， 我们将创建一个新的文件系统，data。
并要在上面存储些重要的文件， 所以文件系统需要被设置成把每一个数据块都保存两份拷贝：

```
# zfs create example/data
# zfs set copies=2 example/data
```

现在可以再次使用 df 查看数据和空间的使用状况：

```
# 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
```

请注意存储池上的每一个文件系统都有着相同数量的可用空间。 这就是我们在这些例子中使用 df 的原因，
是为了文件系统都是从相同的存储池取得它们所需的空间。 ZFS 去掉了诸如卷和分区此类的概念，
并允许多个文件系统占用同一个存储池。 不再需要文件系统与存储池的时候能像这样销毁它们：

```
# zfs destroy example/compressed
# zfs destroy example/data
# zpool destroy example
```

磁盘无法避免的会坏掉和停止运转。 当这块磁盘坏掉的时候，上面的数据都将丢失。
一个避免因磁盘损坏而丢失数据的方法是使用 RAID。ZFS 在它的存储池设计中支持这样的特性，
这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，这一章节将假设存在 3 个 SCSI 设备， da0， da1 和 da2 (或者 ad0
和超出此例使用了 IDE 磁盘)。 使用如下的命令创建一个 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为 3 至 9 块。
如果你要求在单独的一个存储池中使用 10 块或更多的磁盘， 请考虑分拆成更小 RAID-z 组。 如果你只有 2 块磁盘， 并仍然需要冗余， 请考虑使用 ZFS 的 mirror 特性。
更多细节请参考 zpool(8) 手册页。

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) 确认。
如需配给更多的磁盘设备则把它们加这个列表的后面。 在存储池上创建一个叫 home 的文件系统，
用户的文件最终都将被保存在上面：

```
# zfs create storage/home
```

像前文中提到的那样，用户的目录与文件也可启用压缩并保存多份拷贝， 可通过如下的命令完成：

```
# zfs set copies=2 storage/home
# zfs set compression=gzip storage/home
```

把用户的数据都拷贝过来并创建一个符号链接， 让他们开始使用这个新的目录：

```
# cp -rp /home/* /storage/home
# rm -rf /home /usr/home
# ln -s /storage/home /home
# ln -s /storage/home /usr/home
```

现在用户的数据应该都保存在新创建的 /storage/home 上了。 测试添加一个新用户并以这个身份登录。

尝试创建一个可日后用来回退的快照：

```
# zfs snapshot storage/home@08-30-08
```

请注意快照选项将只会抓取一个真实的文件系统， 而不是某个用户目录或文件。@
字符为文件系统名或卷名的分隔符。 当用户目录被损坏时，可用如下命令恢复：

```
# zfs rollback storage/home@08-30-08
```

获得所有可用快照的列表，可使用 ls 命令查看文件系统的 .zfs/snapshot
目录。例如，执行如下命令来查看之前抓取的快照：

```
# ls /storage/home/.zfs/snapshot
```

可以编写一个脚本来每月定期抓取用户数据的快照，久而久之， 快照可能消耗掉大量的磁盘空间。
之前创建的快照可用以下命令删除：

```
# zfs destroy storage/home@08-30-08
```

在所有这些测试之后，我们没有理由再把 /store/home 这样放置了。让它称为真正的 /home 文件系统：

```
# zfs set mountpoint=/home storage/home
```

使用 df 和 mount 命令将显示现在系统把我们的文件系统真正当作了 /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
```

这样就基本完成了 RAID-Z 的配置了。使用夜间 periodic(8) 获取有关文件系统创建之类的状态更新，执行如下的命令：

```
# echo 'daily_status_zfs_enable="YES"' >> /etc/periodic.conf
```

21.2.2.3. 修复 RAID-Z

每一种软 RAID 都有监测它们 状态 的方法。 ZFS 也不例外。 可以使用如下的命令查看 RAID-Z 设备：

```
# zpool status -x
```

如果所有的存储池处于健康状态并且一切正常的话， 将返回如下信息：

```
all pools are healthy
```

如果存在问题，可能是一个磁盘设备下线了， 那么返回的存储池的状态将看上去是类似这个样子的：

```
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，这是由管理员把此设备下线后的状态。 可以使用如下的命令将磁盘下线：

```
# zpool offline storage da1
```

现在切断系统电源之后就可以替换下 da1 了。 当系统再次上线时，使用如下的命令替换磁盘：

```
# zpool replace storage da1
```

至此可用不带 -x 标志的命令再次检查状态：

```
# 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) 来检查存储数据的完整性。
这时在文件系统创建时自动启用的，可使用以下的命令禁用：

```
# zfs set checksum=off storage/home
```

这不是个明智的选择，因为校验和 不仅非常有用而且只需占用少量的存储空间。
并且启用它们也不会明显的消耗过多资源。 启用后就可以让 ZFS 使用校验和校验来检查数据的完整。
这个过程通常称为 "scrubbing"。 可以使用以下的命令检查 storage 存储池里数据的完整性：

```
# zpool scrub storage
```

这个过程需花费相当长的时间，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执行一个这样的操作。 在 scrub 完成之后，状态就会被更新，
可使用如下的命令查看：

```
# 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
```

这个例子中完成时间非常的清楚。 这个特性可以帮助你在很长的一段时间内确保数据的完整。

Z 文件系统有更多的选项，请参阅 zfs(8) 和 zpool(8) 手册页。

Chapter 22. Vinum 卷管理程序

22.1. 概述

无论您有什么样的磁盘，总会有一些潜在问题：

- 它们可能容量太小。
- 它们可能速度太慢。
- 它们可能也太不可靠。

针对这些问题， 人们提出并实现了许多不同的解决方案。 为了应对这些问题， 一些用户采用了多个，
有时甚至是冗余的磁盘这类方法。 除了支持许多种不同的硬件 RAID 控制器之外， FreeBSD
的基本系统中包括了 Vinum 卷管理器， 它是一个用以实现虚拟磁盘驱动器的块设备。 Vinum 是一种称为
卷管理器， 或者说用于解决前面这三种问题的虚拟磁盘驱动程序。 Vinum
能够提供比传统磁盘系统更好的灵活性、 性能和可靠性， 并实现了能够单独或配合使用 RAID-0、 RAID-1
和 RAID-5 模型。

这一章对传统磁盘存储的潜在问题进行了简要说明，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. 模块化磁盘变换框架)， 同时保留其原有的设计创意、 术语，
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实现。
新版本中所有的指令都应通过 gvinum 命令来操作， 而对应的内核模块的名字， 也由
vinum.ko 改为了 geom_vinum.ko， 而在 /dev/vinum 中的所有设备节点，
也改为放到了 /dev/gvinum。 从 FreeBSD 6 开始， 旧版的 Vinum 实现已不再提供。

22.2. 磁盘容量太小

磁盘越大，存储的数据也就越多。您经常会发现您需要 一个比您可使用的磁盘大得多的文件系统。
无可否认，这个问题 已经没有十年前那样严峻了，但它仍然存在。通过创建一个在许多
磁盘上存储数据的抽象设备，一些系统可以解决这个问题。

22.3. 访问瓶颈

现代系统经常需要用一个高度并发的方式来访问数据。 例如，巨大的FTP 或HTTP
服务器可以支持数以千计的并发会话， 可以有多个连到外部世界的100 Mbit/s ，
这远远地超过了绝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,
但这个值在一个有许多不受约束的进程访问一个驱动器的环境中变得并不重要，
它们可能只完成了这些值的一小部分。这样一种情况下，从磁盘子
系统的角度来看问题就更加有趣：重要的参数是在子系统上的负
荷，换句话说是传输占用了驱动器多少时间。

在任何磁盘传输中, 驱动器必须先寻道, 等待磁头访问第一个扇区, 然后执行传输. 这些动作看起来可能很细小:
我们不会感有任何中断。

假设传输10 kB数据， : 现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，，
所以平均寻址时间为2ms. 在70 MB/s的速度传输时, 数据的传输时间大约150 µs, 几乎无法和寻址时间相比.
在这样一种情况下, 高效的传输也会降低到 1 MB/s 显然传输的快慢依赖与所传输数据的大小。

对于这个瓶颈的一般和明显的解决方法是采用 "多个磁盘":而不是只使用一个大磁盘,
它使用几个比较小的磁盘联合起来形成一个大的磁盘.
每个磁盘都可以独立地进行传输数据，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。
尽管每个驱动器并行传输数据，但没有办法确保请求能够平均

分配到每个驱动器上。不可避免一个驱动器的负载可能比另一个要高得多。

磁盘的负载平衡很大程度依赖于驱动器上数据的共享方式. 在下面的讨论中,
将磁盘存储想象成一个巨大的数据扇区，像一本书的页 那样用编号来设定地址.
最明显的方法是把虚拟磁盘分成许多连续的扇区组，
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 串联 它有一个优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较小的区域的时候，性能的提高没有显著的改进。 串联组织
举例说明了用串联组织的方式来分配存储单元的顺序。

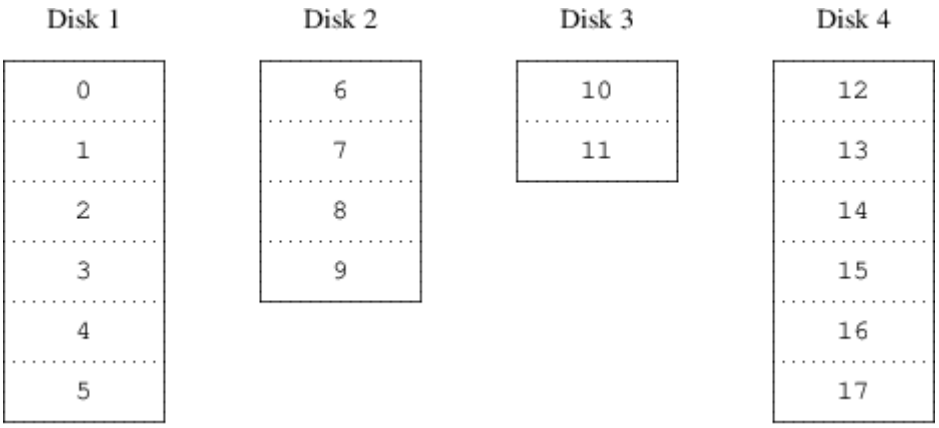


图 103. 串联组织

另外一种影射方法是把地址空间分布在比较小的容量相同的磁盘上，
从而能够在不同的设备上存储它们。例如，前256 个扇区可能存储在第一 个磁盘上，接着的256
个扇区存储在另一个磁盘上等等。 写满最后一个磁
盘后，进程会重复以前的工作，直到所有的磁盘被写满。这个影射叫做 分段(striping) 或者 RAID-0.
分段要求很精确地寻址，通过多个磁盘进行数据传输的时候，它 可能会引起额外的I/O
负载，但它也可能提供更多的连续负载。 分段组织 显示了用分段形式分配的存储单元的顺序。

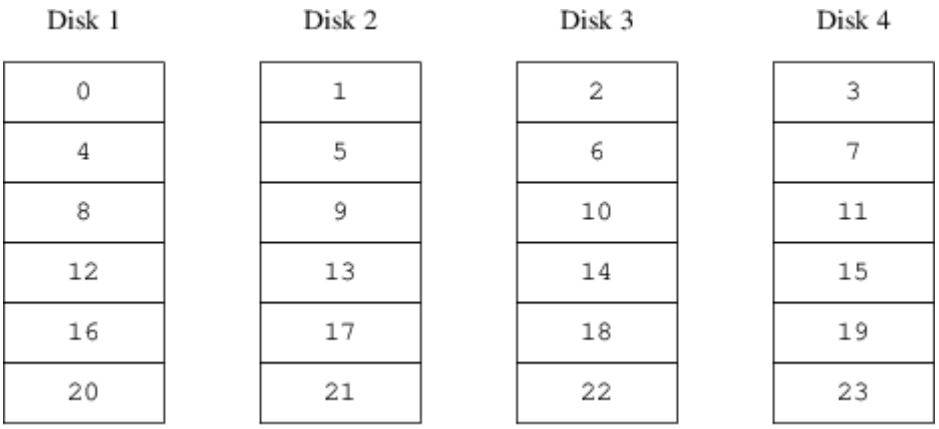


图 104. 分段组织

22.4. 数据的完整性

现时磁盘的最后一个问题是它们不太可靠。 虽然磁盘驱动器的可靠性在过去几年有了很大的提高，
但它们仍然是服务器中最容易损坏的核心组件。 当它们发生故障的时候， 结果可能是灾难性的:
替换坏的磁盘驱动器并恢复数据可能要花费几天时间。

解决这个问题的传统方法是建立 镜象， 在不同的物理硬件上对数据做两个副本。 根据 RAID
级别出现的时间顺序， 这个技术也被叫做 RAID 级别1 或者 RAID-1。 任何写到卷的数据也会被写到镜象上，
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， 数据也还可以从其他驱动器上访问到。

镜象有两个问题:

- 价格. 它需要两倍的存储容量。

- 性能影响。 写入操作必须在两个驱动器上执行，所以它们 花费两倍的带宽。读取数据并不会影响性能：它们甚至看起来会更快。

一个可选的方案采用 奇偶校验 的方式， 用以实现 RAID 2、 3、 4 和 5。 这其中， RAID-5 是我们最感兴趣的。 在 Vinum 的实现中， 这是一个条带组织结构的变体， 其中，每一个条带中都以一个专用的块， 来保存其它块的奇偶校验值。 这样， RAID-5 plex 除了在每个块中都包含了一个奇偶校验块之外， 实现 RAID-5 时也就和普通的条带 plex 一样了。 作为 RAID-5 的一项要求， 奇偶校验块在每一个条带中的顺序都是不同的。 数据块的编号， 决定了它的相对块号。

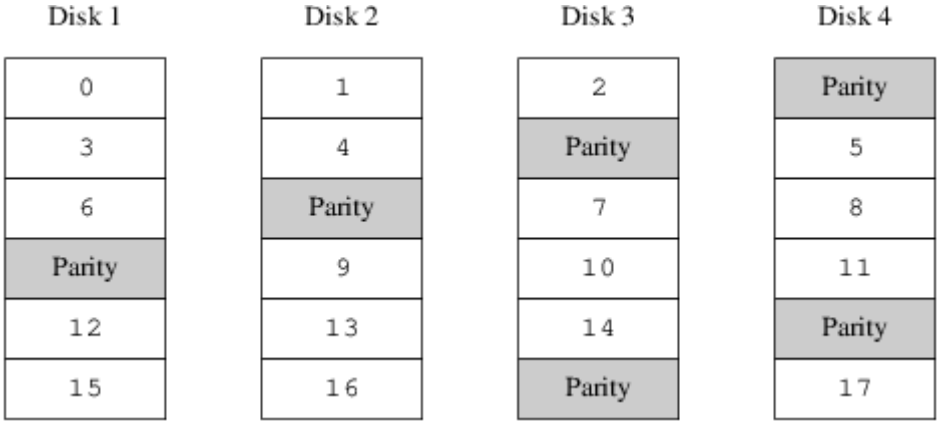


图 105. RAID-5 的组织

与镜像相比， RAID-5 最显著的优势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组织，但写入会慢得多， 大约仅相当于读性能的 25%。 如果一个驱动器失效，则阵列仍然可以在降级的模式运行： 读取来自正常的驱动器数据的操作照常进行，但读取失效的驱动器的数据， 则来自于余下驱动器上相关的计算结果。

22.5. Vinum 目标

为了解决这些问题，Vinum 提出了一个四层的目标结构：

- 最显著的目标是虚拟磁盘, 叫做 卷(volume). 卷本质上与一个UNIX 磁盘驱动器有同样的属性，虽然它们是有些不太一样。它们没有大小的限制。
- 卷下面是 plexes, 每一个表示卷的所有地址空间。在层次结构中的这个水平能够提供冗余功能。可以把plex 想象成用一个镜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, 但事实上这并不可靠:UNIX 磁盘只能有有限数量的分区。 取而代之，Vinum 把一个简单的UNIX 分区 (the drive) 分解成叫做subdisks的相邻区域, 它可以使用这个 来为plex 建立块。
- Subdisks 位于 Vinum 驱动器上, 当前的UNIX 分区。Vinum 驱动器可以包含很多的subdisks。除了驱动器开始的一小块区域用来存储配置和描述信息以外，整个 驱动器都可以用于存储数据。

下面的章节描述了这些目标提供了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 分布在所有的驱动上。 结果,每个独立的驱动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. 多余的数据存储

Vinum 通过给一个卷连上多个plex 来完成镜象的功能。 每个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。

虽然一个plex 描述了一个卷的所有数据，, 但可能描述的部分被物理地丢失了。可能是设计的问题（没有为plex 部分定义一个subdisk）也可能是意外的故障（由于驱动器的故障导致）。只要至少有一个plex 能够为 卷的完全地址范围提供数据，卷就能够正常工作。

22.5.3. 性能问题

Vinum 在plex 水平既执行串联也执行分段:

- 一个串连的plex轮流使用 每个subdisk 的地址空间。
- 一个 分段的plex 在每个subdisk 上 划分数据. Subdisk 必须是大小一样的，为了从一个连接的plex 中区分开它，必须至少有两个subdisk。

22.5.4. 哪种plex 组织更有效?

FreeBSD 12.0提供的Vinum 版本能实现两种plex:

- 串联的plex 更加灵活：它们可以包含任何数量的subdisk， subdisk 也可能有不同的长度。Plex 可以通过添加额外的subdisk 来得到扩展。 与分段 plex 不同， 它们需要的 CPU 时钟更少， 尽管 CPU 上的负载差异是不可测量的。 另一方面，它们的负载可能不平衡，一个磁盘可能负载很重，而其他的可能很空闲。
- 分段(RAID-0) plexes 的最大优点是 它们减少了负载不平衡的情况: 通过选择一个最合适大小的分段(大约是256 kB), 您甚至可以在各个组成的驱动器上降低负载 . 这种方法的缺点是在subdisk 上受到非常复杂的编码限制 : 它们必须是同样大小, 通过添加新的subdisk 来扩展一个plex 是非常复杂的,以至Vinum 当前没有实现它. Vinum 利用一个额外 的，代价不高的限制：一个分段的plex 必须有至少两个subdisk， 否则， 它就无法区分连接的 plex 了。

Vinum Plex组织图 总结一下每个plex 组织 的优点和缺点.

表 8. Vinum Plex组织图

Plex 类型	最少subdisks	可否添加subdisks	尺寸相同	应用
串联	1	可以	不必须	带有很大弹性和适中性能的大数据量存储。
分段	2	不可以	必须	大量并发访问时,具有较高性能。

22.6. 一些例子

Vinum 维护着一个描述本系统中对象的 配置数据库。 开始时， 用户可以在 gvinum(8) 工具来从若干配置文件生成配置数据库。 Vinum 在其控制的每个磁盘分区 (在 Vinum 中称为 device) 上都保存配置数据库的副本。 这一数据库在每次状态变化时均会更新， 因而重启每个 Vinum 对象时，都能够恢复其状态。

22.6.1. 配置文件

配置文件描述了独立的 Vinum.一个简单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 目标:

- drive 行描述了一个磁盘分区（驱动器） 和与下面的硬件相关的它的位置。它给出了一个符号名 a. 这个与设备名称分开的符号名允许 磁盘从一个位置移动到另一个位置而不会搞混。
- volume 行描述了一个卷。 唯一的必须属性是名称，在这个例子中是 myvol.
- plex 行定义了一个plex。 唯一需要的参数是组织,在这个例子中是 concat. 没有名称是必然的:

系统自动通过添加suffix .px 来从卷名称产生一个名字,这里的x 是在卷中的plex 的编号。而这个plex 将被 叫做myvol.p0。

- sd 行描述了一个subdisk。 最小的说明是存储subdisk 的驱动器名称，和subdisk 的长度。
  对于plex，没有名称也是必然的：系统自动通过添加 suffix .sx 来分配源自plex 的名称，这里 x是plex 中subdisk 的编号。 Vinum 给这个subdisk 命名为myvol.p0.s0。

处理完这个文件后， gvinum(8) 会产生下面的输出：

```
# 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       PO:        0  B Size:        512 MB
```

这些输出内容展示了 gvinum(8) 的简要列表格式。 在 一个简单的Vinum 卷 中用图形展示了这个配置。

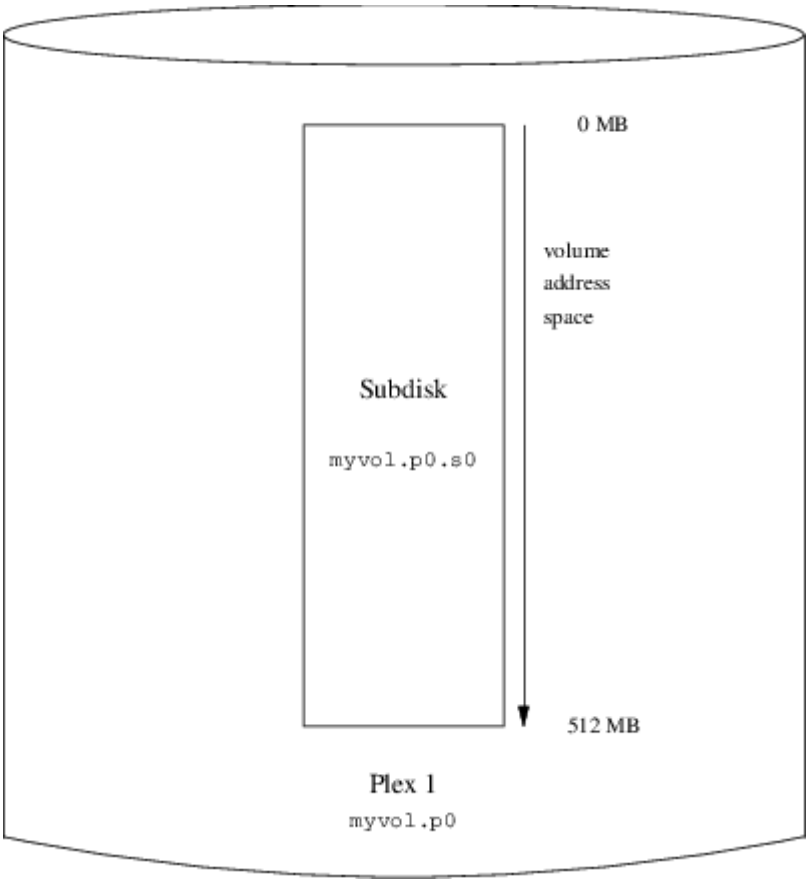


图 106. 一个简单的Vinum 卷

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， 因此这和从磁盘分区分配存储没什么两样。 接下来的几节，
将介绍一些更有用的配置方法。

22.6.2. 提高容错性： 镜像

卷的容错性可以通过镜像来提高。 在配置镜像卷时， 确保 plex 分布在不同的驱动器上十分重要，
这样一个驱动器坏掉时， 就不会同时影响两个 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， 并不需要再次指定驱动器 a， 因为 Vinum 监控所有其配置数据库的对象。 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

S myvol.p0.s0       State: up     PO:      0  B Size:      512 MB
S mirror.p0.s0      State: up     PO:      0  B Size:      512 MB
S mirror.p1.s0      State: empty    PO:      0  B Size:      512 MB
```

镜像 Vinum 卷 以图形方式展示了其结构。

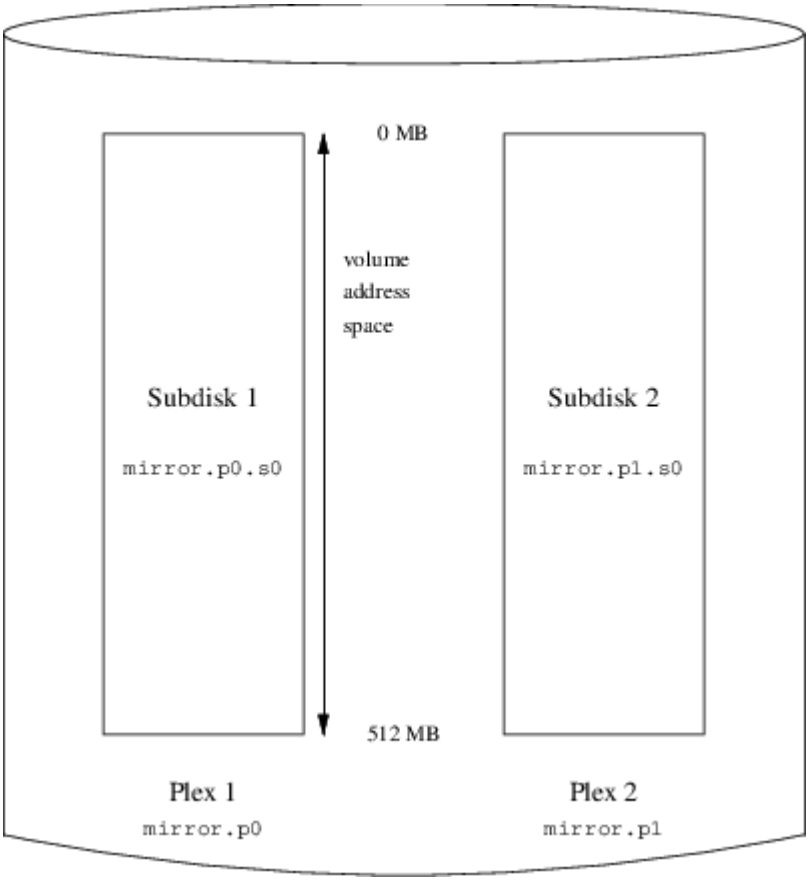


图 107. 镜像 Vinum 卷

这个例子中， 每一个 plex 包含了完整的 512 MB 地址空间。 在前面的例子中， plex 则只包括一个子盘。

22.6.3. 优化性能

前面例子中的镜像卷要比没有镜像的卷具有更好的容灾能力， 但它的性能要差一些： 每一次写入卷时，
需要同时写到两个驱动器上， 因而也就需要更大的磁盘访问带宽。 如果希望非常好的性能，
则需要另外一种方式： 不做镜像， 而将数据分成条带放到尽可能多的、不同的磁盘上。
下面给出了一个跨越四个磁盘驱动器的 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 已经知道的驱动器。 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
P striped.p1       State: up     Subdisks:    1 Size:      512 MB

S myvol.p0.s0      State: up     PO:      0  B Size:      512 MB
S mirror.p0.s0     State: up     PO:      0  B Size:      512 MB
S mirror.p1.s0     State: empty    PO:      0  B Size:      512 MB
S striped.p0.s0    State: up     PO:      0  B Size:      128 MB
S striped.p0.s1    State: up     PO:    512 kB Size:      128 MB
S striped.p0.s2    State: up     PO:   1024 kB Size:      128 MB
S striped.p0.s3    State: up     PO:   1536 kB Size:      128 MB
```

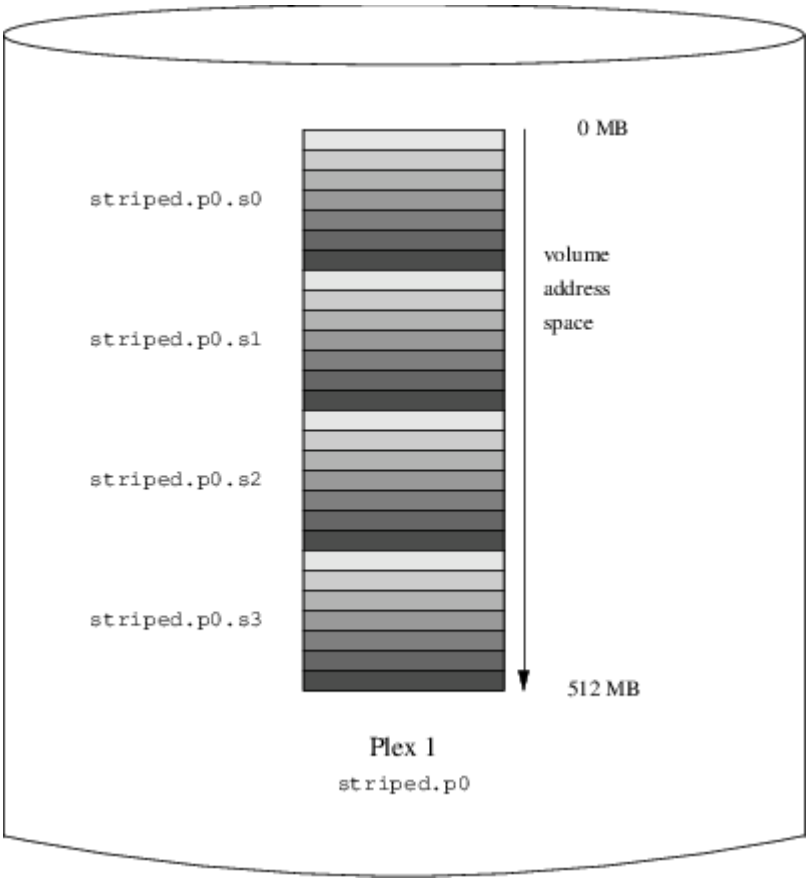


图 108. 条带化的 Vinum 卷

这个卷在 条带化的 Vinum 卷 中给出。 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， 也能够构建比标准 UNIX® 分区同时提高了容灾性和性能的卷。 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 中的子盘和第一个 plex 中的错开了两个驱动器： 这能够帮助确保即使同时访问两个驱动器，
写操作也不会同时发生在同一个盘上。

镜像并条带化的 Vinum 卷 给出了该卷的结构。

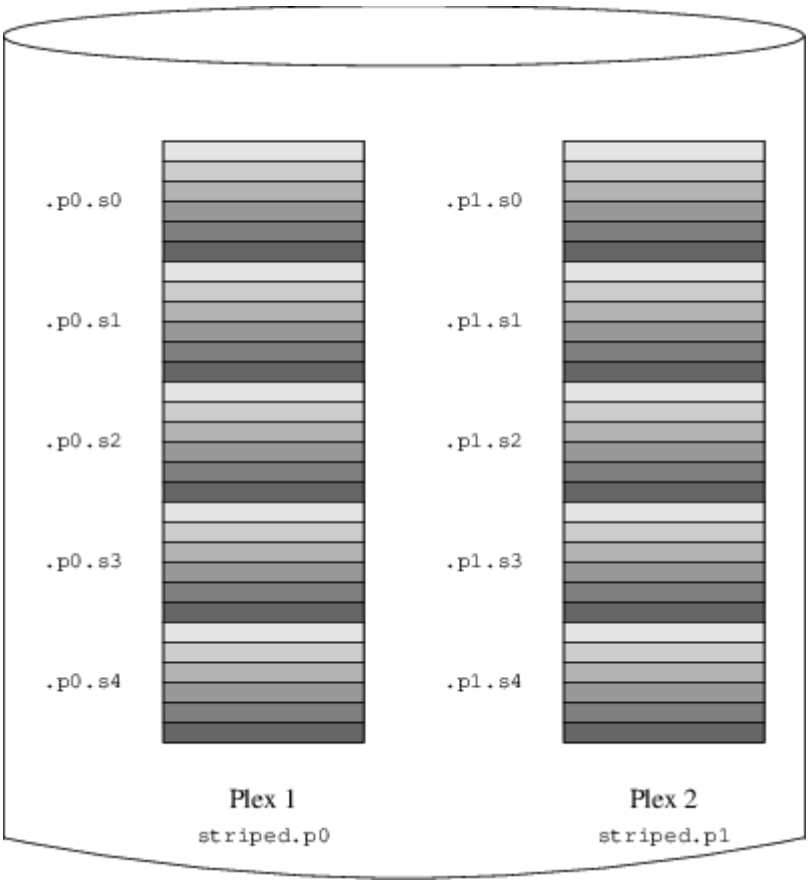


图 109. 镜像并条带化的 Vinum 卷

22.7. 对象命名

如前面所描述的那样， Vinum 会给 plex 和子盘指定默认的名字， 而这些名字也是可以定制的。
不推荐修改默认的名字： 使用允许给对象任意命名的 VERITAS 卷管理器的经验证明，
这一灵活性并没有带来太多的好处， 相反， 它很容易导致对象的混淆。

名字中可以包括任何非空白的字符， 但一般来说， 建议只使用字母、 数字和下划线。 卷、 plex，
以及子盘的名字， 可以包含最多 64 个字符， 而驱动器的名字， 则最长可以使用 32 个字符。

Vinum 对象会在 /dev/gvinum 之下生成设备节点。 前述的配置将使 Vinum 创建以下设备节点：

- 每个卷对应的设备项。 这些是 Vinum 使用的主要设备。 因此， 前述配置包括下列设备：
  /dev/gvinum/myvol、 /dev/gvinum/mirror、 /dev/gvinum/striped、 /dev/gvinum/raid5 以及
  /dev/gvinum/raid10。
- 所有卷的直接项都存放在 /dev/gvinum/ 中。
- 目录 /dev/gvinum/plex， 以及 /dev/gvinum/sd 中相应地存放了每个 plex 以及子盘的设备节点。

例如， 考虑下面的配置文件：

```
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
```

处理这个文件之后， gvinum(8) 将在 /dev/gvinum 中建立下面的结构：

```
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 和子盘一般并不推荐指定名字，但还是必须给 Vinum 驱动器命名。这样，
当把驱动器转移到不同的地方时，它仍然能够被自动地识别出来。驱动器名最长可以包含 32 个字符。

22.7.1. 创建文件系统

对于系统而言，卷和磁盘是一样的。唯一的例外是，与 UNIX® 驱动器不同，Vinum 并不对卷进行分区，
因而它也就不包含分区表。这要求修改某些磁盘工具，特别是 newfs(8)，它会试图将 Vinum
卷名当作分区标识。例如，磁盘驱动器的名字可能是 /dev/ad0a 或 /dev/da2h。
这些名字分别表达在第一个 (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。

要在这个卷上创建文件系统，则需要使用 newfs(8)：

```
# newfs /dev/gvinum/concat
```

22.8. 配置 Vinum

在 GENERIC 内核中，并不包含 Vinum。可以编译一个定制的包含 Vinum 的内核，然而并不推荐这样做。
启动 Vinum 的标准方法，是使用内核模块 (kld)。甚至不需要使用 kldload(8) 来启动 Vinum：在启动
gvinum(8) 时，它会检查这一模块是否已经加载，如果没有，则会自动地加载它。

22.8.1. 启动

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。当从配置数据库中读取时，Vinum
会识别一系列在配置文件中不可用的关键字。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 driveoff set
```

```
1573129b plexoffset 0b
sd name bigraid.p0.s1 drive b plex bigraid.p0 state initializing len 4194304b driveoff set
1573129b plexoffset 4194304b
sd name bigraid.p0.s2 drive c plex bigraid.p0 state initializing len 4194304b driveoff set
1573129b plexoffset 8388608b
sd name bigraid.p0.s3 drive d plex bigraid.p0 state initializing len 4194304b driveoff set
1573129b plexoffset 12582912b
sd name bigraid.p0.s4 drive e plex bigraid.p0 state initializing len 4194304b driveoff set
1573129b plexoffset 16777216b
```

这里最明显的区别是， 指定了配置的位置信息、名称 (这些在配置文件中还是可用的，
但不鼓励用户自行指定) 以及状态信息 (这是用户不能指定的)。 Vinum
并不在配置信息中保存关于驱动器的信息： 它会扫描已经配置的磁盘驱动器上包含 Vinum 标识的分区。
这使得 Vinum 能够在 UNIX® 驱动器被指定了不同的 ID 时也能够正确识别它们。

22.8.1.1. 自动启动

Gvinum 在通过 loader.conf(5) 加载了内核模块之后就能自动启动。 在启动时加载 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 驱动器中读取配置数据库。
正常情况下， 每个驱动器上都包含了同样的配置数据库副本， 因此从哪个驱动器上读取是无所谓的。
但是， 在系统崩溃之后， Vinum 就必须检测哪一个驱动器上的配置数据库是最新的， 并从上面读取配置。
如果需要， 它会更新其它驱动器上的配置。

22.9. 使用 Vinum 作为根文件系统

如果文件系统使用完全镜像的 Vinum 配置， 有时也会希望根文件系统也作了镜像。
这种配置要比镜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
  的基础设施在这一时刻就应该可用了。
- 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， 因此它们必须能够被宿主系统的内建工具
  (例如 PC 机的 BIOS) 识别， 而通常是没办法让它们了解 Vinum 的细节的。

下面几节中， 术语 "根卷" 标识包含根文件系统的 Vinum 卷。 把这个卷命名为 "root"
可能是个不错的主意， 不过从技术上说， 并不严格地要求这样做。 不过，
接下来的命令例子都使用这个名字。

22.9.1. 及早启动 Vinum 以适应对根文件系统的要求

有许多关于它的尺度：

- Vinum 必须在启动时可以被内核使用。 因此， 在 自动启动 中所介绍的方法，
  也就无法适应这一任务的需要了。 在接下来的配置中， 也 不能 设置 start_vinum 参数。
  第一种方法是通过将 Vinum 静态联编到内核中来实现， 这样， 它就在任何时候都可用了，
  虽然一般并不需要这样。 另一种方法是通过 /boot/loader (第三阶段，/boot/loader) 来尽早加载
  vinum 内核模块， 这一操作发生在内核加载之前。 这可以通过将下面的配置：

  ```
  geom_vinum_load="YES"
  ```

  加入到 /boot/loader.conf 文件中来实现。

- 对 Gvinum 而言， 所有的启动过程都是在内核模块加载时自动进行的， 因此上面的操作，
  也就是所要进行的全部工作了。

22.9.2. 让基于 Vinum 的卷在引导时可以访问

因为目前的 FreeBSD 引导程序只有 7.5 KB 的代码，并且已经承担了从 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，并找到引导卷本身的信息。因此，需要一些技巧来为引导代码提供标准的 "a" 分区，而它则包含了根文件系统。

要让这些得以实现，根卷需要满足下面的条件：

- 根卷不能是条带卷或 RAID-5 卷。
- 根卷 plex 不能包含连接的子盘。

需要说明的是，使用多个 plex，每个 plex 都复制一份根文件系统的副本，是需要而且是可行的。然而，引导过程只能使用这些副本中的一个来引导系统，直到内核最终自行挂接根文件系统为止。这些 plex 中的每个子盘，在这之后会有它们自己的 "a" 分区，以表达每一个可以引导的设备。每一个 "a" 分区，尽管并不需要和其它包含根卷的 plex 处于各自驱动器的同一位置。但是，这样创建 Vinum 卷使得镜像卷相互对称，从而能够避免了混淆。

为了创建每一个根卷的 "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员子盘的位置 (从设备开始的偏移量) 和尺寸：

   ```
   # gvinum l -rv root
   ```

   需要注意的是，Vinum 偏移量和尺寸的单位是字节。它们必须是 512 的整数倍，才能得到 bsdlabel 命令所需的块号。

2. 在每一个根卷成员设备上，执行命令：

   ```
   # bsdlabel -e devname
   ```

   这其中，对于没有 slice (也就是 fdisk) 表的磁盘，devname 必须是磁盘的名字 (例如 da0)，或者是 slice 的名字 (例如 ad0s1)。

   如果设备上已经有了 "a" 分区 (比如说，包含 Vinum 之前的根文件系统)，则应改为其它的名字，以便继续访问 (如果需要的话)，但它并不会继续用于启动系统。注意，活动的分区 (类似正挂接的根文件系统) 不能被改名，因此，要完成这项工作，必须从 "Fixit" 盘启动，或者分两步操作，并 (在镜像情形中) 首先操作那些非引导盘。

   然后，设备上 Vinum 分区的偏移 (如果有的话) 必须加到这个设备上根卷对应的子盘上。其结果值，将成为新的 "a" 分区的 "offset" 值。这个分区的 "size" 值，可以根据前面的配置计算得出。"fstype" 应该是 4.2BSD。"fsize"、"bsize"，以及 "cpg" 值，则应与文件系统的实际情况匹配，尽管在配置 Vinum 时并不重要。

   这样，新的 "a" 分区，将创建并覆盖这一设备上的 Vinum 分区的范围。注意，bsdlabel 只有在 Vinum 分区的 fstype 被标记为 "vinum" 时，才允许这样做。

3. 这就成了！所有的 "a" 分区现在都已存在，而且是根卷的一份副本。强烈建议您再次验证其结果，方法是：

   ```
   # fsck -n /dev/devnamea
   ```

务必注意，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。在启动新的 Vinum 根卷时，它们可能和实际在用的根文件系统不匹配。因此，/etc/fstab 和 /boot/loader.conf

这两个文件需要特别地注意。

在下次重启时，引导程序需要从新的基于 Vinum 的根文件系统中获取适当的控制信息，并据此工作。
在内核初始化过程的结尾部分，在所有的设备都被宣示之后，如果显示了下面的信息，则表示配置成功：

```
Mounting root from ufs:/dev/gvinum/root
```

22.9.3. 基于 Vinum 的根文件系统的配置范例

在 Vinum 根卷配置好之后，gvinum l -rv root 的输出可能类似下面的样子：

```
...
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)
```

需要注意的值是 135680，也就是偏移量 (相对于 /dev/da0h 分区)。这相当于 bsdlabel 记法中的 265 个 512-字节的磁盘块。类似地，根卷的尺寸是 245760 个 512-字节的磁盘块。/dev/da1h 中，
包含了根卷的第二个副本，采用了同样的配置。

这些设备的 bsdlabel 类似下面的样子：

```
...
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        # (Cyl.    0 - 4467*)
  h: 71771672      16     vinum                       # (Cyl.    0*- 4467*)
```

可以看到，伪装的 "a" 分区的 "size" 参数和前面的一样，而 "offset" 参数则是 Vinum 分区 "h"，
以及设备中这一分区 (或 slice) 的偏移量之和。这是一种典型的配置，它能够避免在 无法启动，
引导程序发生 panic 中介绍的问题。此外，我们也看到整个 "a" 分区完全处于设备上包含了 Vinum 数据的 "h" 分区之中。

注意，在上面的配置中，整个设备都是 Vinum 专用的，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 配置中的一个子盘。

22.9.4. 故障排除

如果遇到了问题，则需要从中恢复的办法。下面列出了一些常见的缺陷，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

如果由于某种原因系统不再继续启动，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 命令来查看，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预加载的列表，而没有正确加载，则简单使用 load geom_vinum 会有所帮助。

此后，可以使用 boot -as 来继续启动过程。选项 -as 会要求内核询问所挂接的根文件系统 (-a)，
并使引导过程在单用户模式停止 (-s)，此时根文件系统是以只读方式挂接的。这样，即使只挂接了多 plex 卷中的一个 plex，也不会引致 plex 之间数据不一致的问题。

当提示输入要挂接的根文件系统时，可以输入任何一个包含根文件系统的设备。如果正确地配置了 /etc/fstab，则默认的应该是类似 ufs:/dev/gvinum/root。一般可以使用类似 ufs:da0d
这样的设备来代替它，因为它通常包括了 Vinum 之前的根文件系统。需要注意的是，如果在这里输入了 "a" 分区，则它可能表达的实际上是 Vinum 根设备的一个子盘，而在镜像式配置中，
这只会挂接镜像的根设备中的一个。如果之后将这个文件系统以读写方式挂接，则需要从 Vinum 根卷中删去其他的 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 (在启动时，屏幕最左边一列有一个旋转的线)，
则可以尝试在此时中断主引导程序的过程，方法是按 space 键。这将在引导的第二阶段暂停，
具体可以参见 第一阶段，/boot/boot1，和第二阶段，/boot/boot2。此时，可以尝试从另一个分区，
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，引导程序发生 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
目前只在分区开始的地方保留了 4 KB 的空间，之后就开始写 Vinum 头信息了。然而，
目前第一阶段和第二阶段的引导程序，加上 bsdlabel 嵌入的内容则需要 8 KB。因此，如果 Vinum
分区从偏移量 0 开始，而这个 slice 或磁盘能够启动，则 Vinum 的安装将毁掉引导程序。

类似地，如果从上述情形中恢复，例如，从 "Fixit" 盘启动，并通过 bsdlabel -B 按照
第一阶段，/boot/boot1，和第二阶段，/boot/boot2 中介绍的方法来恢复引导程序，则引导程序会覆盖掉 Vinum 头，这样 Vinum 也就找不到它的磁盘了。尽管这并不会真的毁掉 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，
但从这种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 头和系统的引导程序不再冲突。

Chapter 23. 虚拟化

23.1. 概述

虚拟化软件能够让同一台机器上同时运行多个操作系统。 在 PC 上，
这种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。

读完这章， 您将了解：

- 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。
- 如何针对虚拟化环境对 FreeBSD 系统进行性能调优。

在阅读这章之前， 您应：

- 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

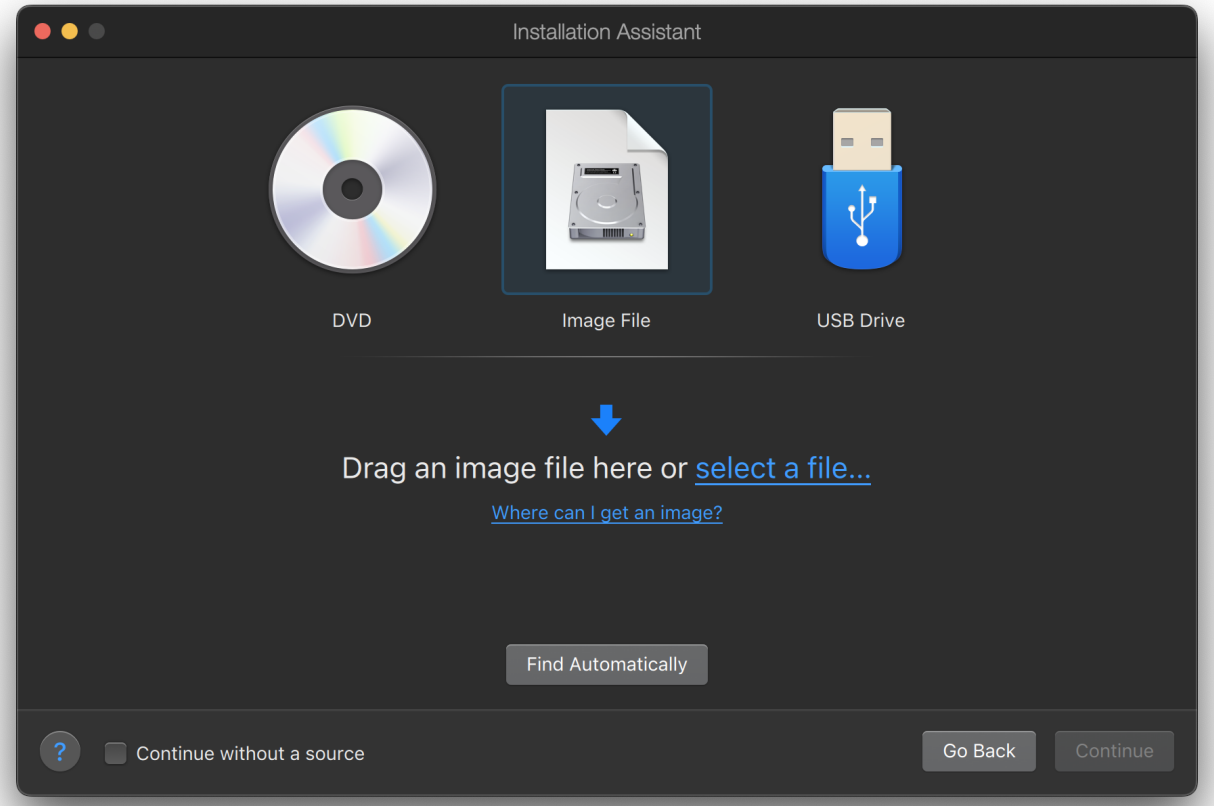
23.2.1. MacOS 上的 Parallels

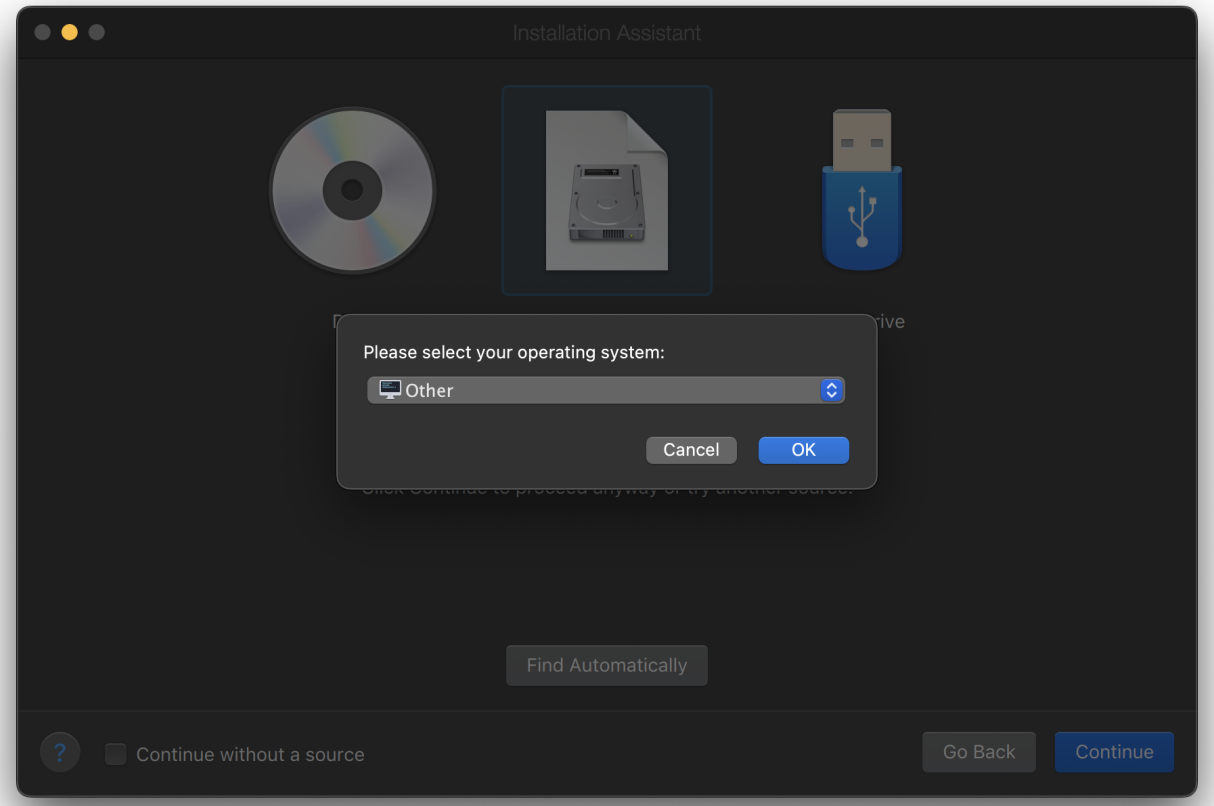
为 Mac® 设计的 Parallels Desktop 是一种可用于采用 Intel® 处理器， 并运行 Mac OS® 10.4.6
或更高版本的 Apple® Mac® 计算机的商业软件。 它为 FreeBSD 系统提供了完整的支持。 在 Mac OS® X
上安装了这个软件之后， 用户需要配置虚拟机并安装所需的客户操作系统。

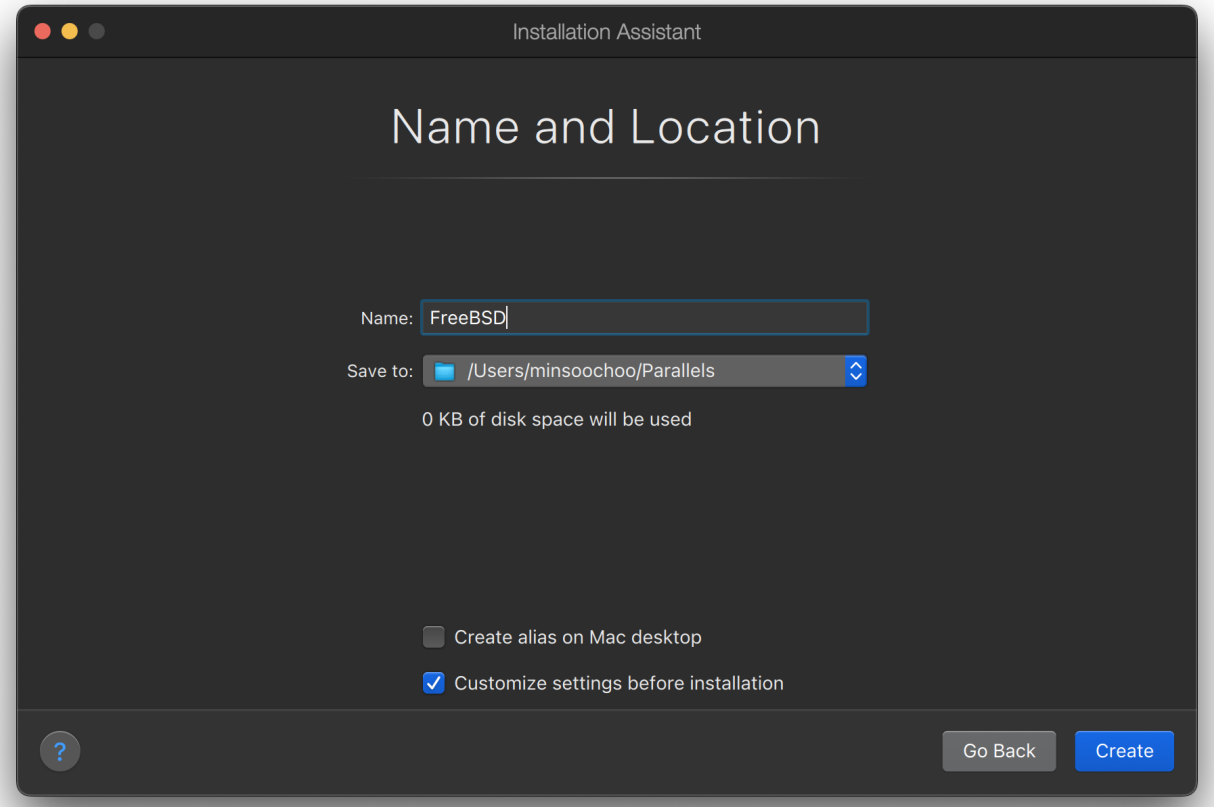
23.2.1.1. 在 Parallels/Mac OS® X 上安装 FreeBSD

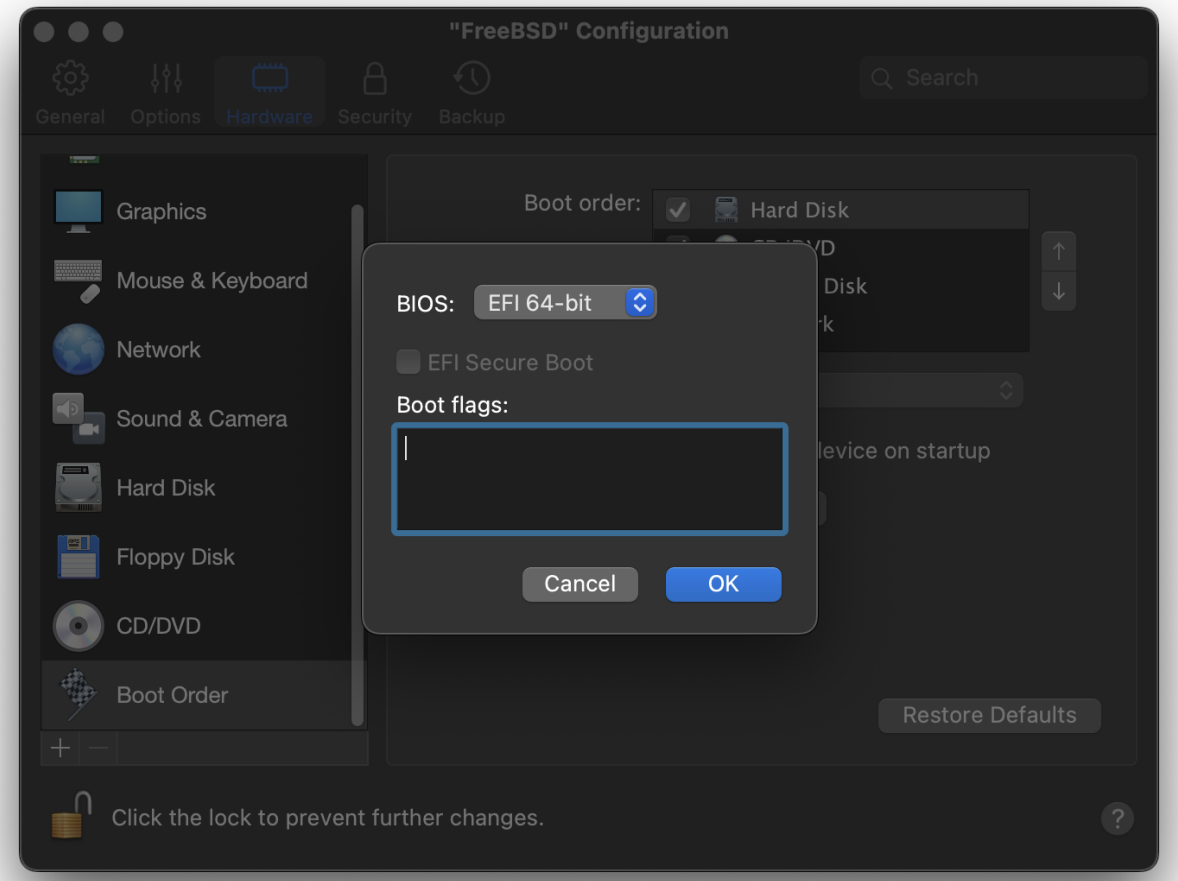
在 Mac OS® X/Parallels 上安装 FreeBSD 的第一步是创建一个新的虚拟机。 在系统提示选择客户 OS 类型
(Guest OS Type) 时选择 FreeBSD， 并根据您使用 FreeBSD 虚拟实例的需要分配磁盘和内存：

对多数在 Parallels 上使用 FreeBSD 的情形而言， 4GB 磁盘空间和 512MB 的 RAM 就够用了：

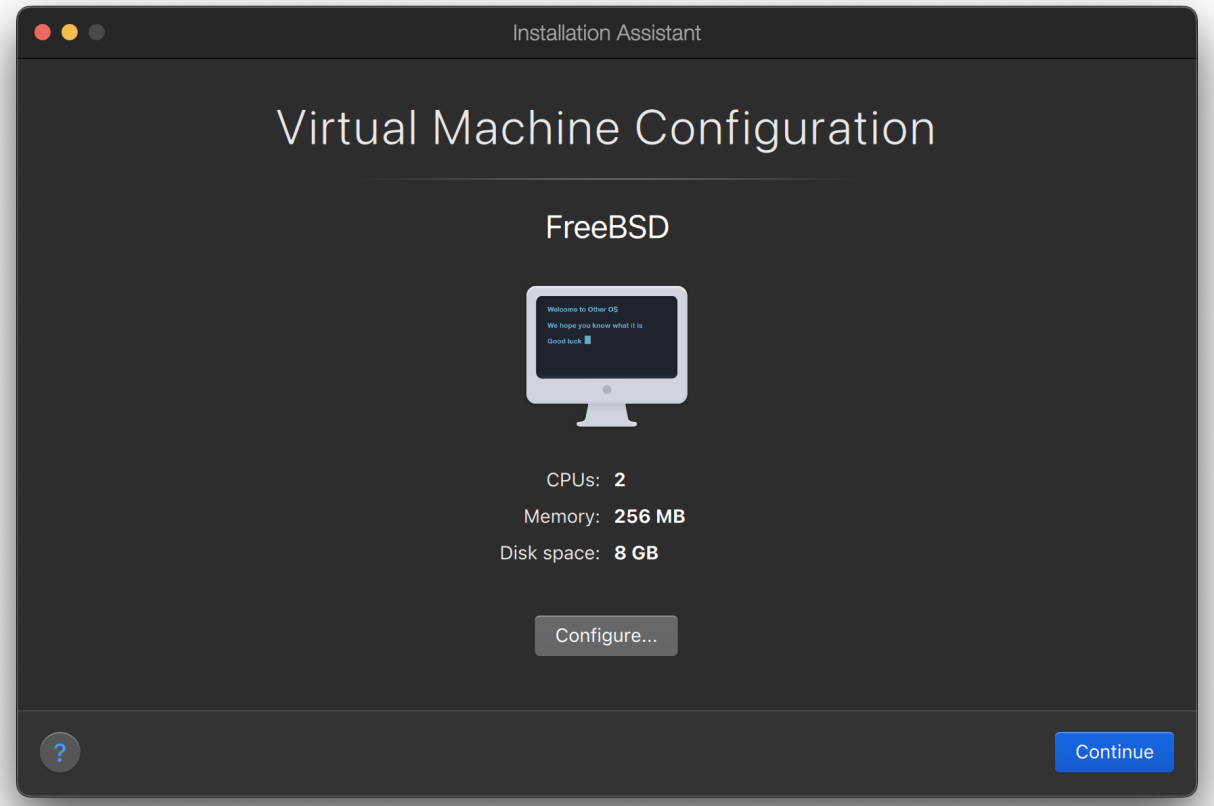



Installation Assistant
Please select your operating system:
Other
Cancel
OK
Find Automatically
Continue without a source
Go Back
Continue


Installation Assistant
Name and Location
Name:
FreeBSD
Save to:
/Users/minsoochoo/Parallels
0 KB of disk space will be used
Create alias on Mac desktop
Customize settings before installation
Go Back
Create

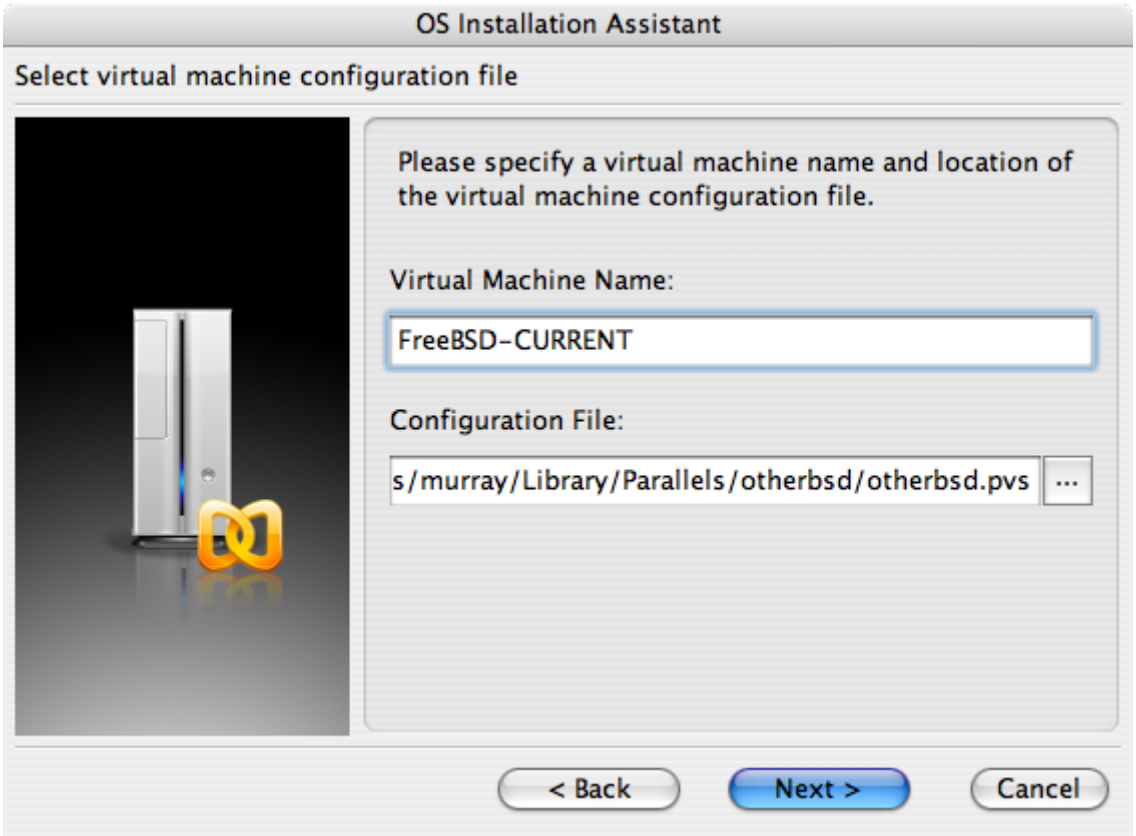


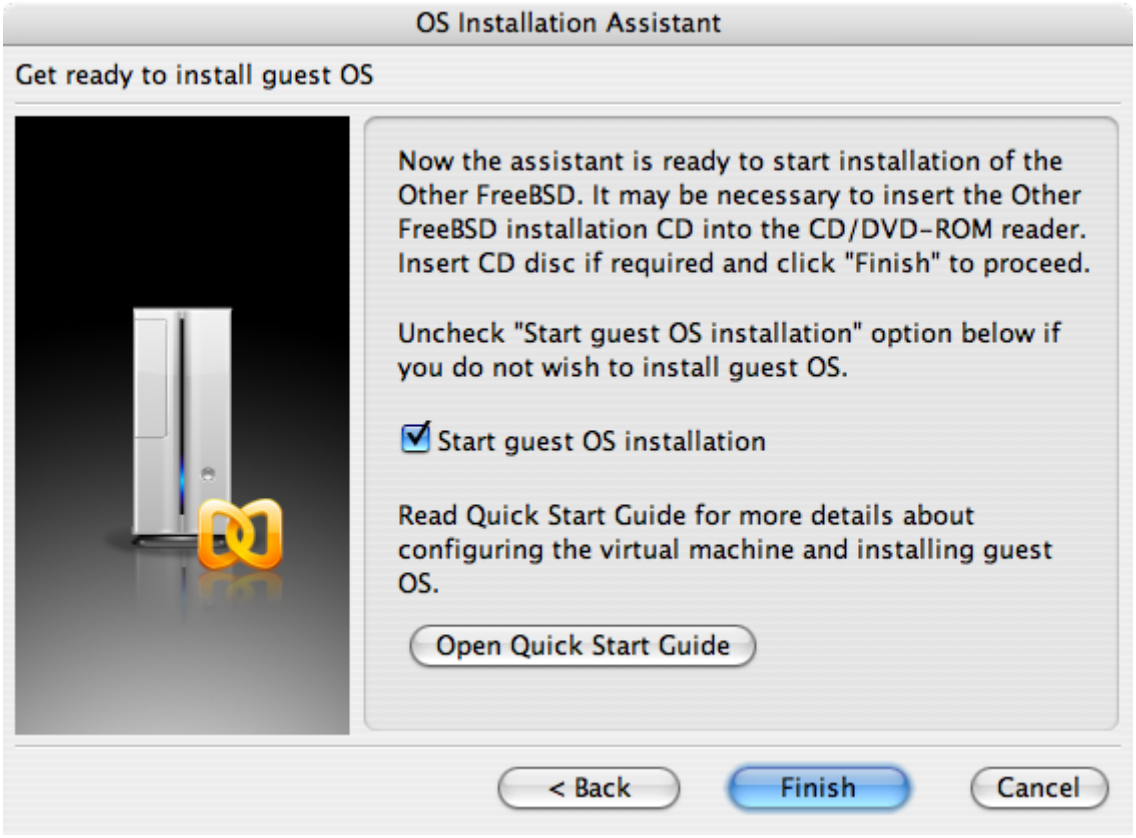
选择使用的网络和网卡类型：


Installation Assistant
Virtual Machine Configuration
FreeBSD
Welcome to Other OS
We hope you know what it is
Good luck
CPUs: 2
Memory: 256 MB
Disk space: 8 GB
Configure...
Continue

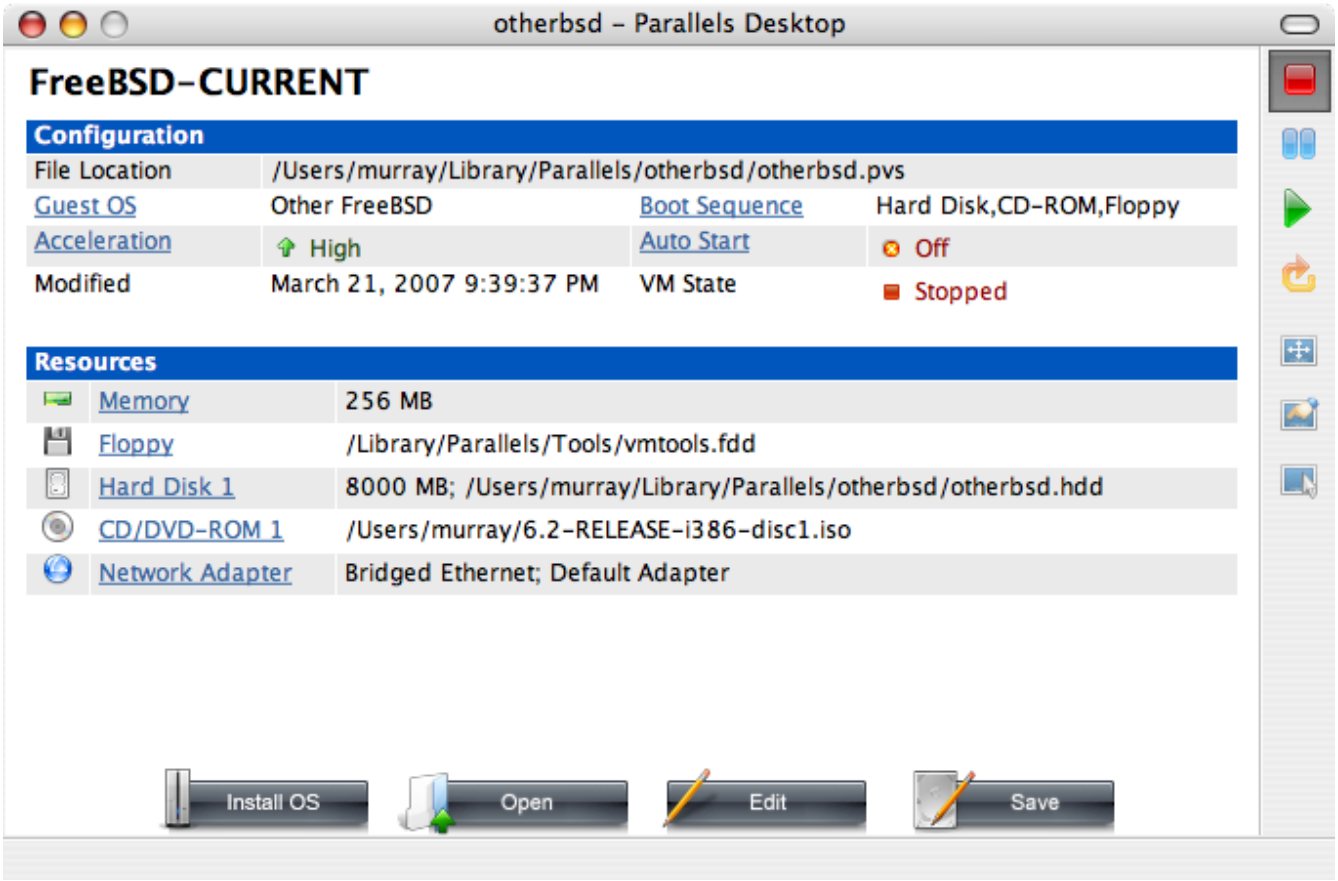

FreeBSD
FreeBSD
Welcome to FreeBSD
1. Boot Multi user [Enter]
2. Boot Single user
3. Escape to loader prompt
4. Reboot
5. Cons: Video
Options:
6. Kernel: default/kernel (1 of 1)
7. Boot Options
Autoboot in 9 seconds. [Space] to pause

保存并完成配置：


OS Installation Assistant
Select virtual machine configuration file
Please specify a virtual machine name and location of the virtual machine configuration file.
Virtual Machine Name:
FreeBSD-CURRENT
Configuration File:
s/murray/Library/Parallels/otherbsd/otherbsd.pvs
...
< Back
Next >
Cancel

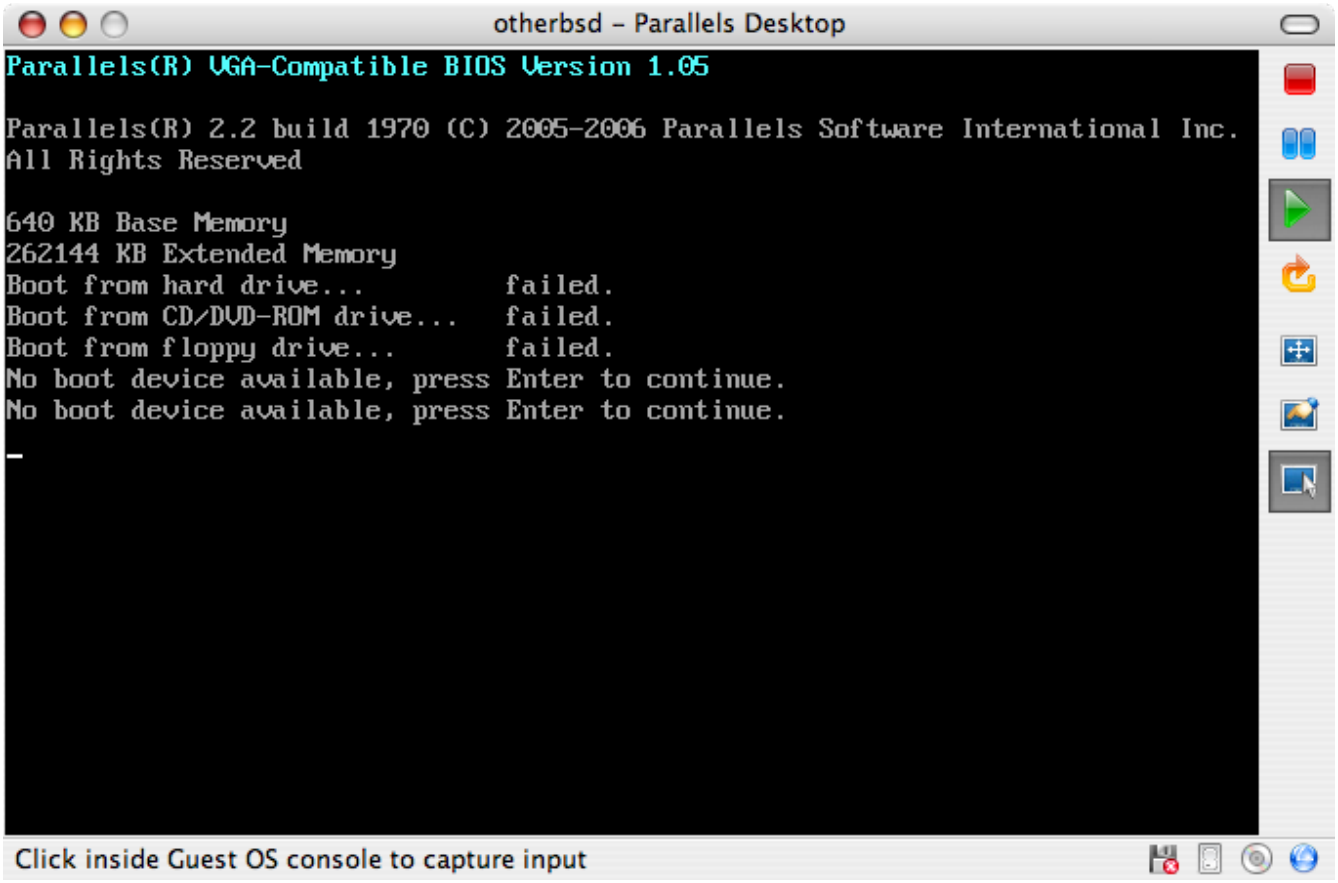


在创建了 FreeBSD 虚拟机之后， 还需要在其中安装 FreeBSD。 最好的做法是使用官方的 FreeBSD CDROM 或从官方 FTP 站点下载的 ISO 镜像来完成这个任务。 如果您的本地 Mac® 文件系统中有 ISO 映像文件，或您的 Mac® 的 CD 驱动器中有 CDROM， 就可以在 FreeBSD Parallels 窗口的右下角点击光盘图标。 之后， 系统将给出一个窗口， 供您完成将虚拟机中的 CDROM 驱动器连接到本地的 ISO 文件或真正的 CDROM 驱动器上。

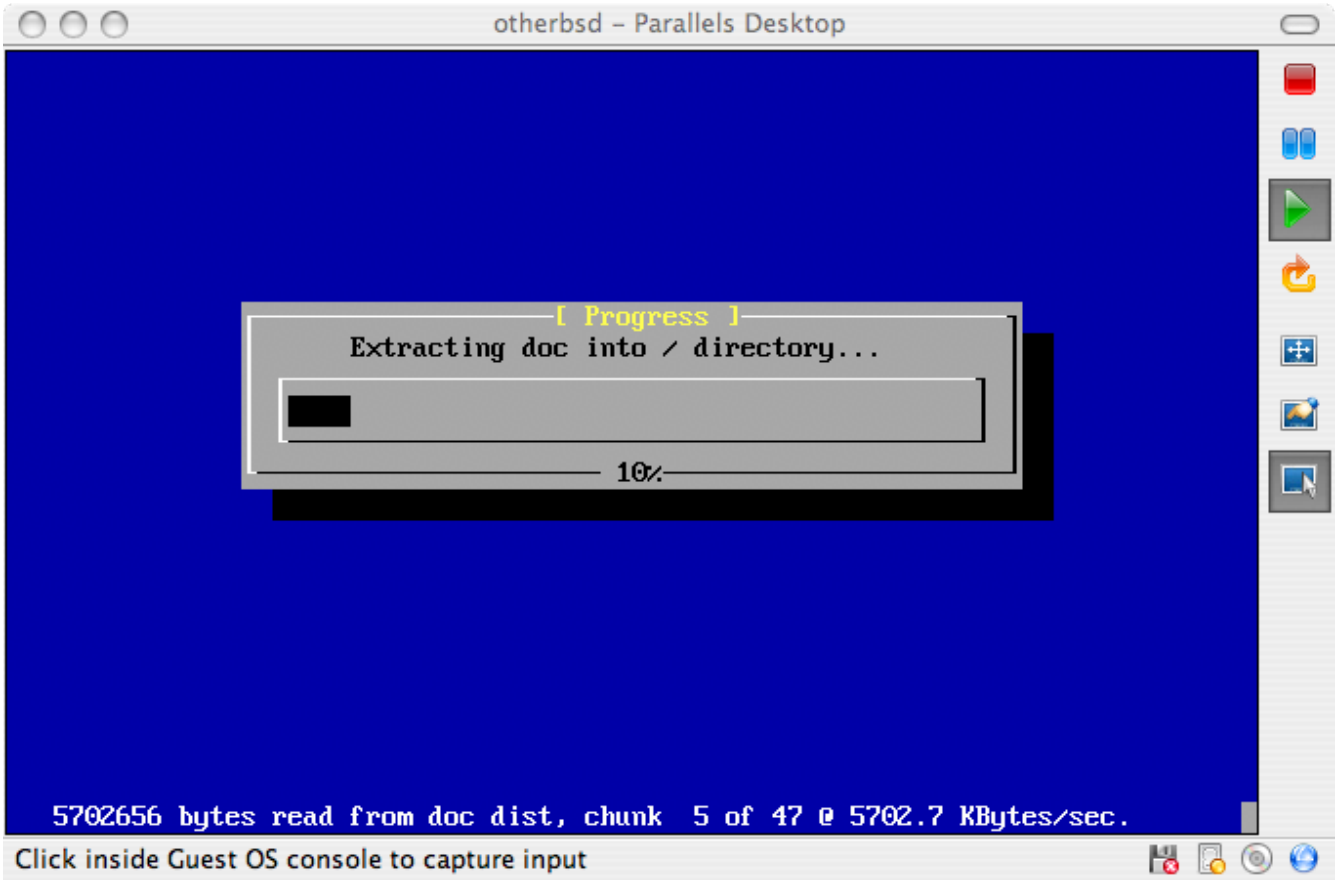


在完成了将 CDROM 与您的安装源完成关联之后， 就可以按重启 (reboot) 图标来重启 FreeBSD 虚拟机了。

Parallels 将配合一个特殊的 BIOS 启动， 后者能够像普通的 BIOS 一样检查系统中是否有 CDROM 驱动器。

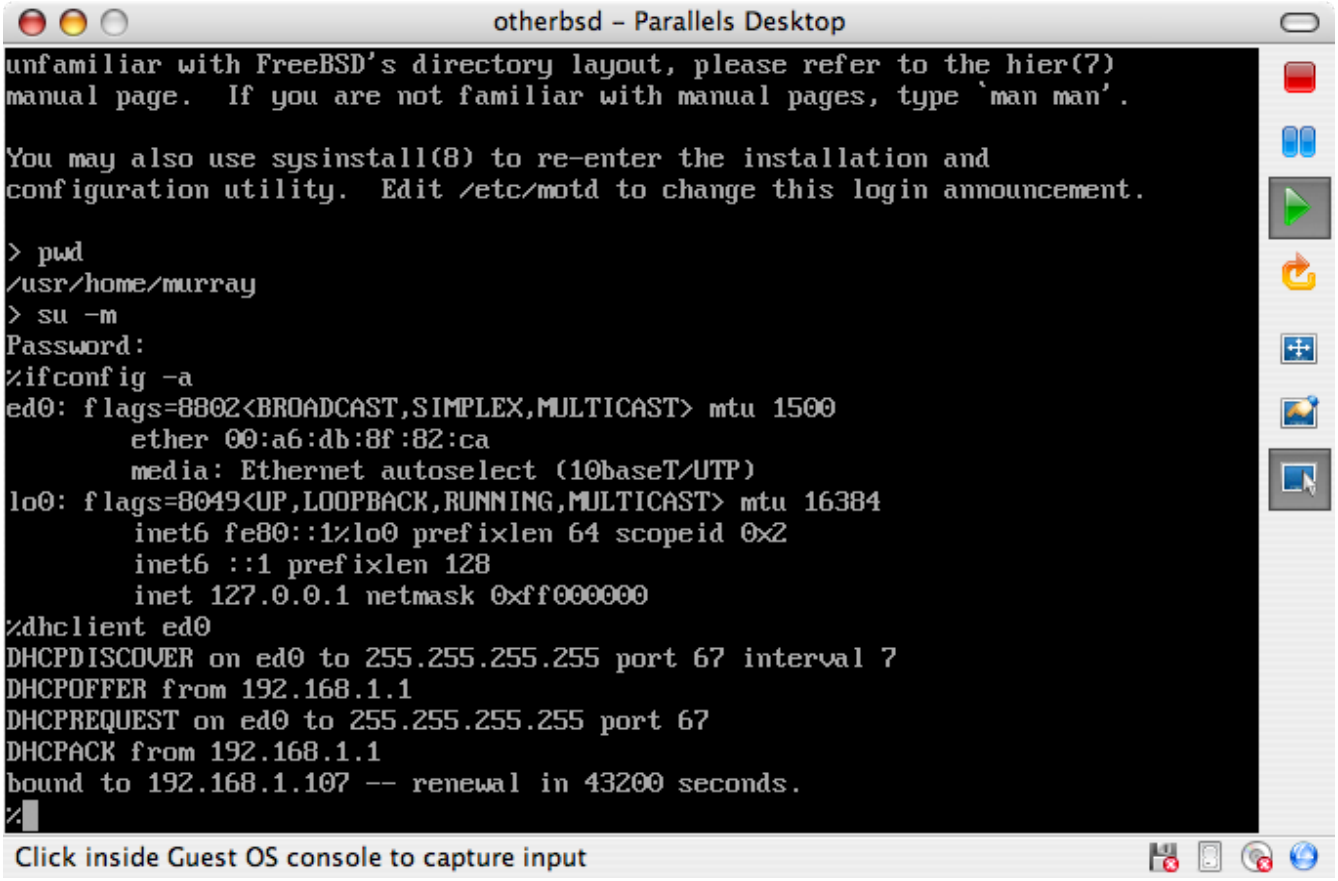


此时， 它就能够找到 FreeBSD 安装介质并开始 安装 FreeBSD 中所介绍的标准的基于 sysinstall 安装的过程。



此时您可以安装 X11， 但暂时不要对它进行配置。 在完成安装之后， 重启并进入新安装的 FreeBSD

虚拟机。



23.2.1.2. 在 Mac OS® X/Parallels 上配置 FreeBSD

在您将 FreeBSD 安装到 Mac OS® X 的 Parallels 上之后， 还需要进行一系列的配置，
以便为系统的虚拟化操作进行优化。

1. 配置引导加载器变量

   最重要的一步是通过调低 kern.hz 变量来降低 Parallels 环境中的 FreeBSD 对 CPU 的使用。
   这可以通过在 /boot/loader.conf 中增加下述配置来完成：

   ```
   kern.hz=100
   ```

   如果不使用这个配置， 闲置的 FreeBSD Parallels 客户 OS 会在单处理器的 iMac® 上使用大约 15%
   的 CPU。 如此修改之后， 空闲时的使用量就减少到大约 5% 了。

2. 创建新的内核配置文件

   您可以删去全部 SCSI、 FireWire， 以及 USB 设备驱动程序。 Parallels 提供了一个由 ed(4)
   驱动的虚拟网卡， 因此， 除了 ed(4) 和 miibus(4) 之外的其他网络接口驱动都可以从内核中删去。

3. 配置网络

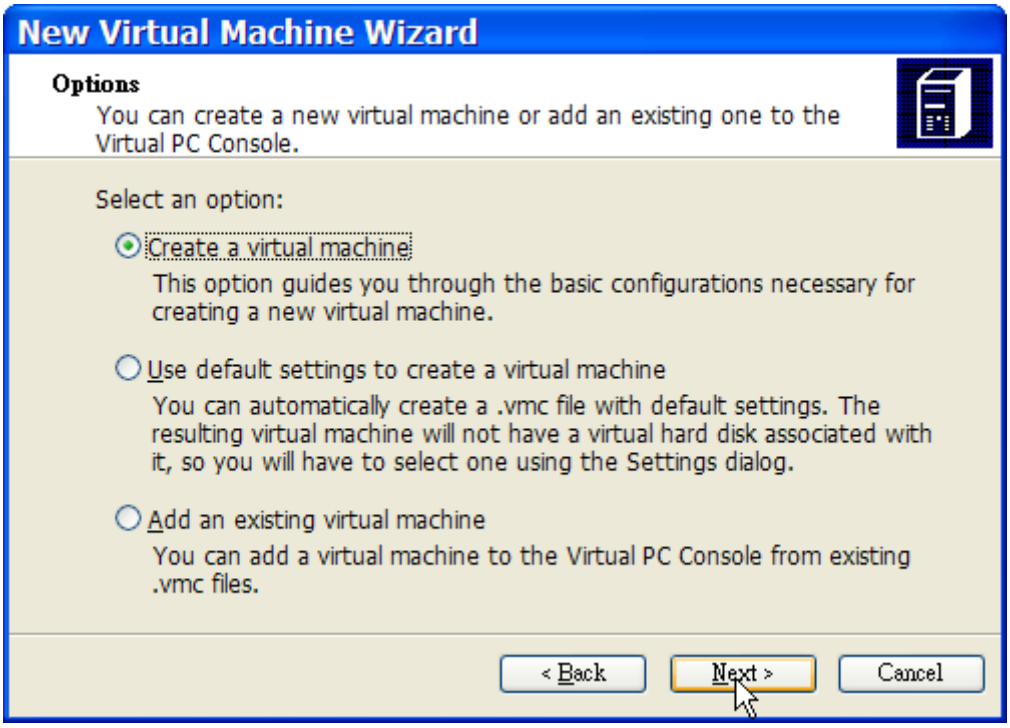
   最基本的网络配置， 是通过使用 DHCP 来将您的虚拟机与宿主 Mac® 接入同一个局域网。
   这可以通过在 /etc/rc.conf 中加入 ifconfig_ed0="DHCP" 来完成。 更高级一些的网络配置方法，
   请参见 高级网络 中的介绍。

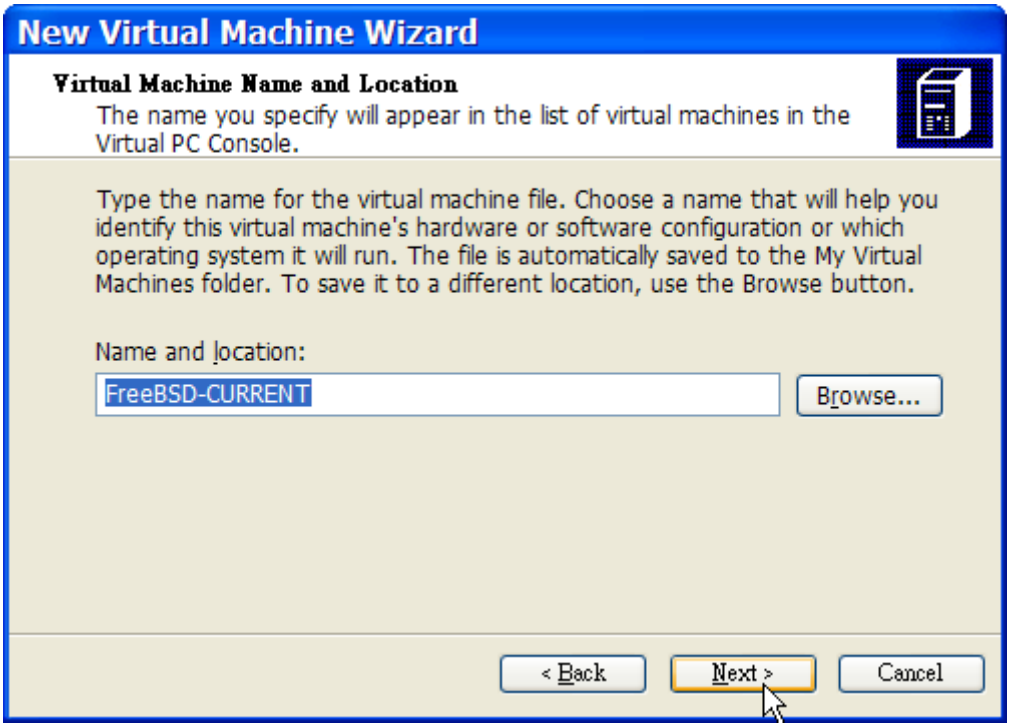
23.2.2. Windows® 上的 Virtual PC

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，请参阅 system requirements 说明。 在 Microsoft® Windows® 装完 Virtual PC 之后，
必须针对所安装的虚拟机器来做相应设定。

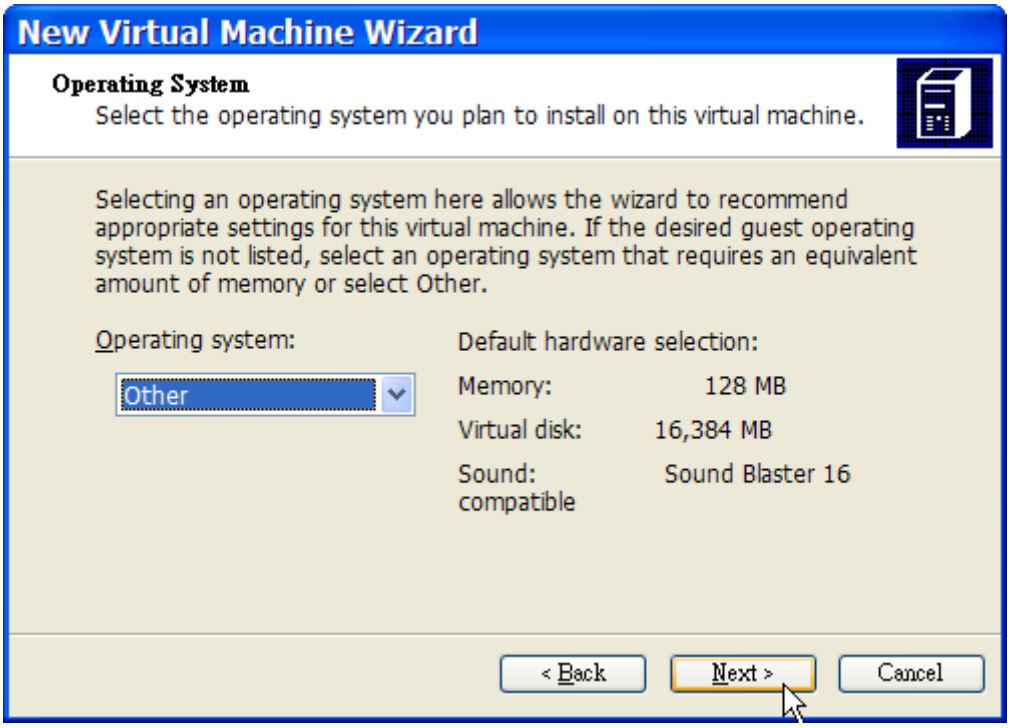
23.2.2.1. 在 Virtual PC/Microsoft® Windows® 上安装 FreeBSD

在 Microsoft® Windows®/Virtual PC 上安装 FreeBSD 的第一步是新增虚拟器。
如下所示，在提示向导中请选择 Create a virtual machine：

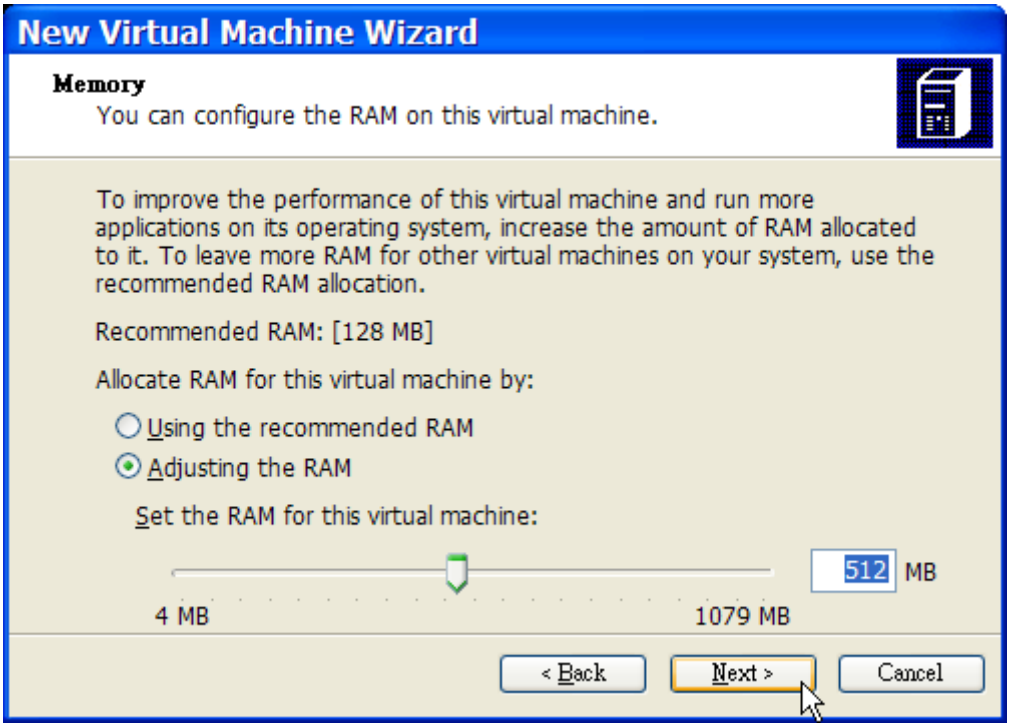


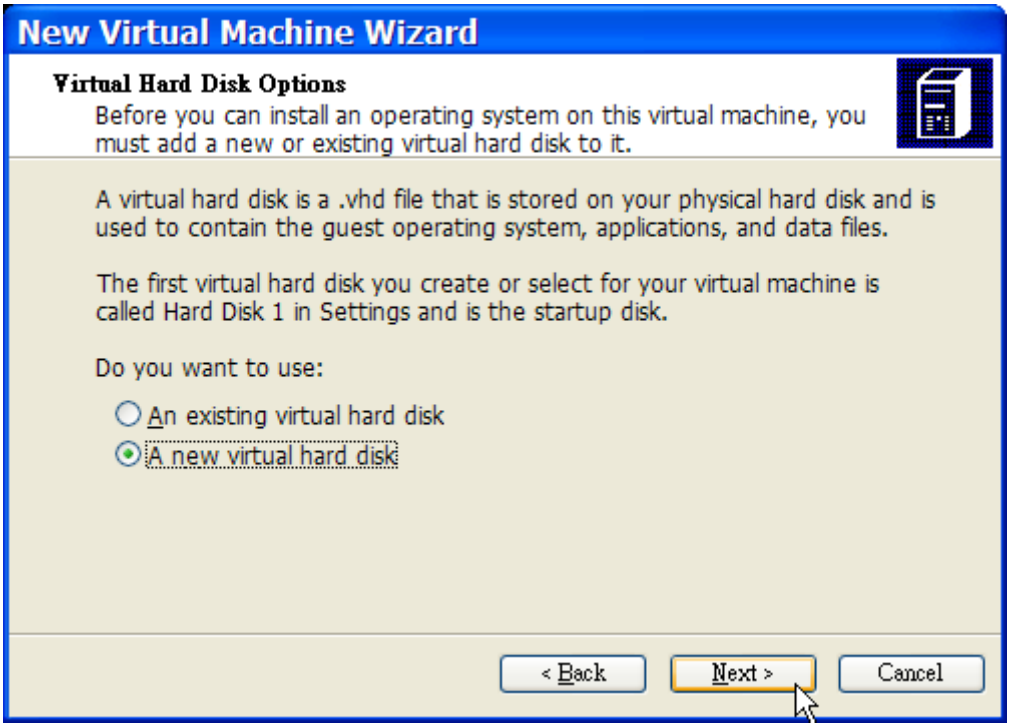


然后在 Operating system 处选 Other：

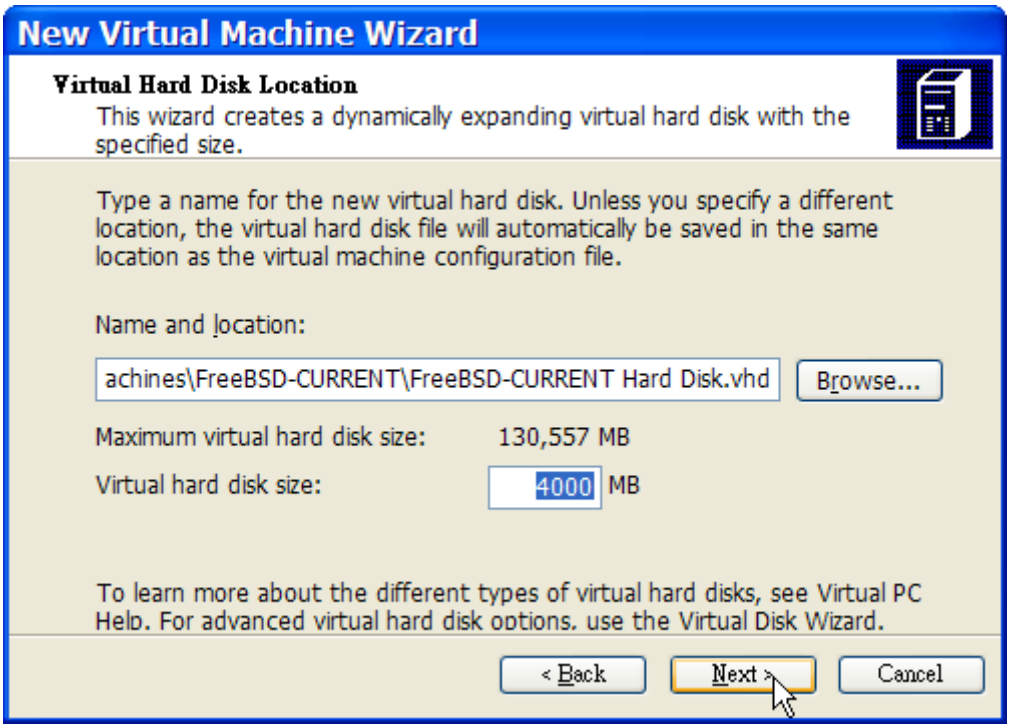


并依据自身需求来规划硬盘容量和内存的分配。对大多数在 Virtual PC 使用 FreeBSD 的情况而言， 大约 4GB 的硬盘空间以及 512MB 的内存就够用了。

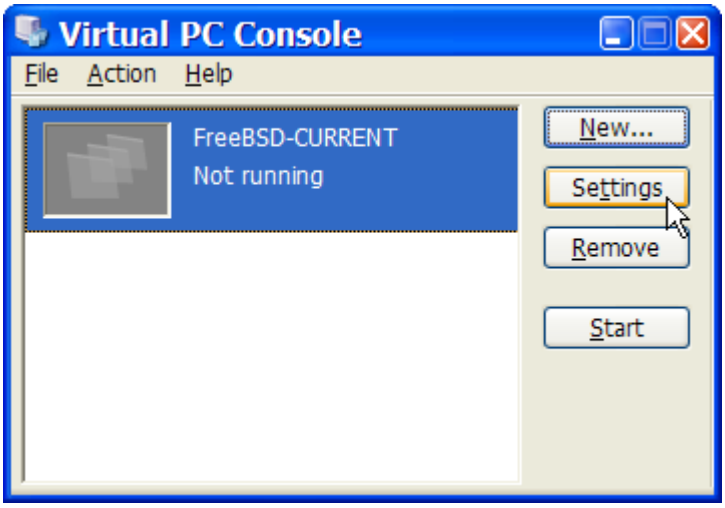


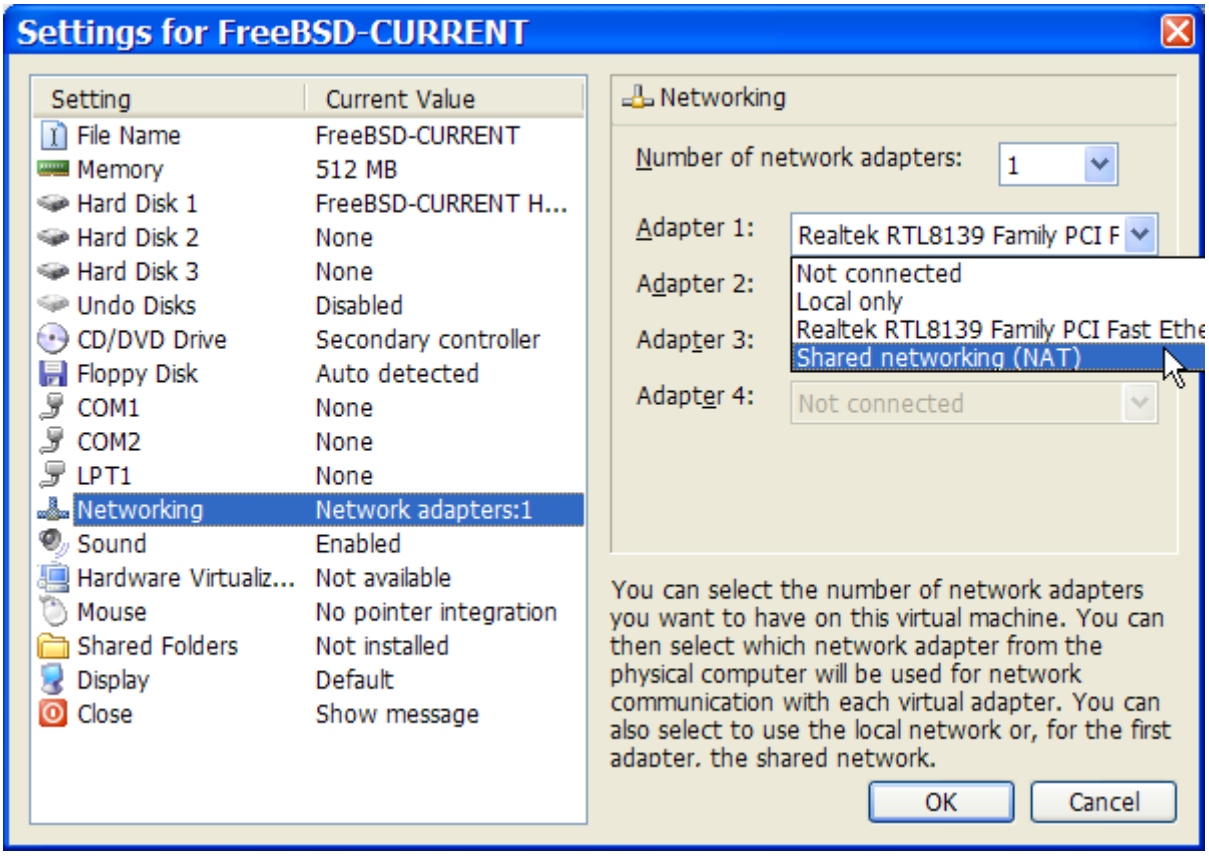


保存并完成配置：

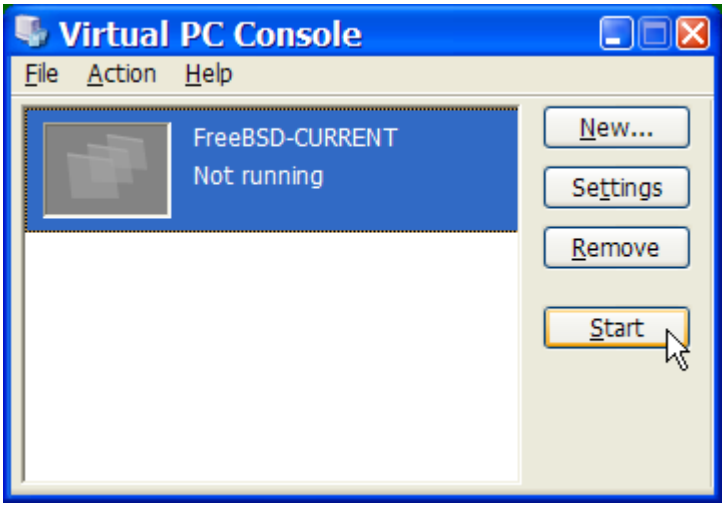


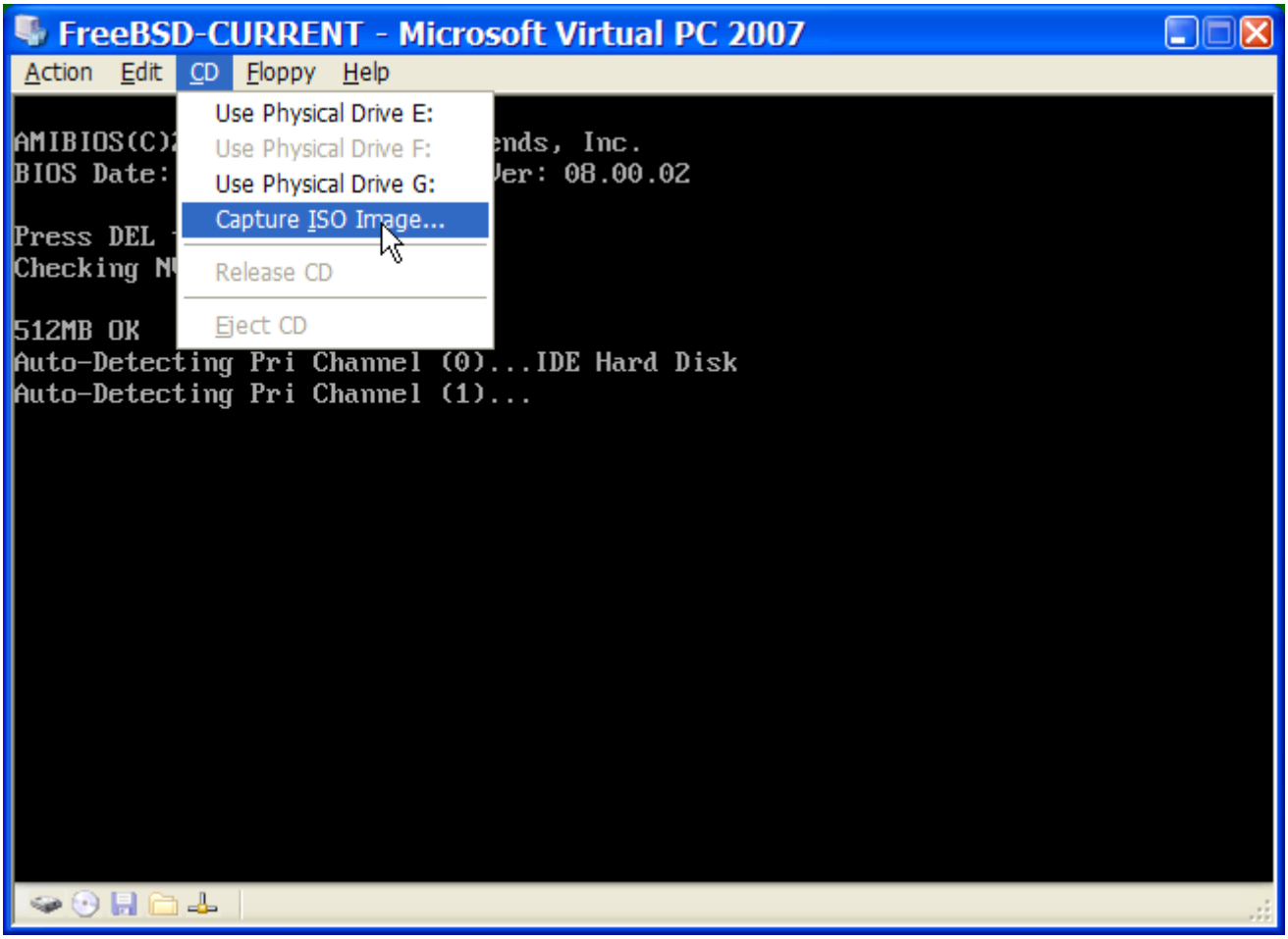
接下来选择新建的 FreeBSD 虚拟机器，并单击 Settings， 以设定网络种类以及网卡：



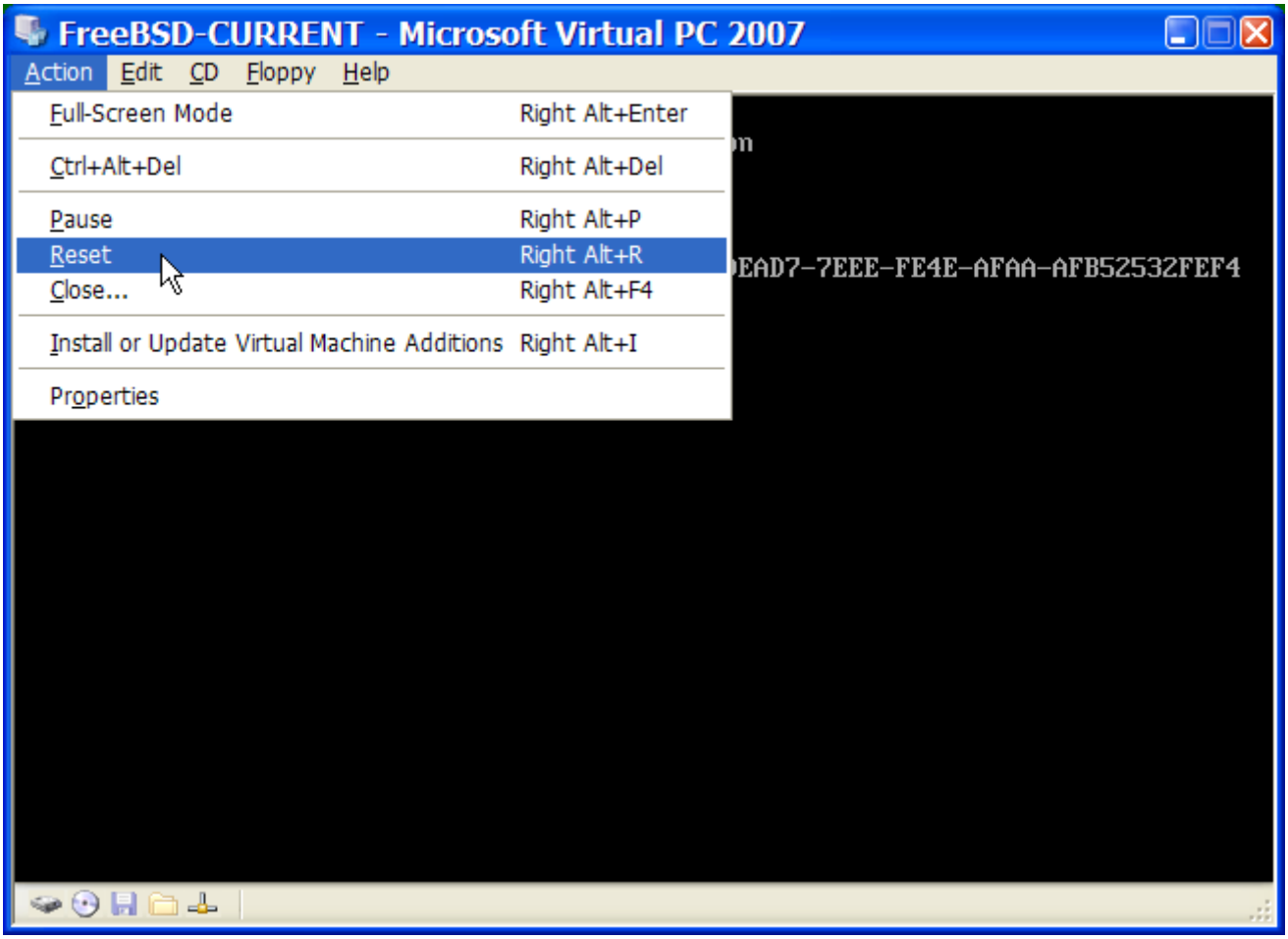


在新建 FreeBSD 虚拟机器以后， 就可以继续以其安装 FreeBSD。 安装方面， 比较好的作法是使用官方的 FreeBSD 光盘或从官方 FTP 站下载 ISO 镜像。 若您的 Windows® 系统 内已有该 ISO 镜像， 那么就可以在 FreeBSD 虚拟机器上双击， 以开始启动。 接着在 Virtual PC 窗口内按 CD 再按 Capture ISO Image…。 接着出现一个对话框， 可以把虚拟机器内的光驱设定到该 ISO 镜像， 或者是真实的光驱。

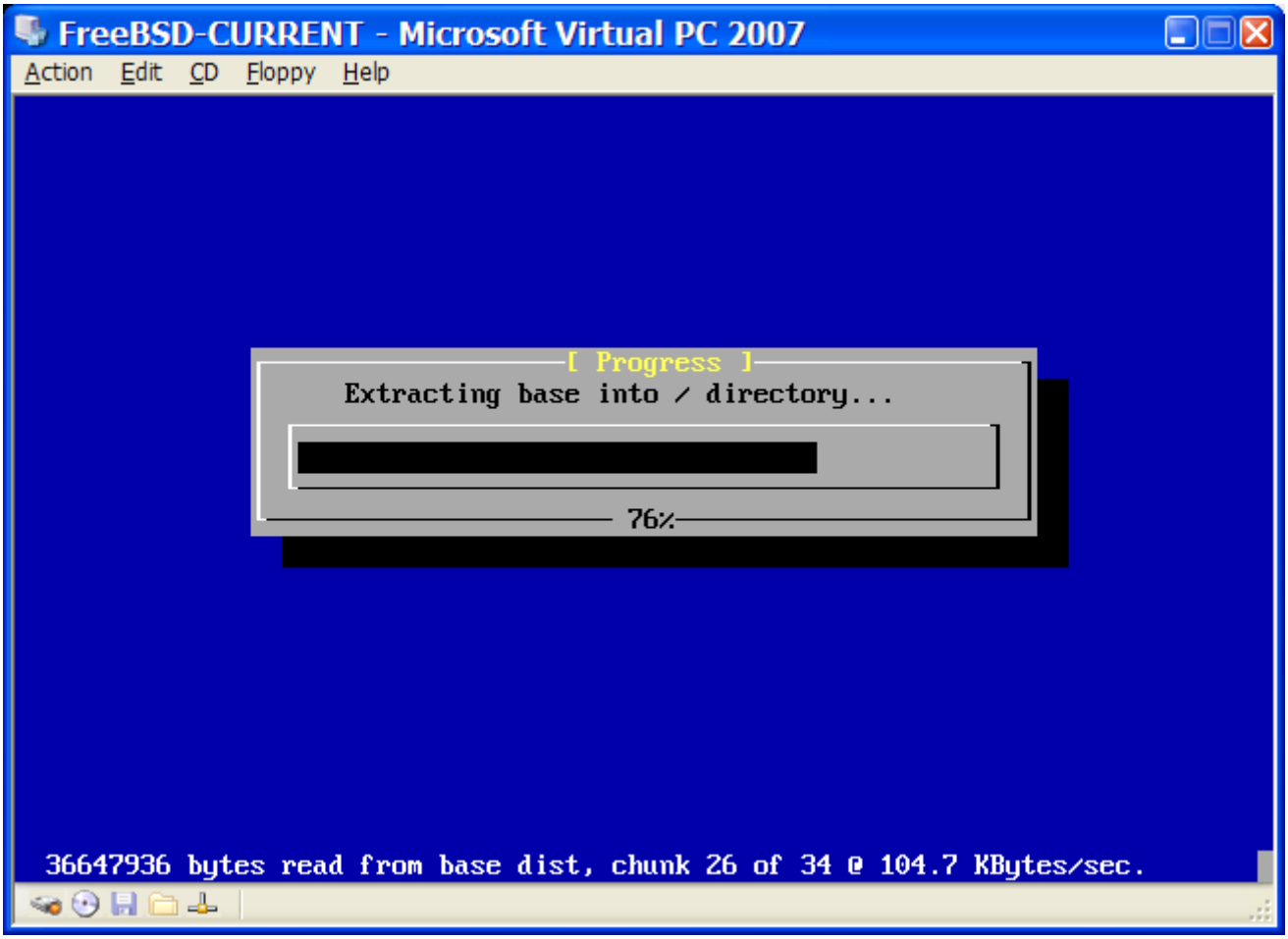




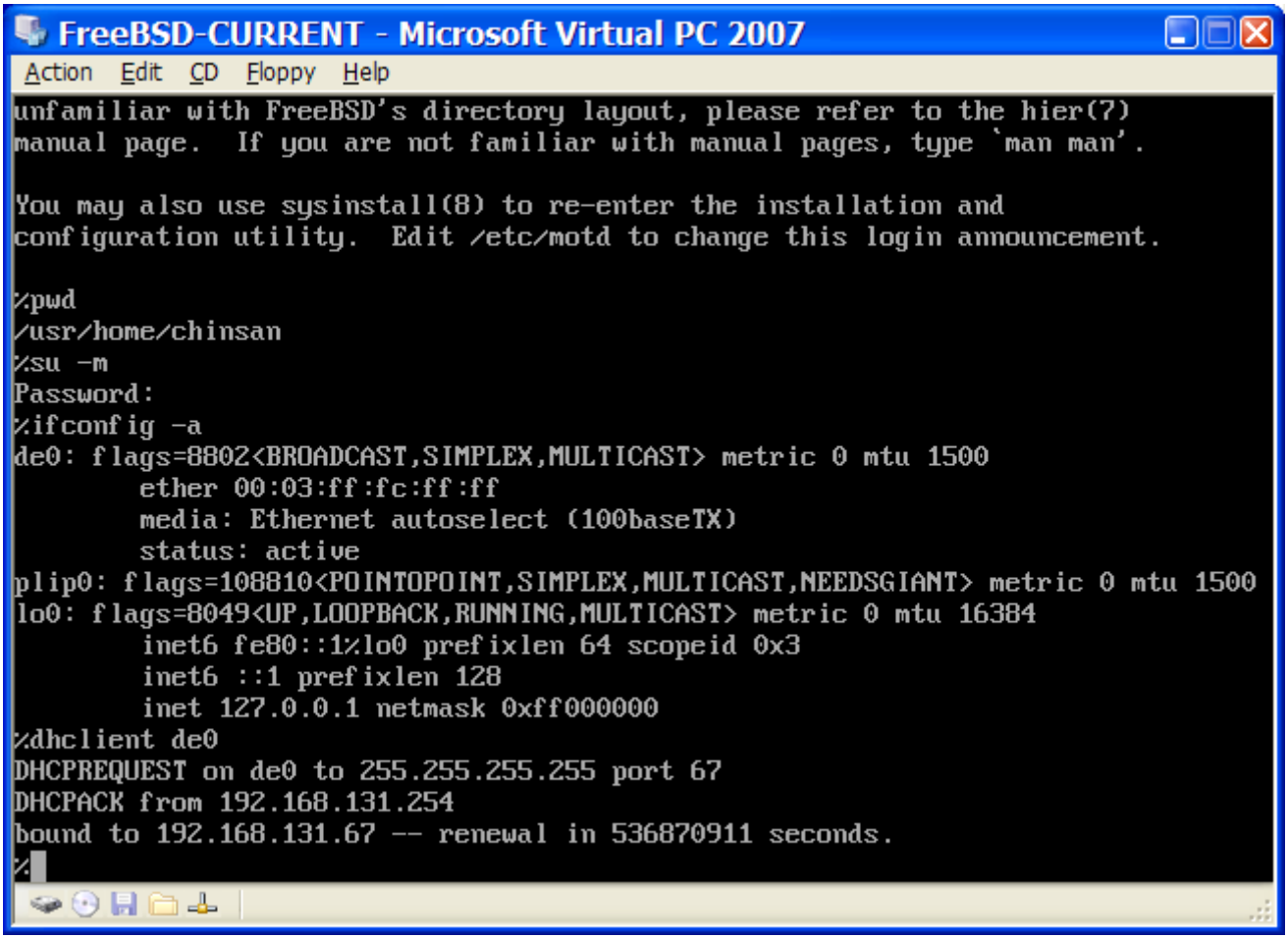
设好光盘来源之后，就可以重新开机， 也就是先按 Action 再按 Reset 即可。 Virtual PC 会以特殊 BIOS 开机， 并与普通 BIOS 一样会先检查是否有光盘驱动器。



此时， 它会找到 FreeBSD 安装光盘， 并开始在 安装 FreeBSD 内所介绍的 sysinstall 安装过程。
这时候也可以顺便安装 X11， 但不要进行相关设定。



完成安装之后， 记得把安装光盘或者 ISO 镜像退出。 最后， 把装好的 FreeBSD 虚拟机器重新开机即可。



23.2.2.2. 调整 Microsoft® Windows®/Virtual PC 上的 FreeBSD

在 Microsoft® Windows® 上以 Virtual PC 装好 FreeBSD 后， 还需要做一些设定步骤， 以便将虚拟机内的 FreeBSD 最佳化。

1. 设定 boot loader 参数

   最重要的步骤乃是藉由调降 kern.hz 来降低 Virtual PC 环境内 FreeBSD 的 CPU 占用率。 在 /boot/loader.conf 内加上下列设定即可：

   ```
   kern.hz=100
   ```

   若不作这设定， 那么光是 idle 状态的 FreeBSD Virtual PC guest OS 就会在单一处理器的电脑上大约有 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 设备驱动都移除。 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. 设定网络

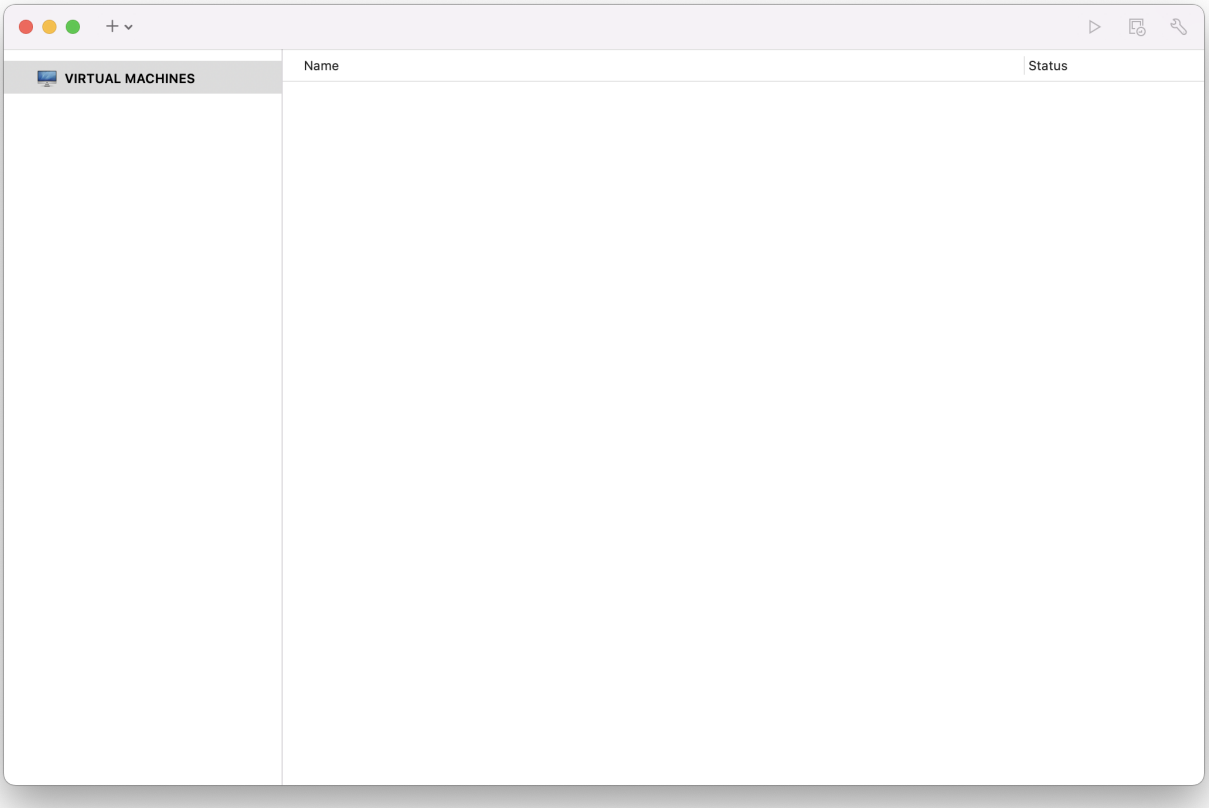
   可以给虚拟机器简单得使用 DHCP 来设定与 host (Microsoft® Windows®) 相同的本地网络环境， 只要在 /etc/rc.conf 加上 ifconfig_de0="DHCP" 即可完成。 其他的高级网络设置， 可参阅 高级网络.

23.2.3. 运行于 MacOS 的 VMware

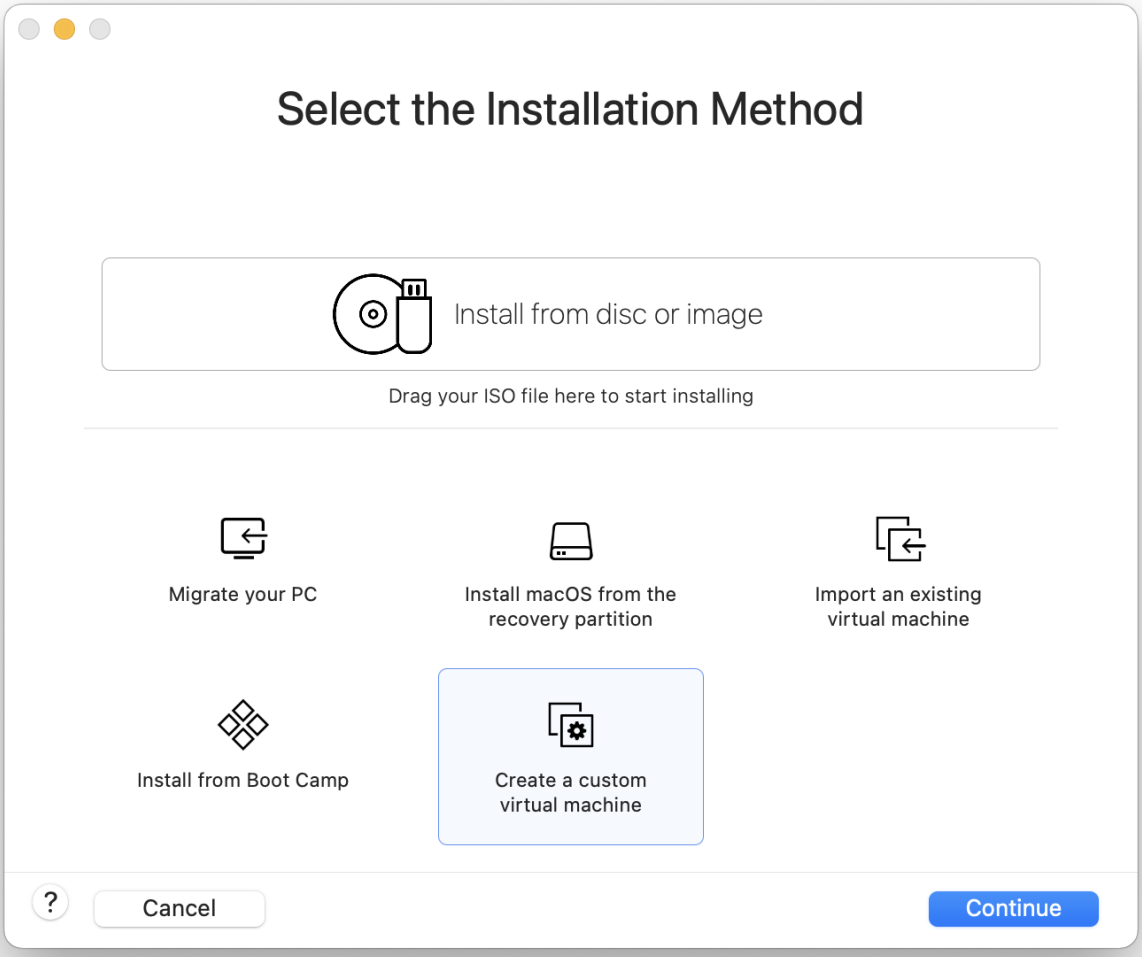
Mac® 版本的 VMware Fusion 是一个商业软件，运行在基于 Intel® 的 Apple® Mac® 计算机的 Mac OS® 10.4.9 或更版本的操作系统上。 FreeBSD 是一个完全被支持的客户操作系统。 在 Mac OS® X 上安装了 VMware Fusion 之后， 用户就可以着手配置一个虚拟机器并安装客户操作系统。

23.2.3.1. 在 VMware/Mac OS® X 上安装 FreeBSD

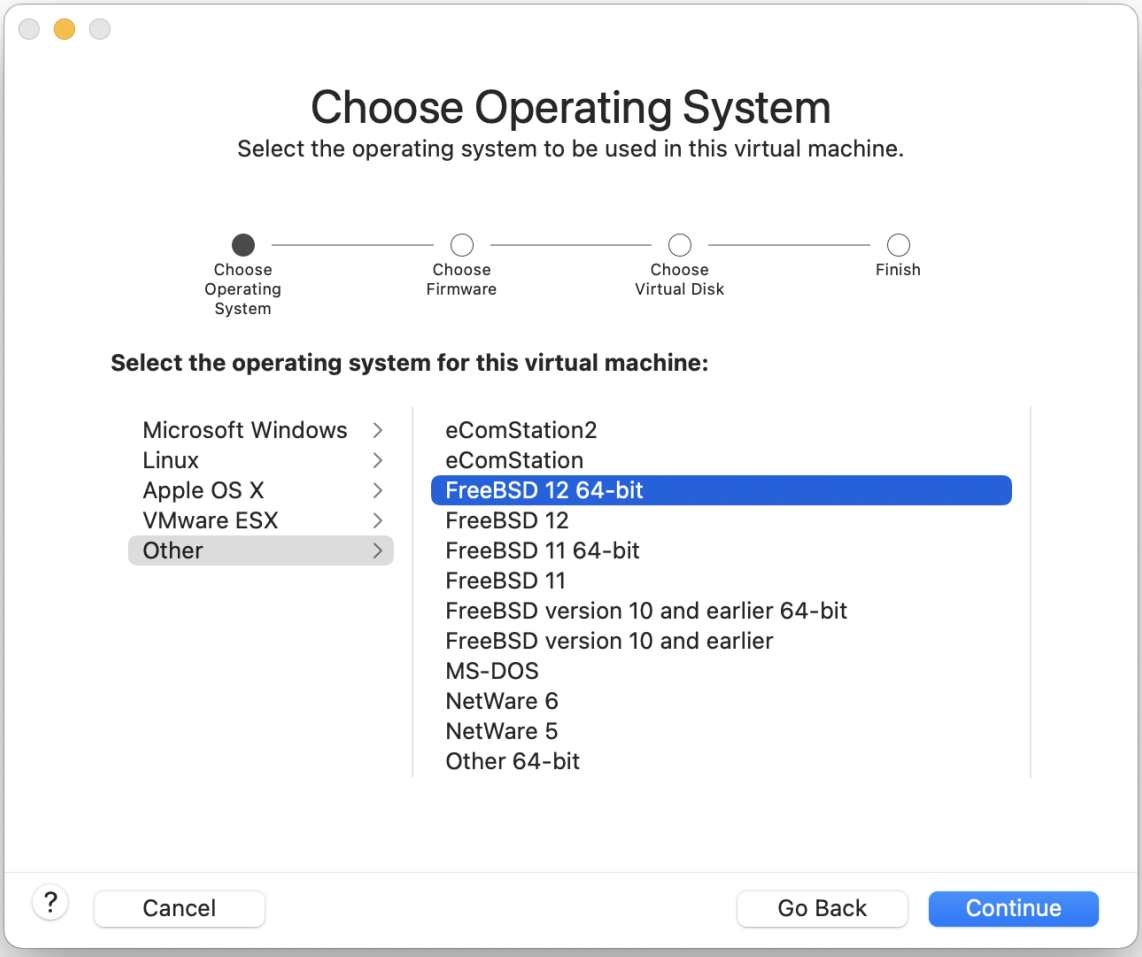
第一步是运行 VMware Fusion， 虚拟机器库将被装载。 单击 "New" 创建 VM：



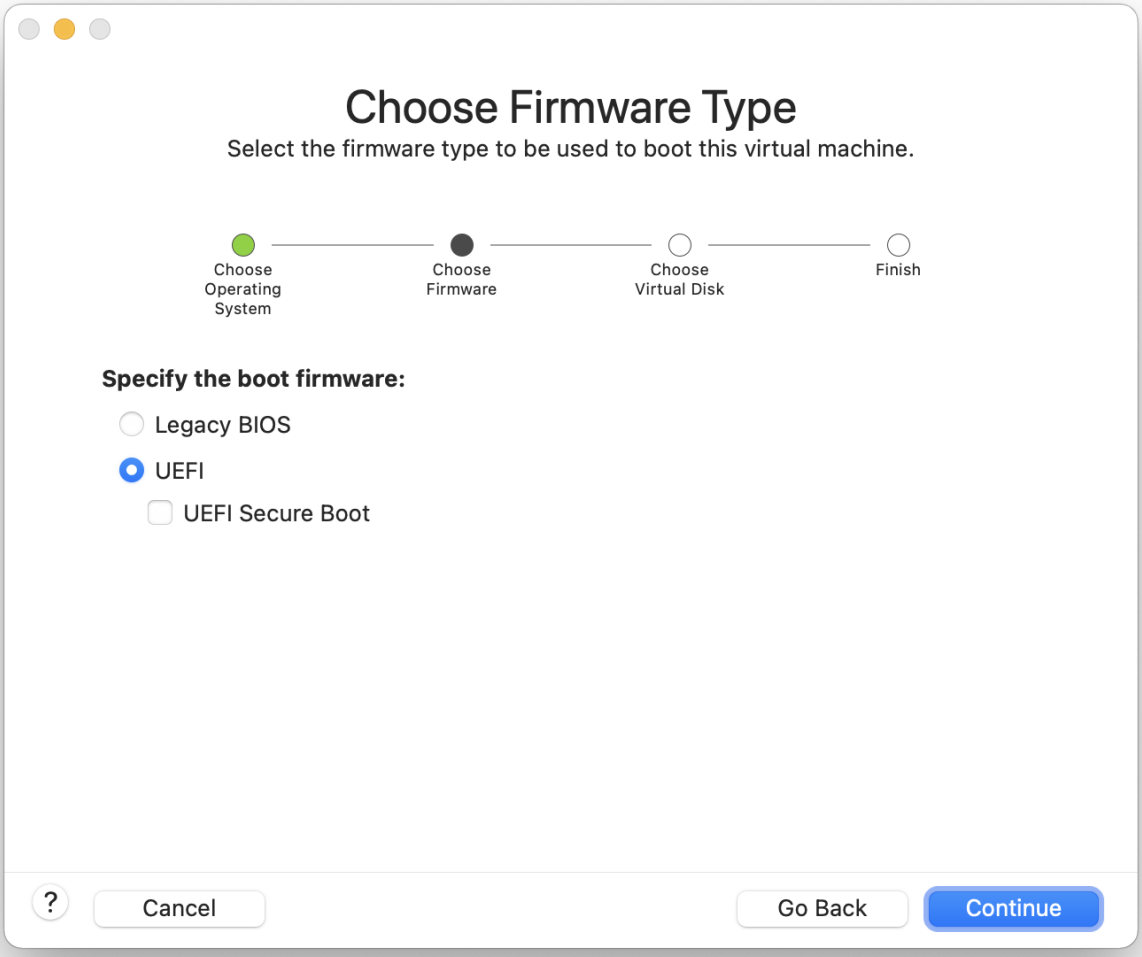
New Virtual Machine Assistant 将被运行来帮助你创建 VM， 单击 Continue 继续：



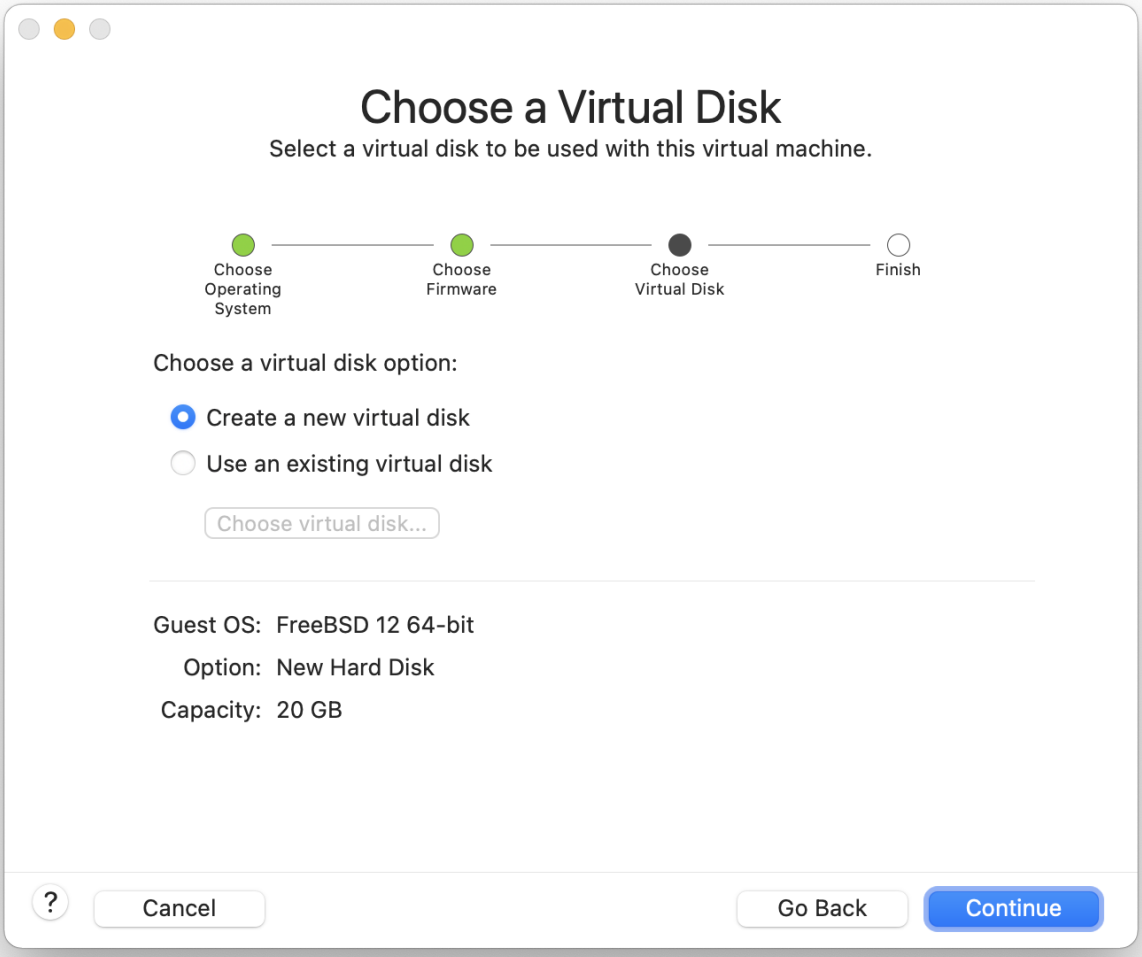
在 Operatiing System 项选择 Other，Version 项可选 FreeBSD 或 FreeBSD 64-bit。



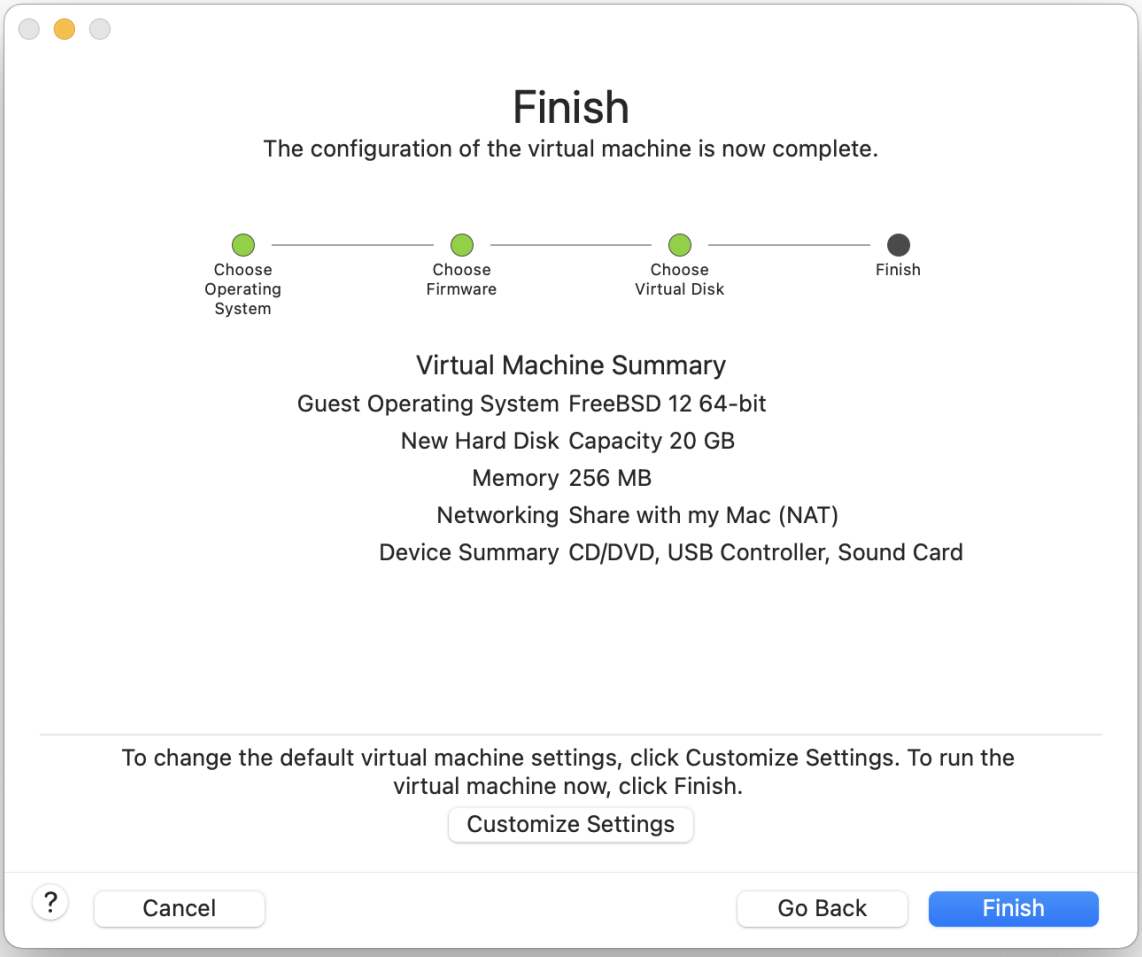
选一个你想要的 VM 镜像名字和存储的目录位置。



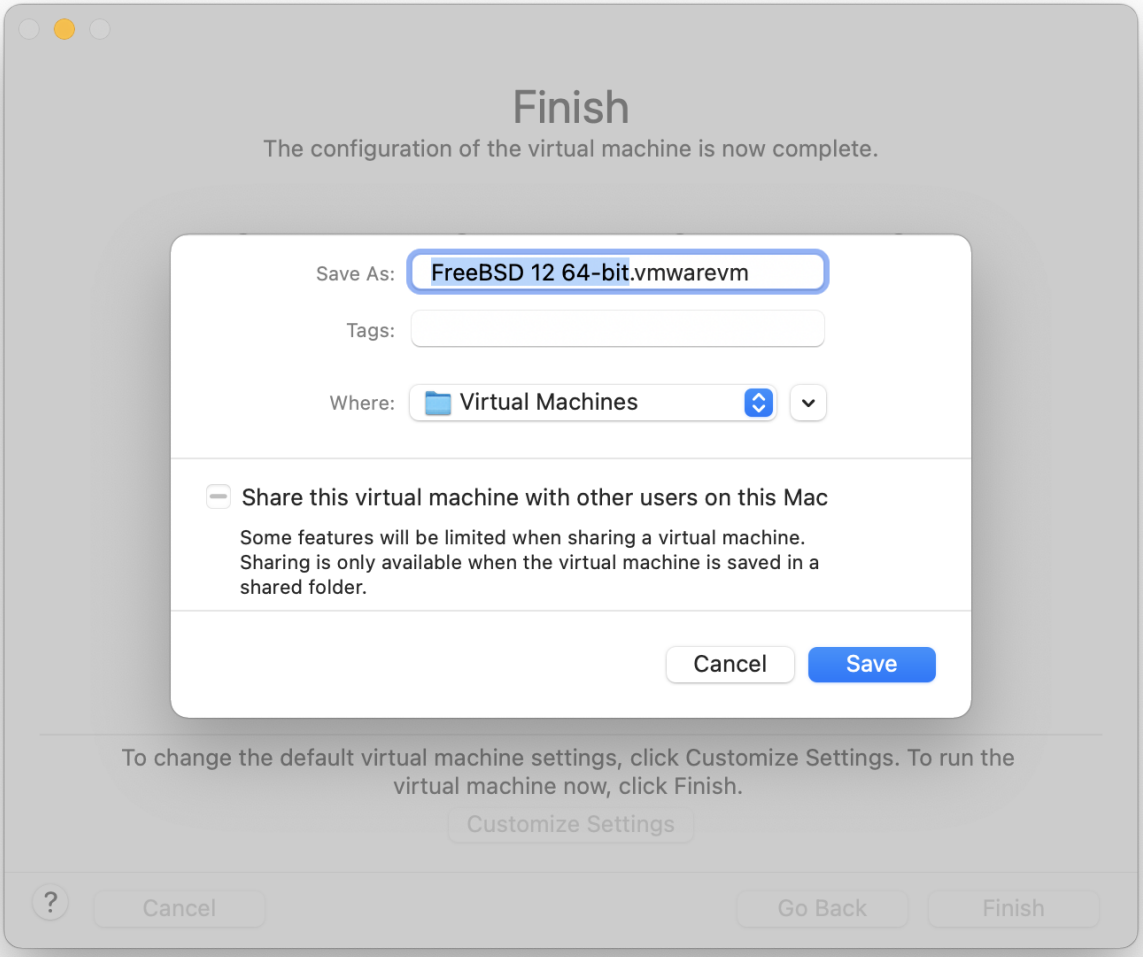
选择 VM 虚拟硬盘的大小：



选择安装 VM 的方式，从一个 ISO 镜像或一张 CD 安装：



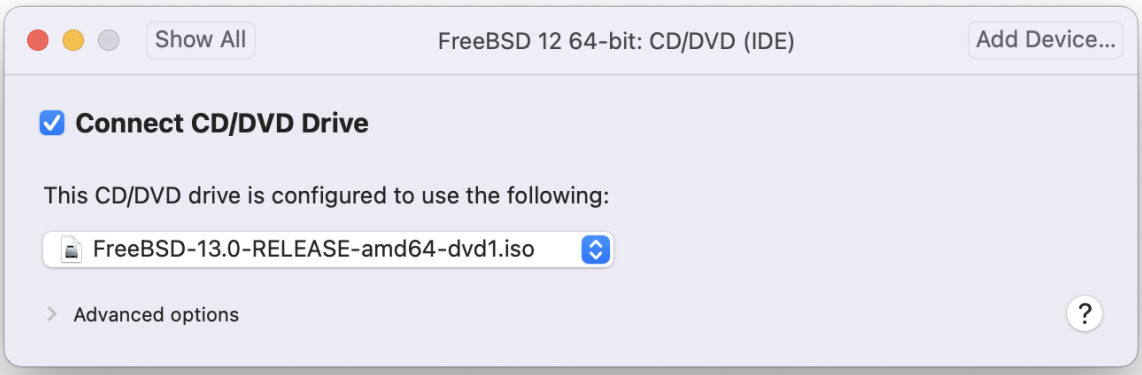
一旦你点击了 Finish， VM 就会启动了：



以你通常的方式安装 FreeBSD 或者参照 安装 FreeBSD 中的步骤：

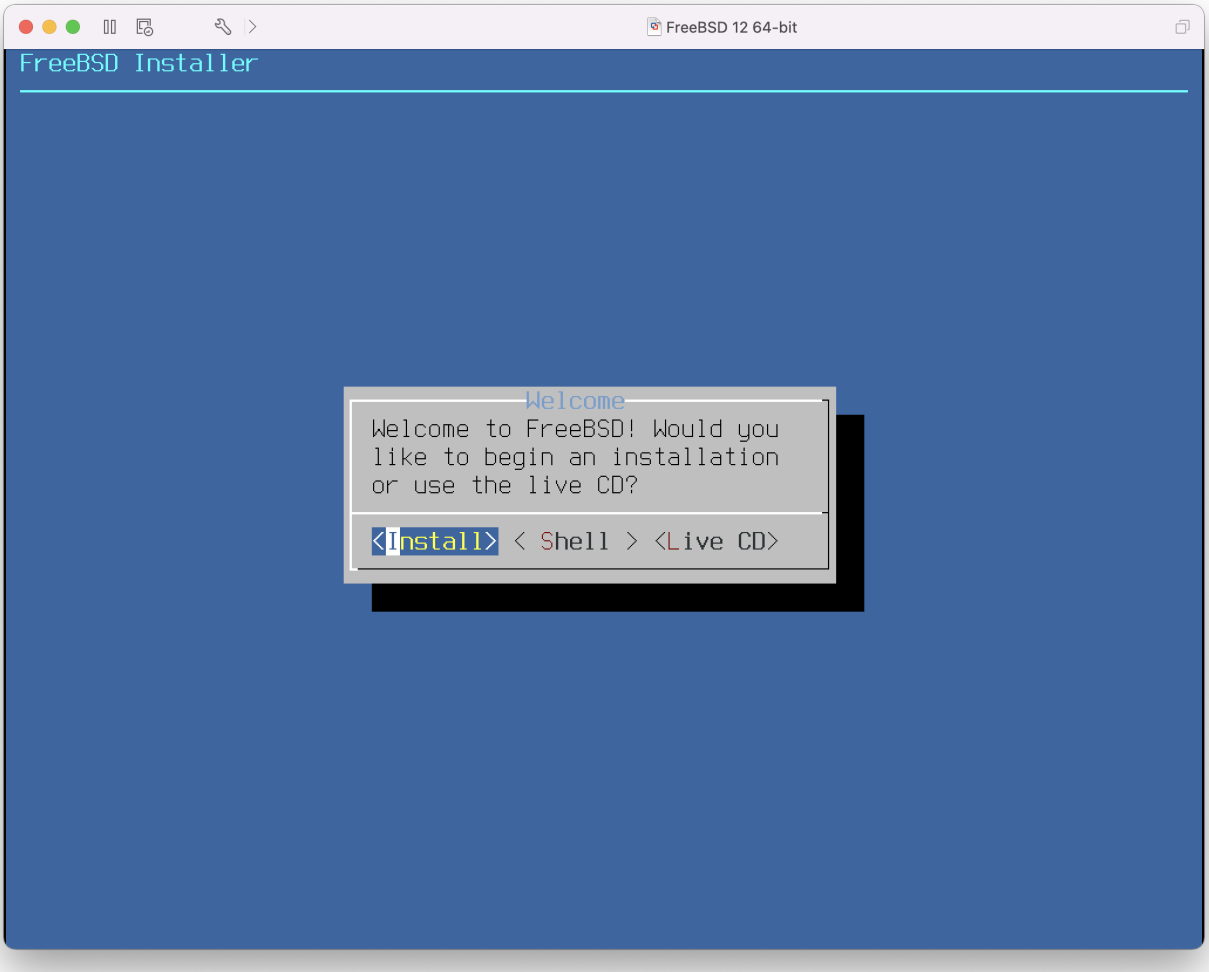
安装完成之后，你就可以修改一些 VM 的设定，比如内存大小：

在 VM 运行的时候，VM 系统硬件的设置是无法修改的。

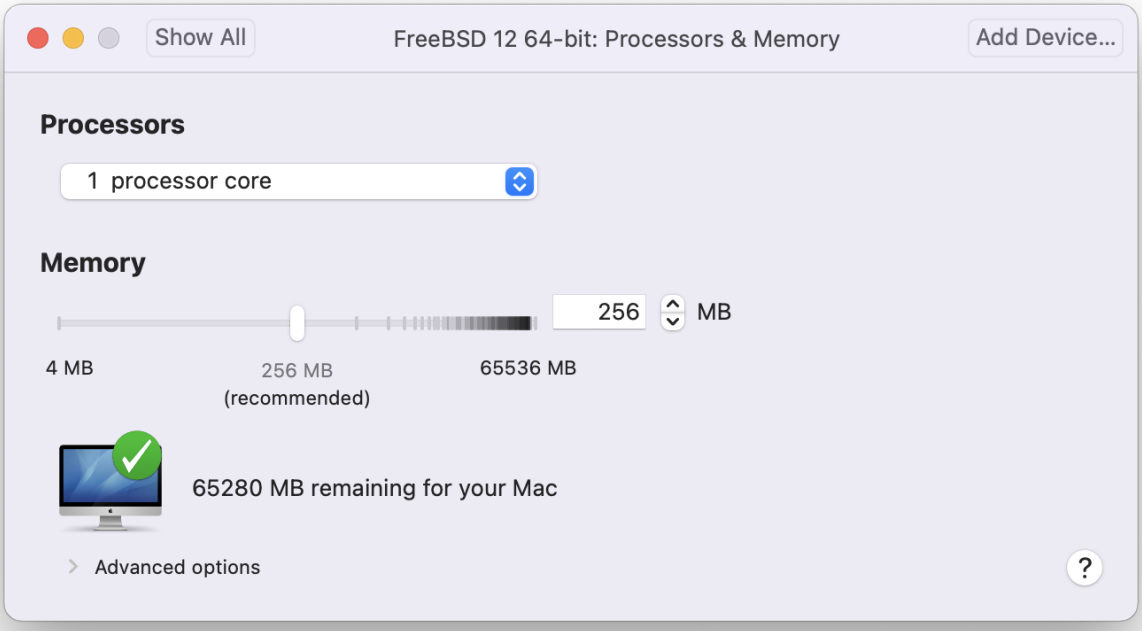


配置 VM 的 CPU 数量：

CD-ROM设备的状态。通常当你不在需要 CDROM/ISO 的时候可以切断他们跟 VM 的连接。



最后一项需要修改的是 VM 与网络连接的方式。 如果你希望除了宿主以外的机器也能连接到 VM， 请选择 Connect directly to the physical network (Bridged)。选择 Share the host’ s internet connection (NAT) 的话， VM 可以连接上网络，但是不能从外面访问。



在你修改完设定之后，就可以从新安装的 FreeBSD 虚拟机器启动了。

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，有些配置的步骤可用来优化虚拟系统。

1. 设置 boot loader 变量

   最重要的步骤是降低 kern.hz 来减少 VMware 上 FreeBSD 的 CPU 使用率。这需要在 /boot/loader.conf 里加入以下这行设定：

   ```
   kern.hz=100
   ```

   如果没有这项设定，VMware 上的 FreeBSD 客户 OS 空闲时将占用 iMac® 上一个 CPU 大约 15% 的资源。在修改此项设定之后仅为 5%。

2. 创建一个新的内核配置文件

   你可以去掉所有的 FireWire, USB 设备的驱动程序。 VMware 提供了一个 em(4) 支持的虚拟网络适配器，所以除了 em(4) 之外的网卡驱动都可以被剔除。

3. 设置网络

   最基本的网络设定包括简单的使用 DHCP 把你的虚拟机器连接到宿主 Mac® 相同的本地网络上。 在 /etc/rc.conf 中加入： ifconfig_em0="DHCP"。 更多有关网络的设置可以参阅 高级网络。

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。 一些用户曾时使用过利用 Linux® 二进制兼容层运行的 VMware 陈旧并多半已过时的版本 (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™ 是一个开发非常活跃， 完全虚拟化的软件， 并且可在大部份的操作系统上使用， 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两种版本。 从用户的角度来看， OSE 版本最主要的限制也许是缺乏 USB 的支持。 其他更多的差异可以通过链接 http://www.virtualbox.org/wiki/Editions 查看 "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
```

在配置对话框中的一个有用的选项是 GusetAdditions 程序套件。 这些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间使用鼠标， 而不用事先按下某个特定的快捷键来切换) 和更快的视频渲染， 特别是在 Windows® 客户系统中。 在安装了客户操作系统之后， 客户附加软件可在 Devices 菜单中找到。

在第一次运行 VirtualBox™ 之前还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， 因为它们使用了由 sysctl(3) 库提供的功能。

当使用旧版本的 port 时， 需要使用下面的步骤来挂载 proc：

```
# mount -t procfs proc /proc
```

为了使配置能在重启后始终生效， 需要在 /etc/fstab 中加入以下这行：

```
proc   /proc   procfs  rw  0   0
```

如果在运行 VirtualBox™ 的终端中发现了类似如下的错误消息：

```
VirtualBox: supR3HardenedExecDir: couldn't read "", errno=2 cchLink=-1
```

此故障可能是由 proc 文件系统导致的。 请使用 mount 命令检查文件系统是否正确挂载。

在安装 VirtualBox™ 时会自动创建 vboxusers 组。 所有需要使用 VirtualBox™ 的用户必须被添加为此组中的成员。 可以使用 pw 命令添加新的成员：

```
# pw groupmod vboxusers -m yourusername
```

运行 VirtualBox™， 可以通过选择你当前图形环境中的 Sun VirtualBox，
也可以在虚拟终端中键入以下的命令:

```
% VirtualBox
```

获得更多有关配置和使用 VirtualBox™ 的信息， 请访问官方网站 http://www.virtualbox.org。 鉴于 FreeBSD port 非常新， 并仍处于开发状态。请查看 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查细则。

Chapter 24. 本地化－I18N/L10N使用和设置

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。
这章将讨论FreeBSD的国际化和本地化的问题,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，主要是通过执行i18N标准来实现的，所以这里我们将为读者提供详细的介绍。

读完这一章，您将了解：

- 不同的语言和地域是如何在现代操作系统上进行编码的。
- 如何为您的登入shell设置本地化。
- 如何配置您的控制台为非英语语言。 languages.
- 如何使用不同的语言来有效地使用X Windows。
- 在哪里可以找到更多有关开发符合i18N标准的应用程序的信息。

阅读这章之前，您应当了解：

- 怎样安装额外的第三方程序（安装应用程序. Packages 和 Ports）。

24.2. 基础知识

24.2.1. I18N/L10N 是什么?

开发人员把internationalization简写成I18N,中间的数字是前后两个字母间的字母个数。
L10N依据"localization" 使用同样的命名规则。
I18N/L10N方法、协议和应用结合在一起，允许用户使用他们自己所选择的语言。

I18N应用程序使用I18N工具来编程。它允许开发人员写一个简单的文件，
就可以将显示的菜单和文本翻译成本地语言。我们非常鼓励程序员遵循这种规则。

24.2.2. 为什么要使用I18N/L10N?

I18N/L10N标准能够很好地支持您查看、输入或处理非英语语言。

24.2.3. I18N支持哪些语言?

I18N和L10N不是FreeBSD特有的。当前，它能支持世界上绝大部分主力语言，
包括但不限于：中文，德文，日文，朝鲜文，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，它是一个规则。我们鼓励您帮助FreeBSD完善这一规则。

本地化设置需要具备三个条件：语言代码 (Language Code)、 国家代码 (Country Code)
和编码(Encoding)。 本地名字可以用下面这些部分来构造：

```
语言代码_国家代码.编码
```

24.3.1. 语言和国家代码

为了用特殊的语言来对FreeBSD系统进行本地化（或其他类UNIX®系统），
用户必须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规范）。
此外，WEB浏览器，SMTP/POP服务器，web服务器等都是以这个为基础的。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. 编码

一些语言不使用 ASCII 编码，它们使用8-位， 宽或多字节的字符， 更多的信息请参考 multibyte(3)。
比较老的应用程序可能会无法识别它们， 并误认为是控制字符。 比较新的应用程序通常会认出 8-位字符。
随实现的不同， 用户可能不得不将宽或多字节字符支持编入应用程序， 或进行一些额外的配置，
才能够正常使用它们。 要输入和处理宽或多字节字符， FreeBSD Ports Collection
已经为每种语言提供了不同的程序。 请参考各个 FreeBSD Port 中的 I18N 文档。

特别需要指出的是， 用户可能需要查看应用程序的文档， 以确定如何正确地配置它， 或需要为
configure/Makefile/编译器 指定什么样的参数。

记住下面这些:

- 特定语言的简单C字符集 (参见 multibyte(3))，例如 ISO8859-1, ISO8859-15, KOI8-R, CP437。
- 宽字节或多字节编码，如EUC, Big5。

您可以在IANA Registry检查一下现行的字符集列表。

与此不同的是， FreeBSD 使用与 X11-兼容的本地编码模式。

24.3.3. I18N应用程序

在FreeBSD Ports和Package系统里面，I18N应用程序已经使用I18N
来命名。然而它们不是总支持需要的语言。

24.3.4. 本地化设置

通常只要在登入shell里面设置LANG为本地化， 一般通过设置用户的 ~/.login_conf
或用户shell的启动文件（~/.profile，~/.bashrc, ~/.cshrc）。没有必要设置 LC_CTYPE，LC_CTIME。
更多的信息请参考特定语言的FreeBSD文档。

您应当在您的配置文件中设置下面两个变量:

- LANG 为POSIX®设置本地化语言功能。
- MM_CHARSET应用程序的MIME字符集。

这包括用户的shell配置，特定的应用配置和X11配置。

24.3.4.1. 设置本地化的方法

有两种方法来设置本地化，接下来都会描述。 第一种 (推荐) 就是在 登入分类里面指定环境变量。
第二种方法是把环境变量加到shell的启动文件里面。

24.3.4.1.1. 登入分类方法

这种方法允许把本地化名称和MIME字符集的环境变量赋给可能的shell，
而不是加到每个特定shell的启动文件里面。 用户级设置 Level Setup
允许普通用户自己完成这个设置，而管理员级设置需要超级用户权限。

24.3.4.1.1.1. 用户级设置

这有一个设置用户根目录文件.login_conf的小例子， 它为上述两个变量设置了Latin-1编码。

```
me:\
  :charset=ISO-8859-1:\
  :lang=de_DE.ISO8859-1:
```

这是一个为.login_conf设置繁体中文的BIG-5编码的例子。应该设置下面的大部分变量，
因为很多软件都没有为中文，日文和韩文设置正确的本地化变量。

```
#Users who do not wish to use monetary units or time formats
#of Taiwan can manually change each variable
me:\
  :lang=zh_TW.Big5:\
  :setenv=LC_ALL=zh_TW.Big5:\
  :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员级设置和login.conf(5)

24.3.4.1.2. 管理员级设置

检查用户的登入分类在 /etc/login.conf里面是否设置了正确的语言。主要确定下面的几个设置：

```
language_name|Account Type Description:\
  :charset=MIME_charset:\
  :lang=locale_name:\
  :tc=default:
```

再次使用前面的Latin-1编码的例子：

```
german|German Users Accounts:\
  :charset=ISO-8859-1:\
  :lang=de_DE.ISO8859-1:\
  :tc=default:
```

在修改用户的登入类型之前， 应首先执行下面的命令：

```
# cap_mkdb /etc/login.conf
```

以便使在 /etc/login.conf 中新增的配置生效。

24.3.4.1.3. 使用 vipw(8) 改变登入类型。

使用vipw添加新用户，看起来像下面这样：

```
user:password:1111:11:language:0:0:User Name:/home/user:/bin/sh
```

24.3.4.1.4. 用adduser(8)改变登入类型。

用adduser添加新用户看起来像下面这样：

- 在/etc/adduser.conf里面设置defaultclass = 语言。应该记住，您必须为使用其它语言的所有用户设置缺省类别。
- 每一次使用adduser(8)的时候，一个特定语言的可选择性回答会像下面这样给出：

  ```
  Enter login class: default []:
  ```

- 如果您打算给每一个用户使用另外一种语言，您应该这样：

  ```
  # adduser -class language
  ```

24.3.4.1.5. 使用pw(8)改变登入类型。

如果您使用pw(8)来添加新用户，应该这样使用：

```
# pw useradd user_name -L language
```

24.3.4.1.6. Shell启动文件方法

不推荐使用这种方法，因为它需要给每一个可能的shell程序一个不同的启动文件。应该用登入分类方法来代替这种方法。

为了设置本地化名称和MIME字符集，只要在/etc/profile或/etc/csh.login启动文件里面设置这两个变量。下面我们使用德语做例子：

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
```

另外，您可以把上面的设置添加到/usr/shared/skel/dot.profile （和前面的/etc/profile一样），或者/usr/shared/skel/dot.login （和前面的/etc/csh.login一样）。

对于X11：

在$HOME/.xinitrc里面：

```
LANG=de_DE.ISO8859-1; export LANG
```

或者：

```
setenv LANG de_DE.ISO8859-1
```

依赖您的shell(看上面）。

24.3.5. 控制台设置

对于所有的简单C字符集，在/etc/rc.conf中用正在讨论的语言设置正确的控制台字符：

```
font8x16=font_name
font8x14=font_name
font8x8=font_name
```

这儿的font_name来自于/usr/shared/syscons/fonts目录， 不带.fnt后缀。

如果需要的话， 还应通过 sysinstall 来配置与单字节 C 字符集对应的 keymap 和 screenmap。 在 sysinstall 中， 选择 Configure 之后选择 Console 即可进行配置。 除此之外， 您也可以在 /etc/rc.conf 中加入类似下面的配置：

```
scrnmap=screenmap_name
keymap=keymap_name
keychange="fkey_number sequence"
```

这儿的screenmap_name是来自/usr/shared/syscons/scrnmaps目录， 不带.scm后缀。
一个带影射字体的屏幕布局通常被作为一个工作区， 用来在VGA适配器字体矩阵上扩展8位到9位。
如果屏幕字体是使用一个8位的排列，要移动这些字母离开这些区域。

如果您在/etc/rc.conf里面启用了moused daemon：

```
moused_enable="YES"
```

那么需要在下一段检查鼠标指针信息。

默认情况下， syscons(4)驱动程序的鼠标指针在字符集中占用0xd0-0xd3的范围。
如果您的语言使用这个范围，您必须把指针范围移出这个范围。 要绕过这个问题， 需要在 /etc/rc.conf 中加入：

```
mousechar_start=3
```

这里， keymap_name 来自于 /usr/shared/syscons/keymaps 目录， 但去掉了 .kbd 后缀。
如果不确定应该使用哪一个键盘布局， 则可以使用 kbdmap(1) 来测试， 而无需反复重启。

通常， keychange 是设定功能键时， 匹配选定的终端类型来说是必需的，
因为功能键序列无法在键盘布局中定义。

此外您还应该检查并确认在 /etc/ttys 中已经为所有的 ttyv* 项配置了正确的终端类型。 目前，
相关的默认定义是：

字符集设置	终端类型
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

对于多字节字符语言，可以您的在 /usr/ports/language 目录中使用正确的FreeBSD
port。一些port以控制台出现， 而系统把它作为串行vtty终端，因此， 必须为 X11
和伪串行控制台准备足够的vtty终端。 下面是在控制台中使用其他语言的应用程序的部分列表：

语言	特定区域
Traditional Chinese (BIG-5)	chinese/big5con
Japanese	japanese/kon2-16dot or japanese/mule-freewnn
Korean	korean/han

24.3.6. X11设置

虽然X11不是FreeBSD计划的一部分， 但我们已经为FreeBSD用户包含了一些信息。
具体细节可以参考Xorg Web 站点 或是您使用的 X11 Server 的网站。

在~/.Xresources里面，您可以适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)， 然后安装对应语言的 TrueType® 字体。
请设置正确的地区信息， 这将让您能够在菜单和其它地方看到所选择的语言。

24.3.6.2. 输入非英语字符

X11输入方法（XIM）协议是所有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。 不同的语言有几种XIM服务器可用。

24.3.7. 打印机设置

一些简单的C字符集通常是用硬编码来编码进打印机的。更宽或多位的字符集需要特定的设置，
我们推荐使用apsfilter。您也可以使用特定语言转换器把文档转换为 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简单的 C 字符集 (参见
multibyte(3))， 但在文件系统中不会保存字符集的名字； 也就是说， 它不加修改地保存 8-位信息，
而并不知道如何编码。 正式说来， FFS 目前还不支持任何形式的宽或多字节字符集。 不过，
某些宽或多字符集提供了独立的针对 FFS 的补丁来帮助启用关于它们的支持。 目前这些要么是无法移植的，

要么过于粗糙，因此我们不打算把它们加入到源代码中。请参考相关语言的 Web 站点，以了解关于这些补丁的进一步情况。

FreeBSD MS-DOS®已经能够配置成用在MS-DOS®上，Unicode字符集和可选的FreeBSD文件系统字符集的更多信息，请参考 mount_msdosfs(8) 联机手册。

24.4. 编译I18N程序

许多FreeBSD Ports已经支持I18N了。他们中的一些都用-I18N作标记。这些和其他很多程序已经内建I18N的支持，不需要考虑其他的事项了。

然而一些像MySQL这样的应用程序需要重新配置字符集，可在Makefile里面设置，或者直接把参数传递给configure。

24.5. 本地化FreeBSD

24.5.1. 俄语（KOI8-R编码）

关于KOI8-R编码的更多信息请查阅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 里面增加如下设置：

  ```
  keymap="ru.utf-8"
  scrnmap="utf-82cp866"
  font8x16="cp866b-8x16"
  font8x14="cp866-8x14"
  font8x8="cp866-8x8"
  ```

- 对于/etc/ttys里面的ttyv*记录，要使用 cons25r作为终端类型。

参看前面的设置控制台的例子。

24.5.1.3. 打印机设置

既然绝大多数带俄语字符的打印机遵循CP866的标准， 那么需要一个针对KOI8-R到CP866转换的特定输出过滤器。这样的一个过滤器默认的安装在 /usr/libexec/lpr/ru/koi2alt。一个支持俄语的打印机的/etc/printcap记录看起来是这样的：

```
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)手册页。

24.5.1.4. MS-DOS®文件系统和俄语文件名

下面的例子是在挂上MS-DOS® 文件系统后，启用对俄语文件名支持的fstab(5)记录：

```
/dev/ad0s2     /dos/c  msdos   rw,-Wkoi2dos,-Lru_RU.KOI8-R 0 0
```

选项 -L 用于选择地区名称， 而 -W 则用于设置字符转换表。 要使用 -W 选项， 则一定要首先挂接 /usr，然后再挂接 MS-DOS® 分区， 因为转换表是放在 /usr/libdata/msdosfs 的。 要了解进一步的细节， 请参考 mount_msdosfs(8) 联机手册。

24.5.1.5. X11设置

1. 首先请进行前面介绍的 非-X 的本地化设置。
2. 如果您正使用 Xorg， 请安装 x11-fonts/xorg-fonts-cyrillic package。

   检查您 /etc/X11/xorg.conf 文件中的 "Files" 小节。 下面的行， 应加到任何其它 FontPath 项之前：

   ```
   FontPath   "/usr/local/lib/X11/fonts/cyrillic"
   ```

   请查看 ports 中的其它西里尔字体。

3. 要激活俄语键盘， 需要在 xorg.conf 文件的 "Keyboard" 小节中加入下列内容：

   ```
   Option "XkbLayout"   "us,ru"
   Option "XkbOptions"  "grp:toggle"
   ```

   要确信XkbDisable 已经关闭 (注释掉) 了。

   RUS/LAT的切换用CapsLock。老的CapsLock功能可以通过 Shift + CapsLock 来模拟（只有在LAT模式的时候）。

   使用 grp:toggle 时， RUS/LAT 切换键将是 右 Alt， 而使用 grp:ctrl_shift_toggle 则表示切换键是 Ctrl + Shift。 使用 grp:caps_toggle 时， RUS/LAT 切换键则是 CapsLock。 旧的 CapsLock 功能仍可通过 Shift + CapsLock (只对 LAT 模式有效)。 由于不明原因， grp:caps_toggle 在 Xorg 中无法使用。

   如果您的键盘上有 "Windows®" 键， 但发现 RUS 模式下， 某些非字母键映射不正常， 则应在您的 xorg.conf 文件中加入下面这行：

```
Option "XkbVariant" ",winkeys"
```

俄语的 XKB 键盘可能并不为某些不具备本地化功能的应用程序所支持。

本地化程序最低限度应在程序启动时调用 XtSetLanguageProc (NULL, NULL, NULL); 函数。

参见 KOI8-R for X Window 以获得关于对 X11 应用进行本地化的指导。

24.5.2. 设置繁体中文

FreeBSD-Taiwan计划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 的维护人员是 沈俊兴 statue@freebsd.sinica.edu.tw。

沈俊兴 statue@freebsd.sinica.edu.tw 利用 FreeBSD-Taiwan 的 zh-L10N-tut建立了 Chinese FreeBSD Collection (CFC)。 相关的 packages 和脚本等可以在 ftp://freebsd.csie.nctu.edu.tw/pub/taiwan/CFC/ 找到。

24.5.3. 德语本地化（适合所有的ISO 8859-1语言）

Slaven Rezic eserte@cs.tu-berlin.de 写了一个在 FreeBSD 机器下如何使用日尔曼语言的德语指南。 这份德语教程可以在 http://user.cs.tu-berlin.de/~eserte/FreeBSD/doc/umlaute/umlaute.html 找到。

24.5.4. 希腊语本地化

Nikos Kokkalis nickkokkalis@gmail.com 撰写了关于在 FreeBSD 上支持希腊语的完整文章， 在 http://www.freebsd.org/doc/el/articles/greek-language-support/。 请注意这篇文章 只有 希腊语的版本。

24.5.5. 日语和韩语本地化

日语本地化请参考http://www.jp.FreeBSD.org/，韩语参考 http://www.kr.FreeBSD.org/。

24.5.6. 非英语的FreeBSD文档

一些 FreeBSD 的贡献者已经将部分 FreeBSD 文档翻译成了其他语言。 您可在 主站 以及 /usr/shared/doc 找到。

Chapter 25. 更新与升级 FreeBSD

25.1. 概述

FreeBSD 在发行版之间始终是持续开发的。 一些人喜欢使用官方发行的版本，
另一些喜欢与最新的开发保持同步。 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。 不论你使用了何种版本， FreeBSD 都提供了所有更新系统所需的工具， 让你轻松的在不同版本间升级。
这一章节将帮助你决定是跟踪开发系统还是坚持使用某个发行的版本。
同时还列出了一些保持系统更新所需的基本工具。

读了本章后，您将了解到：

- 使用哪些工具来更新系统与 Ports Collection。
- 如何使用 freebsd-update, CVSup, CVS, or CTM 让你的系统保持更新。
- 如何比较已安装的系统与原来已知拷贝的状态。
- 如何使用 CVSup 或者文档 ports 来更新本地的文档。
- 两个开发分支 FreeBSD-STABLE 和 FreeBSD-CURRENT 的区别。
- 如何通过 make buildworld 重新编译安装整个基本系统(等等)。

在读本章这前，您应该了解的：

- 正确设置网络连接 (高级网络)。
- 知道怎样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 客户端的话，则可以安装 net/cvsup-without-gui port)。 你也可以使用 csup(1) 代替，
它现在已经是基本系统的一部分了。

25.2. FreeBSD 更新

打安全补丁是对于维护计算机软件的一个重要部分， 特别是对于操作系统。对于 FreeBSD 来说，
很长的一段时间以来这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为二进制，
然后再重新安装编译后的程序。

FreeBSD 引入了 freebsd-update 工具之后这便不再是问题了。这个工具提供了 2 种功能。
第一，它可以把二进制的安全和勘误更新直接应用于 FreeBSD
的基本系统，而不需要重新编译和安装。第二， 这个工具还支持主要跟次要的发行版的升级。

由安全小组支持的各种体系结构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， 应先阅读一下当前发行版的声明，
因为它们可能包含有关于你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查阅： 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

有些用户可能希望通过调整配置文件 /etc/freebsd-update.conf 中的默认配置来更好地控制升级的过程。
可用的参数在文档中介绍的很详细， 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 的哪一部分将被保持更新。 默认的是更新源代码，整个基本系统还有内核。
这些部件跟安装时的那些相同，举例来说， 在这里加入 world/games 就会允许打入游戏相关的补丁。 使用
src/bin 则是允许更新 src/bin 目录中的源代码。

最好的选择是把这个选项保留为默认值， 因为如果要修改它去包含一些指定的选项，
就需要用户列出每一个想要更新的项目。 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 让这些指定的目录在更新过程中不被修改。 这个选项能够防止本地的修改被
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。 用户的任何修改都会使这些文件的自动更新失效。
还有另外一个选项， 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 应该尝试合并的配置文件的列表。文件合并的过程是 一系列的 diff(1)
补丁类似于更少选项的 mergemaster(8) 合并的选项是接受，打开一个文本编辑器，或者 freebsd-update
会被中止。 在不能确定的时候，请先备份 /etc 然后接受合并。更多关于 mergemaster 的信息请参阅
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。 用户做一个版本升级的话，请确认此处至少有 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
```

当设置成 yes 时， freebsd-update 将假设这个 Components 列表时完整的，
并且对此列表以外的项目不会修改。实际上就是 freebsd-update 会尝试更新 Componets
列表里的每一个文件。

25.2.2. 安全补丁

安全补丁存储在远程的机器上， 可以使用如下的命令下载并安装：

```
# freebsd-update fetch
# freebsd-update install
```

如果给内核打了补丁，那么系统需要重新启动。 如果一切都进展顺利，系统就应该被打好了补丁而且
freebsd-update 可由夜间 cron(8) 执行。在 /etc/crontab 中加入以下条目足以完成这项任务：

```
@daily                  root    freebsd-update cron
```

这条记录是说明每天运行一次 freebsd-update 工具。 用这种方法， 使用了 cron 参数， freebsd-update
仅检查是否存在更新。 如果有了新的补丁，就会自动下载到本地的磁盘， 但不会自动给系统打上。root
会收到一封电子邮件告知需手动安装补丁。

如果出现了错误，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话， 就需要重新启动系统。这将使 FreeBSD
装载新的二进制程序进内存。

freebsd-update 工具只能自动更新 GENERIC 内核。 如果您使用自行联编的内核， 则在 freebsd-update
安装完更新的其余部分之后需要手工重新联编和安装内核。 不过， freebsd-update 会检测并更新位于
/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。

保存一份 GENERIC 内核的副本到 /boot/GENERIC 是一个明智的主意。
在诊断许多问题， 以及在 重大和次要的更新 中介绍的使用 freebsd-update
更新系统时会很有用。

除非修改位于 /etc/freebsd-update.conf 中的配置， freebsd-update
会随其他安装一起对内核的源代码进行更新。 重新联编并安装定制的内核可以以通常的方式来进行。

通过 freebsd-update 发布的更新有时并不会涉及内核。 如果在执行 freebsd-update
install 的过程中内核代码没有进行变动， 就没有必要重新联编内核了。 不过， 由于
freebsd-update 每次都会更新 /usr/src/sys/conf/newvers.sh 文件， 而修订版本
(uname -r 报告的 -p 数字) 来自这个文件， 因此， 即使内核没有发生变化，
重新联编内核也可以让 uname(1) 报告准确的修订版本。
在维护许多系统时这样做会比较有帮助，
因为这一信息可以迅速反映机器上安装的软件更新情况。

25.2.3. 重大和次要的更新

这个过程会删除旧的目标文件和库， 这将使大部分的第三方应用程序无法删除。 建议将所有安装的 ports
先删除然后重新安装，或者稍后使用 ports-mgmt/portupgrade 工具升级。
大多数用户将会使用如下命令尝试编译：

```
# portupgrade -af
```

这将确保所有的东西都会被正确的重新安装。 请注意环境变量 BATCH 设置成 yes
的话将在整个过程中对所有询问回答 yes，这会帮助在编译过程中免去人工的介入。

如果正在使用的是定制的内核， 则升级操作会复杂一些。 您会需要将一份 GENERIC 内核的副本放到
/boot/GENERIC。 如果系统中没有 GENERIC 内核， 可以用以下两种方法之一来安装：

- 如果只联编过一次内核， 则位于 /boot/kernel.old 中的内核， 就是 GENERIC 的那一个。
  只需将这个目录改名为 /boot/GENERIC 即可。
- 假如能够直接接触机器， 则可以通过 CD-ROM 介质来安装 GENERIC 内核。 将安装盘插入光驱，
  并执行下列命令：

  ```
  # mount /cdrom
  # cd /cdrom/X.Y-RELEASE/kernels
  # ./install.sh GENERIC
  ```

  您需要将 X.Y-RELEASE 替换为您正在使用的版本。 GENERIC 内核默认情况下会安装到
  /boot/GENERIC。

- 如果前面的方法都不可用， 还可以使用源代码来重新联编和安装 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 能够正确地将内核识别为 GENERIC， 您必须确保没有对 GENERIC
  配置文件进行过任何变动。 此外， 建议您取消任何其他特殊的编译选项 (例如使用空的
  /etc/make.conf)。

上述步骤并不需要使用这个 GENERIC 内核来引导系统。

重大和次要的更新可以由 freebsd-update 命令后指定一个发行版本来执行，
举例来说，下面的命令将帮助你升级到 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获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检测到与未检测到的组件列表。 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 将会尝试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进行之类的问题。

当使用定制内核时， 前面的步骤会产生类似下面的警告：

```
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"
```

此时您可以暂时安全地无视这个警告。 更新的 GENERIC 内核将在升级过程的中间步骤中使用。

在下载完针对本地系统的补丁之后， 这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负载。 这个过程中将会对配置文件所做的变动进行合并 -
这一部分需要用户的参与， 文件可能会自动合并， 屏幕上也可能会给出一个编辑器，
用于手工完成合并操作。 在处理过程中， 合并成功的结果会显示给用户。 失败或被忽略的合并，
则会导致这一过程的终止。 用户可能会希望备份一份 /etc 并在这之后手工合并重要的文件， 例如
master.passwd 和 group。

系统至此还没有被修改，所有的补丁和合并都在另外一个目录中进行。
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，
我们就已经完成了整个升级过程中最困难的部分， 下面就需要用户来安装这些变更了。

一旦这个步骤完成后，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。 此时必须重新启动计算机。 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 内核启动之前， 请确信它包含了用于引导系统所需的全部驱动程序
(如果您是在远程进行升级操作， 还应确信网卡驱动也是存在的)。 特别要注意的情形是，
如果之前的内核中静态联编了通常以内核模块形式存在的驱动程序， 一定要通过
/boot/loader.conf 机制来将这些模块加载到 GENERIC 内核的基础上。 此外，
您可能也希望临时取消不重要的服务、 磁盘和网络挂载等等， 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
```

在系统重新上线后，需要再次运行 freebsd-update。 升级的状态被保存着，这样 freebsd-update 就无需重头开始，但是会删除所有旧的共享库和目标文件。 执行如下命令继续这个阶段的升级：

```
# freebsd-update install
```

取决与是否有库的版本更新，通常只有 2 个而不是 3 个安装阶段。

现在需要重新编译和安装第三方软件。
这么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库。 可使用 ports-mgmt/portupgrade 自动化这个步骤，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
```

一旦这个完成了以后，再最后一次运行 freebsd-update 来结束升级过程。
执行如下命令处理升级中的所有细节：

```
# freebsd-update install
```

如果您临时用过 GENERIC 内核来引导系统，
现在是按照通常的方法重新联编并安装新的定制内核的时候了。

重新启动机器进入新版本的 FreeBSD 升级过程至此就完成了。

25.2.4. 系统状态对照

freebsd-update 工具也可被用来对着一个已知完好的 FreeBSD 拷贝测试当前的版本。
这个选项评估当前的系统工具，库和配置文件。 使用以下的命令开始对照：

```
# freebsd-update IDS >> outfile.ids
```

这个命令的名称是 IDS， 它并不是一个像 security/snort 这样的入侵检测系统的替代品。因为 freebsd-update 在磁盘上存储数据，很显然它们有被篡改的可能。 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 和不使用时把 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。

现在系统将会被检查，生成一份包含了文件和它们的 sha256(1) 哈希值的清单，已知发行版中的值与当前系统中安装的值将会被打印到屏幕上。 这就是为什么输出被送到了 outfile.ids 文件。 它滚动的太块无法用肉眼对照，而且会很快填满控制台的缓冲区。

这个文件中有非常长的行，但输出的格式很容易分析。

举例来说，要获得一份与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
```

这份输出时删节缩短后的，其实是有更多的文件。 其中有些文件并非人为修改，比如 /etc/passwd 被修改是因为添加了用户进系统。在某些情况下， 还有另外的一些文件，诸如内核模块与 freebsd-update 的不同是因为它们被更新过了。 为了指定的文件或目录排除在外，把它们加到 /etc/freebsd-update.conf 的 IDSIgnorePaths 选项中。

除了前面讨论过的部分之外， 这也能被当作是对升级方法的详细补充。

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，它会连上一个远程网站， 校验安全密钥，然后下载一份 Ports Collection 的拷贝。 密钥是用来校验所有下载文件的完整性，确保它们在传输是未被修改。 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) 发现并校验了几个用于当前 ports 的补丁。这还表明以前运行过， 如果是第一次运行的话，那么仅仅只会下载 Ports Collection。

在 portsnap(8) 成功地完成一次 fetch 操作之后， 会将校验过的 Ports 套件和后续的补丁保存在本地。 首次执行 portsnap 之后， 你必须使用 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 可以作为连续的动作执行， 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. 更新系统附带的文档

除了基本系统和 Ports 套件之外， 文档也是 FreeBSD 操作系统的一个组成部分。 尽管您总是可以通过 FreeBSD 网站 来访问最新的 FreeBSD 文档， 一些用户的网络连接可能很慢， 甚至完全没有网络连接。 幸运的是， 有很多方法可以用来更新随发行版本附带的 FreeBSD 文档的本地副本。

25.4.1. 使用 CVSup 来更新文档

FreeBSD 文档的源代码和安装版本都可以通过 CVSup 来以与基本系统 (参考 重新编译 "world") 类似的方法来升级。 这一节中将会介绍：

- 如何安装联编文档所需的工具集， 用于从源代码来联编 FreeBSD 文档所需的那些工具。
- 如何使用 CVSup 将文档下载到 /usr/doc。
- 如何从源代码联编 FreeBSD 文档， 并将其安装到 /usr/shared/doc。
- 联编文档的过程中支持的一些编译选项， 例如只联编某些语言的版本， 或只联编特定的输出格式。

25.4.2. 安装 CVSup 和文档工具集

从源代码联编 FreeBSD 文档需要大量的工具。 这些工具并不是 FreeBSD 基本系统的一部分， 因为这些工具需要占用大量的磁盘空间， 而且并不是对所有 FreeBSD 用户都有用； 只有活跃地撰写 FreeBSD 新文档， 或经常从源代码更新文档的用户才需要这些工具。

全部所需的工具， 均可通过 Ports 套件来安装。 textproc/docproj port 是由 FreeBSD 文档计划开发的方便安装和更新这些工具的主 port。

如果不需要 PostScript® 或 PDF 文档的话， 也可以考虑安装 textproc/docproj-nojadetex port。 这套文档工具集包含除了 teTeX typesetting 引擎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 输出的话， 排除它会节省很多时间和磁盘空间。

如欲了解关于安装和使用 CVSup 的进一步信息， 请参阅 使用 CVSup。

25.4.3. 更新文档源代码

CVSup 工具能够下载文档源代码的原始副本， 您可使用 /usr/shared/examples/cvsup/doc-supfile 文件作为配置模板来修改。 在 doc-supfile 中的默认主机名是一个无效的占位主机名， 但 cvsup(1) 能够通过命令行来指定主机名， 因此文档源代码可以使用下面的命令从 CVSup 服务器获得：

```
# cvsup -h cvsup.FreeBSD.org -g -L 2 /usr/shared/examples/cvsup/doc-supfile
```

您应将 cvsup.FreeBSD.org 改为最近的 CVSup 服务器。 参见 CVSup 站点 关于镜像站点的完整列表。

初始的文档源代码下载需要一些时间， 您需要耐心等待它完成。

后续的更新可以用同样的命令来进行。 由于 CVSup 工具只下载上次运行之后所发生过的更新， 因此在首次运行之后再运行 CVSup 应该是很快的。

在签出源代码之后， 还可以使用另一种由 /usr/doc 目录中的 Makefile 支持的方法来更新它。 通过在 /etc/make.conf 中配置 SUP_UPDATE、 SUPHOST 和 DOCSUPFILE， 可以通过运行：

```
# cd /usr/doc
# make update
```

来完成更新。 典型的 /etc/make.conf 中的 make(1) 选项是：

```
SUP_UPDATE= yes
SUPHOST?= cvsup.freebsd.org
DOCSUPFILE?= /usr/shared/examples/cvsup/doc-supfile
```

将 SUPHOST 和 DOCSUPFILE 的值使用 ?= 来指定的好处是使 make 命令行能够覆盖这些选项。 在向 make.conf 中增加选项时推荐这样做， 以避免在测试时反复修改这个文件。

25.4.4. 文档源代码中可调的选项

FreeBSD 文档的更新和联编系统支持一些方便只更新一部分文档， 或只联编特定格式及译文的选项。 这些选项可以在 /etc/make.conf 文件中配置， 也可以通过 make(1) 工具来指定。

这些选项包括：

DOC_LANG
: 准备联编和安装的语言列表。 例如， 指定为 en_US.ISO8859-1 表示只联编英文版的文档。

FORMATS
: 准备输出的格式列表。 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。 默认为 /usr/shared/doc。

如欲了解 FreeBSD 中其他可供配置的全局 make 变量， 请参阅 make.conf(5)。

关于 FreeBSD 文档联编系统的其他详情， 请参阅 FreeBSD 文档计划入门之新手必读部分。

25.4.5. 从源代码安装 FreeBSD 文档

在 /usr/doc 中下载了最新的文档源代码快照之后， 就可以开始动手联编文档了。

要更新全部 DOC_LANG 中定义的语言的文档， 需要执行下面的命令：

```
# cd /usr/doc
# make install clean
```

如果在 make.conf 中配置了正确的 DOCSUPFILE、 SUPHOST 和 SUP_UPDATE 选项，
则可以将更新源代码和安装一步完成：

```
# 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 中与之对应的目录中运行 make(1)：

```
# cd /usr/doc/en_US.ISO8859-1
# make update install clean
```

此外， 还可以透过 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， 我们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对于所有的 FreeBSD 系统都可行有效。
编译文档源代码需要一大堆的工具， 文档工具链， 对于 CVS 的一定了解和从仓库中检出源代码，
还有一些编译已检出代码的手工步骤。 这一章节我们将介绍一种使用 Ports 来更新已安装的 FreeBSD
文档：

- 下载并安装预编译好的文档快照， 而不用在本地编译任何部份
  (这样便不再需要安装整个文档工具链了)。
- 下载文档的源代码并使用 ports 框架编译 (使得检出和编译的步骤更容易些)。

这两种更新 FreeBSD 文档的方法都由一组 文档工程组 <doceng@FreeBSD.org> 每月更新的 文档 ports
提供支持。 这些都列在了 FreeBSD Ports docs 虚拟分类下面。

25.4.6.1. 编译和安装文档 Ports

文档 ports 使用 ports 的构建框架使得文档的编译变得更加容易。 自动化了检出文档源代码，
配以适合的环境设置和命令行参数运行 make(1)， 它们使得安装或卸载文档变得就像安装 FreeBSD 其他
port 或二进制包那样容易。

另一个特性便是当在本地编译文档 ports 时， 文档工具链 ports 会被列入依赖关系，
并自动安装。

文档 ports 按以下的方式组织：

- 一个 "主 port"， 在 misc/freebsd-doc-en 下可以找到这个文档 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。
- 最后是各种翻译的 "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 并会安装上对应语言的翻译文档。

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
```

这将会安装分章节的英文版本 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. 常见的调节选项

文档 ports 有许多用来修改默认行为的选项。 以下是一段简要列表：

WITH_HTML
: 允许构建 HTML 格式： 每份文档为一个单一的 HTML 文件。 此种文档的文件名视情况而定通常是 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 阅读器查阅。 此种文档的文件名视情况而定通常是 article.pdf 或 book.pdf。

DOCBASE
: 文档将被安装到的目录。默认值 /usr/local/shared/doc/freebsd。

请注意默认的目录与 CVSup 方法种所使用的目录不同。
这是因为我们正在安装的是一个 port， 而 ports 通常会被安装到 /usr/local 目录。
这可以指定 PREFIX 变量覆盖默认值。

这是一份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份文档工具链和一些编译所需的磁盘空间。
当不够资源安装文档工具链， 或者从源代码编译需要太多的磁盘空间时，
我们仍然可以安装预编译好的文档快照的 ports。

文档工程组 <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。

当使用二进制包时， 将安装所指定语言相关的 FreeBSD 文档的 所有 可用格式。

举例来说， 以下的命令将安装最新预编译的匈牙利语文档：

```
# pkg_add -r hu-freebsd-doc
```

二进制包使用了以下与对应 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 表示匈牙利语， 或者 zh_cn 表示简体中文。

25.4.6.3. 更新文档 Ports

任何用于更新 ports 的工具都可以被用来更新已安装的文档 port。 举例来说， 下面的命令通过 ports-mgmt/portupgrade 工具来更新已安装的匈牙利语文档二进制包。

```
# portupgrade -PP hu-freebsd-doc
```

25.5. 追踪开发分支

FreeBSD 有两个开发分支： FreeBSD-CURRENT 和 FreeBSD-STABLE。
这一章节将对每个分支作相应介绍与如何保持你的系统更新。 我们将先介绍 FreeBSD-CURRENT 然后是 FreeBSD-STABLE。

25.5.1. 使用最新的 FreeBSD CURRENT

这里再次强调， FreeBSD-CURRENT 是 FreeBSD 开发的 "最前沿"。 FreeBSD-CURRENT 用户要有较高的技术能力， 并且应该有能力自已解决困难的系统问题。 如果您是个 FreeBSD 新手， 那么在安装之前最好三思。

25.5.1.1. FreeBSD-CURRENT 是什么?

FreeBSD-CURRENT 是 FreeBSD 的发展前沿。 包括了在下一个官方发行的软件中可能存在， 也可能不存在的发展、 试验性改动、 以及过渡性的机制。 尽管许多 FreeBSD 开发者每天都会编译 FreeBSD-CURRENT 源代码， 但有时这些代码仍然会是不能编译的。 虽然这些问题会很快解决， 但 FreeBSD-CURRENT 是带来破坏还是您正希望的功能性改善， 很可能完全取决于您获取源代码的的时机!

25.5.1.2. 谁需要 FreeBSD-CURRENT?

FreeBSD-CURRENT 适合下边三种主要兴趣团体:

1. FreeBSD 社区的成员： 积极工作在源码树的某部分的人和为保持 "最新" 为绝对需求的人。
2. FreeBSD 社区的成员： 为促使 FreeBSD-CURRENT 保持尽可能的健全而愿花时间去解决问题的积极的测试者； 以及那些愿意提出关于 FreeBSD 变化和总体方向的建设性建议并且提供补丁实现它们的人们。
3. 那些只是想关注或为了参考目的使用当前 (current) 源码的人们 (如，为了阅读，而不是执行)。 这些人也偶尔做做注释或贡献代码。

25.5.1.3. FreeBSD-CURRENT 不是什么?

1. 追求最新功能， 您听说里面有一些很酷的新功能， 并希望成为您周围的人中第一个尝试它们的人。 尽管您能够因此首先了解到最新的功能， 但这也意味着在出现新的 bug 时您也首当其冲。
2. 修复错漏的快捷方式。任何 FreeBSD-CURRENT 的既定版本在修复已知错漏的同时又可能会产生新的错漏。
3. 无所不在的"官方支持"。 我们尽最大努力在3个"合法的" FreeBSD-CURRENT 组之一真诚给人们提供帮助，但是我们 没有时间提供技术支持。 这并不是因为我们是那种不喜欢帮助人解困的无耻之徒 (如果我们是的话，就不会制作 FreeBSD 了)。 我们不能每天简单地回复上百的消息，而且 我们继续发展 FreeBSD！ 在改善 FreeBSD 和回复大量关于实验代码的问题之间如果要做个选择的话， 开发人员会选择前者。

25.5.1.4. 使用 FreeBSD-CURRENT

1. 加入 FreeBSD-CURRENT 邮件列表 和 SVN src 树 head/-current 分支的修订讯息 列表。 这个不仅仅是个好主意，而且很 重要。如果您不去 FreeBSD-CURRENT 邮件列表，

您就不会看到人们所做的关于系统当前状态的说明，
这样您就有可能在别人已经发现并解决了的一大堆问题面前难倒。
更重要的是您会错过一些重要的公告---对于您的系统安全可能是至关重要的。

SVN src 树 head/-current 分支的修订讯息 列表允许您看到每个变化的提交记录，
因为这些记录与其它相关信息是同步的。

要加入这些列表，或其它可能的列表，请访问 https://lists.freebsd.org ，并且点击您想订阅的列项。
关于其它步骤的说明那里有提供。 如果你有兴趣追踪整个原代码树的变更记录， 我们建议你订阅 SVN 整个 src 树的修订讯息 (除了 "user" 与 "projects") 邮件列表。

2. 从FreeBSD 镜像站点 获取源码。 您有两种方式选择： ..

   与称作 standard-supfile 的 supfile 一起使用 cvsup，这个可以从
   /usr/shared/examples/cvsup得到。 这是最被推荐的方式，因为它允许您一次获取整个集合，
   以后就只取更改过的部分。许多人从 cron 运行 cvsup，以保持他们的源码自动更新。
   您须要定制上边的 supfile 样本，并且配置 cvsup 以适应您的环境。

   standard-supfile 例子是为追踪指定的 FreeBSD 安全分支而指定的， 而不是 FreeBSD-CURRENT。 你需要编辑这个文件并把如下这行：

   ```
   *default release=cvs tag=RELENG_X_Y
   ```

   替换为：

   ```
   *default release=cvs tag=.
   ```

   可以参阅手册中的 CVS Tags 章节获得更多可用 tag 的详细说明。

   使用工具 CTM。 如果您的连接性能不太好(高价连接或只能通过电子邮件存取)， CTM 是个选择。
   但这也颇有争议并且常常得到到坏文件。因此很少使用它， 这也注定了不能长期用它来工作。对于使用 9600 bps 或更快连接的人，我们推荐使用 CVSup。

3. 如果您获取源码是用于运行，而不只是看看，那么就获取 整个 FreeBSD-CURRENT，不要选部分。
   这样做的原因是源码的大部分都依赖于其他部分，
   要是您试着只编译其中一部分的话，保证您会陷入麻烦。

   在编译 FreeBSD-CURRENT 之前，请仔细阅读 /usr/src 里的 Makefile 文件。
   尽管是部分的升级过程，您至少也要首先安装新的内核和重建系统。阅读 FreeBSD-CURRENT 邮件列表
   邮件列表和 /usr/src/UPDATING， 会让您在其它循序渐进的过程中保持最新，
   这对于我们向下一个发行版转移是很有必要的。

4. 热心一点！如果您正运行 FreeBSD-CURRENT， 我们很想知道您关于它的一些想法，
   尤其是关于错漏修复或增进的建议。 非常欢迎带有代码的建议！

25.5.2. 使用最新的 FreeBSD STABLE

25.5.2.1. FreeBSD-STABLE 是什么?

FreeBSD-STABLE 是我们的发展分支，我们的主要发行版就由此而来。
这个分支会以不同速度变化，并且假定这些是第一次进入 FreeBSD-CURRENT 进行测试。然而，这 仍然是个发展中的分支，这意味着在一定的时候，FreeBSD-STABLE 源码可能或不可能满足一些特殊的要求。
它只不过是另一个工程发展途径，并不是终端用户的资源。

25.5.2.2. 谁需要 FreeBSD-STABLE?

如果您有兴趣追随 FreeBSD 的开发过程或为其做点贡献， 尤其是和下一个 "非计划" 的 FreeBSD

发行版有关时， 您应该考虑采用 FreeBSD-STABLE。

尽管安全更新也会进入 FreeBSD-STABLE 分支，但您并不 必须 使用 FreeBSD-STABLE
来达到这样的目的。 每一个 FreeBSD 的安全公告都会解释如何修复受到影响的发行版中的问题
，而因为安全原因而去采用一个开发分支显然可能会同时引入一些不希望的修改。

尽管我们尽力确保 FreeBSD-STABLE 分支在任何时候都能够正确编译和运行，
但没有人能够担保它在任何时候都总可以。 此外， 尽管代码在进入 FreeBSD-STABLE 之前都是在
FreeBSD-CURRENT 上完成开发， 但使用 FreeBSD-STABLE 的人要比使用 FreeBSD-CURRENT 的更多。
有证据显示， 犄角旮旯里的各种问题有些时候仍然会由于在 FreeBSD-CURRENT 不那么明显 而在
FreeBSD-STABLE 暴露出来。

基于这些原因， 不 推荐您盲目地追随 FreeBSD-STABLE， 并且，
在粗略地测试过代码之前不要更新任何生产服务器到 FreeBSD-STABLE 也非常重要。

如果您没有用于完成这些工作的资源， 我们推荐您使用最新的 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迁移。

25.5.2.3. 使用FreeBSD-STABLE

1. 加入 FreeBSD-STABLE; 邮件列表 列表。让您随时了解可能出现在 FreeBSD-STABLE 里的"build
   依赖性"或其它需要特别注意的问题。 当开发员正在考虑某些有争议的修复或更新时，
   他们就会在这个邮件列表里发表声明，给用户机会回应， 看他们对于提出的变化是否还有什么问题。

   加入相关的 SVN 列表来追踪你所关心的分支。比如，如果你在追踪 7-STABLE 分支，加入 svn-src-stable-7 列表。 这样每次这个分支上有改动的时候就能让你看到提交记录，
   还包括了修改可能引起的副作用之类的相关信息。

   要加入这些列表或其他可用的，访问 https://lists.freebsd.org
   并点击您希望订阅的列表。关于其它步骤的说明可以在那里看到。
   如果你有兴趣追踪整个原代码树的变更记录， 我们建议你订阅 SVN 整个 src 树的修订讯息 (除了 "user" 与 "projects") 邮件列表。

2. 如果您正安装一个新系统， 并希望它运行每月从 FreeBSD-STABLE 编译的快照， 请察看 Snapshots
   网页以了解更多信息。 另外， 也可以从 镜像站点 安装最新的 FreeBSD-STABLE 发行版，
   并按照其中的说明将系统更新到最新的 FreeBSD-STABLE 源代码。

   如果您已经在运行较早的 FreeBSD 版本， 并希望通过源代码方式升级， 则可以通过 FreeBSD 镜像站点 来完成。 这可以通过两种方式来进行： ..

   与称作 stable-supfile 的 supfile 一起使用 cvsup，这个可以从 /usr/shared/examples/cvsup 得到。
   这是最被推荐的方式，因为它允许您一次获取整个集合， 以后就只取更改过的部分。许多人从 cron
   运行 cvsup，以保持他们的源码自动更新。 您须要定制上边的 supfile 样本，并且配置 cvsup
   以适应您的环境。 ..

   使用工具 CTM。 如果您的连接性能不太好(高价连接或只能通过电子邮件存取)， CTM 是个选择。
   但这也颇有争议并且常常得到到坏文件。因此很少使用它， 这也注定了不能长期用它来工作。对于使用
   9600 bps 或更快连接的人，我们推荐使用 CVSup。

3. 本质上说，如果您需要快速存取源码并且不计较通信宽带的话，可以使用 cvsup 或 ftp。否则，就使用
   CTM。 .

   在编译 FreeBSD-STABLE 之前，请仔细阅读 /usr/src 里的 Makefile。
   您至少应该安装一个新的内核并重建系统， 首先做为升级过程的一部分。阅读 FreeBSD-STABLE; 邮件列表 邮件列表和 /usr/src/UPDATING， 可能让您在其它循序渐进的过程中保持更新，
   这在我们向下一发行版转移时是很有必要的。

25.6. 同步您的源码

有许多方式通过互联网(或电子邮件)与 FreeBSD
项目源码特定领域或所有领域保持更新，主要依赖于您的兴趣。 我们提供的主要服务是匿名 CVS、
CVSup，和 CTM。

虽然只更新源码树中的部分是可能的，
唯一被支持的更新过程是更新整个树、并且重编译用户区
(如：在用户空间运行的所有程序，像 /bin 和 /sbin下边的)和内核源码。
只更新源码树中的部分，或只有内核，或只有用户区 (userland)
通常会出现错误。这些问题包括有编译错误、内核崩溃 (kernel panics)、数据出错。

匿名 CVS 和 CVSup 使用 下拉(pull) 模式来更新源代码。 在 CVSup 中， 用户 (或者 cron 脚本) 会调用 cvsup 程序， 后者会同某一个 cvsupd 服务进行交互， 以更新您的文件。
您接到的更新是更新时刻最新的， 并且您只会收到那些需要的更新。 您可以很容易地限制更新的范围，
只更新那些您需要的文件。 服务器端会根据您手头已经有的文件即时地生成更新内容。 匿名 CVS 相对于 CVSup 而言要简单一些， 因为它只是对 CVS 的一种扩展， 让您可以从远程的 CVS 代码库得到更新。
CVSup 相对而言， 要比 匿名 CVS 更有效率， 然而后者却更容易使用。

另一种方法是 CTM。 这种方法并不能将您手头的代码与中央代码库中的版本进行比较， 也不能下载它们。
在主 CTM 服务器上运行的脚本会每天执行多次，
每次运行都能够自动地识别所有文件自上次运行以来所发生的变化， 如果发现有文件发生了变动，
就会压缩、 标上一个序列号， 并进行便于使用电子邮件进行传送的编码操作 (其中只包括可打印的 ASCII 字符)。 一旦接收到， 这些"CTM deltas"就会被传送给 ctm_rmail(1) 工具---
可以自动进行解码、校验和应用这些变化到用户的复制的源码里。 这个过程比 CVSup 更为有效，
而且更少占用我们的服务器资源，因为它不仅仅采用 下拉(pull) 模式，还采用 上推(push) 模式。

当然， 这样做也会带来一些不便。 如果您不经意删除了您的压缩包的部分内容， CVSup
会检测到并为您重建破坏的部分。 CTM 是不会这样做的，
如果您删除了您的源码树中的某部分(并已不能恢复)， 那么您就必须从破坏处 (从最新的CVS "base delta") 开始，使用 CTM 或 匿名 CVS 进行重建，仅仅删除坏的数据并再同步。

25.7. 重新编译 "world"

只要您根据一定版本的 FreeBSD (FreeBSD-STABLE、FreeBSD-CURRENT 等等)，
已经同步了您本地的源码树，那么您就可以使用这些源码树来重建系统。

做好备份

无需强调在行动 之前 备份整个系统是多么的重要。 尽管重新编译系统是
(如果您按照文档的指示做的话) 一件很容易完成的工作， 但出错也是在所难免的， 另外，
别人在源码里面引入的错误也可能造成系统无法引导。

请确信自己已经做过备份， 并且在手边有恢复软盘或可以引导的光盘。
您可能永远也不会用到它， 但安全第一嘛!

订阅恰当的邮件列表

FreeBSD-STABLE 和 FreeBSD-CURRENT 分支自然是 发展中的。为 FreeBSD
做贡献的都是人，偶尔也会犯错误。

有时这些错误没什么危害，只是引起您的系统生成新的诊断警告。
有时是灾难性的，并导致您的系统不能启动或破坏您的文件系统 (甚至更糟)。

如果出现了类似的问题， 贴一封"小心(heads up)"帖到相关的邮件列表里，
讲清问题的本质以及受影响的系统。在问题解决后，再贴封"解除(all clear)"声明。

如果使用 FreeBSD-STABLE 或 FreeBSD-CURRENT 而又不阅读 FreeBSD-STABLE; 邮件列表 和 FreeBSD-CURRENT 邮件列表 各自的邮件列表， 那么您是自找麻烦。

不要使用 make world

许多较早的文档推荐使用 make world 来完成这项工作。 这样做会跳过一些必要的步骤，
因此只有在您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， 您应该使用这里描述的方法。

25.7.1. 更新系统的规范方法

在更新系统时， 一定要首先查看 /usr/src/UPDATING 文件， 以便了解在 buildworld 之前需要进行的操作， 然后按照下面列出的步骤进行操作：

这些更新步骤假定您使用的是包含旧编译器、 内核以及用户态工具及配置的旧版 FreeBSD。 我们使用 "world" 来表示系统中的核心执行文件、 函数库和程序文件。 编译器是 "world" 的一部分， 但有其特殊性。

此外， 我们还假定您已经获得了较新版本操作系统的源代码。 如果您正更新的系统中的源代码也是旧版系统所附带的， 您还需要参阅 同步您的源码 来把代码同步到较新的版本。

从源代码更新系统， 有时会比初看上去的时候更麻烦一些， 另一方面， FreeBSD 的开发人员有时会不得不修改推荐的更新步骤， 特别是当出现了一些无法避免的依赖关系的时候。 这一节余下的部分， 将介绍目前推荐的更新步骤背后的原理。

成功的更新操作必须解决下面的这些问题：

- 旧的编译器可能无法编译新的内核。 (另一方面， 旧的编译器很可能有 bug。) 因此， 新的内核应该以新的编译器编译。 更具体地说， 新的编译器应在新内核开始联编之前已经完成了联编步骤。 请注意， 新的编译器并不一定需要在联编新内核之前 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。

这两个问题就是为什么我们将在后面的章节中介绍的， 需要按照 buildworld、 buildkernel、 installkernel、 installworld 的顺序来更新系统的原因。 这并不是您需要遵守推荐的更新操作的全部原因， 除了这两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， 您在安装完新内核之后， 应尽快将 world 也随之更新。
- 有些配置文件的变动必须在安装新的 world 之前完成， 而另一些配置文件的变动则有可能导致旧 world 工作不正常。 因此， 通常而言会需要两次不同的配置文件更新步骤。
- 多数情况下， 更新步骤只会替换或增加文件； 换言之， 现有的旧文件并不会被删除。 有时， 这可能会导致一些其他问题。 因此， 有时安装操作会指明， 必须在某些操作之前手工删除一些文件。 这些在未来可能会被自动化， 也可能不会自动化。

由于有这些考虑， 因此一般情况下我们建议使用下列更新步骤。 请注意， 具体的更新操作中可能会需要一些附加的步骤， 但核心的过程应该是不会轻易发生变化的：

1. make buildworld

   这步操作会联编新的编译器， 以及少量相关工具， 并在随后使用新的编译器来联编 world。 联编的结果会存放在 /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。 这种做法使得您免去了由于编译器与内核源代码不一致导致的问题。

3. make installkernel

   安装新的内核及其模块， 使系统能够以更新后的内核启动。

4. 重启系统并进入单用户模式。

   单用户模式使得更新正在运行的软件可能导致的问题减到最少。 此外， 它也使配合新内核运行旧 world 可能出现的问题减到最少。

5. mergemaster -p

这步操作会进行完成安装新的 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库中添加新的用户组或用户。
这些操作通常在上次更新之后增加了新的用户组或特殊系统用户之后是需要的， 因为 installworld
这步操作会需要这些用户或组才能顺利完成。

6. make installworld

   从 /usr/obj 中复制 world。 这步操作之后， 您在盘上的系统， 包括内核和 world 就都是新的了。

7. mergemaster

   更新余下的配置文件， 因为您的 world 已经更新完成了。

8. 重启系统。

   这步操作将加在新的内核， 以及新的 world 和更新过的配置文件。

注意， 如果您正从同一 FreeBSD 版本分支升级， 例如， 从 7.0 到 7.1， 则上述过程可能没有那么必要，
因为您不太可能遇到严重的编译器、 内核源代码、 用户态程序源代码或配置文件不匹配的情形。 旧式的
make world 然后再联编新内核的升级方法， 很可能有机会能够正常运作而完成升级工作。

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时， 便很可能遇到一些问题。

此外， 还需要注意的是， 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步骤 (例如在
installworld 之前更名或删除一些文件)。 请仔细阅读 /usr/src/UPDATING 这个文件，
特别是它的结尾部分所介绍的推荐的升级操作顺序。

由于开发人员发现不可能完全避免一些不匹配方面的问题， 这个过程一直在演化过程中。 不过幸运的是，
目前推荐的这个升级步骤， 应该能够在很长一段时间内不需要做任何调整。

总结一下， 目前推荐的从源代码升级 FreeBSD 的方法是：

```
# cd /usr/src
# make buildworld
# make buildkernel
# make installkernel
# shutdown -r now
```

有时， 可能需要额外地执行一次 mergemaster -p 才能够完成 buildworld 步骤。
这些要求， 会在 UPDATING 中进行描述。 一般而言， 您可以简单地跳过这一步，
只要进行的不是大跨度的 FreeBSD 版本升级。

在 installkernel 成功完成之后， 您需要引导到单用户模式 (举例而言， 可以在加载器提示后输入 boot -s)。
接下来执行：

```
# adjkerntz -i
# mount -a -t ufs
# mergemaster -p
# cd /usr/src
# make installworld
# mergemaster
# reboot
```

阅读进一步的说明

前面所给出的， 只是帮助您开始工作的简要说明。 要清楚地理解每一步，
特别是如果打算自行定制内核配置， 就应阅读下面的内容。

25.7.2. 阅读 /usr/src/UPDATING

在您做其它事之前，请阅读 /usr/src/UPDATING (或在您的源码里的等效的文件)。
这个文件要包含有关于您可能遇到的问题的重要信息， 或指定了您可能使用到的命令的执行顺序。如果 UPDATING 与您这里读到相矛盾，那就先依据 UPDATING。

正如先前所述，阅读 UPDATING
并不能替代订阅正确的邮件列表。两都是互补的，并不彼此排斥。

25.7.3. 检查 /etc/make.conf

检查 /usr/shared/examples/etc/make.conf 以及 /etc/make.conf。 第一个文件包含了一些默认的定义 - 它们中的绝大多数都注释掉了。 为了在重新编译系统时能够使用它们， 请把这些选项加入到 /etc/make.conf。 请注意在 /etc/make.conf 中的任何设置同时也会影响每次运行 make 的结果，
因此设置一些适合自己系统的选项是一个好习惯。

一般的用户通常会从 /usr/shared/examples/etc/make.conf 复制 CFLAGS 和 NO_PROFILE 这样的设置到 /etc/make.conf 中并令它们生效。

请考虑其他的一些选项 (例如 COPTFLAGS、 NOPORTDOCS 等等)， 看看是否合用。

25.7.4. 更新 /etc 里的文件

/etc 目录包含有除了您的系统启动时执行的脚本外大部分的系统配置信息。 有些脚本随 FreeBSD 的版本而不同。

有些配置文件在天天运行的系统里也是要使用到的。尤其是 /etc/group。

偶尔， 作为安装过程的一部分， make installworld 会要求事先创建某些特定的用户或组。
在进行升级时， 它们可能并不存在。 这会给升级造成问题。 有时， make buildworld 会检查它们是否已经存在。

最近就有个这样的例子， 当时新增了 smmsp 用户。 当用户尝试完成安装操作时， 在 mtree(8) 尝试建立 /var/spool/clientmqueue 时失败了。

解决办法是通过使用 -p 选项以构建前 (pre-buildworld) 模式运行 mergemaster(8)。
这表示只对比那些对于成功执行 buildworld 或 installworld 起关键作用的文件。 在第一次这样做时，
如果使用的是早期的不支持 -p 的 mergemaster 版本的话， 使用源码中的新版本即可。

```
# cd /usr/src/usr.sbin/mergemaster
# ./mergemaster.sh -p
```

如果您是个偏执狂 (paranoid)，
您可以检查您的系统看看哪个文件属于您已更名或删除了的那个组。

```
# find / -group GID -print
```

将显示所有 GID 组 (可以是组名也可以是数字地组 ID)所有的文件。

25.7.5. 改为单用户模式

您可能想在单用户模式下编译系统。 除了对更快处理事情显然有好处外，
重装系统将触及许多重要的系统文件， 包括所有标准系统二进制文件、库文件、包含 (include) 文件等等。
在正运行的系统 (尤其是在有活跃的用户的时候) 中更改这些文件是自寻烦恼。

另一种模式是在多用户模式下编译系统，然后转换到单用户模式下安装。 如果您喜欢这种方式，只需在建立 (build) 完成后才执行下边的步骤。 您推迟转换到单用户模式下直到您必须 installkernel 或 installworld。

从运行的系统里，以超级用户方式执行：

```
# shutdown now
```

这样就会转换到单用户模式。

除此之外， 也可以重启系统， 并在启动菜单处选择 "single user"(单用户) 选项。
这样系统将以单用户模式启动。 接着， 在 shell 提示符处执行：

```
# fsck -p
# mount -u /
# mount -a -t ufs
# swapon -a
```

这会检查文件系统，重新将 / 以读/写模式挂接， 参考 /etc/fstab 挂接其它所有的 UFS
文件系统，然后启用交换区。

如果您的 CMOS 时钟是设置为本地时间，而不是 GMT (如果 date(1)
命令输出不能显示正确的时间和地区也确有其事)， 您可能也需要执行下边的命令：

```
# adjkerntz -i
```

这样可以确定您正确的本地时区设置-不这样做， 您以后可能会碰到一些问题。

25.7.6. 删除 /usr/obj

随着重新构建系统的进行， 编译结果会放到 (默认情况下) /usr/obj 下。 这些目录会映射到 /usr/src。

通过删除这个目录， 可以加速 make buildworld 的过程， 并避免相互依赖关系等复杂的问题。

/usr/obj 中的某些文件可能设置了不可改标记 (详情参见 chflags(1))， 需要首先去掉这些标志。

```
# cd /usr/obj
# chflags -R noschg *
# rm -rf *
```

25.7.7. 重新编译基本系统

25.7.7.1. 保存输出

建议把执行 make(1) 后得到的输出存成一个文件。 如果什么地方出了错，您就会有个错误信息的备份。
尽管这样不能帮您分析哪里出了错， 但如果您把您的问题贴到某个邮件列表里就能帮助其他的人。

这样做最简单的办法是使用 script(1) 命令，同是带上参数指定存放输出的文件名。
您应在重建系统之前立即这样做，然后在过程完成时输入 exit。

```
# script /var/tmp/mw.out
Script started, output file is /var/tmp/mw.out
# make TARGET
… compile, compile, compile …
# exit
Script done, …
```

如果您这样做，就 不要 把文件存到 /tmp 里边。下次启动时，这个目录就会被清除掉。 存放的最好地方是 /var/tmp (如上个实例)或 root 的主目录。

25.7.7.2. 编译基本系统

您必须在/usr/src目录里边：

```
# cd /usr/src
```

(当然，除非您的源码是在其它地方，真是这样的话更换成那个目录就行了)。

使用 make(1) 命令重建系统。这个命令会从 Makefile (描述组成 FreeBSD 的程序应该怎样被重建，以什么样的顺序建立等等) 里读取指令。

输入的一般命令格式如下：

```
# make -x -DVARIABLE target
```

这个例子里，-x 是会传递给 make(1) 的一个选项。查看 make(1) 手册有您可用的选项例子。

-D_VARIABLE_ 传递一个变量给 Makefile。这些变量控制了 Makefile 的行为。这些同 /etc/make.conf 设置的变量一样， 只是提供了另一种设置它们的方法。

```
# make -DNO_PROFILE target
```

是另一种指定不被建立 (built) 的先定库 (profiled libraries) 的方式，协同 /etc/make.conf 里的

```
NO_PROFILE=   true    #   避免编译性能分析库
```

一起使用。

目标 (target) 告诉 make(1) 什么该做。每个 Makefile 定义了一定数量不同的"目标 (targets)"，然后您选择的目标就决定了什么会发生。

有些目标列在 Makefile 里的，但并不意味着您要执行。相反，建立过程 (build process) 利用它们把重建系统的一些必要的步骤分割成几个子步骤。

大部分的时间不需要向 make(1) 传递参数，因此您的命令看起来可能象这样：

```
# make target
```

此处 target 表示的是若干编译选项。 多数情况下， 第一个 target 都应该是 buildworld。

正如名字所暗示的，buildworld 在 /usr/obj 下边建立了一个全新的树， 然后使用另一个 target，
installworld 在当前的机器里安装它。

将这些选项分开有两个优点。 首先， 它允许您安全地完成建立 (build)，
而不对正在运行的系统的组件产生影响。 构建过程是 "自主的 (self hosted)"。 因为这样，
您可以安全地在以多用户模式运行的机器里执行 buildworld ，而不用当心不良影响。
但是依然推荐您在单用户模式时运行 installworld。

第二，允许您使用 NFS 挂接 (NFS mounts) 升级您网络里的多台计算机。如果您有三台 A、B 和 C
想进行升级，在A 执行 make buildworld 和 make installworld。 然后将 A 上的 /usr/src 和 /usr/obj 通过
NFS 挂接到 B 和 C 上， 接下来， 只需在 B 和 C 上使用 make installworld 来安装构建的结果就可以了。

尽管 world target 仍然存在，强烈建议您不要用它。

运行

```
# make buildworld
```

我们提供了一个试验性的功能， 可以在构建过程中为 make 指定 -j 参数，
令其在构建过程中同时启动多个并发的进程。 对于多 CPU 的机器而言， 这样做有助于发挥其性能。 不过，
由于编译过程中的瓶颈主要是在 IO 而不是 CPU 上， 因此它也会对单 CPU 的机器带来好处。

对典型的单 CPU 机器， 可以使用：

```
# make -j4 buildworld
```

这样， make(1) 会最多同时启动 4 个进程。 从发到邮件列表中的经验看， 这样做能带来最佳的性能。

如果您使用的机器有多颗 CPU， 并且配置了 SMP 的内核， 也可以试试看 6 到 10 的数值，
并观察是否能带来构建性能上的改善。

25.7.7.3. 耗时

联编基本系统所需的时间会受到很多因素的影响， 不过， 较新的机器应该都能在一两个小时之内完成
FreeBSD-STABLE 源代码的构建， 而无须任何技巧或捷径。 完成 FreeBSD-CURRENT 源代码的联编，
则通常需要更长一些的时间。

25.7.8. 编译和安装新内核

要充分利用您的新系统，您应该重新编译内核。 这是很有必要的，因为特定的内存结构已经发生了改变，像
ps(1) 和 top(1) 这样的程序会不能工作， 除非内核同源码树的版本是一样的。

最简单、最安全的方式是 build 并安装一个基于 GENERIC 的内核。虽然 GENERIC
可能没有适合您的系统的所有必要的设备， 但它包括了启动您的系统到单用户模式所必需的内容。
这是个不错的检测新系统是否工作正常的测试。在从 GENERIC 启动、核实系统可以工作后， 您就可以建立
(build) 一个基于您的正常内核配置文件的新的内核了。

在 FreeBSD 中， 首先完成 build world 然后再编译新内核非常重要。

如果您想建立一个定制内核，而且已经有了配置文件， 只需象这样使用
KERNCONF=MYKERNEL：

```
# cd /usr/src
# make buildkernel KERNCONF=MYKERNEL
# make installkernel KERNCONF=MYKERNEL
```

注意，如果您已把 内核安全级别(kern.securelevel) 调高到了 1 以上，而且还设置了 noschg 或相似的标识到了您的内核二进制里边，您可能会发现转换到单用户模式里使用 installkernel 是很有必要的。 如果您没有设置它， 则应该也能毫无问题地在多用户模式执行这两个命令。 请参考 init(8) 以了解更多关于 内核安全级(kern.securelevel) 的信息；查看 chflags(1) 了解更多关于不同文件标识的信息。

25.7.9. 重启到单用户模式

您应该单用户模式测试新内核。照改为单用户模式处的说明去做。

25.7.10. 安装编译好的新系统

您现在应使用 installworld 来安装新的系统二进制。

执行

```
# cd /usr/src
# make installworld
```

如果在 make buildworld 的命令行指定了变量，您就必须在 make installworld 命令行里指定同样的变量。 对于其它的选项并不是必需的，如，-j 就不能同 installworld 一起使用。

举例，您执行了：

```
# make -DNO_PROFILE buildworld
```

您就必须使用：

```
# make -DNO_PROFILE installworld
```

来安装结果，否则就要试着安装先定 (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

重新编译整个系统不会使用新的或改过的配置文件更新某些目录 (尤其像 /etc、/var 和 /usr)

更新这些文件最简单的方式就是使用 mergemaster(8)，手工去做也是可以的，只要您愿意。 不管您选择哪一种，一定记得备份 /etc 以防出错。

25.7.11.1. mergemaster

mergemaster(8) 工具是个 Bourne 脚本，用于检测 /etc 和 /usr/src/etc 源码树里边的配置文件的不同点。 这是保持系统配置文件同源码树里的一起更新的推荐方式。

在提示符里简单地输入 mergemaster 就可以开始，并观看它的开始过程。mergemaster

会建立一个临时的根(root)环境，在 / 下，
放置各种系统配置文件。这些文件然后同当前安装到您系统里的进行比较。 此时，不同的文件会以 diff(1)
格式进行显示，使用 + 符号标识增加或修改的行，- 标识将完全删除的行或将被替换成新行。查看 diff(1)
手册可以得到更多关于 diff(1) 语法和文件不同点怎样显示的信息。

mergemaster(8) 会给您显示每个文件的不同处， 这样您就可以选择是删除新文件 (相对临时文件)，
是以未改状态安装临时文件，是以当前安装的文件合并临时文件， 还是再看一次 diff(1) 结果。

"选择删除临时文件"将使 mergemaster(8) 知道我们希望保留我们当前的文件不改，并删除新的。
并不推荐这个选择，除非您没有更改当前文件的理由。任何时候在 mergemaster(8) 提示符里输入
?，您就会得到帮助。 如果选择跳过文件，将在其它文件处理完后再次进行。

"选择安装未修改临时文件"将会使新文件替换当前的。 对大部分未改的文件，这是个最好的选择。

"选择合并文件"将为您打开一个文本编辑器， 里边是两个文件的内容。您现在就可以一边合并它们，
一边在屏幕里查看，同时从两者中选取部分生成最终文件。 当两个文件一起比较时，l
键会选择左边的内容， r 会选择右边的。最终的输出是由两个部分组成的一个文件，
用它就可以安装了。这个选项通常用于用户修改了设置的文件。

"选择再次查看 diff(1) 结果"将会在提供给选择之前， 显示文件的不同处，就象 mergemaster(8)
所做的一样。

在 mergemaster(8) 完成了对系统文件的处理后， 您会得到其它的选项。mergemaster(8)
可能会问您是否要重建密码文件， 并在最后提示您是否要删除余下的临时文件。

25.7.11.2. 手动更新

如果想要手工更新，但不要只是从 /usr/src/etc 把文件复制到 /etc 就了事。有些文件是必须先"安装"的。
这是因为 /usr/src/etc 目录并 不是 想像的那样是 /etc 目录的一个复制。事实上，有些是文件是 /etc
有的，而 /usr/src/etc 里边没有。

如果您使用 mergemaster(8) (作为推荐)，您可以向前跳到 下一节。

手工做最简单的方式是安装这些文件到一个新的目录，完成后再来查找不同处。

备份您已有的 /etc

虽然，理论上，没有什么会自动访问这个目录， 事情还是做稳操胜当一点。复制已有 /etc
到一个安全的地方，如：

```
# cp -Rp /etc /etc.old
```

-R 完成递归复制 (译者注：即可以复制目录以下的所有内容)，-p
保留文件的时间、所属等等。

您需要建立一个虚目录 (a dummy set of directories) 来安装新的 /etc 和其它文件。 /var/tmp/root
是个不错的选择， 除此之外，还有一些子目录是需要的。

```
# mkdir /var/tmp/root
# cd /usr/src/etc
# make DESTDIR=/var/tmp/root distrib-dirs distribution
```

这样就建好了需要的目录结构，然后安装文件。在 /var/tmp/root 下建立的大部分子目录是空的，
而且要删除掉。最简单的方式是：

```
# cd /var/tmp/root
```

```
# find -d . -type d | xargs rmdir 2>/dev/null
```

这样会删除所有的空目录。(标准的错误信息被重定向到了 /dev/null，以防止关于非空目录的警告。)

/var/tmp/root 现在包含了应放在 / 下某个位置的所有文件。
您现在必须仔细检查每一个文件，检测它们与您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"."在开头。在写的时候，像这样唯一的文件是
/var/tmp/root/ 和 /var/tmp/root/root/ 里 shell
启动文件，尽管可能有其它的(依赖于您什么时候读取这个)。 确信使用 ls -a 可以看到它们。

最简单的方式是使用 diff(1) 去比较两个文件：

```
# diff /etc/shells /var/tmp/root/etc/shells
```

这会显示出 /etc/shells 文件和新的 /var/tmp/root/etc/shells 文件的不同处。
用这些来决定是合并您已做的变化还是复制您的旧文件过来。

使用日戳 (Time Stamp) 命名新的 Root(根)目录(
/var/tmp/root)，这样您可以轻松地比较两个版本的不同
频繁重建系统意味着必须频繁更新 /etc，而这可能会有点烦琐。

在合并到 /etc 的文件里， 最新更改的您可以做个复制，由此加快这个(指更新)过程。
下边就给出了一个怎样做的主意。

1. 像平常一样建立系统 (Make the world)。当您想更新 /etc 和其它目录里，
   给目标目录一个含有当前日期的名字。假如您是 1998 年 2 月 14
   日做的，您可以执行下边的：

   ```
   # mkdir /var/tmp/root-19980214
   # cd /usr/src/etc
   # make DESTDIR=/var/tmp/root-19980214 \
     distrib-dirs distribution
   ```

2. 如上边列出的，从这个目录合并变化。

   在您完成后，不要 删除 /var/tmp/root-19980214 目录。

3. 在您下载了最新版的源码并改过后，执行第一步。
   这样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. 您现在能看到两个目录间的不同了---在隔周的时间里使用 diff(1) 建立递归 diff
   产生的不同：

   ```
   # cd /var/tmp
   # diff -r root-19980214 root-19980221
   ```

   一般情况下，这两种间的不同处比 /var/tmp/root-19980221/etc 和 /etc
   之间的不同要小很多。 因为不同点更小，也就更容易把这些变化移到您的 /etc
   目录里边。

5. 您现在可以删除早先的两个 /var/tmp/root-* 目录：

```
# rm -rf /var/tmp/root-19980214
```

6. 每次您需要合并这些变化到 /etc 里，就重复这个流程。

您可以使用 date(1) 自动产生目录的名称：

```
# mkdir /var/tmp/root-`date "+%Y%m%d"`
```

25.7.12. 重启

现在完成了。在您检查所有内容都放置正确后， 您可以重启系统了。只是简单的 shutdown(8) 可以这样做：

```
# shutdown -r now
```

25.7.13. 结束

恭喜！您现在成功升级了您的 FreeBSD 系统。

如果还有轻微的错误，可以轻易地重建系统的选定部分。 例如，在部分升级或合并 /etc 时，您不小心删除了 /etc/magic，file(1) 命令就会停止工作。这种情况下，执行下边进行修复：

```
# cd /usr/src/usr.bin/file
# make all install
```

25.7.14. 问题

25.7.14.1. 每个变化您都须要重建系统吗?

这个不好说，因为要看变化的情况。如，如果您刚运行了 CVSup，并得到下边更新的文件：

```
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
```

这就不必重建整个系统。您只需到相关的子目录里执行 make all install，仅此而已。
但是，如果有重大变化，如 src/lib/libc/stdlib， 那么您就要重建系统或至少静态连接的那些部分 (除了您增加的部分都是静态连接的)。

在这天后，就是您的事了。要是说每两个星期重建一下系统的话，
您可能会高兴。或者您可能只想重做改变过的部分， 确信您能找出所有依赖关系。

当然，所有这些依赖于您想升级的频率，和您是否想跟踪 FreeBSD-STABLE 或 FreeBSD-CURRENT。

25.7.14.2. 我的编译失败，并伴随有许多 11信号 11 (或其它的数字信息) 号错误。是怎么回事呀?

这个通常表示硬件错误。 (重)建系统是个强压测试系统硬件的有效地方式， 并且常常产生内存错误。
这些正好表示它们自已做为编译器离奇地死于收到的奇怪信息。

一个确信的指示器是如果重新开始 make，并且整个过程中会死在不同的点上。

对于这种情况，您没有什么可做的，除了更换机器里的部件，看是哪一个坏了。

25.7.14.3. 我完成后可以删除 /usr/obj 吗?

简短地说，可以。

/usr/obj 包含了所有在编译阶段生成的目标文件。通常， 在 make buildworld
过程中第一步之一就是删除这个目录重新开始。 这种情况下，在您完成后，保留 /usr/obj
没有多大意义，还可释放一大堆磁盘空间(目前是 2 GB 左右)。

不过， 如果您很了解整个过程， 也可以让 make buildworld 跳过这一步。
这会让后续的联编过程执行得更快， 因为大部分的源码都不必再进行编译了。
这样做的负面效果是它可能会触发一些由于敏感的依赖关系导致的问题，
这些问题会导致联编以奇怪的方式出错并失败。 这在 FreeBSD 邮件列表里经常引起沸腾， 当有人抱怨他们 build 失败时，并没意识到这是因为自已是想抄近路。

25.7.14.4. 中断的 build 可以被恢复吗?

依赖于您在您找到问题之前整个过程进行了多远。

一般而言 (当然这并不是硬性规定)， make buildworld 的过程中将会首先构建新版的基本构建工具 (例如 gcc(1)， 以及 make(1)) 和系统库。 随后会安装这些工具和库。
这些新版本的工具和库在随后将被用于重新编译和连接它们本身。 整个系统 (现在包括了常规的用户程序，例如 ls(1) 或 grep(1)) 会同新版的系统文件一起被重新构建。

如果您正处于最后一个阶段， 并且了解它 (因为您已经看过了所保存的输出) 则可以 (相当安全地) 做:

```
… 问题修复 …
# cd /usr/src
# make -DNO_CLEAN all
```

这样就不会取消先前的 make buildworld 所做的工作了。

在"make buildworld"的输出中如果看到如下信息:

```
----------------------------------------------------------
Building everything..
----------------------------------------------------------
```

出现在 make buildworld 的输出中， 则这样做应该不会有什么问题。

如果没有看到这样的信息， 或者您不确定， 则从头开始构建将是万无一失的做法。

25.7.14.5. 我怎样加快建立系统的速度?

- 以单用户模式运行
- 把 /usr/src 和 /usr/obj
  目录放到不同磁盘里的独立文件系统里。如果可能，这些磁盘在不同的磁盘控制器里。
- 更好的，是把这些文件系统放置到多个使用 ccd(4) (连接磁盘驱动器—concatenated disk

driver)设备的磁盘里。

- 关掉 profiling (在 /etc/make.conf 里设置 "NO_PROFILE=true")。您差不多用不了它。
- 在 /etc/make.conf 里也为 CFLAGS 设置上 -O -pipe。 最佳优化 -O2 会更慢，而且 -O 和 -O2 之间的优化差别基本上可以忽略。 -pipe 让编译器使用管道而不用临时文件进行通信，这样可以减少磁盘存取 (以内存作为代价)。
- 传递 -jn 选项给 make(1) 以便并发运行多个进程。 这样就不会考虑您的是否是单个或多个处理器机器。
- 存放 /usr/src 的文件系统可以使用 noatime 选项来挂接 (或重新挂接)。这样会防止文件系统记录文件的存取时间。 您可能并不需要这些信息。

```
# mount -u -o noatime /usr/src
```

这个例子里假定 /usr/src 是在它自己的文件系统里。如果不是 (例如假设它是 /usr 的部分)，那么您就需要那个文件系统挂接点， 而不是 /usr/src。

- 存放 /usr/obj 的文件系统可以使用 async 选项被挂接 (或重新挂接)。 这样做将启用异步写盘。换句话说， 对应用程序而言写会立即完成， 而数据则延迟几秒才会写到盘里。这样做能够成批地写下数据， 从而极大地改善性能。

注意， 这个选项会使您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时，
文件系统进入不可恢复状态的概率。

如果在这个文件系统里 /usr/obj
是很关键的，这不是问题。如果您有其它有价值的数据在同一个文件系统，
那么在您使用这个选项这前，确认备份一下。

```
# mount -u -o async /usr/obj
```

同上，如果 /usr/obj
不在自己的文件系统里，使用相关挂接点的名字把它从例子里边替换掉。

25.7.15. 如果出现了错误我该怎么办?

绝对确信您的环境没有先前 build 留下的残余。这点够简单。

```
# chflags -R noschg /usr/obj/usr
# rm -rf /usr/obj/usr
# cd /usr/src
# make cleandir
# make cleandir
```

不错，make cleandir 真的要执行两次。

然后重新开始整个过程，使用 make buildworld 开始。

如果您还有问题，就把错误和 uname -a 的输出发送到 FreeBSD 一般问题邮件列表
邮件列表。准备回答其它关于您的设置的问题!

25.8. 删除过时的文件、 目录和函数库

在 FreeBSD 的开发过程中， 随时可能会出现一些文件或其内容过时的情况。
这种情况有可能是由于其功能在其它地方实现了， 函数库的版本号增加， 或完全从基本系统中删去，
等等。 一般的联编和更新过程并不会删去这些旧的文件、 函数库或目录， 在更新系统之后，
应及时予以清理。 清理的好处是这些文件不会再继续占用存储 (以及备份) 空间， 另外，
如果旧的函数库或文件中存在安全或可靠性问题， 您也应更新到新的函数库，
以避免安全隐患或崩溃情形的发生。 过时的文件、 目录和函数库会列在 /usr/src/ObsoleteFiles.inc 中。
接下来将介绍在系统更新过程中如何删去这些过时的文件。

我们假定您已经按照 更新系统的规范方法 介绍的步骤完成了更新操作。 在 make installworld 和 mergemaster 命令完成之后， 您应使用下面的命令检查系统中是否存在过时的文件或库：

```
# cd /usr/src
# make check-old
```

如果有过时的文件， 则可以用下面的命令来删除：

```
# make delete-old
```

参阅 /usr/src/Makefile 可以了解其他 target 的功用。

在删除文件时， 系统会针对每个文件都给出提示。 您可以跳过这些提示， 并让系统自动完成删除操作，
方法是使用 make 变量 BATCH_DELETE_OLD_FILES， 具体做法如下：

```
# make -DBATCH_DELETE_OLD_FILES delete-old
```

您也可以用 yes 命令和管道来达到类似的目的：

```
# yes|make delete-old
```

删去过时的文件， 有可能会破坏现有的依赖这些文件的应用程序。 对于旧的函数库来说，
这种问题出现的可能性更大。 绝大多数情况下， 您应重新联编使用旧库的所有的程序、
port 或函数库之后再执行 make delete-old-libs。

在 Ports Collection 中提供了一些检测动态连接库依赖关系的工具， 例如 sysutils/libchk 和 sysutils/bsdadminscripts。

过时的动态连接库可能会与新库冲突， 导致类似这样的警告消息：

```
/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， 需要确认安装这个库的 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、 重新联编并安装 port。 您可以使用 ports-mgmt/portmaster 或 ports-mgmt/portupgrade 工具来自动完成这些操作。 在确认所有的 port 都重新联编， 并且不再使用旧库以后，
您就可以用下面的命令来删除它们了：

```
# make delete-old-libs
```

25.9. 跟踪多台机器

如果您有多台机器想跟踪同样的源码树， 那么让它们都下载源码并重建所有东西，看起有点浪费资源：
磁盘空间、网络带宽以及 CPU 周期。 解决的办法是让一台机器处理大部分的工作，而其它的机器通过 NFS 挂接 (mount) 这些工作。这部分列举了一种这样做的方法。

25.9.1. 准备

首先，确定一批机器，运行的二进制代码是同一套---我们称作 构建集群 (build set)。
每台机器可以使用不同的定制内核， 但它们运行的是相同的用户区二进制文件(userland binaries)。
从这批机器中选择一台机器做为 构建机器(build machine)。
这将是用于构建(build)系统和内核的机器。想像一下，它应该是一台快速的机器， 有足够的空余的 CPU 来执行make buildworld。 您也想要选一台机器做为 测试机器(test machine)，
这个将用于软件的更新生成产品之前对他们进行测试。这个 必须 是一台您能提供的平时也可使用的机器。
它可以是"构建机器"，但没这个必要。

在这个"构建集群"里的所有机器需要从同一台机器、 同一个点上挂接 /usr/obj 和 /usr/src。理想地，
它们在"构建机器"上的两个不同的驱动器里， 但是在那台机器上可以进行 NFS
挂接。如果您有多个"构建集群"， /usr/src 应该在某个"构建机器"上， 而在其它机器上进行 NFS 挂接。

最后，确认"构建集群"里所有机器上的 /etc/make.conf 和 /etc/src.conf 与"构建机器"里的相同。
这意味着"构建机器"必须构建部分基本系统用于 "构建集群"里所有机器的安装。同样，
每台"构建机器"要有它自己的内核名字，使用 /etc/make.conf 里的 KERNCONF
进行设置，并且每台"构建机器"应该把它们列在 KERNCONF 里，同时把自己的内核列在最前。
"构建机器"的 /usr/src/sys/arch/conf
里一定要有每台机器的内核配置文件，如果它想构建它们的内核的话。

25.9.2. 基本系统

既然所有的妥当了，就准备构建所有的东西。如编译基本系统中描述的一样在"构建机器"上构建内核和系统， 但是什么也不安装。在构建结束后，转到"测试机器"上， 安装您刚构建的内核。如果这台机器通过 NFS 挂接了 /usr/src 和 /usr/obj， 在您重启到单用户模式里，您需要启动网络然后挂接他们。
最简单的方式是启动到多用户模式下，然后执行 shutdown now
转到单用户模式。一旦进入，您就可以安装新的内核和系统，并执行
mergemaster，就像平常一样。完成后， 重启返回到一般多用户模式操作这台机器。

在您确信所有在 "测试机"里都工作正常后， 就使用相同的过程在
"构建集群"里的其它机器里安装新的软件。

25.9.3. Ports

类似的想法是使用 ports 树。 第一个关键的步骤是从同一台计算机上挂接 /usr/ports 到 "构建集群"
里的全部计算机。 然后正确设置 /etc/make.conf 共享 distfiles。您应把 DISTDIR
设置到一个共享的目录里， 那里可以被任何一个 root 用户写入， 并且是由您的 NFS 挂接映射的。
设置每一台机器的 WRKDIRPREFIX 到一个本地构建 (build) 目录。最后，如果您要构建和发布包 (packages)，那么您应该设置 PACKAGES 到一个类似于 DISTDIR 的目录。

Chapter 26. DTrace

26.1. 概述

DTrace，也称为动态跟踪，是由 Sun™ 开发的一个用来在生产和试验性生产系统上找出系统瓶颈的工具。
在任何情况下它都不是一个调试工具， 而是一个实时系统分析寻找出性能及其他问题的工具。

DTrace 是个特别好的分析工具，带有大量的帮助诊断系统问题的特性。
还可以使用预先写好的脚本利用它的功能。 用户也可以通过使用 DTrace D
语言创建他们自己定制的分析工具， 以满足特定的需求。

在阅读了这一章节之后，你将了解：

- DTrace 是什么，它提供了些哪些特性。
- DTrace 在 Solaris™ 与 FreeBSD 上的实现的差别。
- 如何在 FreeBSD 上开启和使用 DTrace。

在阅读这一章节之前，你应该了解：

- 了解 UNIX® 和 FreeBSD 的基本知识 (UNIX 基础)。
- 熟悉基本的内核配置/编译 (配置FreeBSD的内核).
- 熟悉 FreeBSD 有关的安全知识 (安全)。
- 了解如何获取和重新编译 FreeBSD 源代码 (更新与升级 FreeBSD)。

这项特性目前仍被认为是试验性的。
有些选项功能性缺失，另有一些可能还无法运行。最终，
这个特性会适合用于生产，届时这篇文档也会做些适当的修改。

26.2. 实现上的差异

虽然 FreeBSD 上的 DTrace 与 Solaris™ 上的非常相似， 在继续深入之前我们需要说明一下存在的差异。
用户首先会注意到的便是 FreeBSD 上的 DTrace 需要明确地被启用。 DTrace
相关的内核选项和模块必须开启后才能正常工作。 稍后我们会作详细介绍。

有一个 DDB_CTF 内核选项用来开启从内核与内核模块加载 CTF 数据。 CTF 是 Solaris™ Compact C Type Format 封装了类似于 DWARF 和 venerable stabs 简化的调试信息。CTF 数据是由 ctfconvert 和 ctfmerge 工具加入二进制文件的。ctfconvert 工具分析由编译器生成的 DWARFELF 调试 section，ctfmerge 合并目标文件的 CTFELF section 到可执行文件或共享库。更多关于在启用 FreeBSD 内核上启用此项的详细内容即将完成。

比起 Solaris™， FreeBSD 有几个不同提供器。 最值得注意的是 dtmalloc 提供器， 可以让你根据类型追踪 FreeBSD 内核中的 malloc()。

只有 root 可以使用 FreeBSD 上的 DTrace。 这是由系统安全上的差异造成的，Solaris™ 提供了一些 FreeBSD 上还未实现的低层的安全检查。 同样， /dev/dtrace/dtrace 也被严格的限制为仅供 root 用户访问。

最后，DTrace 为 Sun™ CDDL 许可下发布的软件。随 FreeBSD 发行的 Common Development and Distribution License 可以在查阅 /usr/src/cddl/contrib/opensolaris/OPENSOLARIS.LICENSE 或者通过 http://www.opensolaris.org/os/licensing 查看在线版本。

这个许可表示带有 DTrace 选项的 FreeBSD 内核仍为 BSD 许可； 然而， 以二进制发布模块，
或者加载二进制模块则需遵守 CDDL。

26.3. 启用 DTrace 支持

在内核配置文件中加入以下几行来开启对 DTrace 的支持：

```
options     KDTRACE_HOOKS
options     DDB_CTF
```

使用 AMD64 架构的需要在内核配置文件中加入如下这行：

```
options     KDTRACE_FRAME
```

此选项提供了对 FBT 特性的支持。 DTrace 可以在没有此选项的情况下正常工作，但是函数边界跟踪便会有所限制。

所有的源代码都必须重新使用 CTF 选项编译安装。重新编译 FreeBSD 源代码可以通过以下的命令完成：

```
# cd /usr/src

# make WITH_CTF=1 kernel
```

系统需要重新启动。

在重新启动和新内核载入内存之后，需要添加 Korn shell 的支持。因为 DTrace 工具包有一些工具是由 ksh 写的。安装 shells/ksh93。 同样也可以通过 shells/pdksh 或者 shells/mksh 使用这些工具。

最后是获得最新的 DTrace 工具包。 当前版本可以通过下面的链接找到 http://www.opensolaris.org/os/community/dtrace/dtracetoolkit/。
这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们。

26.4. 使用 DTrace

在使用 DTrace 的功能之前，DTrace 设备必须存在。 使用如下的命令装载此设备：

```
# kldload dtraceall
```

DTrace 支持现在应该可以使用了。 管理员现在可以使用如下的命令查看所有的探测器：

```
# dtrace -l | more
```

所有的输出都传递给 more 工具， 因为它们会很快超出屏幕的显示区域。此时，DTrace 应该被认为是能够正常工作的了。现在是该考察工具包的时候了。

工具包是实现写好的一堆脚本，与 DTrace 一起运行来收集系统信息。
有脚本用来检查已打开的文件，内存，CPU 使用率和许多东西。使用如下的命令解开脚本：

```
# 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。

所有这些脚本都需要修改它们的内容。那些指向 /usr/bin/ksh 需要修改成 /usr/local/bin/ksh，另外使用 /usr/bin/sh 需要变更为 /bin/sh，最后还有使用 /usr/bin/perl 的需要变更为 /usr/local/bin/perl。

此刻还需谨慎提醒一下读者 FreeBSD 的 DTrace 支持仍是 不完整的 和 试验性 的。
这些脚本中的大多数都无法运行，因为它们过于针对 Solaris™
或者使用了目前还不支持的探测器。

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。这两个脚本便是我们下一部分将要探讨的：

hotkernel 被设计成验明哪个函数占用了内核时间。 正常运行的话，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。
紧接着中止之后，脚本便会一张内核函数与测定时间的列表， 使用增量排序输出：

```
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`_isitmyx                          4  0.0%
kernel`find_instance                     4  0.0%
kernel`_mtx_unlock_flags                 5  0.0%
kernel`syscall                           5  0.0%
kernel`DELAY                             5  0.0%
0xc108a253                               6  0.0%
kernel`witness_lock                      7  0.0%
kernel`read_aux_data_no_wait             7  0.0%
kernel`Xint0x80_syscall                  7  0.0%
kernel`witness_checkorder                7  0.0%
kernel`sse2_pagezero                     8  0.0%
kernel`strncmp                           9  0.0%
kernel`spinlock_exit                    10  0.0%
kernel`_mtx_lock_flags                  11  0.0%
kernel`witness_unlock                   15  0.0%
kernel`sched_idletd                    137  0.3%
0xc10981a5                           42139 99.3%
```

这个脚本也能与内核模块一起工作。要使用此特性， 用 -m 标志运行脚本：

```
# ./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 脚本捕捉并打印给定 PID 的系统调用时间。 在下面的例子中，新生成了一个 /bin/csh 实例。procsystime 执行后则等待在新运行的 csh 上键入一些命令。 这是测试的结果：

```
# ./procsystime -n csh
Tracing... Hit Ctrl-C to end...
^C

Elapsed Times for processes csh,

      SYSCALL       TIME (ns)
       getpid           6131
     sigreturn           8121
        close          19127
        fcntl          19959
          dup          26955
      setpgid          28070
         stat          31899
    setitimer          40938
       wait4          62717
    sigaction          67372
  sigprocmask         119091
 gettimeofday         183710
        write         263242
       execve         492547
        ioctl         770073
```

```
      vfork        3258923
  sigsuspend       6985124
       read     3988049784
```

正如显示的那样，read 系统调用似乎使用了最多的纳秒单位时间， getpid() 系统调用使用了最少的时间。

26.5. D 语言

DTrace 工具包包括了很多由 DTrace 特殊语言写成的脚本。 在 Sun™ 的文档中称这类语言为 "D 语言"，它与 C++ 非常类似。对此语言更深入的讨论则超出了这篇文章的范围。 更多相关的讨论可以在 http://wikis.sun.com/display/DTrace/Documentation 找到。

Part IV: 网络通讯

FreeBSD 是目前以高性能网络服务为目的而部署范围最广的操作系统之一。 讨论这些话题的章节包括：

- 串口通讯
- PPP 和以太网上的 PPP
- 电子邮件
- 运行网络服务
- 防火墙
- 其他进阶网络话题

这些章节主要供您在需要时参考。 不必按特定的顺序来阅读它们， 此外， 您开始在网络中使用 FreeBSD 之前也不需要先把它们都读完。

Chapter 27. 串口通讯

27.1. 概述

UNIX® 一直都是支持串口通讯的。事实上， 早期的 UNIX® 系统就是利用串口线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键盘， 现在这些已经发生了很大的变化。
这章将介绍一些利用 FreeBSD 进行串口通讯的方法。

读完这章，您将了解到：

- 如何通过终端连接到您的FreeBSD系统。
- 如何使用modem拨号到远程主机。
- 如何允许远程用户通过modem登录到您的系统。
- 如何从串口控制台引导您的系统。

阅读这章之前，您应当了解：

- 如何配置和安装一个新的内核 (配置FreeBSD的内核)。
- 理解 UNIX® 的权限和进程 (UNIX 基础)。
- 准备您打算在 FreeBSD 中使用的串口设备 (modem 或多插口卡)的技术参考手册。

27.2. 介绍

从 FreeBSD 8.0 开始， 用于串口的设备节点从 /dev/cuadN 改为了 /dev/cuauN； 从 /dev/ttydN 改为了 /dev/ttyuN。 FreeBSD 7.X
用户需要根据实际情况对这份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 - 如您的计算机

DCE
: 数据通讯设备 - 如您的modem

RS-232
: 用于硬件串口通讯的EIA标准

当讨论通讯数据速度的时候，这节不会使用术语
"baud"。Baud指电气标准传输率，它已经使用了很长时间， 而 "bps" (bits per second)
才是正确使用的术语 (至少它不会打扰那些爱争吵的家伙)。

27.2.2. 线缆和端口

要将 modem 或终端与您的 FreeBSD 系统相连， 您的计算机需要一个串口，
以及用于连接串口设备所需的线缆。 如果您比较熟悉硬件及所需要的电缆， 则可以跳过这节。

27.2.2.1. 线缆

串口线缆有许多不同的种类。 最常见的两种类型是 null-modem 线缆和标准 ("直联") RS-232 线缆。
您的硬件说明书中会介绍应使用的线缆种类。

null-modem 电缆会直接传送某些信号， 如 "Signal Ground" (信号地)， 但对其他信号进行交换。 例如， "Transmitted Data" (数据发送) 引脚是连到另一端 "Received Data" (数据接收) 引脚的。

也可以自行制作 null-modem 电缆给终端使用 (例如， 为了品质的要求)。 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) 线， 但这通常都被省掉。 某些终端在只有引脚 2、 3 和 7 的时候， 就已经能够正常使用了， 而其他一些， 则需要下面例子中所展示的不同的配置。

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

这里还有两种目前比较流行的其他接线方式。

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
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 电缆

信号	引脚 #		引脚 #	信号
RD	2	DB-9 到 DB-25 Null-Modem 电缆	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

信号	引脚 #		引脚 #	信号
CTS	8	接到	4	RTS

当某一段连接器上的一个引脚需要连接到对端的一对引脚时，
通常是将那一对引脚使用一短线连接，而使用长线接到另一端的那个引脚。

上面的设计似乎更为流行。在其他变种中 (在 RS-232 Made Easy 这本书中进行了详细介绍) 则是 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线缆

标准的串口电缆会直接传送所有 RS-232C 信号。也就是说，一头的 "Transmitted Data" 引脚，会直接接到另一头的 "Transmitted Data" 引脚。这包括将调制解调器接到您的 FreeBSD 系统上的那种电缆，同样也适用于某些型号的终端。

27.2.2.2. 端口

串口是FreeBSD主机与终端传输数据的设备。这节描述了端口的种类和它们在 FreeBSD 上是如何编址的。

27.2.2.2.1. 端口的种类

有好几种串口。在采购或制作线缆之前，您应确认它能够适合您的终端以及 FreeBSD 系统。

绝大多数终端都提供 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。如果您的 PC 上有多插口串口卡，则可能有 RJ-12 或 RJ-45 口。

请参见您硬件的文档以了解所用接口的规格。此外，您也可以通过观察外观来了解所用的端口。

27.2.2.2.2. 端口名称Port Names

在FreeBSD中，您可以通过 /dev 目录中的一个记录来访问每个串口。有两种不同的记录：

- 呼入端口的名字是 /dev/ttyuN，其中 N 是端口的编号，从零开始计数。一般来说，您使用呼入端口作为终端。呼入端口要求数据线使用载波检测 (DCD) 信号来工作。
- 呼出端口的名字是 /dev/cuauN。通常并不使用呼出端口作为终端，而只用于调制解调器。如果串口线或终端不支持载波检测信号，则可能必须要使用呼出端口。

如果您已经连接一个终端到第一个串口 (在 MS-DOS® 上是COM1)，则可以使用 /dev/ttyu0 来作为终端。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，这些是 COM1，COM2，COM3，和 COM4。FreeBSD 目前支持 "dumb" 多口串口卡，如 BocaBoard 1008 和 2016，以及许多 Digiboard 和 Stallion Technologies 制造的智能多接口卡。不过，默认的内核只会寻找标准的COM端口。

要看看您的内核是否支持您的串口，只要在内核启动时查看一下启动信息，或使用 /sbin/dmesg 命令重新检测内核启动信息。特别的，寻找以sio字符启动的信息。

如果想只察看包含 sio 一词的消息，可以使用下面的命令：

```
# /sbin/dmesg | grep 'sio'
```

例如，在一个带有4个串口的系统上，这些是串口特定的内核启动信息：

```
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
```

如果内核未能认出所有的串口， 可能需要通过修改 /boot/device.hints 文件来进行一些配置。 此外，
也可以注释或完全删除掉您没有的设备。

请参见 sio(4) 联机手册来了解关于串口， 以及多插口卡配置的进一步细节。
如果您正使用一个在不同版本的 FreeBSD 上的文件请务必小心， 因为设备参数和语法发生了变化。

这里端口 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，这些都是各自端口相应的端口地址。
中断4，3，5，9都是经常用的中断。也要注意有些正常的串口可能 无法
在一些ISA总线的PC上共享中断 (多插口板卡有板载的电子设备，允许在板上所有 16550A 的设备共享一个或两个中断请求)。

27.2.4. 设备特殊文件

在内核中， 大多数设备都是通过 "设备特殊文件" 来访问的， 这些文件一般位于 /dev 目录中。 sio 是通过 /dev/ttyuN (呼入) 和 /dev/cuauN (呼出) 设备来访问的。 此外， FreeBSD 也提供了初始化设备 （/dev/ttyuN.init 和 /dev/cuauN.init） 以及锁设备 （/dev/ttyuN.lock 和 /dev/cuauN.lock）。
初始化设备用于在打开端口时初始化其通讯参数， 例如使用 RTS/CTS 信号进行流控制的调制解调器的 crtscts。 锁设备则用于在端口上提供一个锁标志， 防止用户或程序改变特定的参数； 请参见 termios(4)、 sio(4)， 以及 stty(1) 的联机手册， 以了解关于终端配置、 锁和初始化设备，
以及配置终端参数的详细信息。

27.2.5. 串口配置

ttyuN (或 cuauN) 设备是您将要打开的应用程序的一般设备。 当进程打开某个设备时， 它将有一个终端 I/O 设置的默认配置。 您可以在命令行看看这些设置：

```
# stty -a -f /dev/ttyu1
```

当您修改了这个设备的设置，这个设置会生效，除非设备被关闭。 当它被重新打开时，它将回到默认设置。
要修改默认设置，您可以打开和调整 "初始状态" 设备的设置。例如， 要为ttyu5 打开 CLOCAL
模式，8位通讯和默认的 XON/XOFF 流控制， 输入：

```
# stty -f /dev/ttyu5.init clocal cs8 ixon ixoff
```

串口设备的系统级初始化， 是由 /etc/rc.d/serial 来控制的。 这个文件会影响串口设备的默认设置。

为了防止应用程序修改某些设置， 应修改 "lock state"(锁状态) 设备。 例如， 要把 ttyu5 的速率锁定为 57600 bps， 输入：

```
# stty -f /dev/ttyu5.lock 57600
```

现在，一个打开ttyu5 和设法改变端口速度的应用程序将被固定在57600bit/s。很自然地，
您需要确定初始状态，然后用root帐户锁定状态设备的写入功能。

很显然，您应该只让 root 用户可以初始化或锁定设备的状态。

27.3. 终端

从 FreeBSD 8.0 开始， 用于串口的设备节点从 /dev/cuadN 改为了 /dev/cuauN； 从 /dev/ttydN 改为了 /dev/ttyuN。 FreeBSD 7.X
用户需要根据实际情况对这份文档中的例子进行必要的调整。

当您在计算机控制台或是在一个连接的网络上时， 终端提供了一个方便和低成本的访问 FreeBSD 系统的方法。 这节描述了如何在 FreeBSD 上使用终端。

27.3.1. 终端的用法和类型

早期的 UNIX® 系统没有控制台。 人们通过将终端连接到计算机的串口来登录和使用程序。 它很像用 modem 和一些终端软件来拨号进入一个远程的系统， 只能执行文本的工作。

今天的 PC 已经可以使用高质量的图形了，
但与今天的其他UNIX®操作系统一样，建立一个登录会话的能力仍然存在。
通过使用一个终端连接到一个没有使用的串口， 您就能登录和运行任何文本程序或在 X 视窗系统中运行一个 xterm 窗口程序。

对于商业用户，您可以把任何终端连接到 FreeBSD 系统， 然后把它们放在员工的桌面上。
对于家庭用户，则可以使用一台比较老的 IBM PC 或 Macintosh 运行一个终端连接到一台运行 FreeBSD 的高性能机器上。

对于FreeBSD，有三种终端：

- 哑终端
- 充当终端的PC
- X 终端

下面一小节将描述每一种终端。

27.3.1.1. 哑终端

哑终端需要专门的好几种硬件，让您通过串口线连接到计算机。 它们被叫做 "哑" 是因为它们只能够用来显示， 发送和接收文本。 您不能在它上面运行任何程序。

有好几百种哑终端，包括Digital Equipment Corporation 的VT-100和Wyse的WY-75。只有几种可以在FreeBSD上工作。
一些高端的终端可以显示图形，但只有某些软件包可以使用这些高级特性。

哑终端被广泛用于那些不需要图形应用的工作中。

27.3.1.2. 充当终端的PC

假如 哑终端 的功能仅限于显示、 发送和接收文本的话， 那么显然任何一台闲置的个人计算机，都完全能够胜任哑终端的工作。 因此您需要的是合适的线缆， 以及一些在这台计算机上运行的 终端仿真软件。

这种配置在家庭中应用十分广泛。 例如， 如果您的爱人正忙于在您的 FreeBSD 系统的控制台上工作时，您就可以将一台功能稍弱的计算机挂在这个 FreeBSD 系统上来同时完成一些文本界面的工作。

在 FreeBSD 的基本系统中至少有两个能用于进行串口连接的工具： cu(1) 和 tip(1)。

如果要从运行 FreeBSD 的计算机上通过串口连接到另一系统， 可以使用：

```
# cu -l 串口设备
```

此处 "串口设备" 表示您计算机上某个串口对应的设备名。 /dev/cuauN。

此处的 "N" 表示串口的编号。

请注意在 FreeBSD 中设备的编号是从零而非一开始的 (这一点与另一些系统， 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另一些来自 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 终端

X终端是最复杂的终端系统。它们通常需要使用以太网来连接。 它们能显示任何X应用程序。

我们介绍X终端只是为了感兴趣。然而， 这章_不会_涉及X终端的安装，配置或使用。

27.3.2. 配置

这节描述了您在一个终端上启用一个登录会话时， 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 引导过程 中曾经提到， init 进程依赖于系统启动时所有的处理控制和初始化。 通过 init 来执行的一些任务将先读取 /etc/ttys文件， 然后在可用的终端上启用一个 getty 进程。 getty 进程可用来阅读一个登录名和启动 login程序。

然而，要为您的FreeBSD系统配置终端，您需要以 root 身份执行下面的步骤：

1. 如果它不在那里， 您需要为串口在 /dev 目录下添加一行记录到 /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. 指定默认的终端类型。
4. 设置端口为 "on"。
5. 确定端口是否为 "secure"。
6. 迫使init 重新读取 /etc/ttys文件。

作为可选的步骤，您可以通过在 /etc/gettytab 中建立一个记录，在第2步创建一个定制的 getty 类型来使用。这章不会介绍如何做。 您可以参考 gettytab(5) 和 getty(8) 的联机手册了解更多信息。

27.3.2.1. 添加一个记录到/etc/ttys

/etc/ttys文件列出了您 FreeBSD系统上允许登录的所有端口。 例如， 第一个虚拟控制台 ttyv0 在这个文件中有一个记录。 您可以使用这个记录登录进控制台。
这个文件也包含其他虚拟控制台的记录，串口，和伪 ttys 终端。 对于一个硬连线的终端， 只要列出串口的 /dev 记录而不需要 /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。
如果您从那些端口中某一个使用终端，您不需要添加另一个记录。

例 32. 在 /etc/ttys 中增加终端记录

假设我们连接两个终端给系统： 一个 Wyse-50 和一个老的运行 Procomm 终端软件模拟一个 VT-100 终端的286IBM PC。 在 /etc/ttys 文件中的相应的记录是这样的：

```
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。
- 第二部分是在这行执行的命令，通常是 getty(8)。 getty 初始化然后打开一行，设置速度，用户名的命令和执行登录程序。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 文件读取这些特征。文件/etc/gettytab 包含了许多老的和新的终端行记录。
在很多例子中，启动文本 std 的记录将用硬连线终端来工作。 这些记录忽略了奇偶性。
这是一个从110到115200 bit/s的 std 记录。 当然，您可以添加您自己的记录到这个文件。
gettytab 的联机手册提供了更多的信息。当在/etc/ttys中设置 getty 类型的时候，
确信在终端上的通讯设置匹配。 在我们的例子中， Wyse-50 不使用奇偶性， 用38400 bit/s 来连接。286 PC不使用奇偶性，用19200bit/s来连接。
- 第三部分是通常连接到那个tty行的终端类型。对于拨号端口， unknown 或 dialup 通常被用在这个地方。 对于硬连线的终端，终端类型不会改变，
所以您可以从termcap数据库文件中放一个真正的终端类型。在我们的例子中， Wyse-50 使用真正的终端类型， 而运行 Procomm 的286 PC将被设置成在VT-100上的模拟。
- 如果端口被启用，可以指定第四个部分。在第二部分， 把它放在这儿将执行初始化进程来启动程序 getty。如果您在这部分拖延， 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。 标记一个安全的端口意味着您信任它允许用 root 帐户从那个端口登录。 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帐户登录， 然后使用 su 或一个相似的机制来获得超级用户的权限。

27.3.2.2. 重新读取/etc/ttys来强制init

对/etc/ttys文件做一个必要的修改后，您必须发送一个 SIGHUP
信号给初始化进程来迫使它重新读取配置文件，例如：

```
# kill -HUP 1
```

init 总是系统运行时的第一个进程，因此它总是PID 1。

如果能够正确设置，所有的线缆都是适当的，终端将可以启用了， 然后一个 getty 进程将在每个终端运行，您将在您的终端上看到登录命令行。

27.3.3. 您的连接可能出现的问题

即使您小心翼翼地注意细节，您仍然可能会在设置终端时出错。 这有一个有关问题和解决办法的列表：

27.3.3.1. 没有登录命令出现：

确定终端被嵌入和打开了。如果把一台个人计算机充当一个终端， 确信终端模拟软件运行在正确的串口上。

确信线缆被稳固地连接在终端和 FreeBSD 计算机上。 确信用了正确的电缆。

确定终端和 FreeBSD 的传输速度和奇偶设置已经一致了。
如果您有一个图像显示终端，确信对比度已经调节好了。
如果它是一个可打印的终端，确信纸张和墨水已经就绪了。

确定一个 getty 进程正在运行和服务终端。 例如， 可以用ps 命令得到运行 getty 程序的列表，键入：

```
# ps -axww|grep getty
```

您将看到一个终端的记录。例如，下面的显示表明一个getty 正在第二个串口 ttyu1 运行， 正在

/etc/gettytab 中使用 std.38400 的记录：

```
22189  d1  Is+   0:00.03 /usr/libexec/getty std.38400 ttyu1
```

如果没有 getty 进程运行， 确信您已经在/etc/ttys中启用了端口。 在修改完/etc/ttys文件后，记得运行 kill -HUP 1。

如果 getty 进程确实在运行， 但终端上仍然没有显示出登录提示， 或者虽然显示了单缺不允许您输入， 您的终端或电缆可能不支持硬件握手。 请尝试将 /etc/ttys 中的 std.38400 改为 3wire.38400 (注意在改完 /etc/ttys 之后要 kill -HUP 1)。 3wire 记录和 std 类似， 但忽略硬件握手。 您可能需要在使用 3wire 时减少波特率或启用软件流控制以避免缓冲区溢出。

27.3.3.2. 出现一个 "垃圾" 而不是一个登录命令行

确信终端和 FreeBSD 使用相同的 bit/s 传输率和奇偶校验设置。 检查一下 getty 进程确信当前使用正确的 getty 类型。 如果没有， 编辑/etc/ttys然后运行kill -HUP 1。

27.3.3.3. 当键入密码时，字符两个两个出现

将终端 (或终端模拟软件) 从 "半双工" 或 "本地回显" 换成 "全双工"。

27.4. 拨入服务

从 FreeBSD 8.0 开始， 用于串口的设备节点从 /dev/cuadN 改为了 /dev/cuauN； 从 /dev/ttydN 改为了 /dev/ttyuN。 FreeBSD 7.X
用户需要根据实际情况对这份文档中的例子进行必要的调整。

为拨入服务配置FreeBSD系统与连接到终端是非常相似的，除非您正在使用 modem来拨号而不是终端。

27.4.1. 外置vs.内置modem

外置modem看起来很容易拨号。 因为，外置 modem 可以通过储存在非易失性的RAM中的参数来配置，它们通常提供指示器来显示重要的RS-232信号的状态。 不停闪光的信号灯能给用户留下比较深刻的印象，而且指示器也可以用来查看modem是否正常地工作。

内置modem通常缺乏非易失性的RAM， 所以对它们的配置可能会限制在通过 DIP 开关来设置。
如果您的内置modem有指示灯，您也很难看得到。

27.4.1.1. Modem和线缆

如果您使用一个外置的 modem，那您将需要适当的电缆线。
一个标准的串口线应当足够长以至普通的信号能够连接上：

表 12. 信号名称

缩写	全名
RD	收到数据 (Received Data)
TD	传出数据 (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 对速度超过 2400 bps 的情形需要通过 RTS 和 CTS 信号来完成流控制， 通过 CD 信号来检测呼叫响应和挂机， 并通过 DTR 信号来在会话结束时对调制解调器进行复位。 某些电缆在连接时没有提供全部需要的信号， 这会给您带来问题， 例如在挂断时登录会话不消失， 这就有可能是电缆的问题。

与其它类 UNIX® 操作系统类似， FreeBSD 使用硬件信号来检测呼叫响应， 以及在挂断时挂断并复位调制解调器。 FreeBSD 避免发送命令给调制解调器， 或监视其状态。 如果您熟悉通过调制解调器来连接基于 PC 的 BBS 系统， 这可能看起来有点难用。

27.4.2. 串口的考虑

FreeBSD支持基于 NS8250， NS16450， NS16550 和 NS16550A 的EIA RS-232C通讯接口。 8250和16450设备有单字符缓冲。 16550设备提供了一个 16 个字符的缓冲， 可以提高更多的系统性能。 因为单字符缓冲设备比 16 个字符的缓冲需要更多的系统资源来工作， 所以基于16550A的接口卡可能更好。 如果系统没有活动的串口， 或有较大的负载， 16 字符缓冲的卡对于低错误率的通讯来说更好。

27.4.3. 快速预览

对于终端， init 会在每个配置串口上为每个拨入连接产生一个 getty 进程。 例如， 如果一个 modem 被附带在 /dev/ttyu0 中，用命令ps ax可以显示下面这些：

```
4850 ?? I    0:00.09 /usr/libexec/getty V19200 ttyu0
```

当用户拨上modem， 并使用它进行连接时， CD 线就会被 modem 认出。 内核注意到载波信号已经被检测到， 需要完成 getty 端口的打开。 getty 发送一个登录：在指定的初始线速度上的命令行。 Getty 会检查合法的字符是否被接收， 在典型的配置中， 如果发现 "垃圾"， getty 就会设法调节线速度，直到它接收到合理的字符。

用户在键入他/她的登录名称后， getty执行/usr/bin/login， 这会要求用户输入密码来完成登录， 然后启动用户的shell。

27.4.4. 配置文件

如果希望允许拨入您的 FreeBSD 系统， 在 /etc 目录中有三个系统配置文件需要您关注。 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 设备上应该运行 getty。 最后， 关于端口的初始化命令， 应放到 /etc/rc.d/serial 脚本中。

关于在 UNIX® 上配置拨入调制解调器有两种主要的流派。 一种是将本地计算机到调制解调器的 RS-232 接口配置为固定速率。 这样做的好处是， 远程用户总能立即见到系统的登录提示符， 而其缺点则是， 系统并不知道用户真实的数据速率是多少， 因而， 类似 Emacs 这样的程序， 也就无法调整它们绘制屏幕的方式， 以便为慢速连接改善响应时间。

另一种流派将调制解调器的 RS-232 接口速率配置为随远程用户的连接速率变化。 例如， 对 V.32bis (14.4 Kbps) 连接， 调制解调器会让自己的 RS-232 接口以 19.2 Kbps 的速率运行， 而 2400 bps 连接， 则会使调制解调器的 RS-232 接口以 2400 bps 的速率运行。 由于 getty 并不能识别具体的调制解调器的连接速率反馈信息， 因此， getty 会以初始速度给出一个 login: 提示， 并检查用户的响应字符。 如果用户看到乱码， 则他们应知道此时应按下 Enter 键， 直到看到可以辨认的提示符为止。 如果数据速率不匹配， 则 getty 会将用户输入的任何信息均视为 "乱码"， 并尝试以下一种速率来再次给出 login: 提示符。 这一过程可能需要令人作呕地重复下去， 不过一般而言， 用户只要敲一两下键盘就能看到正确的提示符了。 显然， 这种登录过程看起来不如前面所介绍的 "锁定速率" 方法那样简单明了， 但使用低速连接的用户， 却可以在运行全屏幕程序时得到更好的交互响应。

这一节将尽可能公平地介绍关于配置的信息， 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 风格的文件。 请看看 gettytab

的联机手册了解完整的文件格式和功能列表。

27.4.4.1.1. 锁定速度的配置

如果您把您的modem的数据通讯率锁定在一个特殊的速度上， 您不需要对 /etc/gettytab 文件作任何变化。

27.4.4.1.2. 匹配速度的配置

您将需要在 /etc/gettytab 中设置一个记录来告诉 getty 您希望在 modem 上使用的速度。 如果您的 modem 的速率是 2400 bit/s， 则可以使用现有的 D2400 的记录。

```
#
# 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:
```

如果您有一个更高速度的 modem， 必须在 /etc/gettytab 中添加一个记录。 下面是一个让您可以以最高 19.2 Kbit/s 的用在 14.4 Kbit/s的modem上的接口记录：

```
#
# 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:
```

这样做的结果是 8-数据位， 没有奇偶校验的连接。

上面使用19.2 Kbit/s的连接速度的例子，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。 通讯率的调节使用 nx= ("next table") 来实现。 每条线使用一个 tc= ("table continuation") 的记录来加速对于一个特殊传输率的标准设置。

如果您有28.8 Kbit/s的modem，或您想使用它的 14.4Kbit/s 模式， 就需要使用一个更高的超过 19.2 Kbit/s 的通讯速度的 modem。 这是一个启动 57.6 Kbit/s 的 gettytab 记录的例子：

```
#
# 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:
```

如果您的 CPU 速度较低， 或系统的负荷很重， 而且没有 16550A 的串口，您可能会在57.6 Kbit/s 上得到 sio "silo"错误。

27.4.4.2. /etc/ttys

/etc/ttys文件的配置在 在 /etc/ttys 中增加终端记录中介绍过。 配置 modem 是相似的，
但我们必须指定一个不同的终端类型。 锁定速度和匹配速度配置的通用格式是：

```
ttyu0   "/usr/libexec/getty xxx"   dialup on
```

上面的第一条是这个记录的设备特定文件 - ttyu0 表示 /dev/ttyu0 是这个 getty 将被监视的文件。 第二条 "/usr/libexec/getty xxx" 是将运行在设备上的进程 init。 第三条，dialup，是默认的终端类型。
第四个参数， on， 指出了线路是可操作的 init。 也可能会有第五个参数， secure，
但它将只被用作拥有物理安全的终端 (如系统终端)。

默认的终端类型可能依赖于本地参考。 拨号是传统的默认终端类型，
以至用户可以定制它们的登录脚本来注意终端什么时候拨号， 和自动调节它们的终端类型。 然而，
作者发现它很容易在它的站点上指定 vt102 作为默认的终端类型，
因为用户刚才在它们的远程系统上使用的是VT102模拟器。

您对/etc/ttys作修改之后，您可以发送 init 进程给一个 HUP
信号来重读文件。您可以使用下面的命令来发送信号：

```
# kill -HUP 1
```

如果这是您的第一次设置系统， 您可能要在发信号 init 之前等一下， 等到您的 modem
正确地配置并连接好。

27.4.4.2.1. 锁定速度的配置

对于一个锁定速度的配置，您的 ttys 记录必须有一个为 getty 提供固定速度的记录。 对于一个速度被锁定在 19.2kbit/s 的 modem， ttys 记录是这样的：

```
ttyu0   "/usr/libexec/getty std.19200"   dialup on
```

如果您的 modem 被锁定在一个不同的数据速度， 为 std.speed 使用适当的速度来代替 std.19200。
确信您使用了一个在 /etc/gettytab 中列出的正确的类型。

27.4.4.2.2. 匹配速度的设置

在一个匹配速度的设置中，您的 ttys 录需要参考在 /etc/gettytab 适当的起始 "auto-baud" 记录。 例如，
如果您为一个以 19.2 Kbit/s 开始的可匹配速度的 modem 添加上面建议的记录， 您的 ttys
记录可能是这样的：

```
ttyu0   "/usr/libexec/getty V19200"   dialup on
```

27.4.4.3. /etc/rc.d/serial

高速调制解调器， 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。
您可以在 /etc/rc.d/serial 中增加 stty 命令来在 FreeBSD 内核中， 为调制解调器设置硬件流控制标志。

例如， 在 1 号串口 (COM2) 拨入和拨出设备上配置 termios 标志 crtscts， 可以通过在 /etc/rc.d/serial
增加下面的设置来实现：

```
# Serial port initial configuration
stty -f /dev/ttyu1.init crtscts
stty -f /dev/cuau1.init crtscts
```

27.4.5. Modem 设置

如果您有一个 modem， 它的参数能被存储在非易失性的 RAM 中， 您将必须使用一个终端程序来设置参数
（比如 MS-DOS® 下的 Telix 或者 FreeBSD 下的 tip）。 使用同样的通讯速度来连接 modem 作为初始速度
getty 将使用和配置 modem 的非易失性 RAM 来适应这些要求：

- 连接时宣告 CD
- 操作时宣告 DTR； DTR 消失时挂断线路并复位调制解调器
- CTS 传输数据流控制
- 禁用 XON/XOFF 流控制
- RTS 接收数据流控制
- 宁静模式 (无返回码)
- 无命令回显

请阅读您 modem 的文档找到您需要用什么命令和 DIP 接口设置。

例如，要在一个 U.S. Robotics® Sportster® 14400 的外置 modem
上设置上面的参数，可以用下面这些命令：

```
ATZ
AT&C1&D2&H1&I0&R2&W
```

您也可能想要在 modem 上寻找机会调节这个设置， 例如它是否使用 V.42bis 和 MNP5 压缩。

外置 modem 也有一些用来设置的 DIP 开关， 也许您可以使用这些设置作为一个例子：

- 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)

在拨号 modem 上的结果代码应该被 禁用/抑制， 以避免当 getty 在 modem 处于命令模式并回显输入时错误地给出 login: 提示时可能造成的问题。 这样可能导致 getty 与 modem 之间产生更长的不必要交互。

27.4.5.1. 锁定速度的配置

对于锁定速度的配置， 您需要配置 modem 来获得一个不依赖于通讯率的稳定的 modem到计算机的传输率。 在一个 U.S. Robotics® Sportster® 14400 外置 modem 上， 这些命令将锁定 modem 到计算机的传输率：

```
ATZ
AT&B1&W
```

27.4.5.2. 匹配速度的配置

对于一个变速的配置， 您需要配置 modem 调节它的串口传输率匹配接收的传输率。 在一个 U.S. Robotics® Sportster® 14400 的外置 modem 上， 这些命令将锁定 modem 的错误修正传输率适合命令要求的速度， 但允许串口速度适应没有纠错的连接：

```
ATZ
AT&B2&W
```

27.4.5.3. 检查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。 要看看正确的 modem 操作参数， 可以使用命令 ATZ 然后是 ATI4。

如果您有一个不同牌子的 modem， 检查 modem 的使用手册看看如何双重检查您的 modem 的配置参数。

27.4.6. 问题解答

这儿是几个检查拨号modem的步骤。

27.4.6.1. 检查FreeBSD系统

把您的modem连接到FreeBSD系统， 启动系统， 然后， 如果您的 modem 有一个指示灯， 当登录时看看 modem 的 DTR 指示灯是否亮： 会在系统控制台出现命令行——如果它亮， 意味着 FreeBSD 已经在适当的通讯端口启动了一个 getty 进程， 等待 modem 接收一个呼叫。

如果DTR指示灯不亮， 通过控制台登录到 FreeBSD系统，然后执行一个 ps ax 命令来看 FreeBSD 是否正在正确的端口运行 getty进程。 您将在进程显示中看到像这样的一行：

```
114 ??  I    0:00.10 /usr/libexec/getty V19200 ttyu0
115 ??  I    0:00.10 /usr/libexec/getty V19200 ttyu1
```

如果您看到是这样的：

```
114 d0  I    0:00.10 /usr/libexec/getty V19200 ttyu0
```

modem 不接收呼叫， 这意味着 getty 已经在通讯端口打开了。 这可以指出线缆有问题或 modem 错误配置， 因为 getty 无法打开通讯端口。

如果您没有看到任何 getty 进程等待打开想要的 ttyuN 端口， 在 /etc/ttys 中双击您的记录看看那儿是否有错误。 另外，检查日志文件 /var/log/messages 看看是否有一些来自 init 或 getty 的问题日志。 如果有任何信息， 仔细检查配置文件 /etc/ttys 和 /etc/gettytab，还有相应的设备文件 /dev/ttyuN， 是否有错误，丢失记录，或丢失了设备指定文件。

27.4.6.2. 尝试接入Try Dialing In

设法拨入系统。 确信使用8位， 没有奇偶检验， 在远程系统上的1阻止位。 如果您不能立刻得到一个命令行， 试试每隔一秒按一下 Enter。 如果您仍没有看到一个登录： 设法发送一个 BREAK。 如果您正使用一个高速的 modem 来拨号， 请在锁定拨号 modem 的接口速度后再试试。

如果您不能得到一个登录：prompt，再检查一下 /etc/gettytab，重复检查：

- 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。
- 每个 nx= 记录与另一个 gettytab 可用名称匹配。
- 每个 tc= 记录与另一个 gettytab可用名称相匹配。

如果您拨号但 FreeBSD 系统上的 modem 没有回应， 确信 modem 能回应电话。 如果 modem 看起来配置正确了， 通过检查 modem 的指示灯来确认 DTR 线连接正确。

如果您做了好几次，它仍然无法工作，打断一会，等会再试试。 如果还不能工作，也许您应该发一封电子邮件给 FreeBSD 一般问题邮件列表 寻求帮助。

27.5. 拨出设备

从 FreeBSD 8.0 开始， 用于串口的设备节点从 /dev/cuadN 改为了 /dev/cuauN； 从 /dev/ttydN 改为了 /dev/ttyuN。 FreeBSD 7.X 用户需要根据实际情况对这份文档中的例子进行必要的调整。

下面将让您的主机通过 modem 连接到另一台计算机上。 这只要适当地建立一个终端作为远程主机就可以。

这可以用来登录进一个BBS。

如果您用 PPP 有问题， 那这种连接可以用来从 Internet 上下载一个文件。 如果您必须 FTP 一些东西， 而 PPP 断了， 使用终端会话来 FTP 它们。 然后使用 zmodem 来把它们传输到您的机器上。

27.5.1. 我的Stock Hayes Modem不被支持，我该怎么办?

事实上， 联机手册对于这个的描述已经过时了。 一个通用的 Hayes拨号已经内建其中。 只要在您的 /etc/remote 文件中使用 at=hayes。

Hayes 驱动不够 "聪明" 只能认出一些比较新的 modem 的高级特性 - 如 BUSY、 NO DIALTONE， 或 CONNECT 115200 的信息将被搞乱。 当您使用的时候， 您必须把这些信息关掉。(通过 ATX0&W)。

另外，拨号的延迟是 60 秒。 您的 modem 可能使用另外的时间或提示认为有其他的通讯问题。 试试 ATS7=45&W。

27.5.2. 我如何输入这些 AT 命令?

在 /etc/remote 文件中增加一个 "direct" 项。 举例而言， 如果您的调制解调器挂在第一个串口， 即

/dev/cuau0 上， 则应添加下面这行：

```
cuau0:dv=/dev/cuau0:br#19200:pa=none
```

此处应使用您的 modem 所支持的最高 br bps 速率。 接下来， 输入 tip cuau0 就可以连到 modem 上了。

此外， 也可以 root 身份执行 cu 命令：

```
# cu -lline -sspeed
```

line 是串口 (例如 /dev/cuau0) 而 speed 则是速率 (如 57600)。 当您输入完 AT 之后， 按 ~. 即可退出。

27.5.3. 现在pn @标记不能工作?

在电话号码中的 @ 标记告诉计算机在 /etc/phones 文件中查找一个电话号码。 但 @ 标记也是一个在像 /etc/remote 这样的可用文件中的特殊字符。 用一个反斜线符号退出：

```
pn=\@
```

27.5.4. 我如何在命令行拨电话号码?

在您的 /etc/remote 文件中通常放着一个叫做 "generic" 的记录。 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， 可以执行：

```
# tip -115200 5551234
```

如果您更喜欢cu而不是tip，使用一个通用的cu记录：

```
cu115200|Use cu to dial any number at 115200bps:\
	:dv=/dev/cuau1:br#57600:at=hayes:pa=none:du:
```

然后键入：

```
# cu 5551234 -s 115200
```

27.5.5. 这么做时是否每次都需要重新输入 bps 速率?

添加一项 tip1200 或 cu1200， 并将 bps 速率换成更合适的值。 tip 的默认值是1200 bps，也就是为什么会有 tip1200 这条记录的原因。 虽然您并不需要使用 1200 bps。

27.5.6. 我通过一个终端服务器访问了很多主机。

除非每次都要等到您连接到主机然后键入 CONNECT host， 否则使用 tip 的 cm 功能。 例如， 在 /etc/remote 中的这些记录：

```
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:
```

将让您键入 tip pain 或 tip muffin 连接到主机 pain 或 muffin， 和 tip deep13 连接到终端服务器。

27.5.7. tip能为每个站点试用多个线路吗?

经常有一个问题， 一个大学有几个modem线路， 几千个学生设法使用它们。

在 /etc/remote 中为您的大学添加一个记录， 然后为 pn 功能使用 @ 标记：

```
big-university:\
    :pn=\@:tc=dialout
dialout:\
    :dv=/dev/cuau3:br#9600:at=courier:du:pa=none:
```

接着， 在 /etc/phones 中列出大学的电话号码：

```
big-university 5551111
big-university 5551112
big-university 5551113
big-university 5551114
```

tip 将按顺序试用每一个，然后就停止。 如果想继续测试， 隔一段时间再运行 tip。

27.5.8. 为什么我必须键入 Ctrl + P 两次才能发出 Ctrl + P 一次?

Ctrl + P 是默认的"强制"字符，被用来告诉 tip 下一个字符是文字的数据。您可以用 ~s 给任何其他的字符设置强制字符，这意思是 "设置一个变量"。

在新的一行键入 ~sforce=single-char。 single-char 是任何简单的字符。 如果您遗漏了 single-char， 那强制字符就是空字符， 这可以键入 Ctrl + 2 或 Ctrl + Space 来完成。 更好的 single-char 是 Shift + Ctrl + 6， 这只用在一些终端服务器上。

通过在您的 $HOME/.tiprc 文件中指定下面这行， 就可以得到您想要的任何强制字符：

```
force=single-char
```

27.5.9. 突然我键入的每一样东西都变成了大写??

您一定是键入了 Ctrl + A， 即 tip 的 "raise character"， 会临时地指定成坏掉的 caps-lock键。 使用上面的 ~s 来合理地设置各种 raisechar。 事实上，
如果您不想使用这些特性的话，您可以用同样的方法设置强制字符。

这儿有一个很好的示例 .tiprc 文件，对 Emacs用户来说，需要经常按 Ctrl + 2 和 Ctrl + A：

```
force=^^
raisechar=^^
```

^^ 是 Shift + Ctrl + 6.

27.5.10. 如何用 tip 做文件传输?

如果您正在与另一台 UNIX® 系统对话， 您可以用 ~p(put) 和 ~t (take) 发送和接收文件。
这些命令可以在远程系统上运行 cat 和 echo 来接收和发送文件。 语法是这样的：

~p local-file [remote-file]

~t remote-file [local-file]

由于没有错误校验， 所以您需要使用其他协议， 如 zmodem。

27.5.11. 我如何用tip运行zmodem?

要接收这些文件，可以在远程终端启动发送程序。然后，键入 ~C rz 在本地开始接收它们。 要发送文件，
可以在远程终端启动接收程序。 然后， 键入 ~C sz files 把它们发送到远程系统。

27.6. 设置串口控制台

从 FreeBSD 8.0 开始， 用于串口的设备节点从 /dev/cuadN 改为了 /dev/cuauN； 从 /dev/ttydN 改为了 /dev/ttyuN。 FreeBSD 7.X
用户需要根据实际情况对这份文档中的例子进行必要的调整。

27.6.1. 介绍

FreeBSD可以通过一个串口只使用一个哑 (dumb) 终端就可以启动一个系统。
这样一种配置只有两种人能使用： 希望在机器上安装 FreeBSD 的系统管理员， 他没有键盘或显示器，
还有就是要调试内核或设备驱动程序的开发人员。

就象 FreeBSD 引导过程 描述的， FreeBSD 采用一个三步的启动过程。 最先两步储存在 FreeBSD 启动磁盘的启动 slice 的启动代码块中。 引导块然后就被加载， 接着运行第三步启动引导器 (/boot/loader)。

为了设置串口控制台， 您必须配置启动代码块， 启动引导器代码和内核。

27.6.2. 串口控制台的配置， 简明版

这一节假定您使用默认的配置， 只希望迅速地获得关于配置串口控制台的概览。

1. 使用串口电缆连接 COM1 和控制终端。
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份执行下面的命令：

```
# 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口令， 从而导致潜在的安全漏洞。
4. 重新启动并观察是否生效。

如果需要不同的配置， 更进一步的配置讨论可以在 串口控制台的设置 找到。

27.6.3. 串口控制台的设置

1. 准备一根串口线缆。

   您需要使用一个 null-modem 的线缆或标准的串口线和一个 null-modem 适配器。 请参考 线缆和端口 中有关串口线的讨论。

2. 拔掉键盘。

   绝大多数的PC在开机检测的时候会检测到键盘， 如果没有检测到键盘， 则会出现错误。
   一些机器会提示缺少键盘， 就不会继续引导系统。

   如果您的计算机出现错误， 但仍能继续启动， 您可以不必理它。

   如果您的计算机没有键盘拒绝启动， 那您需要配置 BIOS 来避免它。
   请参考您的主板的使用说明了解更多细节。

   在 BIOS 中将键盘设为 "Not installed" (未安装)。 现在您仍然无法使用键盘。
   这样做只是告诉 BIOS 在启动时不要探测键盘。 您的 BIOS
   不应抱怨键盘不存在。 即使这一标志设置为 "Not installed" 时，
   只要把键盘插上， 它就仍可使用。 如果以上的选项不存在于 BIOS 中，
   可尝试寻找 "Halt on Error" 选项。 把这一项设置为 "All but Keyboard" 或者是 "No Errors"， 都能器到相同的作用。

   如果系统有 PS/2 鼠标， 如果幸运的话， 您也可以象键盘一样把它拔下来，
   这是因为 PS/2 鼠标与键盘的一些硬件是共享的， 您的鼠标插上去，
   系统会认为键盘仍在那儿。

3. 插一个哑 (dumb) 终端到COM1： （sio0）。

   如果您没有哑终端， 可以使用一个比较老的带有一个 modem 程序的PC/XT机器， 或在其他 UNIX® 机器上的串口。 如果您没有 COM1： (sio0)， 去找一个。 这时， 您就不能只能选择 COM1： 来启动系统。 如果您已经在另一台设备上使用 COM1， 您必须临时删除那个设备，
   然后安装一个新的系统引导块和内核。

4. 确信您的内核配置文件已经为 COM1： (sio0) 设置了适当的标记：

   有关的标记是：

   0x10
   : 启用控制台支持。 如果没有设置它， 则其他的控制台标记都会被忽略。 现在，
   绝大多数的设置都有控制台的支持。 这个标记的第一个就是首选的。
   这个单独选项是不能确保串口适用于控制台的， 设置下面的标记或加上下面描述的 -h 选项，
   和这个放在一起。

   0x20
   : 无论是否使用了下面将要讨论的 -h 选项， 都强制这个单元作为控制台
   (除非使用了更高优先级的控制台)。 标志 0x20 必须与 0x10 一起使用。

0x40
预留这个单元 (配合 0x10) 并让它不能用于普通的使用。
您不应在希望作为控制台的串口单元上设置这个标志。 这一标志是为内核远程调试准备的。
参见 开发者手册 以了解关于远程调试更进一步的情况。

例如：

```
device sio0 flags 0x10
```

看看 sio(4) 的联机手册了解更多信息。

如果标记没有被设置， 您必须运行UserConfig或重新编译内核。

5. 在启动磁盘的 a 分区的根目录创建 boot.config 文件。

这个文件将指导引导块代码如何启动系统。 为了激活串口控制台，
您必须有一个或多个下面的选项——如果您要多个选项， 在同一行必须都包含它们：

-h
切换内部和串口控制台。 您使用这个来交换控制台设备。 例如， 如果您从内部控制台启动，
您可以使用 -h 来直接使用启动引导器和内核来使用串口作为它的控制台设备。 另外，
如果您从串口启动， 您可以使用 -h 来告诉启动引导器和内核使用显示设备作为控制台。

-D
切换单一和双重控制台配置。 在单一配置中， 控制台将是本机的控制台 (显示设备) 或串口。
在双重控制台配置中， 显示设备和串口将同时成为控制台， 无论 -h 的选项的情形。 然而，
双控制台配置只在引导块运行的过程中起作用。 一旦启动引导器获得控制， 由 -h
选项指定的控制台将成为唯一的控制台。

-P
在启动时，探测键盘。如果键盘找不到， -D 和 -h 选项会自动设置。

由于当前版本引导块的空间限制， -P 选项只能探测扩展的键盘。
少于101键的键盘将无法被探测到。 如果您碰到这个情况， 您必须避免使用
-P 选项。 目前还没有绕过这个问题的办法。

使用 -P 选项来自动选择控制台， 或使用 -h 选项来激活控制台。

您也可以使用boot联机文档中所描述的其他选项。

除了 -P 选项， 所有选项将被传给启动引导器 (/boot/loader)。 启动引导器将通过检查 -h
选项的状态来决定是显示设备成为控制台， 还是串口成为控制台。 这表示如果您指定 -D 选项，
但在 /boot.config 中没有 -h 选项， 您在启动代码块时使用串口作为控制台。
启动引导器将使用内部显示设备作为控制台。

6. 启动机器

当您启动您的FreeBSD时，引导块将把 /boot.config 的内容发给控制台。例如：

```
/boot.config: -P
Keyboard: no
```

如果您把 -P 放在 /boot.config 中并指出键盘存在或不存在， 那将只出现第二行。
这些信息会被定位到串口或内部控制台， 或两者同时， 这完全取决于 /boot.config 中的选项。

选项	送出消息的设备
none	内部控制台
-h	串口控制台
-D	串口控制台和内部控制台
-Dh	串口控制台和内部控制台
-P，有键盘	内部控制台
-P，无键盘	串口控制台

出现上面信息后，在引导块加载启动引导器和更多信息被映到屏幕之前将有一个小小的停顿。
在通常情况下，您不需要打断启动进程，但为了确信设置是否正确，您也可以这样做。

在控制台上按 Enter 以外的任意键就能打断启动进程。引导块将进入命令行模式。您将看到：

```
>> FreeBSD/i386 BOOT
Default: 0:ad(0,a)/boot/loader
boot:
```

检验上面出现的信息，可能是串口，或内部控制台，或两个同时，完全取决于您在 /boot.config 中的选项。如果信息出现在正确的控制台，按 Enter 继续启动进程。

如果您要使用串口控制台，但您没有看到命令行，那可能设置有问题。这时，输入 -h 然后按 Enter 或 Return 来告诉引导块 (然后是启动引导器和内核) 选择串口作为控制台。
一旦系统起来了，就可以回去检查一下是什么出了问题。

启动引导器加载完后，您将进入启动进程的第三步，
您仍然可以在启动引导器通过设定您喜欢的环境来切换内部控制台和串口控制台。参考
从启动引导器修改控制台。

27.6.4. 摘要

这是几个在这章要讨论的几个设置和选择的控制台的摘要。

27.6.4.1. 例1：为 sio0 设置标记 0x10

```
device sio0 flags 0x10
```

在 /boot.config 中的选项	引导块执行时所用的控制台	引导加载器执行时所用的控制台	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，有键盘	内部	内部	内部
-P，没有键盘	串口和内部	串口	串口

27.6.4.2. 例2：您为 sio0 设置标记为 0x30

```
device sio0 flags 0x30
```

在 /boot.config 中的选项	引导块执行时所用的控制台	引导加载器执行时所用的控制台	内核所用的控制台
无	内部	内部	串口
-h	串口	串口	串口
-D	串口和内部	内部	串口
-Dh	串口和内部	串口	串口
-P，有键盘	内部	内部	串口
-P，没有键盘	串口和内部	串口	串口

27.6.5. 串口控制台的提示

27.6.5.1. 设置更高的串口速度

在默认配置中，串口的设置是：速率 9600 波特、8 数据位、无奇偶校验位、1 停止位。
如果您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 配置为希望的速率，并重新编译引导块。请参见 使用 sio0 以外的串口 作为控制台 以了解如何联编和安装新的引导块。

  如果串口控制台已配置为使用 -h 以外的其它方式引导，或者内核使用的速率与引导块不同，则必需在内核配置文件中加入下述设置，并重新联编新内核：

  ```
  options CONSPEED=19200
  ```

- 使用内核引导选项 -S. -S 这个命令行选项可以加到 /boot.config 中。请参见联机手册 boot(8) 以获得如何在 /boot.config 中增加选项，以及其它的可用选项。
- 在您的 /boot/loader.conf 文件中启用 comconsole_speed 选项。

  使用这个选项时，您还需要在 /boot/loader.conf 中配置 console、boot_serial，以及 boot_multicons。下面是一个利用 comconsole_speed 改变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 作为控制台

使用串口而不是 sio0 作为控制台需要做一些重编译。如果您无论如何都要使用另一个串口，重新编译引导块，启动引导器和内核。

1. 取得内核源代码 (参考 更新与升级 FreeBSD)。
2. 编辑 /etc/make.conf 文件，然后设置 BOOT_COMCONSOLE_PORT作为您要使用 (0x3f8、0x2f8、0x3E8 或 0x2E8) 端口的地址。只有 sio0 到 sio3 (COM1 到 COM4) 都可以使用；但多口串口卡将不会工作。不需要任何中断设置。
3. 创建一个定制的内核配置文件，在您要使用的串口添加合适的标记。例如，如果要将 sio1 (COM2) 作为控制台：

```
device sio1 flags 0x10
```

或

```
device sio1 flags 0x30
```

其他端口的控制台标记也不要设。

4. 重新编译和安装引导块：

```
# cd /sys/boot
# make clean
# make
# make install
```

5. 重建和安装内核。
6. 用 bsdlabel(8) 将引导块写到启动盘上，然后从新内核启动。

27.6.5.3. 通过串口线进入DDB调试器

```
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

您可能希望通过串口线进入登录提示， 现在您可以看到启动信息，
通过串口控制台键入内核调试信息。可以这样做。

用一个编辑器打开 /etc/ttys 文件， 然后找到下面的行：

```
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。 如果您已经改变了串口的速度，
还必须改掉标准的 9600 与当前的例如 19200 相匹配。

您也可以改变终端的类型从不知名的到您串口终端的真实类型。 编辑完这个文件， 您必须 kill -HUP 1
来使这个修改生效。

27.6.6. 从启动引导器修改控制台

前面一节描述了如何通过调整引导块来设定串口控制台。
这节将讲到在启动引导器中通过键入一些命令和环境变量来指定控制台。
由于启动引导器会被启动进程的第三步所调用， 引导块以后，
在启动引导器中的设置将忽略在引导块中的设置。

27.6.6.1. 配置串口控制台

您可以很容易地指定启动引导器和内核来使用串口控制台， 只需要在 /boot/loader.ronf中写入下面这行：

```
console="comconsole"
```

无论前一节中的引导块如何配置， 这个设置都会生效。

您最好把上面一行放在 /boot/loader.conf
文件的第一行，以便尽早地在启动时看到串口控制台的启动信息。

同样地，您可以指定内部控制台为：

```
console="vidconsole"
```

如果您不设置启动引导环境变量控制台， 启动引导器和内核将使用在引导块时用 -h 选项指定的控制台。

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) 的联机手册了解更多信息。

目前， 引导块尚不提供与引导加载器的 -P 选项等价的选项， 另外，
它也不能根据是否有键盘存在自动决定选择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作为控制台

要使用一个串口而不是 sio0 作为串口控制台 需要重新编译启动引导器。下面的步骤跟 使用 sio0 以外的串口作为控制台 描述的相似。

27.6.7. 警告

这篇文章本意是想告诉人们如何设定没有显示设备或键盘的专用服务器。 不幸的是，
绝大多数系统没有键盘可以让您启动， 而没有显示设备就不让您启动。 使用 AMI BIOS 的机器可以通过在 CMOS 中将 "graphics adapter" 项设为 "Not installed" 来在启动时不要求显示适配器。

然而， 许多机器并不支持这个选项， 如果您的系统没有显示硬件就拒绝启动。 对于这些机器，
即使您没有显示器， 也必须在机器上插上显示适配器。 建议您试试采用 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。 通过使用拨号 modem 来建立网络或 Internet 连接， 或允许其他人通过您的机器来连上网络， 这些都要求使用 PPP 或 SLIP。
这章将详细介绍设置这些基于 modem 的通信服务的方法。

读完这一章， 您将了解：

- 如何设置用户级 PPP。
- 如何设置内核级 PPP。 (仅限 FreeBSD 7.X)。
- 如何设置 PPPoE (PPP over Ethernet)。
- 如何设置 PPPoA (PPP over ATM)。
- 如何配置和安装 SLIP 客户端和服务器。 (仅限 FreeBSD 7.X)。

在阅读这章之前， 您应：

- 熟悉基本的网络术语。
- 理解拨号连接和 PPP、 SLIP 的基础知识。

您可能想知道用户级 PPP 与内核级 PPP 之间的不同之处。 回答很简单： 用户级 PPP 处理用户级的输入和输出数据， 而不是内核级。 在内核与用户区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实现。 用户级PPP使用 tun 设备与外界通信而内核级 PPP 使用 ppp 设备。

在这章中， 如果没有特殊说明， 则 ppp 指的是用户态 PPP， 除非需要和其它 PPP 软件， 例如 pppd (仅限 FreeBSD 7.X) 加以区分。 另外， 若没有额外的注明， 本章所介绍的所有命令都需要以 root 身份来运行权限。

28.2. 使用用户级 PPP

从 FreeBSD 8.0 开始， uart(4) 驱动取代了 sio(4) 驱动。 用以表示串口的设备节点由分别 /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应之对配置文件进行必要的更改。

28.2.1. 用户级 PPP

28.2.1.1. 前提条件

本章假定您具备如下条件：

- 您有一个 ISP 提供的用于连接使用 PPP 的帐号。
- 您需要有连接在系统上， 并做了正确配置的 modem， 或其他能够连接您 ISP 的设备。
- ISP 的拨号号码。
- 您的登录名称和密码 (可能是一般的 UNIX 风格的登录名和密码对， 也可能是 PAP 或 CHAP 登录名和密码对)。
- 一个或多个域名服务器 IP 地址。 通常， 您会从ISP处得到两个这样的IP地址。 如果您至少得到了一个， 就可以在文件 ppp.conf 中加入 enable dns 命令使 ppp 设置域名服务。 这个功能取决于 ISP 对支持 DNS 协商的具体实现。

下面的信息由您的 ISP 提供， 但不是必需的：

- ISP的网关IP地址。 网关是您准备连接， 并设为 默认路由 的主机。 如果您没有这个信息， 您可以虚构一个， 在连接时 ISP 的 PPP 服务器会自动告诉您正确的值。

这个虚构的 IP 地址在 ppp 中记做 HISADDR。

- 准备使用的子网掩码。 如果ISP没有提供， 一般使用 255.255.255.255 是没有问题的。
- 如果 ISP 提供了静态的IP地址和主机名， 可以输入它们。 反之， 则应让对方主机指定它认为合适的 IP 地址。

如果您不知道这些信息， 请与您的 ISP 联系。

在这节中， 所有作为例子展示的配置文件中都有行号。
这些行号只是为了使解释和讨论变得方便， 在真实的文件中并不存在。 此外，
在必要时应使用 Tab 和空格来进行缩进。

28.2.1.2. PPP自动化配置

ppp和pppd(PPP的内核级实现， 仅限 FreeBSD 7.X) 都使用 /etc/ppp 目录中的配置文件。 用户级 PPP 的例子可以在 /usr/shared/examples/ppp/ 中找到。

配置ppp要求根据您的需要编辑几个文件。 编辑哪几个文件取决于您的 IP 是静态分配 (每次都使用同一个地址) 还是动态分配的 (每次连接到 ISP 都会获得不同的 IP 地址)。

28.2.1.2.1. PPP和静态IP地址

您需要编辑配置文件/etc/ppp/ppp.conf， 如下所示。

以冒号:结尾的行从第一列 (行首)开始， 其它所有的行都要使用空格或制表符 (Tab) 来缩进。

```
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:
指定默认的项。 当PPP运行时这个项中的命令将自动执行。

行2：
启用登录参数。 工作正常后， 为避免产生过多的日志文件， 这行应该简化为：

```
set log phase tun
```

行 3：
告诉 PPP 怎样向对方标识自己。 如果在建立或使用连接时遇到任何麻烦，
PPP就会向对方主机自我标识。 对方主机管理员在处理这个问题时， 这些信息会有用。

行 4：
标明modem要连接的端口号。 COM1 对应的设备是 /dev/cuau0 而 COM2 对应的则是 /dev/cuau1。

行 5：
设置连接的速度。 如果 115200 有问题， 试试 38400。

行 6 & 7：
拨号字符串。 用户级 PPP 使用一种与 chat(8)程序相似的语法。
请参考联机手册了解这种语言的相关信息。

注意， 为了便于阅读此命令进行了换行。 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：
设置连接的时间间隔。 默认是 180 秒， 所以这一行是多余的。

行 9：
告诉PPP向对方主机确认本地域名解析设置。 如果您运行了本地的域名服务器，
要注释或删除掉这一行。

行 10：
为了可读性的需要设置一个空行。 空行会被PPP忽略。

行 11：
为 "provider"指定一个项。 可以改成 ISP的名字， 这样您以后就可以使用 load ISP 来开启连接。

行 12：
设置提供商的电话号码。 多个电话号码可以使用冒号 (:) 或管道符号 (|) 隔开。
这两个字符的区别在ppp(8)的联机手册中有介绍。 总的来讲， 如果您要循环使用这些号码，
可以使用冒号。 如果您想使用第一个号码， 当第一个号码失败了再用第二个号码， 就使用管道符号。
如所示的那样， 要给整个电话号码加上引号(")。

如果电话号码里有空格， 必须用引号(")将其括起来。 否则会造成简单却难以察觉的错误。

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时， 这些值可以用带有 \U \P 参数的 set login
命令进行修改。 当使用PAP或CHAP进行连接时， 这些值在验证使用。

行 15：
如果您使用的是PAP或者CHAP， 在这里就不会有登录。 要注释或删除掉这一行。 请参考 PAP 和
CHAP认证 以了解更多细节。

登录命令是的语法是chat类型的。 在这个例子中是这样的：

```
J. Random Provider
login: foo
```

```
password: bar
protocol: ppp
```

您需要改变这个脚本以适合您自己的需要。 当您第一次写这个脚本时， 应当确保已经启用 "chat" 并处于登录状态， 这样您才能确认通信是否正在按计划进行。

行16：

设置默认的超时时间。 这里， 连接若在 300 秒内无响应将被断开。如果您不想设置成超时， 将这个值设置成0， 或在命令行使用 -ddial 选项。

行 17：

设置接口地址。 您需要用 ISP 提供给您的 IP 地址替换字符串 x.x.x.x， 用 ISP 的网关 IP 地址 (即您要连接的主机) 替换字符串 y.y.y.y。 如果ISP没有给您提供网关地址， 可以使用 10.0.0.2/0。 如果您需要使用一个 "猜到"的地址， 请确保在 /etc/ppp/ppp.linkup 中为每个 PPP和动态IP地址 指令创建了这一项。 如果没有这一行， ppp 将无法以 -auto 模式运行。

第18行：

添加一个到ISP网关的默认路由。 HISADDR这个关键字会被第17行所指定的网关地址替换。 这行必须出现在第17行之后，以免在 HISADDR 初始化之前使用它的值。

如果您不想使用 -auto 的 PPP，则这行应挪到 ppp.linkup 文件中。

若您有一个静态IP地址， 且使用-auto 模式运行ppp(因为在连接之前已经正确设置了路由表项)， 那就不需要再向ppp.linkup 添加项。 您可能希望在连接以后创建一个项来调用程序。 这在以后的sendmail的例子中会解释。

示例配置文件可以在目录 /usr/shared/examples/ppp/ 中找到。

28.2.1.2.2. PPP和动态IP地址

如果ISP没给您指定静态的IP地址， ppp要被配置成能够与对方协商确定本地和远程地址。 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配置协议(IPCP)进行正确配置。 ppp.conf的配置是与 PPP和静态IP地址一样的， 除了以下的改变：

```
17    set ifaddr 10.0.0.1/0 10.0.0.2/0 255.255.255.255 0.0.0.0
```

再次强调， 不要包括行号， 它只是一个引用标记。 缩排一个空格是必需的。

行17：

/ 字符后面是 PPP 所要求的地址掩码。 您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 开始协商地址， 对于有些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 模式时不能设置初始路由。

如果您不运行-auto模式， 就需要在/etc/ppp/ppp.linkup中创建一个项。 连接建立之后， ppp.linkup被启用。 这时候， ppp将指派接口地址， 接着再添加路由表项：

```
1   provider:
2     add default HISADDR
```

行 1：

为了建立连接， ppp 会按照如下规则在 ppp.linkup寻找项:首先， 试图寻找相同的标签 (如同在ppp.conf一样）。 如果失败了， 寻找作为网关 IP 地址的项， 此项是四个八位字节的风格。 如果依旧没有找到， 就寻找 MYADDR 项

行 2：
这行告诉 ppp添加指向 HISADDR的默认路由。 HISADDR由通过IPCP协商得到的IP号替换。

参考/usr/shared/examples/ppp/ppp.conf.sample 和/usr/shared/examples/ppp/ppp.linkup.sample 中的pmdemand项以获取细节化的例子。

28.2.1.2.3. 接收拨入

当要配置 ppp接受来自LAN上的 拨入时， 您需要决定是否将包转给LAN。 如果是的话， 您就必须从 LAN 子网中给对方分配一个IP， 需要在文件 /etc/ppp/ppp.conf 中使用命令 enable proxy。 您还应该确定文件 /etc/rc.conf 中包含以下内容：

```
gateway_enable="YES"
```

28.2.1.2.4. 使用哪个getty?

配置 FreeBSD 的拨号服务 描述了如何用 getty(8) 来启动拨号服务。

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，是按照拨号线的思想设计的。

使用 mgetty 的好处是它能积极地与 modem 进行 会话， 这就意味着如果在/etc/ttys中的端口被关闭，您的moderm就不会回应拨入。

较新版本的 mgetty (从 0.99beta 起) 也支持自动检测 PPP 数据流，
这样即便客户端不使用脚本也能访问服务器了。

参考Mgetty 和 AutoPPP的联机手册了解更多信息。

28.2.1.2.5. PPP 权限

ppp 命令通常必须以 root 用户的身份运行。 如果希望以普通用户的身份启动 ppp 服务 (就像下面描述的那样)， 就必须把此用户加入 network 组， 使其获得运行 ppp 的权限。

您还需要使用allow命令使用户能访问配置文 件的一个或多个部分：

```
allow users fred mary
```

如果这个命令被用在 default 部分中， 您可以让指定的用户访问任何东西。

28.2.1.2.6. 动态IP用户的PPP Shell

创建一个名为/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
```

这个脚本要有可执行属性。 然后通过如下命令创建一个指向此脚本且名为 ppp-dialup的符号链接：

```
# ln -s ppp-shell /etc/ppp/ppp-dialup
```

您应该将这个脚本作为所有拨入用户的 shell。 以下是在文件 /etc/passwd 中关于 PPP 用户 pchilds 的例子(切记， 不要直接修改这个密码文件， 用 vipw(8) 来修改它)。

```
pchilds:*:1011:300:Peter Childs PPP:/home/ppp:/etc/ppp/ppp-dialup
```

创建一个名为 /home/ppp 的目录作为拨入用户的主目录， 其中包含以下这些空文件：

```
-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用户的Shell

像上面那样创建ppp-shell文件， 为每个静态分配IP用户创建一个到 ppp-shell的 符号链接。

例如， 如果您希望为三个拨号用户， fred， sam， 和 mary 路由 /24 CIDR 的网络，
则需要键入以下内容：

```
# 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
```

每个用户的Shell必须被设成一个符号链接(例如用户 mary的Shell应该是/etc/ppp/ppp-mary)。

28.2.1.2.8. 为动态IP用户设置ppp.conf

/etc/ppp/ppp.conf文件应该包含下面 这些行：

```
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
```

缩进得必须的。

default:项在每次会话时都会加载。 每个在 /etc/ttys 中启用的行都必须为其创建一个相似于 ttyu0: 的项。 每一行应该从动态 IP 地址池中取得唯一的IP地址。

28.2.1.2.9. 为静态 IP 用户配置 ppp.conf

根据上面 /usr/shared/examples/ppp/ppp.conf 文件的内容， 您必须为每个静态拨号用户添加一个项。 我们继续以 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 也应该包括每个静态IP用户的的路由信息。 下面这一行为客户连接添加了到 203.14.101.0/24 网络的路由。

```
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 在编译时启用了 AUTO_PPP 选项， 它使 mgetty 能够检测 PPP 连接的 LCP 状态， 并自动产生 PPP shell。 不过， 由于在默认配置中的 login/password 序列并不出现， 因此， 就必须使用 PAP 或 CHAP 来严重用户身份。

这节假定用户已经在系统中成功地编译并安装了 comms/mgetty+sendfax。

确认您的 /usr/local/etc/mgetty+sendfax/login.config 文件中包含以下内容：

```
/AutoPPP/ -    -         /etc/ppp/ppp-pap-dialup
```

这行告诉mgetty运行 ppp-pap-dialup脚本来侦听PPP连接。

创建/etc/ppp/ppp-pap-dialup文件写入以下内容 (此文件应该是可执行的)：

```
#!/bin/sh
exec /usr/sbin/ppp -direct pap$IDENT
```

对应于每个在/etc/ttys的启用行， 都要在/etc/ppp/ppp.conf 中创建相应的项。
这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
```

每个以这种方式登录的用户， 都必须在 /etc/ppp/ppp.secret 文件中给出用户名/口令，
或者使用以下选项， 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
```

如果您想为某些用户分配静态IP， 可以在 /etc/ppp/ppp.secret 中将IP号作为第三个参数指定。 请参见
/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 版本中启用这些扩展， 需要在 /etc/ppp/ppp.conf 的对应项中加入下列配置：

```
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
```

这将告诉客户端首选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中的值。

28.2.1.2.12. PAP 和 CHAP 验证

一些 ISP 将系统配置为使用 PAP 或 CHAP 机制来完成连接验证。 如果遇到这种情况， 在您连接时 ISP
就不会看到 login: 提示符， 而是立即开始 PPP 对话。

PAP 安全性要比 CHAP 差一些， 但在这里安全性并不是问题， 因为密码 (即使用明文传送)
只是通过串行线传送， 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：
这一行指明您的PAP/CHAP用户名。 您需要为_MyUserName_输入正确的值。

第 14 行：
这一行指明您的 PAP/CHAP password密码。 您需要为 MyPassword 输入正确的值。
另外，您可能希望加入一些额外的选项，例如：

```
16    accept PAP
```

或

```
16    accept CHAP
```

以明确您的意图， 不过， 默认情况下 PAP 和 CHAP 都会被接受。

行 15：
如果您使用的是 PAP 或 CHAP， 一般来说 ISP 就不会要求您登录服务器了。 这时， 就必须禁用 "set login" 设置。

28.2.1.2.13. 即时改变您的ppp 配置

与后台运行的ppp程序进行对话是可能的， 前提是设置了一个合适的诊断端口。 做到这一点，
需要把下面的行加入到您的配置中：

```
set server /var/run/ppp-tun%d DiagnosticPassword 0177
```

这行告诉 PPP在指定的UNIX®域socket中侦听， 当用户连接时需要给出指定的密码。 %d用
tun设备号替换。

一旦启用了socket， 就可以在脚本中调用程序pppctl(8)来处理正在运行的 的PPP。

28.2.1.3. 使用PPP网络地址翻译

PPP 可以使用内建的 NAT， 而无需内核支持。 您可以在 /etc/ppp/ppp.conf 中加入如下配置来启用它：

```
nat enable yes
```

PPP NAT也可以使用命令行选项 -nat启动。 在 /etc/rc.conf 文件中也有 ppp_nat 项， 并默认启用。

如果您使用了这个特性， 您还会发现在 /etc/ppp/ppp.conf中以下 选项对于启用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

现在您已配置了ppp， 但在真正工作之前还有一些事情要做。 即修改 /etc/rc.conf。

从上依次往下看， 确认已经正确地配置了 hostname=， 例如：

```
hostname="foo.example.com"
```

如果您的ISP提供给您一个静态的IP和名字， 将这个名字设为hostname是最合适的。

寻找 network_interfaces 变量。 如果要配置系统通过拨号连入ISP， 一定要将tun0设备加入这个列表，
否则就删除它。

```
network_interfaces="lo0 tun0"
ifconfig_tun0=
```

ifconfig_tun0变量应该是空的， 且要创建一个名为 /etc/start_if.tun0的文件。
这个文件应该包含这一行：

```
ppp -auto mysystem
```

此脚本在网络配置时被执行， 开启PPP守护进程进入自动模式。
如果这台机子充当一个LAN的网关， 您可能希望使用 -alias。
参考相关联机手册了解更多细节。

务必在 /etc/rc.conf 中， 把路由程序设置为 NO：

```
router_enable="NO"
```

不启动 routed 服务程序非常重要， 因为 routed 总会删掉由 ppp 所建立的默认路由。

此外， 我们建议您确认一下 sendmail_flags 这一行中没有指定 -q 参数， 否则 sendmail
将会不断地尝试查找网络， 而这样做将会导致机器不断地进行拨号。 可以考虑：

```
sendmail_flags="-bd"
```

替代的做法是当每次 PPP 连接建立时您必须通过键入以下命令强制 sendmail 重新检查邮件队列：

```
# /usr/sbin/sendmail -q
```

您也可以在ppp.linkup使用!bg命令自动完成这些工作：

```
provider:
 delete ALL
 add 0 0 HISADDR
 !bg sendmail -bd -q30m
```

如果您不喜欢这样做， 可以设立一个 "dfilter" 以阻止 SMTP 传输。 参考相关文件了解更多细节。

现在您唯一要做的事是重新启动计算机。 重启之后，可以输入：

```
# ppp
```

然后是dial provider以开启 PPP会话。 或者如果您想让ppp自动建立会话， 因为您有一条广域网连接 (且没有创建 start_if.tun0 脚本)， 键入：

```
# ppp -auto provider
```

28.2.1.5. 总结

当第一次设置PPP时， 下面几步是必须的：

客户端：

1. 确保 tun编译进了进核。
2. 确保 /dev 目录中名为 tunN 的设备文件是可用的。
3. 在 /etc/ppp/ppp.conf中创建一个项。 pmdemand示例应该适合于绝大多数ISP。
4. 如果您使用动态IP地址， 在/etc/ppp/ppp.linkup创建一个项。
5. 更新/etc/rc.conf 文件。
6. 如果您要求按需拨号， 创建一个start_if.tun0脚本。

服务器端：

1. 确保tun设备已编译入内核。
2. 确保 /dev 目录中名为 tunN 的设备文件是可用的。
3. 在/etc/passwd中创建一个项 (使用vipw(8)程序)。
4. 在用户的home目录创建一个运行 ppp -direct direct-server或相似命令的profile。
5. 在/etc/ppp/ppp.conf中创建一个项。 direct-server示例应该能满足要求。
6. 在 /etc/ppp/ppp.linkup中创建一个项。
7. 更新 /etc/rc.conf 文件。

28.3. 使用内核级PPP

这节内容只在 FreeBSD 7.X 上可用。

28.3.1. 设立内核级PPP

在开始配置 PPP 之前， 请确认 pppd 已经存放在 /usr/sbin 中， 并且 /etc/ppp 目录是存在的。

pppd能在两种模式下工作：

1. 作为一个 "客户" - 您要通过PPP串行线或modem线把您的机器连接到互联网上。
2. 作为"服务器" -计算机已经位于网络上， 且被用于通过PPP与其它计算机连接。

两种情况您都需要设立一个选项文件， (/etc/ppp/options 或者是 ~/.ppprc
如果您的计算机有多个用户使用PPP)。

您还需要一些modem/serial软件(comms/kermit就很适合)， 使您能够拨号并与远程主机建立连接。

28.3.2. 使用pppd作为客户端

下面这个 /etc/ppp/options选项文件能够被用来与CISCO终端服务器的 PPP线连接。

```
crtscts       # enable hardware flow control
modem         # modem control line
noipdefault    # remote PPP server must supply your IP address
          # if the remote host does not send your IP during IPCP
          # negotiation, remove this option
passive       # wait for LCP packets
domain ppp.foo.com     # put your domain name here

:remote_ip    # put the IP of remote PPP host here
          # it will be used to route packets via PPP link
          # if you didn't specified the noipdefault option
          # change this line to local_ip:remote_ip

defaultroute    # put this if you want that PPP server will be your
          # default router
```

连接：

1. 使用 Kermit (或其他 modem 程序来拨号)， 然后输入您的用户名和口令 (或在远程主机上启用 PPP 所需的其他信息)。
2. 退出 Kermit (并不挂断连接)。
3. 键入下面这行：

   ```
   # /usr/sbin/pppd /dev/tty01 19200
   ```

   一定要使用正确的速度和设备名。

现在您的计算机已经用PPP连接。 如果连接失败， 您可在文件 /etc/ppp/options 中添加 debug 选项，
并查看控制台信息以跟踪问题。

下面这个/etc/ppp/pppup脚本能自动完成这三个步骤：

```
#!/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， 它会完成拨号， 并在远程主机上完成所有需要的身份验证过程(这份文档的最后有一个脚本实例)。

使用下面这个脚本/etc/ppp/pppdown断开PPP连线：

```
#!/bin/sh
pid=`pgrep pppd`
if [ X${pid} != "X" ] ; then
        echo 'killing pppd, PID=' ${pid}
        kill -TERM ${pid}
fi

pgrep -l kermit
pid=`pgrep kermit`
if [ "X${pid}" != "X" ] ; then
        echo 'killing kermit, PID=' ${pid}
        kill -9 ${pid}
fi

/sbin/ifconfig ppp0 down
/sbin/ifconfig ppp0 delete
```

```
kermit -y /etc/ppp/kermit.hup
/etc/ppp/ppptest
```

通过执行/usr/etc/ppp/ppptest， 看看pppd 是否仍在运行：

```
#!/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
```

执行脚本 /etc/ppp/kermit.hup以挂起moderm， 这个文件包含：

```
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
```

```
crtscts    # enable hardware flow control
modem      # modem control line
connect "/usr/bin/chat -f /etc/ppp/login.chat.script"
noipdefault # remote PPP serve must supply your IP address
          # if the remote host doesn't send your IP during
            # IPCP negotiation, remove this option
passive       # wait for LCP packets
domain your.domain  # put your domain name here

:     # put the IP of remote PPP host here
          # it will be used to route packets via PPP link
            # if you didn't specified the noipdefault option
            # change this line to local_ip:remote_ip

defaultroute    # put this if you want that PPP server will be
          # your default router
```

/etc/ppp/login.chat.script：

以下的内容应该放在一行内。

```
ABORT BUSY ABORT 'NO CARRIER' "" AT OK ATDTphone.number
 CONNECT "" TIMEOUT 10 ogin:-\\r-ogin: login-id
 TIMEOUT 5 sword: password
```

一旦这些被安装且修改正确， 您所要做的就是运行pppd， 就像这样：

```
# pppd
```

28.3.3. 使用pppd作为服务器

/etc/ppp/options要包括下面这些内容：

```
crtscts                 # Hardware flow control
netmask 255.255.255.0        # netmask (not required)
192.114.208.20:192.114.208.165  # IP's of local and remote hosts
                        # local ip must be different from one
                        # you assigned to the Ethernet (or other)
                        # interface on your machine.
                        # remote IP is IP address that will be
                        # assigned to the remote machine
domain ppp.foo.com          # your domain
```

```
passive                 # wait for LCP
modem                    # modem line
```

下面这个脚本/etc/ppp/pppserv 使pppd以服务器方式启动：

```
#!/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) 能够启用/禁用您 modem 的自动应答模式。
其内容类似下面这样：

```
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
```

一个名为/etc/ppp/kermit.dial的脚本用于向远程主机 进行拨号和验证。 您要根据需要定制它。
要加入您的登寻名和密码， 您还要根据 modem 和远程主机的反应修改输入语句。

```
;
; 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.

:slloop
increment \%x             ; Count the seconds
clear                   ; Clear unread characters from input buffer
output \13
;
; put your expected login prompt here:
;
input 1 {Username: }
if success goto sluid
reinput 1 {\255}
if success goto slhup
reinput 1 {\127}
if success goto slhup
if < \%x 10 goto slloop       ; try 10 times to get a login prompt
else goto slhup             ; hang up and start again if 10 failures
```

```
:sluid
;
; put your userid here:
;
output ppp-login\13
input 1 {Password: }
;
; put your password here:
;
output ppp-password\13
input 1 {Entering SLIP mode.}
echo
quit

:slnodial
echo \7No dialtone.  Check the telephone line!\7
exit 1

; local variables:
; mode: csh
; comment-start: "; "
; comment-start-skip: "; "
; end:
```

28.4. PPP 连接故障排除

从 FreeBSD 8.0 开始， uart(4) 驱动取代了 sio(4) 驱动。 用以表示串口的设备节点由分别 /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应之对配置文件进行必要的更改。

本节将讲述通过modem连接使用PPP时可能出现的问题。 例如，
您可能需要确切地知道您拨入的系统会出现一个怎样的命令行提示符。 有些 ISP 会提供 ssword提示符，而其它的可能会出现 password； 如果没有根据情况的不同相应地编写 ppp 脚本， 登录就会失败。 诊断 ppp 最常用的方法是手动进行连接。 以下的信息会一步一步地带您完成手动连接。

28.4.1. 检查设备节点

如果使用的是定制内核， 确认在其编译配置中包含下列配置：

```
device  uart
```

默认的 GENERIC 内核中包含了 uart 设备， 因此如果您使用的是它的话， 就不需要担心了。 只要查看 dmesg 输出中是否有 modem 设备：

```
# dmesg | grep uart
```

您应该找到与 uart 设备有关的输出。 这些就是我们需要的 COM 端口。 如果您的 modem 按照标准串行端口工作， 您就会在 uart1 或 COM2 上找到它。 如果 modem 设备连接在 uart1 接口 (在 DOS 中称为COM2)， 那么您的 modem 将会是 /dev/cuau1。

28.4.2. 手动连接

通过手动控制ppp来连接Internet 是诊断连接及获知ISP处理PPP客户端方式的一个快速， 简单的方法。 让我们从PPP 命令行开始， 在所有的例子中我们使用 example 表示运行 PPP 服务的主机名。 键入ppp 命令打开 ppp：

```
# ppp
```

现在我们已经打开了ppp。

```
ppp ON example> set device /dev/cuau1
```

设置modem设备， 在本例子中是 cuau1。

```
ppp ON example> set speed 115200
```

设置连接速度， 在本例中我们使用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不能确定我们的主机名， 可以在稍后设置。

```
ppp ON example> term
```

切换到 "终端"样我们就能手动地控制这台 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给您的号码进行拨号。

```
CONNECT
```

连接配置， 如果我们遇到了与硬件无关的连接问题， 可以在这里尝试解决。

```
ISP Login:myusername
```

这里提示您输入用户名， 输入ISP提供的用户名然后按回车。

```
ISP Pass:mypassword
```

这时提示我们输入密码， 输入 ISP提供的密码。 如同登录入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。 这里我们需要考虑： 是使用运行于提供商端的 Shell，
还是启动 ppp？ 这本例中， 我们选择使用 ppp， 因为我们希望得到 Internet 连接。

```
Ppp ON example>
```

注意在这个例子中， 第一个 p已经大写。 这表示我们已经成功地连接上了 ISP。

```
PPp ON example>
```

我们已经成功通过了 ISP的验证， 正在等待分配IP地址。

```
PPP ON example>
```

我们得到了一个 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器端建立了连接。 如果由于已存在的路由而导致操作失败， 您可以在 add 前加
!号。 除此之外， 您也可以在真正连接之前设置这些 (指 add default HISADDR)， ppp
会根据这项设定协商取得新的路由。

如果一切顺利， 现在我们应该能得到一个活动的 Internet 连接， 可以使用 CTRL + z 使其转入后台。
如果您发现 PPP重新变为 ppp， 则表示连接被断开。 大写的 P 表明建立了到 ISP 的连接， 而小写的 p
则表示连接由于某种原因被断开， 这有助于帮助我们了解连接的状态。 ppp 只有这两个状态。

28.4.2.1. 诊断排错

如果您有一根直连线且似乎不能建立连接， 要使用set ctsrts off以关闭字节流的CTS/RTS。
这种情况一般发生在连接兼容 PPP 的终端服务器时。 当它向通信连接写入数据时， 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。 如果使用了这个选项， 您还应使用 set
accmap选项， 某些存在缺陷的硬件在完成端对端发送特定字符， 特别是 XON/XOFF 时可能会遇到困难。
请参见 ppp(8) 联机手册以了解关于可用选项的更多细节， 以及如何使用它们。

如果您的 modem 比较旧， 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
(当流量大量增加时) 调制解调器和某些 ISP 被用来纠错。 您需要使用这个选项才能使用 Compuserve ISP。

PPP 可能并不返回命令模式， 这通常是 ISP 等待您这一端发起协商时发生了错误。 此时， 使用 ~p 命令将强制 ppp 开始发送配置信息。

如果您没有看到登录提示， 则很可能需要使用 PAP 或 CHAP 验证来代替前面例子中的 UNIX® 风格验证。 要使用 PAP 或 CHAP 只需在进入终端模式之前把下面的选项加入 PPP：

```
ppp ON example> set authname myusername
```

此处 myusername 应改为您的 ISP 分配给您的用户名。

```
ppp ON example> set authkey mypassword
```

此处 mypassword 应该为您的 ISP 分配给您的口令。

如果连接正常， 但无法查找域名， 请尝试 ping(8) 某个 IP 地址来看看是否返回了信息。 如果您发现百分之百 (100%) 丢包， 那么您很可能没有分配默认路由。 请仔细检查选项 add default HISADDR 是否在连接时被设置了。 如果您能连接到远程的 IP 地址则有可能域名解析服务器的地址没有被加入到 /etc/resolv.conf。 这个文件应该是下面的样子：

```
domain example.com
nameserver x.x.x.x
nameserver y.y.y.y
```

此处 x.x.x.x 和 y.y.y.y 应该改为您的 ISP 的 DNS 服务器的 IP 地址。 这一信息在您注册时可能会提供给您， 不过通常只需给 ISP 打个电话就能知道了。

您还可以让 syslog(3) 为您的 PPP 连接提供日志。 只需增加：

```
!ppp
*.*     /var/log/ppp.log
```

到 /etc/syslog.conf 中。 绝大多数情况下， 这个功能默认已经打开了。

28.5. 使用基于以太网的PPP(PPPoE)

本节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， 并没有必须的内核配置。 如果必需的 netgraph 支持没有编译入内核， 它可以由 ppp 动态加载。

28.5.2. 设置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:xl1 # replace xl1 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 身份执行：

```
# ppp -ddial name_of_service_provider
```

28.5.4. 启动时运行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 服务标签

在某些时候， 有必要使用一个服务标签来建立您的连接。 服务标签用于区分同一网络中的不同服务器。

您可以在ISP提供的文档中找到必要的服务标签信息。 若不能找到， 则应向您的 ISP 寻求技术支持。

作为最后的方法， 您可以试试 Roaring Penguin PPPoE， 它可以在 Ports Collection 中找到。
然而需要注意的是， 它可能会清楚 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这个操作。 简单地安装由服务提供商随 modem 提供的程序。 随后， 选择 System 菜单。 您的配置文件应该会在这里列出。 一般来说它的名字应该是 ISP。

配置文件名 (service tag， 服务标签) 将被用于 PPPoE 在 ppp.conf 中的配置项， 作为服务商 set device 命令的一部分 (参见 ppp(8) 联机手册以了解更多细节)。 它应该类似下面的样子：

```
set device PPPoE:xl1:ISP
```

记住将_xl1_换成实际的以太网设备。

记住将 ISP 换成您刚刚找到的profile名。

获得更多的信息， 请参考：

- 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. 带有一个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)， 其作者为 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。 请向 3Com 抱怨， 如果您认为它应该遵守 PPPoE 的规范。

为了让FreeBSD能够与这个设备通信， 必须设置sysctl。 通过更改/etc/sysctl.conf，
这一步可以在启动时自动完成：

```
net.graph.nonstandard_pppoe=1
```

或者， 也可以直接执行下面的命令：

```
# sysctl net.graph.nonstandard_pppoe=1
```

很不幸，由于这是系统全局设置， 无法同时与正常的PPP客户端(或服务器) 和3Com®HomeConnect™ ADSL Modem通信。

28.6. 使用 ATM 上的 PPP (PPPoA)

以下将介绍如何设置基于ATM的PPP(PPPoA)。 PPPoA是欧洲DSL提供商的普遍选择。

28.6.1. 使用 Alcatel SpeedTouch™USB 的 PPPoA

针对这一设备的 PPPoA 支持， 在 FreeBSD 中是作为 port 提供的， 因为其固件使用了 阿尔卡特许可协议，
因而不能与 FreeBSD 的基本系统一起免费地再发布。

使用 Ports 套件 可以非常方便地安装 net/pppoa 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， 阿尔卡特的 SpeedTouch™ USB 需要从主机上下载固件才能够正常工作。 在
FreeBSD 中您可以将此操作自动化， 在有设备插到某个 USB 口的时候自动下载固件。 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传送。 注意， 必须以 root
用户的身份编辑它。

```
device "Alcatel SpeedTouch USB"
  devname "ugen[0-9]+"
  vendor 0x06b9
  product 0x4061
  attach "/usr/local/sbin/modem_run -f /usr/local/libdata/mgmt.o"
```

要启动USB守护进程usbd， 在/etc/rc.conf加入以下行：

```
usbd_enable="YES"
```

也可以将ppp设置成启动时拨号。 向 /etc/rc.conf加入以下这几行。 同样地您需要以root用户登录。

```
ppp_enable="YES"
ppp_mode="ddial"
ppp_profile="adsl"
```

为了使其正常工作， 您需要使用net/pppoa port提供的ppp.conf样例。

28.6.2. 使用mpd

可以使用 mpd 来连接多种类型的服务， 特别是 PPTP 服务。 您可以在 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，而阿尔卡特 SpeedTouch™ Home 正是其中的一种。

首先需要从 port 完成安装， 然后才能配置 mpd 来满足您的需要， 并完成服务商的配置。 port 会把一系列包括了详细注解的配置文件实例放到 PREFIX/etc/mpd/。 注意， 这里的 PREFIX 表示 ports 安装的目录， 默认情况下， 应该是 /usr/local/。 关于配置 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 服务的一个简单例子。 配置被分别放到了两个文件中， 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用来向您的ISP进行验证。

② password用来向您的ISP进行验证。

mpd.links包含连接的信息：

```
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 运行mpd的主机的IP地址。

② ADSL modem的IP地址。 Alcatel SpeedTouch™ Home 默认的是 10.0.0.138。

初始化连接：

```
# mpd -b adsl
```

您可以通过以下命令查看连接状态：

```
% ifconfig ng0
ng0: flags=88d1<UP,POINTOPOINT,RUNNING,NOARP,SIMPLEX,MULTICAST> mtu 1500
    inet 216.136.204.117 --> 204.152.186.171 netmask 0xffffffff
```

使用mpd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。 您需要有 root 权限才能完成这两项操作。 以下是 ppp.conf 中的一个示例项。 参考 ppp 的联机手册 ppp(8)，以了解更多有关 ppp.conf 选项的信息。

```
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 您在 DSL 服务提供商那里的用户名

② 您帐户的口令。

由于您必须将帐号密码以明文的方式放入ppp.conf
您应该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对 root用户可读。 请参见 chmod(1) 和 chown(8) 的联机手册以了解有关如何操作的进一步信息。

```
# chown root:wheel /etc/ppp/ppp.conf
# chmod 600 /etc/ppp/ppp.conf
```

以下将为到 DSL 路由器的会话打开一个 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。 路由器的文档应该会告诉您它使用的地址。

执行以下命令以打开 tunnel 并开始会话：

```
# pptp address adsl
```

您应该在命令的最后加上("&")号， 否则 pptp 无法返回到命令行提示符。

要创建一个 tun虚拟设备用于进程pptp 和ppp 之间的交互。 一旦您回到了命令行， 或者 pptp 进程确认了一个连接， 您可以这样检查tunnel设备：

```
% ifconfig tun0
tun0: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST> mtu 1500
      inet 216.136.204.21 --> 204.152.186.171 netmask 0xffffff00
      Opened by PID 918
```

如果您无法连接， 一般可以通过telnet或者web浏览器检查路由器(modem)的配置。 如果依旧无法连接，您应该检查pptp的输出及ppp的日志文件 /var/log/ppp.log 以获得线索。

28.7. 使用SLIP

这节内容只在 FreeBSD 7.X 上可用。

28.7.1. 设置 SLIP 客户端

下面是在静态主机网络上配置 FreeBSD 机器使用 SLIP 的方法。 对于动态主机名分配 (您的地址会随每次拨号而不同)， 您可能需要稍复杂一些的设置。

首先， 您需要确认调制解调器所连接的串口。 许多人会设置一个符号连接， 例如 /dev/modem，用以指向实际的设备名， 如 /dev/cuadN。 这样您就可以对实际的设备名进行抽象，以备调制解调器换到其他串口时方便调整之用。 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 对应 COM1， 而 /dev/cuad1 则对应 COM2， 等等。

确保您的内核文件包含以下内容：

```
device   sl
```

这包含在GENERIC内核， 所以这应该不会是个问题， 除非您 已经删除了它。

28.7.1.1. 只需做一次的事情

1. 把您本地网络上的机器、 网关以及域名服务器， 都加入到 /etc/hosts 文件中。
   我们的是下面这个样子：

   ```
   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. 请确保在您的 /etc/nsswitch.conf 中的 hosts: 小节里面， files 先于 dns 出现。
   如果不是这样的话， 可能会产生一些不希望的现象。
3. 编辑/etc/rc.conf。
   a. 编辑以下这行设置主机名(hostname)：

   ```
   hostname="myname.my.domain"
   ```

   应该用您主机的Internet全名代替。 ..

   改变这一行以指明默认的路由：

   ```
   defaultrouter="NO"
   ```

   改为：

   ```
   defaultrouter="slip-gateway"
   ```

4. 创建文件/etc/resolv.conf， 写入以下内容：

   ```
   domain CS.Example.EDU
   nameserver 128.32.136.9
   nameserver 128.32.136.12
   ```

   正如您看到的， 这些行设置了域名服务器。 当然， 实际的域名和IP地址取决于您的环境。

5. 设置root和 toor的密码(其它任何没有密码的帐号)。
6. 重启计算机， 然后确认使用了正确的主机名。

28.7.1.2. 创建一个SLIP连接

1. 在命令提示符之后输入 slip 进行拨号， 输入您的机器名和口令。 具体需要输入什么，
   与您的环境密切相关。 如果使用 Kermit， 则可以使用类似下面的脚本：

   ```
   # 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， 您还需要修改用户名和口令来满足实际需要。 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。

将密码以纯文本的形式存放在文件系统无论如何都是个 坏 主意。 请考虑这样做的风险。

2. 在这里退出 Kermit (也可以用 Ctrl + z 将其挂起)， 以 root 用户键入：

```
# slattach -h -c -s 115200 /dev/modem
```

如果您能ping通路由器另一端的主机， 就是连接好了! 如果不行， 您可以使用-a选项代替 -c 作为slattach的参数。

28.7.1.3. 关闭连接

按下面的步骤做：

```
# kill -INT `cat /var/run/slattach.modem.pid`
```

来杀掉 slattach。 切记上述操作只有以 root 身份才能完成。 接下来回到 kermit (如果之前是将它挂起了，则使用 fg) 并退出 (q)。

在 slattach(8) 联机手册中提到， 必须使用 ifconfig sl0 down 才能将接口标记为关闭，但和这样做似乎没有什么区别。 (ifconfig sl0 仍然报告同样的东西。)

有时， 您的 modem 可能会拒绝挂断。 这种情况下， 只需重新启动 kermit 并再次退出它就可以了。一般来说试二次就可以了。

28.7.1.4. 问题解答

如果还不行， 尽管发邮件到 freebsd-net 邮件列表来提问。 常见的问题包括：

- 执行 slattach 时不使用 -c和-a选项 (这应该不是关键的， 但有些用户报告这样做解决了问题)。
- 使用s10替换 sl0 (在一些字体下很难看出不同)。
- 试试ifconfig sl0来查看您的接口状态。 例如， 您可以这样做：

```
# ifconfig sl0
sl0: flags=10<POINTOPOINT>
      inet 136.152.64.181 --> 136.152.64.1 netmask ffffff00
```

- 如果在使用 ping(8) 时得到了 no route to host 这样的提示， 则说明您的路由表可能有问题。 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。 您系统上的这些数字会因网络活动的不同而改变。

28.7.2. 设置SLIP服务器

本文提供了在 FreeBSD 上设置 SLIP 服务， 也就是如何配置您的系统， 使其能在远程 SLIP 客户端登录时自动地开启连接的建议。

28.7.2.1. 前提条件

这一节技术性很强， 所以要求您有一定的背景知识。 本节假定您熟悉 TCP/IP 网络协议， 特别是网络和节点寻址、 子网掩码、 子网划分、 路由、 路由协议 (如RIP) 等知识。 在拨号服务器上配置 SLIP 需要这些概念性的知识。 如果您不熟悉它们， 请先阅读 Craig Hunt 的 TCP/IP 网络管理 由O’ Reilly & Associates, Inc. 出版 (ISBN 0-937175-82-X)， 或 Douglas Comer 有关 TCP/IP 协议的书籍。

此外还假定您已经配置好了您的调制解调器以及相应的系统文件， 以允许通过调制解调器进行登录。 如果您还没有为此配置好系统， 请参见 拨入服务 以了解关于如何进行拨号服务的配置。 您可能也会想看一看 sio(4) 的联机手册， 以了解关于串口设备驱动的进一步信息， 以及 ttys(5)、 gettytab(5)、 getty(8) & init(8) 上关于怎样配置系统来接受来自调制解调器的登录请求的具体情况， 还有 stty(1) 以了解关于设置串口参数 (例如 clocal 表示串口直联) 等。

28.7.2.2. 快速浏览

使用FreeBSD作为SLIP服务器， 在典型配置时， 它是这样工作的： 一个SLIP客户拨号并以专用的login ID登录到FreeBSD SLIP服务器系统。 这个用户使用 /usr/sbin/sliplogin 作为 shell。 sliplogin 程序会在文件 /etc/sliphome/slip.hosts 中查找这个用户的项， 如果找到了匹配项， 就将串行线连接到一个可用的 SLIP 接口， 然后运行 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用户的ID是Shelmerg， 在/etc/master.passwd中Shelmerg的项如下的所示：

```
Shelmerg:password:1964:89::0:0:Guy Helmer -
SLIP:/usr/users/Shelmerg:/usr/sbin/sliplogin
```

Shelmerg登录时， sliplogin在文件 /etc/sliphome/slip.hosts中搜索与用户ID匹配的行;如下所示：

```
Shelmerg      dc-slip sl-helmer      0xfffffc00      autocomp
```

sliplogin找到这条区配行， 并将串行线与另一个可用的SLIP接口连起来，
然后执行/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 绑定的 SLIP 接口上发出 ifconfig (前述的例子中是 SLIP 接口 0， 这是 slip.login 的第一个参数)， 以设置本地 IP 地址 (dc-slip)、 远程 IP 地址 (sl-helmer)、 这一 SLIP 接口的子网掩码 (0xfffffc00)， 以及任何其他标志 (autocomp)。 如果发生错误， sliplogin 通常会通过 syslogd 的 daemon facility 记下有用的信息， 前者会把这些信息保存到 /var/log/messages (参见 syslogd(8) 和 syslog.conf(5) 以及 /etc/syslog.conf 的联机手册， 以了解 syslogd 在记录什么， 以及这些内容将被记在哪里)。

28.7.2.3. 内核配置

FreeBSD 的默认内核 (GENERIC) 提供了 SLIP (sl(4)) 支持； 使用定制的内核时，
您必须把下面的设置加入到配置文件：

```
device   sl
```

默认情况下， 您的 FreeBSD 计算机不会转发包。 如果您希望将 FreeBSD SLIP 服务器作为路由器使用，
就需要修改 /etc/rc.conf 文件， 将 gateway_enable 变量设为 YES。
这样下次系统引导时就能够保持这一配置了。

要立即应用这些配置， 可以 root 的身份运行：

```
# /etc/rc.d/routing start
```

请参阅 配置FreeBSD的内核 以了解如何配置 FreeBSD 内核， 并获得在重新配置内核方面的指导。

28.7.2.4. Sliplogin配置

正如先前所提到的， /etc/sliphome 目录中有三个文件， 它们共同构成 /usr/sbin/sliplogin 的配置 (参考 sliplogin 的联机手册 sliplogin(8))： 用于定义 SLIP 用户和相关的 IP 地址的 slip.hosts、 通常仅用于配置 SLIP 接口的 slip.login， 以及 (可选的) slip.logout， 用以撤销由 slip.login 所执行的动作。

28.7.2.4.1. 配置 slip.hosts

/etc/sliphome/slip.hosts里的每行包含至少四个元素， 元素之间由空格隔开：

- SLIP用户的登录ID
- SLIP连接的本地地址(指SLIP服务器)
- SLIP连接的远程地址
- 网络掩网

本地和远程地址可以是主机名 (通过文件/etc/hosts或者域名服务解析为IP地址，
这取决于文件/etc/nsswitch.conf 中的设置)， 网络掩网可以是一个 能通过文件
/etc/networks解析的名字。 在一个样例系统中， /etc/sliphome/slip.hosts是这样的：

```
#
```

```
# login local-addr    remote-addr    mask        opt1   opt2
#                          (normal,compress,noicmp)
#
Shelmerg  dc-slip     sl-helmerg     0xfffffc00     autocomp
```

在这行末尾是一或多个选项：

- normal -不压缩报头
- compress - 压缩报头
- autocomp -如果远程端允许， 压缩报头
- noicmp -禁用ICMP数据包 (这样就会丢弃所有的"ping"数据包， 不占用您的带宽)

对SLIP连接的本地及远程地址的选择取决是您是准备在SLIP服务器上使用 TCP/IP 子网还是使用"ARP代理" (它并不是"真正的"ARP代理， 而是我们在本节用于介绍的术语)。
如果您不能确定选择何种方式或者如何分配地址， 请参考"前提条件"(前提条件)里列出的TCP/IP书籍 或者向您的IP网络管理员请教。

如果打算为您的 SLIP 客户使用一个独立的子网， 就需要先从分配得到的网络号中取出一个子网号， 然后再在这个子网里给每个 SLIP 客户分配 IP 地址。 接下来， 您还需要通过 SLIP 服务器在最近的 IP 路由器上配置一个指向 SLIP 子网的静态路由。

如果要使用 "代理 ARP" 的方式， 您还需要从 SLIP 服务器的以太子网中为每个 SLIP 客户分配IP地址， 还必须修改/etc/sliphome/slip.login 和 /etc/sliphome/slip.logout脚本以使用 arp(8)来管理在 SLIP 服务器 ARP 表中的 "代理 ARP" 项。

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脚本仅仅为带有相应本地及远程地址和掩码的SLIP接口执行 ifconfig。

如果您决定使用"ARP代理" 方式(而非为您的SLIP客户使用独立的子网)， 您的/etc/sliphome/slip.login 应该是这样：

```
#!/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 表中加入了一个表项。
这个ARP项使得每当这个以太网上的其它 IP 节点对 SLIP 客户端 IP 地址进行 ARP 请求时， SLIP
服务器会以自已的以太网MAC地址作为回应。

当使用以上的例子时， 一定要将 以太网MAC地址 （00:11:22:33:44:55） 替换成您系统网卡的MAC地址，
否则"ARP代理" 将完全无法工作！ 您可以查看 netstat -i 输出结果以取得以太网 MAC 地址;
输出的第二行应该是这样：

```
ed0  1500 <Link>0.2.c1.28.5f.4a      191923   0  129457    0  116
```

这行表明这个系统的以太网MAC地址是00:02:c1:28:5f:4a -netstat -
i输出的以太网MAC地址必须改成用冒号隔开， 并且要单个十六进数前加上。 这是arp(8)要求的格式;
参考arp(8) 的联机手册以获取完整的使用方法。

在编写 /etc/sliphome/slip.login 和 /etc/sliphome/slip.logout 时， 一定要设置
"可执行" (execute) 位 (换言之， chmod 755 /etc/sliphome/slip.login
/etc/sliphome/slip.logout)， 否则 sliplogin将无法执行它。

28.7.2.4.3. slip.logout配置

/etc/sliphome/slip.logout并不是必需的 (除非您使用了"ARP代理")， 如果您准备创建它，
这里有一个基本的 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"， 则可能希望 /etc/sliphome/slip.logout 在用户注销时自动为 SLIP 客户端删除
ARP 项：

```
#!/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 将删除由 "代理 ARP"slip.login 在 SLIP 客户程序登录时所生成的 ARP 项。

再次强调： 建立 /etc/sliphome/slip.logout 之后， 一定要设置可执行位 (也就是说， chmod 755 /etc/sliphome/slip.logout)。

28.7.2.5. 路由考虑

如果没有使用 "代理 ARP" 的方法来在您的 SLIP 客户机和网络的其余部分 (也可能是 Internet) 之间路由数据包， 您可能需要增加离您最近的默认路由器的静态路由， 以便通过 SLIP 服务器来在 SLIP 客户机子网上进行路由。

28.7.2.5.1. 静态路由

向您最近的默认路由添加一个静态路由可以说是很麻烦 (或者说是不可能， 如果您没有权限这么做)。
如果在您的组织中使用多路由器网络， 有些路由器 (比如 Cisco 和 Proteon 生产的) 不但要配置指向 SLIP 子网的路由， 而且还需要配置将哪些静态路由传给其它的路由器。
所以一些专家意见和问题解答对于使基于静态路由表的路由正常工作很有必要。

Chapter 29. 电子邮件

29.1. 概述

"电子邮件"，或通常所说的 email，是现今使用最广泛的通信方式之一。 本章将对如何在 FreeBSD 上运行邮件服务，以及如何使用 FreeBSD 来收发电子邮件作基本的介绍；
然而，它并不是一份完整的参考手册，实际上，许多需要考虑的重要事项都没有提及。 我们推荐读者阅读 参考文献 中的参考书籍，以获得对于这部分的全面认识。

读完这章，您将了解：

- 哪些软件与收发电子邮件有关。
- FreeBSD 下的基本 sendmail 配置文件在哪里。
- 本地和远程邮箱之间的区别。
- 如何阻止垃圾邮件制造者非法地使用您的邮件服务器作为转发中继。
- 如何安装和配置用于替代 sendmail 的其他邮件传输代理。
- 如何处理常见的邮件服务器问题。
- 如何使用 SMTP 和 UUCP。
- 如何设置系统使其只能发送邮件。
- 如何在拨号连接时使用邮件。
- 如何配置 SMTP 验证以增加安全性。
- 如何安装并使用用户邮件代理，如 mutt 来收发邮件。
- 如何从远程的 POP 或 IMAP 服务器上下载邮件。
- 如何在进入的邮件上自动地应用过滤器和规则。

阅读本章之前，您需要：

- 正确地配置您的网络连接 (高级网络).
- 正确地为您的邮件服务器配置 DNS 信息 (网络服务).
- 知道如何安装第三方软件 (安装应用程序. Packages 和 Ports).

29.2. 使用电子邮件

邮件交换可以分为 5 部分。它们是： 用户端程序、服务端守护进程、DNS、远程或本地的邮箱、
当然，还有邮件主机自己。

29.2.1. 用户端程序

这包括一些基于命令行的程序，例如 mutt、 alpine、elm 和 mail，以及类似 balsa、 xfmail 这样的 GUI 程序。 此外，还有我们更"熟悉的"WWW 浏览器这样的程序。 这些程序简单地通过调用 服务守护进程把邮件事务交给本地的 "邮件主机"，或者通过 TCP 把邮件发出去。

29.2.2. 邮件主机上使用的服务程序

FreeBSD 默认情况下采用 sendmail， 但它也支持为数众多的其它邮件服务程序， 这其中包括：

- exim;
- postfix;
- qmail.

邮件服务器后台守护程序通常有两个功能 - 接收外面发来的邮件和把邮件传送出去。 但它 不 负责使用类似 POP 或 IMAP 这样的协议来帮您阅读邮件， 也不负责连接到本地的 mbox 或 Maildir 信箱。

您可能需要其它的 服务程序 来完成这些任务。

较早版本的 sendmail 有一些严重的安全问题，
他们可能导致攻击者从本地和/或远程操作您的电脑。
您应该确认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的投递过程当中扮演着很重要的角色。
为了能够从您的站点向其它的站点传递邮件， 服务程序需要通过 DNS 查找接收邮件的远程站点的位置。
类似地， 在远程站点向您的主机投递邮件时也会发生这样的查找。

DNS 负责将主机名映射为 IP 地址， 同时， 也需要保存递送邮件时所需要的信息， 这些信息称作 MX 记录。
MX (Mail eXchanger，邮件交换) 记录指定了哪个， 或哪些主机能够接收特定域下的邮件。
如果您没有为主机名或域名设置 MX 记录， 则邮件将被直接递交给主机名对应 IP 所在的主机。

您可以通过 host(1) 命令来查找任何域或主机名对应的 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

为您的域接收邮件是通过邮件服务器来完成的。 它收集发送给您的域的那些邮件，并保存到 mbox (存储邮件默认的方法) 或 Maildir 格式， 这取决于您采用的配置。 一旦邮件被保存下来，
就可以在本地通过类似 mail(1) 或 mutt 这样的程序， 或在远程通过 POP 或 IMAP 这样的协议来读取了。
简单地说， 如果您只在本地阅读邮件，那就没有必要安装 POP 或 IMAP 服务。

29.2.4.1. 通过 POP 和 IMAP 访问远程的邮件

如果希望在远程访问邮箱， 就需要访问 POP 或 IMAP 服务器。
这些协议允许用户从远程方便地访问他们的信箱。 尽管 POP 和 IMAP 都允许用户从远程访问信箱， 但 IMAP 有很多优点， 这包括：

- IMAP 既可以从远程服务器上抓取邮件， 也可以把邮件放上去。
- IMAP 支持并发更新。
- IMAP 对于使用低速网络的用户尤为有用， 因为它能够让这些用户把邮件的结构下载下去，
  而无需立即下载整个邮件。 它还可以在服务器端执行类似查找这样的操作，
  以减少客户机和服务器之间的通讯量。

您可以按照下面的步骤来安装和配置 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 来加载 POP 或 IMAP 服务。

此外还应注意的是 POP 和 IMAP 传递的信息，包括用户名和口令等等，通常都是明文的。这意味着如果您希望加密传输过程中的信息，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 中进行了详细阐述，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。这通常是通过 mutt 或 mail(1) 这样的应用程序实现的。

29.2.5. 邮件服务器

邮件服务器是通过服务器给的一个名字 (译注：来识别主机)，这也正是它能在您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 中的默认邮件传输代理 (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们送给配置好的的寄送程序。sendmail 也能接受网络连接，并且发送邮件到本地邮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 访问数据库文件
/etc/mail/aliases	邮箱别名
/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	邮件分发列表
/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

访问数据库定义了什么主机或者 IP 地址可以访问本地邮件服务器和它们是哪种类型的访问。主机可能会列出 OK、REJECT、RELAY 或者简单的通过 sendmail 的出错处理程序检测一个给定的邮件错误。主机默认列出 OK，允许传送邮件到主机，只要邮件的最后目的地是本地主机。列出 REJECT 将拒绝所有的邮件连接。如果带有 RELAY 选项的主机将被允许通过这个邮件服务器发送邮件到任何地方。

例 33. 配置 sendmail 的访问许可数据库

```
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
```

在上面的例子中我们有 5 条记录。与左边列表匹配的发件人受到右边列表动作的影响。前边的两个例子给出了 sendmail 的出错处理程序检测到的错误代码。当一个邮件与左边列表相匹配时，这个信息会被打印到远程主机上。下一条记录拒绝来自 Internet

上的一个特别主机的邮件 another.source.of.spam。接下来的记录允许来自 okay.cyberspammer.com 的邮件连接， 这条记录比上面那行 cyberspammer.com 更准确。更多的准确匹配使不准确的匹配无效。最后一行允许电子邮件从主机和 128.32 开头的 IP 地址转发。 这些主机将被允许通过这台邮件服务器前往其它邮件服务器发送邮件。

当这个文件被升级的时候，您必须在 /etc/mail/ 运行 make 升级数据库。

29.3.2. /etc/mail/aliases

别名数据库包含一个扩展到用户，程序或者其它别名的虚拟邮箱列表。 下面是一些在 /etc/mail/aliases 中使用的例子：

例 34. 邮件别名

```
root: localuser
ftp-bugs: joe,eric,paul
bit.bucket:  /dev/null
procmail: "|/usr/local/bin/procmail"
```

这个文件的格式很简单； 冒号左边的邮箱名， 会被展开成右边的形式。 第一个例子简单地将 root 邮箱扩展为 localuser， 之后将继续在别名数据库中进行查找。 如果没有找到匹配的记录，则邮件会被发给本地用户 localuser。 第二个例子展示了一个邮件列表。 发送到 ftp-bugs 的邮件会被展开成 joe， eric 和 paul 这三个邮箱。 注意也可以通过 user@example.com 这样的形式来指定远程的邮箱。 接下来的例子展示了如何把邮件写入到文件中， 这个例子中是 /dev/null。 最后一个例子展示了如何将邮件发给一个程序， 具体而言是通过 UNIX® 管道发到 /usr/local/bin/procmail 的标准输入。

更新此文件时， 您需要在 /etc/mail/ 中使用 make 来更新数据库。

29.3.3. /etc/mail/local-host-names

这是一个 sendmail(8) 被接受为一个本地主机名的主机名列表。 可以放入任何 sendmail 将从那里收发邮件的域名或主机。例如，如果这个邮件服务器从域 example.com 和主机 mail.example.com 接收邮件，它的 local-host-names 文件，可以看起来象如下这样：

```
example.com
mail.example.com
```

当这个文件被升级，sendmail(8) 必须重新启动，以便更新设置。

29.3.4. /etc/mail/sendmail.cf

sendmail的主配置文件 sendmail.cf 控制着 sendmail 的所有行为，包括从重写邮件地址到打印拒绝远程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，它的细节部分已经超出了本节的范围。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改动。

sendmail 主配置文件可以用 m4(1) 宏定义 sendmail 的特性和行为。它的细节请看 /usr/src/contrib/sendmail/cf/README。

当这个文件被修改时， sendmail 必须重新启动以便对新修改生效。

29.3.5. /etc/mail/virtusertable

virtusertable 映射虚拟域名和邮箱到真实的邮箱。 这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， 我们映射了一个域 example.com。 这个文件是按照从上到下，
首个匹配的方式来处理的。 第一项将 root@example.com 映射到本地邮箱 root。 下一项则将 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， 则将使用最后一条映射， 它表示将所有的其它邮件发给 example.com 域的某个人。 这样， 将映射到本地信箱 joe。

29.4. 改变您的邮件传输代理程序

先前已经提到，FreeBSD 中的 sendmail 已经安装了您的 MTA
(邮件传输代理程序)。因此它负责着您的收发邮件的工作。

然而，基于不同的理由，一些系统管理员想要改变他们系统的 MTA。这些理由从简单的想要尝试另一个 MTA，到需要一个特殊的特性或者 package
依赖某个邮寄程序等等。幸运的是，不管是什么理由，FreeBSD 都能容易的改变它。

29.4.1. 安装一个新的 MTA

对于可用的 MTA 您有很多的选择。一个好的出发点是 FreeBSD Ports Collection，在那里您能找到很多。
当然您可以从任何位置不受任何限制的使用 MTA，只要您能让它运行在 FreeBSD 下。

开始安装您的新 MTA。一旦它被安装， 它可以让您有机会判定它是否能满足您的需要， 并且在它接管 sendmail 之前让您有机会配置您的新软件。
当完成这些之后，您应该确信安装的新软件不会尝试更改系统的二进制文件例如 /usr/bin/sendmail。
除此以外， 您的新邮件软件启用之前要已经配置好它。

具体配置请参考您所选择的 MTA 软件的配置文档或其它相关资料。

29.4.2. 禁用 sendmail

如果您打算禁用 sendmail 的邮件发出服务， 保持系统中有一个替代它的、可用的邮件递送系统就非常重要。 如果您不这样做的话， 类似 periodic(8) 这样的系统功能就无法如预期的那样， 通过邮件来传送其执行结果。
您系统中的许多部分可能都假定有可用的 sendmail-兼容 系统。
如果这些应用程序继续使用 sendmail 的执行文件来发送邮件， 而您又禁用了它，则邮件将进入 sendmail 的非活跃 (inactive) 队列， 而永远不会被送达。

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， 您应该在 /etc/rc.conf 文件中设置

```
sendmail_enable="NO"
```

更多的有关 sendmail 可用的启动选项，参看 rc.sendmail(8) 联机手册.

29.4.3. 机器引导时运行您的新 MTA

可以向 /etc/rc.conf 中加入配置项使新的 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
```

这样 MTA 就能在系统启动是自动运行了。

29.4.4. 替换系统默认的邮寄程序 sendmail

因为 sendmail 程序是一个在 UNIX®
系统下普遍存在的一个标准的软件，一些软件就假定它已经被安装并且配置好。 基于这个原因，许多其它的 MTA 提供者都提供了兼容 sendmail 的命令行界面来执行。 这使它们像"混入"sendmail
一样变的很容易掌握。

因此，如果您使用其它的邮寄程序， 您必须确定这个软件是去尝试运行标准的 sendmail 二进制，就象 /usr/bin/sendmail，还是运行您自己选择的替换邮寄程序。 幸运的是，FreeBSD 提供了一个系统调用 mailwrapper(8)，它能为您做这件工作。

当 sendmail 安装后被运行，您可以在 /etc/mail/mailer.conf 中找到如下行：

```
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
```

这个的意思就是当这些公共命令 (例如 sendmail 它本身) 运行时， 系统实际上调用了一个 sendmail 指定的 mailwrapper 的副本，它检查 mailer.conf 并且运行 /usr/libexec/sendmail/sendmail
做为替代。当默认的 sendmail 功能被调用， 系统将很容易的改变实际上运行的二进制文件。

因此如果您想要 /usr/local/supermailer/bin/sendmail-compat 替换 sendmail 被运行，您应该改变 /etc/mail/mailer.conf 文件为：

```
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

一旦做完您想要配置的每件事，您应该杀掉 sendmail 进程并且启动属于您的新软件的进程，
或者简单的重启。 重启也将给您提供了确认您的系统已经进行了正确的配置的机会。
在引导的时候自动的运行您新的 MTA。

29.5. 疑难解答

29.5.1. 为什么必须在我的站点的主机上使用 FQDN?

您可能会发现主机实际上是在另外一个域里面， 例如，如果您是在 foo.bar.edu 里，而您要找一台叫 mumble 的主机，它在 bar.edu 域里，您就必须用完整的域名 mumble.bar.edu，而不是用 mumble。

传统上，这在 BSD BIND resolvers 中是可行的。 然而目前随 FreeBSD 附带的 BIND
已不为同一域外提供缩写服务。所以，这个不完整的主机名 mumble 必须以 mumble.foo.bar.edu
这种形式才能被找到， 或者将在根域中搜索它。

这跟以前的处理是不同的，以前版本将会继续寻找 mumble.bar.edu 和 mumble.edu。
如果您想要了解这种方式是否是好，或者它有什么安全方面的漏洞， 请参阅 RFC 1535 文档。

如果您想要一个好的工作环境，您可以使用如下行：

```
search foo.bar.edu bar.edu
```

替换先前旧的版本：

```
domain foo.bar.edu
```

把这行放在您的 /etc/resolv.conf 文件中。然而，请一定要确定这样的搜寻顺序不会造成 RFC 1535
里提到的"boundary between local and public administration" 问题。

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?

您已经通过 MX 记录指定把发送给特定的域 (例如，domain.net)
的邮件被转寄到指定的主机 (在这个例子中，relay.domain.net)，
而这台机器并不认为它自己是 domain.net。请把 domain.net 添加到
/etc/mail/local-host-names 文件中 [在 8.10 版之前是 /etc/sendmail.cw]
(如果您使用 FEATURE(use_cw_file) 的话) 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， 如果您想要对您的邮件做任何的"调整"，则推荐首先看一看它。

29.5.3. 我如何在一个拨号主机上运行一个邮件服务？ PPP

您想要把局域网上的 FreeBSD 主机连接到互连网上，而这台 FreeBSD 主机将会成为这个局域网的邮件网关， 这个拨号连接不必一直保持在连接状态。

最少有两种方法可以满足您的要求。一种方法就是使用 UUCP。

另一种方法是找到一个专职的服务器来为您的域提供副 MX 主机服务。 例如，如果您公司的域名是 example.com，您的互连网服务提供者把 example.net 作为您域的副 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终收信主机 (在 example.com 主机的 /etc/mail/sendmail.cf 文件中添加 Cw example.com)。

当 sendmail 试图分发邮件的时候， 它会尝试通过 modem 连接到您 (example.com)。因为您并不在线，所以总是会得到一个超时的错误。 sendmail 将会把邮件发送到副 MX 主机，也就是说，您的互连网服务提供者 (example.net)。副 MX 主机将周期性的尝试连接并发送邮件到您的主机 (example.com)。

您也许想要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
```

如果您想要为一个用户建立一个分开登录的脚本， 您可以使用 sendmail -qRexample.com 替换上面的脚本。这样将使所有的邮件按照您的 example.com 队列立即被处理。

更深入的方法可以参考下面这段：

这段信息是从 FreeBSD Internet 服务提供商的邮件列表 拿来的。

```
> 我们为用户提供副 MX 主机服务。用户每天都会上线好几次
> 并且自动把信件取回主 MX 主机
> (当有他们的邮件时我们并没有通知他们)。
> 我们的 mailqueue 程序每 30 分钟清一次邮件队列。那段时间他们
> 就必须上线 30 分钟以确保他们的信件送达他们的主 MX 主机。
>
> 有任何指令可以用 sendmail 寄出所有邮件么?
> 普通用户在我们的机器上当然没有 root 权限。

在 sendmail.cf 的privacy flags部分，有这样的设定
Opgoaway,restrictqrun
```

移除 restrictqrun 可以让非 root 用户启动队列处理的程序。
您可能也要重新安排您的 MX 设定。我们是用户的 MX 主机，
而且我们还设定了这个：

If we are the best MX for a host, try directly instead of generating # local config error.
OwTrue

这样的话远程机器会直接把信送给您，而不会尝试连接您的用户的机器。
然后您就可以把邮件发送到您的用户。这个设定只对
主机有效，所以您必须要让您的用户在 DNS 中把他们的邮件主机设置为
customer.com或者
hostname.customer.com。只要为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总是有 Relaying Denied 出错信息?

默认的 FreeBSD 安装中， sendmail 会配置为只发送来自它所在主机上的邮件。 例如，如果有可用的 POP 服务器，则用户将可以从学校、 公司或其他什么别的地方检查邮件，但他们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 错误信息的邮件。

有很多方法可以避免这种现象。 最直截了当的方法是把您的 ISP 的地址放到 /etc/mail/relay-domains 文件中。 完成这项工作的简单的方法是：

```
# echo "your.isp.example.com" > /etc/mail/relay-domains
```

建立或编辑这个文件以后您必须重新启动 sendmail。 如果您是一个管理员并且不希望在本地发送邮件，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。如果您仅有一到两个邮件帐户它也非常的有用。 如果有大量的地址需要添加，
您可以很简单的使用您喜欢的文本编辑器打开这个文件添加域名， 每行一个：

```
your.isp.example.com
other.isp.example.net
users-isp.example.org
www.example.org
```

现在邮件可以通过您的系统传送， 这个列表中存在的主机 (前提是用户在您的系统上已经有一个帐户) 将可以成功的发送。这是一个允许正常的远程用户从您的系统发送邮件，
并且阻止其它非法用户通过您系统发送垃圾邮件的好方法。

29.6. 高级主题

下面这节将介绍邮件配置和为整个域安装邮件。

29.6.1. 基本配置

在邮箱外，只要您设置 /etc/resolv.conf 或者运行您自己的名字服务器，您就可以发送邮件到外部的主机。 如果您想要您的邮件发送给某个特定的 MTA(例如， sendmail) 在您的 FreeBSD 主机上，有两个方法：

- 运行您自己的域名服务器和您自己的域。例如， FreeBSD.org

- 获得直接分发给您主机的邮件。您可以直接使用您当前的 DNS 名称。例如，example.FreeBSD.org。

不管您选择上面那种方法，为了直接在您的主机上发送邮件， 必须有一个静态的 IP 地址(不是象 PPP 拨号一样的动态地址)。 如果您在防火墙后面，它必须让 SMTP 协议通过。
如果您想要在您的主机上直接的收取邮件， 您必须确定两件事：

- 确定在您 DNS 中的 MX 记录(最小编号的)指向您的 IP 地址。
- 确定在您 DNS 中的 MX 记录没有禁止您的主机。

上面的每条记录都允许您在您的主机直接接收邮件。

试试这个：

```
# hostname
example.FreeBSD.org
# host example.FreeBSD.org
example.FreeBSD.org has address 204.216.27.XX
```

如果您看到这些， 则直接发往 yourlogin@example.FreeBSD.org 应该已经可以正常工作了 (假设 sendmail 已经在 example.FreeBSD.org 上正确启动了)。

如果您看到这些：

```
# 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
```

所有发送到主机 (example.FreeBSD.org) 的邮件在相同的用户名下将会被 hub 终止的收集，而不是直接发送到您的主机。

上面的信息是通过您的 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 地址被直接的发送到主机。

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
```

正如您说看到的，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。如果因为一些原因它不可用，其它 (有时会访问"backup MXes")接收信息将会暂时接替并做临时的排列。

为了有效的使用交换式 MX 站点，应当从您的机器上分离一些 Internet 连接。您的 ISP 或者其它友好的站点可以没有任何问题的为您提供这个服务。

29.6.2. Mail for Your Domain

为了设置一个"邮件主机"(又称邮件服务器) 您必须要把许多邮件发送到与它相连的几个工作站中。基本上，您想要"要求"在您域的每个主机的所有邮件 (在这个例子里是 *.FreeBSD.org) 转向到您的邮件服务器，从而使您的用户可以在主邮件服务器里接收他们的邮件。

要使工作最简单，带有同样 用户名 的帐户应该同时存在于两台机器上。使用 adduser(8) 来这样做。

您将使用的邮件主机必须为每个工作站指定一个邮件交换。您可以在 DNS 中这样配置：

```
example.FreeBSD.org A   204.216.27.XX     ; Workstation
        MX  10 hub.FreeBSD.org  ; Mailhost
```

无论 A 记录指向哪，这将为工作站重新定位到邮件主机。邮件将被发送到 MX 主机。

您不能自己这样做除非您运行着一个 DNS 服务器。 如果不是这样，或者不能运行您自己的 DNS 服务器，告诉您的 ISP 或者给您提供 DNS 服务的人。

如果您正在使用虚拟邮件主机，下面的信息将会对您有用。
在这个例子里，我们假定您有一个客户并且他有自己的域， 这个例子中是 customer1.org，您要把 customer1.org 所有的邮件发送到您的邮件主机 mail.myhost.com。 您的 DNS 记录应该是这样：

```
customer1.org     MX  10  mail.myhost.com
```

您 不 需要有个 A 记录， 如果您只为域 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。

最后一件您必须要做的事是告诉 sendmail 接受邮件的是什么域和(或)主机名。
这里有好几种方法。下面方法可以任选一种：

- 添加您的主机到 /etc/mail/local-host-names 文件中，如果您使用的是 FEATURE(use_cw_file)。如果您使用 sendmail 8.10 或者更高版本，文件是 /etc/sendmail.cw。
- 添加一行 Cwyour.host.com 到您的 /etc/sendmail.cf 或 /etc/mail/sendmail.cf 文件，如果您使用 sendmail 8.10 或者更高版本。

29.7. SMTP 与 UUCP

sendmail 的配置，在 FreeBSD 中已经配置好为您的站点直接的连接 Internet。
如果站点希望他们的邮件通过 UUCP 交换，则必须安装其它的 sendmail 配置文件。

手工的配置 /etc/mail/sendmail.cf 是一个高级主题。sendmail 8 版本通过 m4(1) 预处理生成一个配置文件，实际上这个配置发生在一个比较高的抽象层。 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 传送的方法是使用 mailertable 的特点。建立一个资料库让 sendmail 可以使用它自己的路由决策。

首先，您必须建立您自己的 .mc 文件。 /usr/shared/sendmail/cf/cf 目录包含一些例子。
假定您已经命名自己的文件叫做 foo.mc， 您要做的只是把它转换成一个有效的 sendmail.cf：

```
# cd /etc/mail
# make foo.cf
# cp foo.cf /etc/mail/sendmail.cf
```

一个典型的 .mc 文件看起来可能象这样：

```
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
特性将避免在传送邮件时使用DNS的机会。UUCP_RELAY 项是支持 UUCP 传送所必须的。简单的放入一个 Internet 上可以处理 UUCP 虚拟域地址的主机名。通常，您在这里填入您 ISP 邮件的回复处。

一旦您做完这些，您还需要这个 /etc/mail/mailertable 文件。
如果您只有一个用来传递所有邮件的对外通道的话， 以下的文件就足够了：

```
#
# makemap hash /etc/mail/mailertable.db < /etc/mail/mailertable
.                   uucp-dom:your.uucp.relay
```

一个更复杂点的例子象这样：

```
#
# 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:
```

头三行处理域地址邮件，不应该被传送出默认的路由， 而由某些 UUCP
邻居取代的特殊情况，这是为了走"捷径"。 下一行处理本地网的邮件让它可以使用 SMTP 来传送。
最后，UUCP 邻居提起。UUCP 虚拟域的记载， 允许一个 uucp-neighbor !recipient
推翻默认规则。最后一行则以一个单独的句点最为结束， 以 UUCP 传送到提供您所有的邮件网关的 UUCP 邻居。 所有在 uucp-dom: 关键字里的节点名称必须是有效的 UUCP 邻居，您可以用 uuname 去确认。

提醒您这个文件在使用前必须被转换成 DBM 数据库文件。最好在 mailertable
最上面用注解写出命令行来完成这个工作。 当您每次更换您的 mailertable 后您总是需要执行这个命令。

最后提示：如果您不确定某个特定的路径可用， 记得把 -bt 选项加到 sendmail。这会将 sendmail 启动在地址检测模式。只要按下 3,0，接着输入您希望测试的邮件路径位置。 最后一行告诉您使用邮件代理程序，代理程序会通知目的主机以及 (可能转换) 地址。 要离开此模式请按 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. 只发送邮件的配置

许多时候， 可能只希望通过转发服务器来发送邮件。 典型的情况包括：

- 使用桌面机， 但希望通过类似 send-pr(1) 这样的程序发送邮件。 这样就需要使用 ISP 的邮件转发服务器。
- 不在本地处理邮件的服务器， 但它需要把邮件交给转发服务器来进行处理。

几乎任何一个 MTA 都能够胜任这样的工作。 然而不幸的是， 要把一个全功能的 MTA 正确地配置为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 这样的程序，多少有些杀鸡用牛刀的感觉。

此外， 如果您使用典型的 Internet 访问服务， 您的协议可能会包含禁止运行 "邮件服务器" 的条款。

满足这些需要最简单的办法是安装 mail/ssmtp port。 以 root 身份执行下面的命令：

```
# 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_
```

请确认您为 root 使用了真实的电子邮件地址。 用您的 ISP 提供的外发邮件转发服务器名称， 替换掉 mail.example.com (某些 ISP 可能将其称为 "外发邮件服务器" 或 "SMTP 服务器")。

接下来需要确认禁用了 sendmail， 包括邮件发出服务在内。 请参见 禁用 sendmail 以了解进一步的细节。

mail/ssmtp 也提供了一些其他选项。 请参见在 /usr/local/etc/ssmtp 中的示例配置， 或者 ssmtp 的联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， 能够让您计算机上的任何需要发送邮件的软件都正常运转， 而不必冒违反 ISP 的使用政策， 或使您的电脑被劫持用于发送垃圾邮件的风险。

29.9. 拨号连接时使用邮件传送

如果您有静态的 IP 地址， 就应该不用修改任何默认的配置。 将主机名设置为分配给您的 Internet 名称，其他的事情 sendmail 都会替您做好。

如果您的 IP 地址是动态分配的， 并使用 PPP 连接拨入 Internet， 则您可能会从 ISP 的邮件服务器上得到一个信箱。 这里我们假设您的 ISP 的域名是 example.net， 您的用户名是 user， 您把自己的机器称作 bsd.home， 而您的 ISP 告诉您可以使用 relay.example.net 来转发邮件。

为了从邮箱收取邮件， 需要安装一个收信代理。 fetchmail 是一个能够支持许多种不同协议的不错的选择。 这个程序可以通过 package 或 Ports Collection (mail/fetchmail) 来安装。 通常， 您的 ISP 会提供 POP。 如果您使用用户 PPP，您还可以在 Internet 连接建立时自动地抓取邮件， 这可以通过在 /etc/ppp/ppp.linkup 中增加如下的项来实现：

```
MYADDR:
!bg su user -c fetchmail
```

如果您正使用 sendmail (如下所示) 来为非本地用户传送邮件， 则可能需要让 sendmail 在您的 Internet 连接建立时立即传送邮件队列。 要完成这项工作， 应该把下面的命令放到 /etc/ppp/ppp.linkup 中的 fetchmail 之后

```
 !bg su user -c "sendmail -q"
```

假设您在 bsd.home 上有一个 user 用户。 在 bsd.home 上的 user 主目录中创建一个 .fetchmailrc 文件：

```
poll example.net protocol pop3 fetchall pass MySecret
```

因为包含了密码 MySecret， 这个文件应该只有 user 可读。

要使用正确的 from: 头来发送文件， 您必须告诉 sendmail 使用 user@example.net 而不是i user@bsd.home。 另外， 您可能也需要要求 sendmail 通过 relay.example.net 来发送邮件， 以便更快地传送它们。

以下的 .mc 文件应该可以满足您的需求：

```
VERSIONID(`bsd.home.mc version 1.0')
OSTYPE(bsd4.4)dnl
FEATURE(nouucp)dnl
MAILER(local)dnl
MAILER(smtp)dnl
Cwlocalhost
Cwbsd.home
MASQUERADE_AS(`example.net')dnl
FEATURE(allmasquerade)dnl
FEATURE(masquerade_envelope)dnl
FEATURE(nocanonify)dnl
```

```
FEATURE(nodns)dnl
define(`SMART_HOST', `relay.example.net')
Dmbsd.home
define(`confDOMAIN_NAME',`bsd.home')dnl
define(`confDELIVERY_MODE',`deferred')dnl
```

如何转换这个 .mc 文件到 sendmail.cf 文件的细节，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 验证

在您的邮件服务器上启用 SMTP 验证有很多好处。 SMTP 验证可以让 sendmail 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们的邮件客户端配置。

1. 从 ports 安装 security/cyrus-sasl2。 这个 port 位于 security/cyrus-sasl2。 security/cyrus-sasl2 port 支持很多可以在编译时指定的可选项。 由于我们要使用 SMTP 身份验证，
因此要确认没有禁用 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) 并在其中增加下列配置：

   ```
   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时的代理人角色。
这避免了为每个需要使用 SMTP 身份验证的用户建立对应的用户名和口令的麻烦，
也确保了登录与邮件的口令一致。

4. 现在编辑 /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
   ```

   这些配置将告诉系统在联编 sendmail 时使用适当的配置选项来在编译过程中连入 cyrus-sasl2.
在重新编译 sendmail 之前， 请确认已经安装了 cyrus-sasl2。

5. 重新编译 sendmail 运行如下命令：

   ```
   # 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 和共享库没有大的变化并且它们都必须可用，sendmail 编译应该没有任何问题。

6. sendmail 被重新编译和安装后， 编辑您的 /etc/mail/freebsd.mc 文件 (或者无论您选择使用的您的哪个 .mc 文件。许多管理员选择使用跟 hostname(1) 一样的唯一的 .mc 文件输出)。添加这些行在这个文件：

```
dnl set SASL options
TRUST_AUTH_MECH(`GSSAPI DIGEST-MD5 CRAM-MD5 LOGIN')dnl
define(`confAUTH_MECHANISMS', `GSSAPI DIGEST-MD5 CRAM-MD5 LOGIN')dnl
```

这些选项配置有不同的方法，对于 sendmail 验证用户。 如果您想要使用除 pwcheck 之外的方法，请参考相关文档。

7. 最后，在 /etc/mail 运行 make(1)。 它将建立您的新 .mc 文件并建立一个 .cf 文件命名为 freebsd.cf (或者您想使用您的其它名字的 .mc文件)。接着使用命令 make install restart，这将复制文件到 sendmail.cf，并且正确的重新启动 sendmail。 更多有关这个过程的信息，您可以参考 /etc/mail/Makefile 文件。

如果所每个步骤都做对了， 您应该可以通过您的邮件客户端进入您的登录信息并且传送一个测试信息。 更多的分析，设置 sendmail 的 LogLevel 到 13 并且查看 /var/log/maillog 中的信息。

如欲了解更多的信息， 请参看 sendmail 网站上的 关于 SMTP 验证 的介绍。

29.11. 邮件用户代理

邮件用户代理 (MUA) 是一个用于收发邮件的应用程序。 更进一步， 随着电子邮件的 "演化" 并愈发复杂， MUA 在和电子邮件相结合方面变得日趋强大； 这为用户提供了更多的功能和灵活性。 FreeBSD 包含了对于众多邮件用户代理的支持， 所有这些都可以通过 FreeBSD Ports Collection 来轻松安装。 用户可以选择类似 evolution 以及 balsa 这样的图形界面程序， 也可以选择类似 mutt、 alpine 或 mail 这样的控制台程序， 或者某些大型机构使用的 web 界面。

29.11.1. mail

mail(1) 是 FreeBSD 中默认的邮件用户代理 (MUA)。 它是一个基于控制台的 MUA， 提供了所有用于收发文本形式的电子邮件所需的基本功能， 虽然它处理附件的能力有限， 而且只支持本地的信箱。

虽然 mail 没有内建的 POP 或 IMAP 服务器支持， 然而这些信箱可以通过类似 fetchmail 这样的应用程序， 来下载到本地的 mbox 文件中。 这一应用程序在本章的稍后部分 (使用 fetchmail) 进行了介绍。

要收发邮件， 只需简单地使用 mail 命令， 如下所示：

```
% mail
```

用户保存在 /var/mail 中的信箱的内容会被 mail 程序自动地读取。 如果信箱是空的， 程序会退出并给出一个消息表示没有邮件。 一旦读完了信箱， 将启动应用程序的界面， 并列出邮件。 所有的邮件会被自动编号， 类似下面的样子：

```
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"
```

现在， 您通过使用 mail 的 `t` 命令， 并给出邮件的编号， 就可以看到邮件了。 在这个例子中，
我们将阅读第一封邮件：

```
& 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` 键将显示完整的邮件头。 要再次查看邮件的列表， 可以使用 `h` 键。

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` 这两个 mail键。 `R` 键会要求 mail
只回复发送邮件的人， 而 `r` 不仅回复发送邮件的人， 而且也会将回复抄送给原来邮件的其他接收者。
如果需要， 也可以在这些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，
在邮件的最后应该有一个只有一个 `.` 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， 应该使用 `,` 隔开。
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) 联机手册则可以获得更多关于 mail 的帮助信息。

正如前面所提到的那样， mail(1) 命令在设计时没有考虑到要处理附件，
因而在这方面他的功能很弱。 新的 MUA， 如 mutt， 能够更好地处理附件。
但如果您仍然希望使用 mail 命令， 那么 converters/mpack port
则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用户代理， 它提供了许多卓越的功能， 包括：

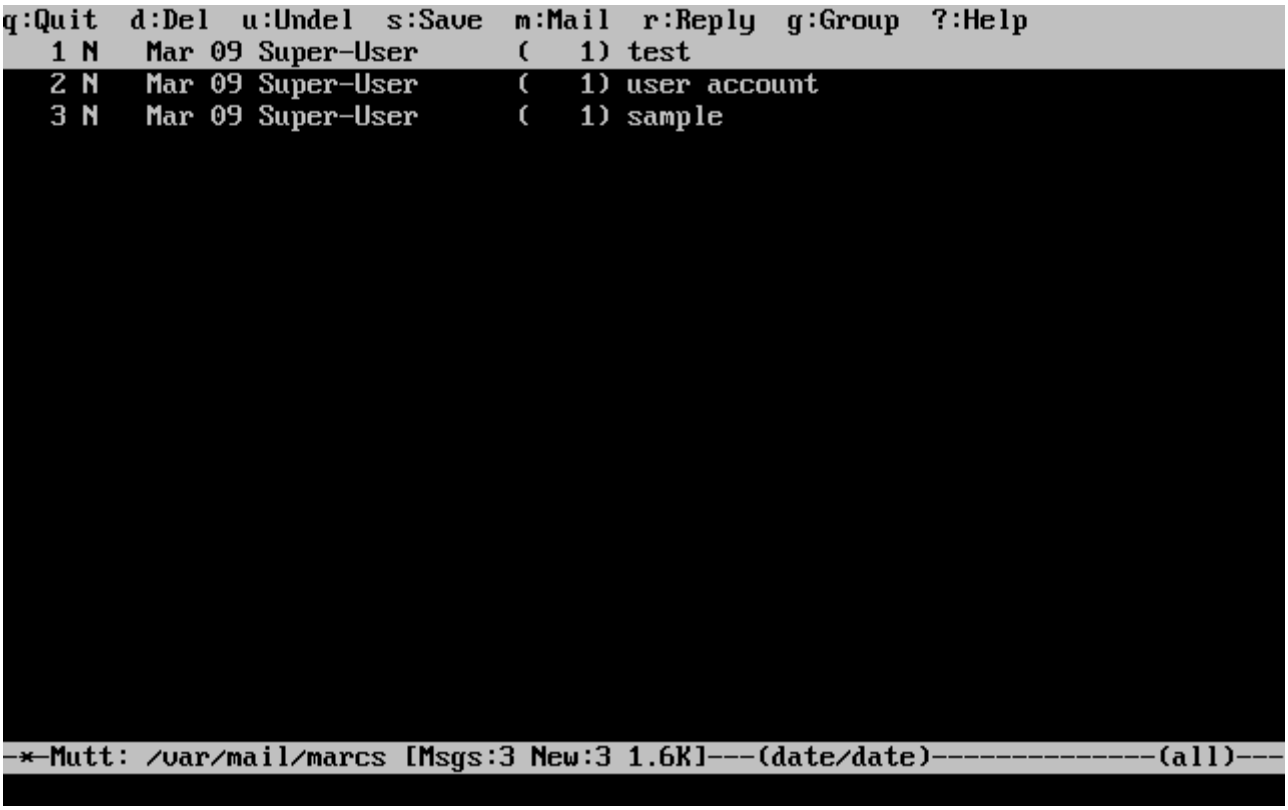
- 能够按线索阅读邮件；
- 支持使用 PGP 对邮件进行数字签名和加密；
- 支持 MIME；
- 支持 Maildir；
- 高度可定制。

所有这些特性， 都使得 mutt 得以跻身于目前最先进的邮件用户代理的行列。 请参考
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， 则可以通过使用 mail/mutt-devel port
安装。 通过 port 安装之后，可以通过下面的命令来启动 mutt：

```
% mutt
```

mutt 会自动读取 /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， 则 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：



要阅读邮件， 只需用光标键选择它， 然后按 Enter 键。 以下是 mutt 显示邮件的例子：

```
i:Exit  -:PrevPg  <Space>:NextPg v: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 receieve it.

-N  - 1/1: Super-User              test                              -- (all)
```

和 mail(1) 类似， mutt 允许用户只回复发件人， 或者回复所有人。 如果只想回复发信人， 使用 `r` 快捷键。 要回复所有人 (group reply)， 可以用 `g` 快捷键。

mutt 会使用 vi(1) 命令作为编辑器， 用于创建和回复邮件。
这一行为可以通过建立用户自己的 .muttrc 文件来订制， 方法是修改 editor 变量或配置 EDITOR 环境变量。 请参见 http://www.mutt.org/ 以了解配置 mutt 的进一步信息。

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 将启动 vi(1)，
您可以在其中撰写邮件。 写好邮件的内容之后， 存盘并退出 vi， 则 mutt 将继续，
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 也提供了相当详尽的帮助， 在绝大多数菜单中， 都可以使用 ? 键将其呼出。
屏幕顶行中也会给出常用的快捷键。

29.11.3. alpine

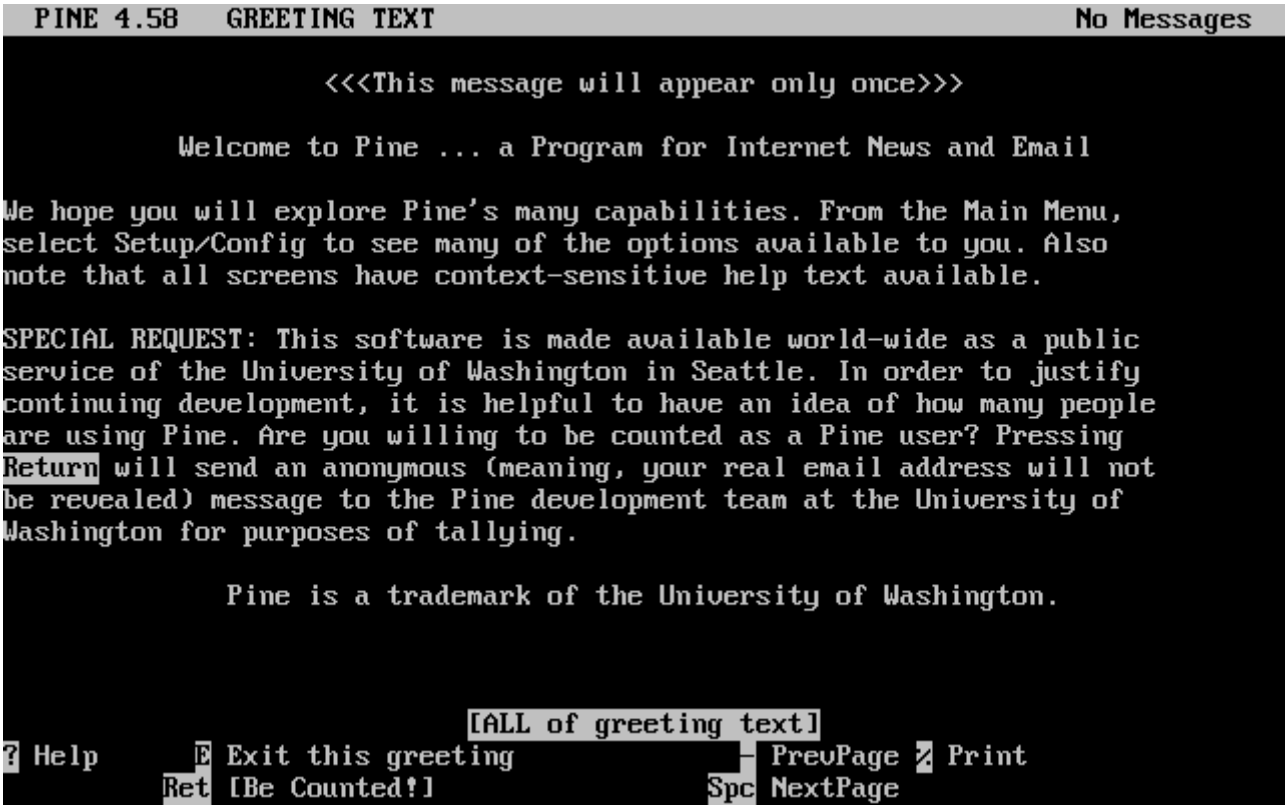
alpine 主要是针对初学者设计的， 但也提供了一些高级功能。

过去， alpine 软件被发现有许多远程漏洞，
这些漏洞会允许远程的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 问题都已经被修正了， 但 alpine 的代码是以很不安全的风格编写的， 并且 FreeBSD
安全官相信仍然有一些尚未被发现的安全漏洞。 您应当考虑并承担安装 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启动：

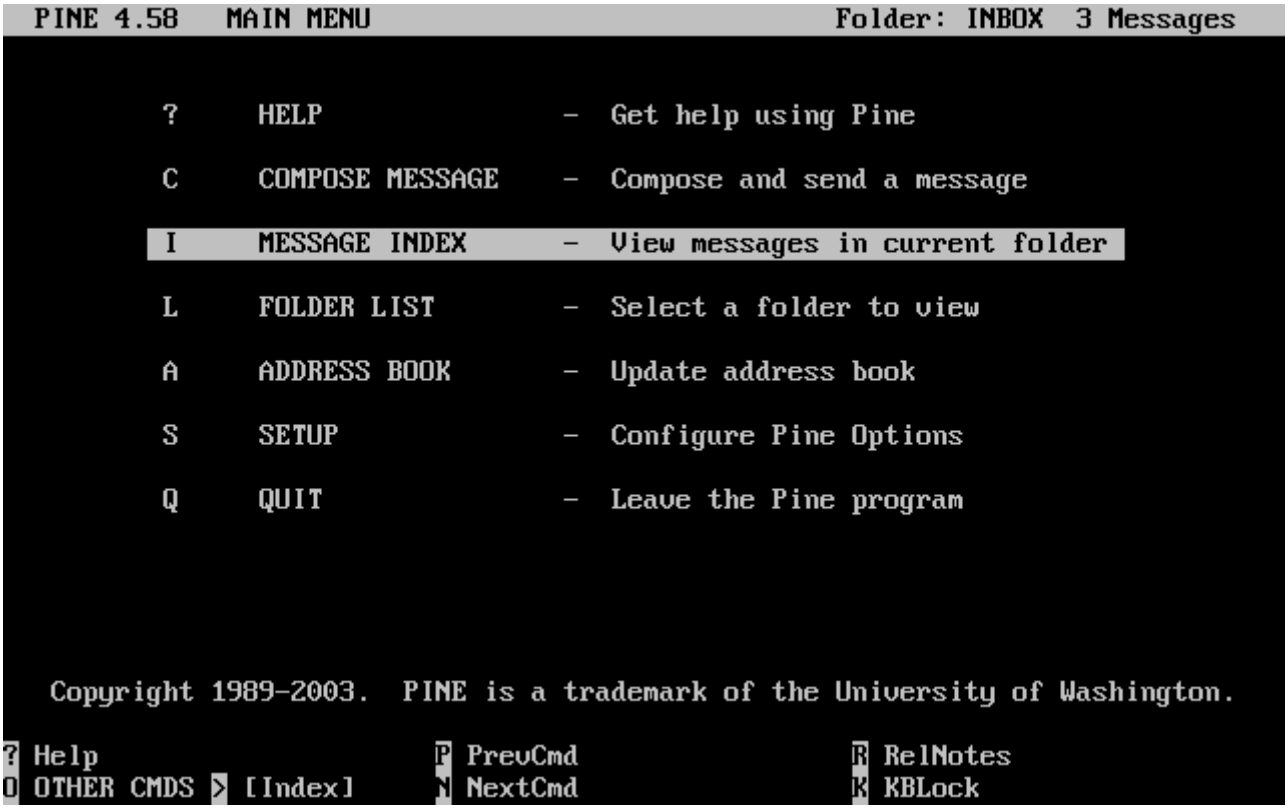
```
% alpine
```

第一次启动 alpine 时， 它会显示出一个欢迎页， 并给出简要的介绍， 以及 alpine
开发小组要求用户匿名发送一封邮件， 以便帮助他们了解有多少用户在使用他们开发的客户程序的请求。
要发送这封匿名的邮件， 请按 Enter， 您也可以按 E 退出， 而不发送匿名邮件。
下面是欢迎页的一个例子：

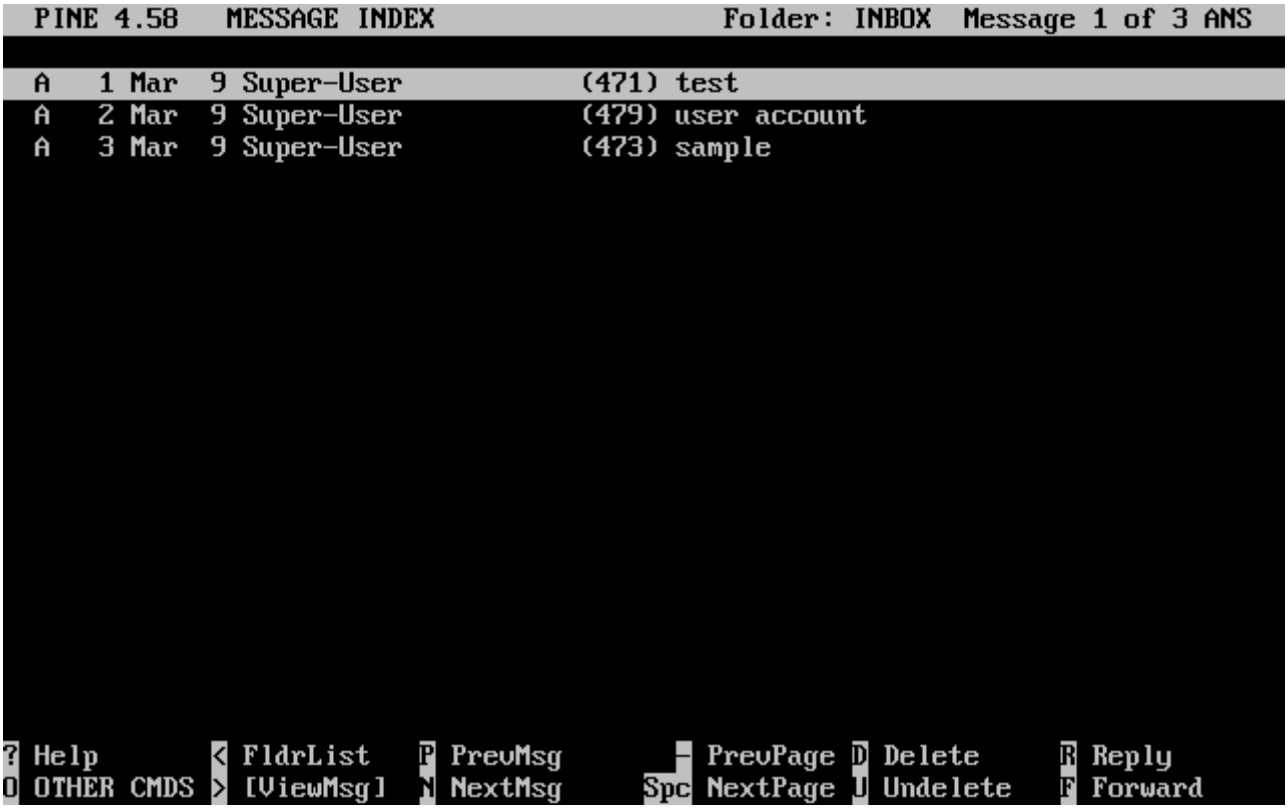


接下来展现给用户的将是主菜单， 可以很容易地通过光标键在上面进行选择。
这个主菜单提供了用于撰写新邮件、 浏览邮件目录， 甚至管理地址簿等等的快捷方式。
主菜单下面是完成各种功能的快捷键说明。

由 alpine 打开的默认目录是 inbox。 要查看邮件索引， 应按 I， 或选择下面所示的 MESSAGE INDEX 选项：

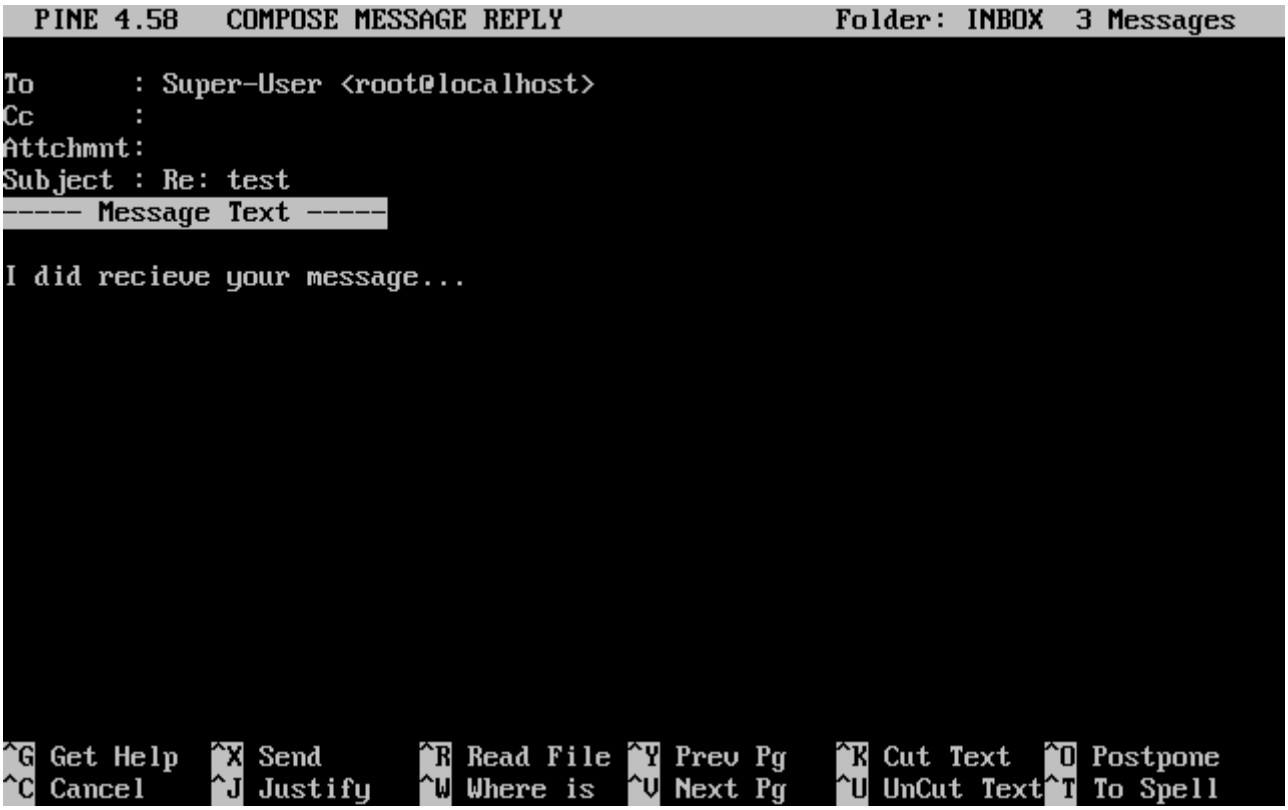


邮件索引展示了当前目录下的邮件， 可以使用光标键翻阅。 按 Enter 键阅读高亮选定的邮件。



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 键， 它告诉 MUA 回复正显示的邮件。

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 工具使得浏览邮件变得更加简单，并且要比 vi(1) 或 mail(1) 更能容忍误操作。回复写好之后，可以用 Ctrl + X 来发出它。此前，alpine 程序会要求确认。



alpine 程序可以通过使用主菜单中的 SETUP 选项来进行定制。请参考 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， 访问这些邮件就变得更方便了。 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 协议。
- 通过 SMTP 转发邮件， 这使得过滤、 转发， 以及邮件别名能够正常工作。
- 能够以服务程序的方式运行， 并周期性地检查邮件。
- 能够从多个信箱收取邮件， 并根据配置， 将这些邮件转发给不同的本地用户。

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， 建议将其设置为只有属主所有， 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 和口令 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， 并根据需要转到不同的本地用户：

```
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， 并给出 fetchmail 在轮询 .fetchmailrc 文件中列出的服务器的时间间隔， 来以服务程序的方式运行。 下面的例子会让 fetchmail 每 600 秒轮询一次：

```
% 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过滤进入邮件的应用程序。 它允许用户定义 "规则"， 并用这些规则来匹配进入的邮件， 进而执行某些特定的功能， 或将这些邮件转发到其他信箱和/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绝大多数 MTA 中； 请参考您使用的 MTA 的文档了解具体的作法。 另外， procmail 可允许通过把下面的设置加入到用户主目录中的 .forward 文件中， 来启用 procmail 功能：

```
"|exec /usr/local/bin/procmail || exit 75"
```

接下来我们将介绍一些基本的 procmail 规则， 以及它们都是做什么的。 各种各样的规则， 都应该写到 .procmailrc 文件中， 而这个文件则必须放在用户的主目录下。

主要的规则， 也可以在 procmailex(5) 联机手册中找到。

将所有来自 user@example.com 的邮件， 转发到外部地址 goodmail@example2.com：

```
:0
* ^From.*user@example.com
! goodmail@example2.com
```

转发所有不超过 1000 字节的邮件到外部地址 goodmail@example2.com：

```
:0
* < 1000
! goodmail@example2.com
```

把所有发送到 alternate@example.com 的邮件放到信箱 alternate 中：

```
:0
* ^TOalternate@example.com
alternate
```

将所有标题为 "Spam" 的邮件发到 /dev/null：

```
:0
^Subject:.*Spam
/dev/null
```

将收到的所有 FreeBSD.org 邮件列表的邮件， 转发到各自的信箱：

```
:0
* ^Sender:.owner-freebsd-\/[^@]+@FreeBSD.ORG
{
  LISTNAME=${MATCH}
  :0
  * LISTNAME??^\/[^@]+
  FreeBSD-${MATCH}
}
```

Chapter 30. 网络服务器

30.1. 概要

本章将覆盖某些在 UNIX® 系统上常用的网络服务。话题将会涉及
如何安装、配置、测试和维护多种不同类型的网络服务。本章节中将提
供大量配置文件的样例，期望能够对您有所裨益。

在读完本章之后，您将会知道：

- 如何管理 inetd。
- 如何设置运行一个网络文件系统。
- 如何配置一个网络信息服务器以共享用户帐号。
- 如何通过DHCP自动配置网络。
- 如何配置一个域名服务器。
- 如何设置Apache HTTP 服务器。
- 如何设置文件传输（FTP）服务器。
- 如何使用Samba为 Windows® 客户端设置文件和打印服务。
- 如何同步时间和日期，以及如何设置使用NTP协议的时间服务器。
- 如何配置标准的日志守护进程， syslogd， 接受远程主机的日志。

在阅读此章节之前，您应当：

- 理解有关/etc/rc中脚本的基本知识。
- 熟悉基本网络术语。
- 懂得如何安装额外的第三方软件（安装应用程序. Packages 和 Ports）。

30.2. inetd"超级服务器"

30.2.1. 总览

inetd(8) 有时也被称作 "Internet 超级服务器"， 因为它可以为多种服务管理连接。 当 inetd 收到连接时，
它能够确定连接所需的程序， 启动相应的进程， 并把 socket 交给它 (服务 socket 会作为程序的标准输入、
输出和错误输出描述符)。 使用 inetd 来运行那些负载不重的服务有助于降低系统负载，
因为它不需要为每个服务都启动独立的服务程序。

一般说来， inetd 主要用于启动其它服务程序， 但它也有能力直接处理某些简单的服务， 例如 chargen、
auth， 以及 daytime。

这一节将介绍关于如何通过命令行选项， 以及配置文件 /etc/inetd.conf 来对 inetd
进行配置的一些基础知识。

30.2.2. 设置

inetd 是通过 rc(8) 系统启动的。 inetd_enable 选项默认设为 NO， 但可以在安装系统时，
由用户根据需要通过 sysinstall 来打开。 将：

```
inetd_enable="YES"
```

或

```
inetd_enable="NO"
```

写入 /etc/rc.conf 可以启用或禁用系统启动时 inetd 的自动启动。 命令：

```
# /etc/rc.d/inetd rcvar
```

可以显示目前的设置。

此外， 您还可以通过 inetd_flags 参数来向 inetd 传递额外的其它参数。

30.2.3. 命令行选项

与多数服务程序类似， inetd 也提供了为数众多的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 选项来传给 inetd。 默认情况下， inetd_flags 设为 -wW -C 60， 者表示希望为 inetd 的服务启用 TCP wrapping， 并阻止来自同一 IP 每分钟超过 60 次的请求。

虽然我们会在下面介绍关于限制连接频率的选项，
但初学的用户可能会很高兴地发现这些参数通常并不需要进行修改。 在收到超大量的连接请求时，
这些选项则有可能会发挥作用。 完整的参数列表， 可以在 inetd(8) 联机手册中找到。

-c maximum
: 指定单个服务的最大并发访问数量，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
: 指定单个服务一分钟内能被单个IP地址调用的最大次数，
默认不限。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
: 指定单个服务一分钟内能被调用的最大次数，默认为256。 设为0 则允许不限次数调用。

-s maximum
: 指定同一 IP 同时请求同一服务时允许的最大值； 默认值为不限制。 您可以通过 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强制 inetd 重新读取配置文件：

例 36. 重新加载 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务程序。 在这个文件中， 前面有 "#" 的内容被认为是注释。
/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
```

下面是针对 IPv4 的 ftpd(8) 服务的例子：

```
ftp    stream  tcp    nowait  root    /usr/libexec/ftpd     ftpd -l
```

service-name
指明各个服务的服务名。其服务名必须与/etc/services中列出的一致。 这将决定inetd会监听哪个port。一旦有新的服务需要添加，必须先在/etc/services里面添加。

socket-type
可以是stream、dgram、raw或者 seqpacket。 stream 用于基于连接的 TCP 服务；而 dgram 则用于使用 UDP 协议的服务。

protocol
下列之一：

协议	说明
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 里头调用的服务是否可以自己处理socket. dgram socket类型必须使用wait，而stream socket daemons， 由于通常使用多线程方式，应当使用 nowait. wait 通常把多个 socket 丢给单个服务进程， 而 nowait 则 会为每个新的 socket 生成一个子进程。

max-child 选项能够配置 inetd 能为本服务派生出的最大子进程数量。
如果某特定服务需要限定最高10个实例， 把/10 放到nowait后头就可以了。 指定 /0 表示不限制子进程的数量。

除了 max-child 之外， 还有两个选项可以限制来自同一位置到特定服务的最大连接数。 max-connections-per-ip-per-minute 可以限制特定 IP 地址每分钟的总连接数， 例如， 限制任何 IP 地址每分钟最多连接十次。 max-child-per-ip 则可以限制为某一 IP 地址在任何时候所启动的子进程数量。 这些选项对于防止针对服务器有意或无意的资源耗竭和拒绝服务 (DoS) 攻击十分有用。

这个字段中， 必须指定 wait 或 nowait 两者之一。 而 max-child、 max-connections-per-ip-per-minute 和 max-child-per-ip 则是可选项。

流式多线程服务， 并且不配置任何 max-child、 max-connections-per-ip-per-minute 或 max-child-per-ip 限制时， 其配置为： nowait。

同一个服务， 但希望将服务启动的数量限制为十个时， 则是： nowait/10。

同样配置， 限制每个 IP 地址每分钟最多连接二十次， 而同时启动的子进程最多十个， 应写作：nowait/10/20。

下面是 fingerd(8) 服务的默认配置：

```
finger stream  tcp    nowait/3/10 nobody /usr/libexec/fingerd fingerd -s
```

最后这个例子中， 将子进程数限制为 100 个， 而任意 IP 最多同时建立 5 个连接： nowait/100/0/5。

user
: 该开关指定服务将以什么用户身份运行。一般而言，服务运行身份是root。基于安全目的，可以看到有些服务以 daemon身份，或者是最小特权的 nobody身份运行。

server-program
: 当连接到来时，执行服务程序的全路径。如果服务是由 inetd内置提供的，以internal代替。

server-program-arguments
: 当server-program调用到时，该开关 的值通过argv[0]通过传递给服务而工作。如果命令行为：mydaemon -d，则 mydaemon -d为server-program-arguments开关的值。同样的，如果服务是由inetd 内置提供的，这里还是 internal。

30.2.5. Security

随安装时所选的模式不同， 许多 inetd 的服务可能已经默认启用。 如果确实不需要某个特定的服务，则应考虑禁用它。 在 /etc/inetd.conf 中， 将对应服务的那行前面加上 "#"， 然后 重新加载 inetd 配置就可以了。 某些服务， 例如 fingerd， 可能是完全不需要的， 因为它们提供的信息可能对攻击者有用。

某些服务在设计时是缺少安全意识的， 或者有过长或压根没有连接请求的超时机制。这使得攻击者能够通过缓慢地对这些服务发起连接， 并耗尽可用的资源。 对于这种情况， 设置 max-connections-per-ip-per-minute、 max-child 或 max-child-per-ip 限制，来制约服务的行为是个好办法。

默认情况下，TCP wrapping 是打开的。参考 hosts_access(5) 手册，以获得更多关于在各种 inetd调用的服务上设置TCP限制的信息。

30.2.6. 杂项

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服务， 它可以配置为提供不同级别的服务，而其它服务则通常只能简单的打开或关闭。

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支持的文件系统中的一种， 也被称为 NFS。NFS允许一个系统在网络上与它人共享目录和文件。通过使用NFS，用户和程序可以象访问本地文件一样访问远端系统上的文件。

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。
- 用户不必在每个网络上机器里头都有一个home目录。Home目录可以被放在NFS服务器上并且在网络上处处可用。
- 诸如软驱，CDROM，和 Zip® 之类的存储设备可以在网络上面被别的机器使用。这可以减少整个网络上的可移动介质设备的数量。

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， 以及至少一台客户机，
客户机远程地访问保存在服务器上的数据。 要让这一切运转起来， 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，为来自NFS客户端的 请求服务。
mountd	NFS挂载服务，处理nfsd(8)递交过来的请求。
rpcbind	此服务允许 NFS 客户程序查询正在被 NFS 服务使用的端口。

客户端同样运行一些进程，比如 nfsiod。 nfsiod处理来自NFS的请求。
这是可选的，而且可以提高性能，对于普通和正确的操作来说并不是必须的。 参考
nfsiod(8)手册获得更多信息。

30.3.2. 配置NFS

NFS的配置过程相对简单。这个过程只需要 对/etc/rc.conf文件作一些简单修改。

在NFS服务器这端，确认/etc/rc.conf 文件里头以下开关都配上了：

```
rpcbind_enable="YES"
nfs_server_enable="YES"
mountd_flags="-r"
```

只要NFS服务被置为enable，mountd 就能自动运行。

在客户端一侧，确认下面这个开关出现在 /etc/rc.conf里头:

```
nfs_client_enable="YES"
```

/etc/exports文件指定了哪个文件系统 NFS应该输出（有时被称为"共享"）。
/etc/exports里面每行指定一个输出的文件系统和
哪些机器可以访问该文件系统。在指定机器访问权限的同时，访问选项
开关也可以被指定。有很多开关可以被用在这个文件里头，不过不会在这
里详细谈。您可以通过阅读exports(5) 手册来发现这些开关。

以下是一些/etc/exports的例子：

下面是一个输出文件系统的例子， 不过这种配置与您所处的网络环境及其配置密切相关。 例如， 如果要把
/cdrom 输出给与服务器域名相同的三台计算机 (因此例子中只有机器名， 而没有给出这些计算机的域名)，
或在 /etc/hosts 文件中进行了这种配置。 -ro 标志表示把输出的文件系统置为只读。 由于使用了这个标志，
远程系统在输出的文件系统上就不能写入任何变动了。

```
/cdrom -ro host1 host2 host3
```

下面的例子可以输出/home给三个以IP地址方式表示的主机。
对于在没有配置DNS服务器的私有网络里头，这很有用。 此外， /etc/hosts
文件也可以用以配置主机名；参看 hosts(5) 。 -alldirs 标记允许子目录被作为挂载点。
也就是说，客户端可以根据需要挂载需要的目录。

```
/home  -alldirs  10.0.0.2 10.0.0.3 10.0.0.4
```

下面几行输出 /a ，以便两个来自不同域的客户端可以访问文件系统。 -maproot=root 标记授权远端系统上的 root 用户在被输出的文件系统上以root身份进行读写。 如果没有特别指定 -maproot=root 标记， 则即使用户在远端系统上是 root 身份， 也不能修改被输出文件系统上的文件。

```
/a  -maproot=root  host.example.com box.example.org
```

为了能够访问到被输出的文件系统，客户端必须被授权。 请确认客户端在您的 /etc/exports 被列出。

在 /etc/exports 里头，每一行里面，输出信息和文件系统一一对应。
一个远程主机每次只能对应一个文件系统。而且只能有一个默认入口。比如，假设 /usr 是独立的文件系统。这个 /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， 有两行指定输出到同一主机， client. 解决这一问题的正确的格式是：

```
/usr/src /usr/ports  client
```

在同一文件系统中， 输出到指定客户机的所有目录， 都必须写到同一行上。
没有指定客户机的行会被认为是单一主机。 这限制了你可以怎样输出的文件系统，
但对绝大多数人来说这不是问题。

下面是一个有效输出列表的例子， /usr 和 /exports 是本地文件系统：

```
# 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， 就必须让 mountd 服务重新检查它， 以便使修改生效。
一种方法是通过给正在运行的服务程序发送 HUP 信号来完成：

```
# kill -HUP `cat /var/run/mountd.pid`
```

或指定适当的参数来运行 mountd rc(8) 脚本：

```
# /etc/rc.d/mountd onereload
```

关于使用 rc 脚本的细节， 请参见 在 FreeBSD 中使用 rc。

另外， 系统重启动可以让 FreeBSD 把一切都弄好。 尽管如此， 重启不是必须的。 以 root 身份执行下面的命令可以搞定一切。

在 NFS 服务器端：

```
# rpcbind
# nfsd -u -t -n 4
# mountd -r
```

在 NFS 客户端：

```
# nfsiod -n 4
```

现在每件事情都应该就绪，以备挂载一个远端文件系统。 在这些例子里头， 服务器名字将是：server ，而客户端的名字将是： client。
如果您只打算临时挂载一个远端文件系统或者只是打算作测试配置正确与否， 只要在客户端以 root 身份执行下面的命令：

```
# mount server:/home /mnt
```

这条命令会把服务端的 /home 目录挂载到客户端的 /mnt 上。 如果配置正确，您应该可以进入客户端的 /mnt 目录并且看到所有服务端的文件。

如果您打算让系统每次在重启动的时候都自动挂载远端的文件系统，把那个文件系统加到 /etc/fstab 文件里头去。下面是例子：

```
server:/home    /mnt    nfs rw  0   0
```

fstab(5) 手册里有所有可用的开关。

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。 要启用它， 需要在服务器和客户机的 /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该程序：

```
# /etc/rc.d/lockd start
# /etc/rc.d/statd start
```

如果并不需要真的在 NFS 客户机和 NFS 服务器间确保上锁的语义， 可以让 NFS 客户机在本地上锁，方法是使用 mount_nfs(8) 时指定 -L 参数。 请参见 mount_nfs(8) 联机手册以了解更多细节。

30.3.4. 实际应用

NFS 有很多实际应用。下面是比较常见的一些：

- 多个机器共享一台CDROM或者其他设备。这对于在多台机器中安装软件来说更加便宜跟方便。
- 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目录可能会带来便利。
  这些目录能被输出到网络以便用户不管在哪台工作站上登录，总能得到相同的home目录。
- 几台机器可以有通用的/usr/ports/distfiles 目录。
  这样的话，当您需要在几台机器上安装port时，您可以无需在每台设备上下载而快速访问源码。

30.3.5. 通过 amd 自动地挂接

amd(8) (自动挂接服务) 能够自动地在访问时挂接远程的文件系统。
如果文件系统在一段时间之内没有活动， 则会被 amd 自动卸下。 通过使用 amd，
能够提供一个持久挂接以外的选择， 而后者往往需要列入 /etc/fstab。

amd 通过将自己以 NFS 服务器的形式， 附加到 /host 和 /net 目录上来工作。
当访问这些目录中的文件时， amd 将查找相应的远程挂接点， 并自动地挂接。 /net 用于挂接远程 IP
地址上导出的文件系统， 而 /host 则用于挂接远程主机名上的文件系统。

访问 /host/foobar/usr 中的文件， 相当于告诉 amd 尝试挂接在主机 foobar 上导出的 /usr。

例 37. 通过 amd 来挂接导出的文件系统

您可以通过使用 showmount 命令来查看远程主机上导出的文件系统。 例如， 要查看 foobar
上导出的文件系统， 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 显示了导出的 /usr。 当进入 /host/foobar/usr 这个目录时，
amd 将尝试解析主机名 foobar 并自动地挂接需要的文件系统导出。

amd 可以通过启动脚本来启动， 方法是在 /etc/rc.conf 中加入：

```
amd_enable="YES"
```

除此之外， 还可以给 amd 通过 amd_flags 选项来传递额外的参数。 默认情况下， amd_flags 为：

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

/etc/amd.map 文件定义了挂接导出文件系统时所使用的默认选项。 /etc/amd.conf 文件，
则定义了更多关于 amd 的高级功能选项。

请参考 amd(8) 和 amd.conf(8) 联机手册， 以了解进一步的情况。

30.3.6. 与其他系统集成时的常见问题

某些特定的 ISA PC 系统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络问题， 特别是与 NFS 配合使用时。 这些问题并非 FreeBSD 所特有的， 但 FreeBSD 系统会受到这些问题的影响。

这样的问题， 几乎总是在当 (FreeBSD) PC 系统与高性能的工作站， 例如 Silicon Graphics, Inc., 和 Sun Microsystems, Inc. 的工作站联网时发生。 NFS 挂接能够正常工作， 而且一些操作也可能成功， 但服务器会很快变得对客户机不太理会， 虽然对其他客户机的请求仍然能够正常处理。 这种情况通常发生在客户端， 无论它是一个 FreeBSD 系统或是终端。 在许多系统上， 一旦发生了这样的问题， 通常没办法正常地关闭客户机。 唯一的办法通常是让终端复位， 因为这一 NFS 状况没有办法被解决。

尽管 "正确的" 解决办法， 是为 FreeBSD 系统配备一块高性能的、 适用的以太网适配器， 然而也有办法绕过问题并得到相对满意的结果。 如果 FreeBSD 系统是 服务器， 则在客户机挂接时， 应该指定 -w=1024。 如果 FreeBSD 系统是 客户机， 则应加入 -r=1024 参数。 这些选项可以通过在对应的 fstab 的第四个字段加入， 以便让客户机能够自动地挂接， 或者通过 mount(8) 的 -o 参数在手工挂接时指定。

还需要注意的是另一个问题， 有时会被误认为是和上面一样的问题。 这个问题多见于 NFS 服务器和客户机在不同的网络上时。 如果是这种情况， 一定要 确定 您的路由器确实把必需的 UDP 信息路由到了目的地， 否则您将什么也做不了。

下面的例子中， fastws 是主机 (接口) 的名字， 它是一台高性能的终端， 而 freebox 是另一台主机 (接口) 的名字， 它是一个使用较低性能的以太网适配器的 FreeBSD 系统。 同时， /sharedfs 将被导出成为 NFS 文件系统 (参见 exports(5))， 而 /project 将是客户机上挂接这一导出文件系统的挂接点。 所有的应用场景中， 请注意附加选项， 例如 hard 或 soft 以及 bg 可能是您的应用所需要的。

关于 FreeBSD 系统 (freebox) 作为客户机的示范 /etc/fstab 文件， 见于 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 系统作为服务器的例子， 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， 都能够在没有上述读写尺寸限制的情况下正常工作。

对于那些关心到底是什么问题的人， 下面是失败如何发生的解释， 同时这也说明了为什么这是一个无法恢复的问题。 典型情况下， NFS 会使用一个 "块" 为单位进行操作， 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 "块" 会分成多个以太网包， 虽然在更高层的代码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 确认。 高性能的工作站， 可以将构成 NFS 单元的包迅速发出， 其节奏会快到标准允许的最大限度。 在容量较小的卡上， 后来的包会冲掉同一单元内的较早的包， 因而整个单元无法被重建或确认。 其结果是， 工作站将超时并重试， 但仍然是完整的 8 K 单元， 这一过程将无休止地重复下去。

如果将单元尺寸限制在以太网包尺寸之下， 我们就能够确保每一个以太网包都能够被独立地接收和确认，

从而避免了上面的死锁情形。

溢出在高性能工作站将数据库投向 PC 系统时仍会发生， 但在更好的网卡上，
能够保证这类溢出不会在每一个 NFS "单元" 上都发生。 当出现溢出时， 被影响的单元被重传，
因而此时有很大的机会它将被正确接收、 重组， 并确认。

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， 它基本上已经成为了业界标准； 所有主流的类 UNIX® 系统 (Solaris™, HP-UX, AIX®, Linux, NetBSD, OpenBSD, FreeBSD, 等等) 都支持 NIS。

NIS 也就是人们所熟知的黄页(Yellow Pages)， 但由于商标的问题， Sun 将其改名为现在的名字。
旧的术语 (以及 yp)， 仍然经常可以看到， 并被广泛使用。

这是一个基于 RPC 的客户机/服务器系统， 它允许在一个 NIS 域中的一组机器共享一系列配置文件。 这样，
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， 并在单点上增加、
删除或修改配置数据。

尽管实现的内部细节截然不同， 这和 Windows NT® 域系统非常类似，
以至于可以将两者的基本功能相互类比。

30.4.2. 您应该知道的术语和进程

有一系列术语和重要的用户进程将在您在 FreeBSD 上实现 NIS 时用到， 无论是在创建 NIS 服务器， 或作为 NIS 客户机:

术语	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® 域名类似， NIS 域名与 DNS 无关。
rpcbind	必须运行这个程序， 才能够启用 RPC (远程过程调用， NIS 用到的一种网络协议)。 如果没有运行 rpcbind， 则没有办法运行 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。 这样， 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-服务器通讯的核心； 如果客户机上的 ypbind 死掉的话， 它将无法访问 NIS 服务器。
ypserv	只应在 NIS 服务器上运行它； 这是 NIS 的服务器进程。 如果 ypserv(8) 死掉的话， 则服务器将不再具有响应 NIS 请求的能力 (此时， 如果有从服务器的话， 则会接管操作)。 有一些 NIS 的实现 (但不是 FreeBSD 的这个) 的客户机上， 如果之前用过一个服务器， 而那台服务器死掉的话， 并不尝试重新连接到另一个服务器。 通常， 发生这种情况时， 唯一的办法就是重新启动服务器进程 (或者， 甚至重新启动服务器) 或客户机上的 ypbind 进程。
rpc.yppasswdd	另一个只应在 NIS 主服务器上运行的进程； 这是一个服务程序， 其作用是允许 NIS 客户机改变它们的 NIS 口令。 如果没有运行这个服务， 用户将必须登录到 NIS 主服务器上， 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，
而从服务器则是保存这些信息的冗余副本。 客户机依赖于服务器向它们提供这些信息。

许多文件的信息可以通过这种方式来共享。 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这些本应在本地的文件中的资料的时候，
它都会向所绑定的 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。 这台服务器， 和 Windows NT® 域控制器类似， 会维护所有 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，
本书不打算对这种配置进行介绍， 因为这种配置， 通常只出现在小规模的 NIS 环境中。

- NIS 从服务器。 这一概念， 与 Windows NT® 的备份域控制器类似。 NIS 从服务器， 用于维护 NIS 主服务器的数据文件副本。 NIS 从服务器提供了一种冗余， 这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 客户机总是挂接到最先响应它们的 NIS 服务器上，
而这也包括来自从服务器的响应。
- NIS 客户机。 NIS 客户机， 和多数 Windows NT® 工作站类似， 通过 NIS 服务器 (或对于 Windows NT® 工作站， 则是 Windows NT® 域控制器) 来完成登录时的身份验证过程。

30.4.4. 使用 NIS/YP

这一节将通过实例介绍如何配置 NIS 环境。

30.4.4.1. 规划

假定您正在管理大学中的一个小型实验室。 在这个实验室中， 有 15 台 FreeBSD 机器，
目前尚没有集中的管理点； 每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， 如果您在实验室中增加一个用户，
将不得不在所有 15 台机器上手工执行 adduser 命令。 毋庸置疑， 这一现状必须改变，
因此您决定将整个实验室转为使用 NIS， 并使用两台机器作为服务器。

因此， 实验室的配置应该是这样的：

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	教员工作站
bird	10.0.0.5	客户机
cli[1-11]	10.0.0.[6-17]	其他客户机

如果您是首次配置 NIS， 仔细思考如何进行规划就十分重要。 无论您的网络的大小如何，
都必须进行几个决策。

30.4.4.1.1. 选择 NIS 域名

这可能不是您过去使用的 "域名(domainname)"。 它的规范的叫法， 应该是 "NIS 域名"。
当客户机广播对此信息的请求时， 它会将 NIS 域的名字作为请求的一部分发出。 这样，
统一网络上的多个服务器， 就能够知道谁应该回应请求。 您可以把 NIS
域名想象成以某种方式相关的一组主机的名字。

一些机构会选择使用它们的 Internet 域名来作为 NIS 域名。 并不推荐这样做， 因为在调试网络问题时，
这可能会导致不必要的困扰。 NIS 域名应该是在您网络上唯一的，
并且有助于了解它所描述的到底是哪一组机器。 例如对于 Acme 公司的美工部门， 可以考虑使用 "acme-art" 这样的 NIS 域名。 在这个例子中， 您使用的域名是 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。
如果您的网络上存在包含这类限制的机器， 就 必须 使用 Internet 域名来作为您的 NIS 域名。

30.4.4.1.2. 服务器的物理要求

选择 NIS 服务器时， 需要时刻牢记一些东西。 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，
则这台机器通常会陷于不可用的状态。 缺少用户和组信息， 会使绝大多数系统进入短暂的冻结状态。
基于这样的考虑， 您需要选择一台不经常重新启动， 或用于开发的机器来承担其责任。
如果您的网络不太忙， 也可以使用运行着其他服务的机器来安放 NIS 服务， 只是需要注意， 一旦 NIS 服务器不可用， 则 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 信息的正规版本， 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库， 称为 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， 因此可以建立很多这样的目录， 所支撑一个域对应一个。
每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， 翻译请求的域， 并将名字映射为相关的数据库文件的路径，
然后将来自数据库的数据传回客户机。

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， 而其具体步骤则取决于您的需要。 FreeBSD 提供了一步到位的 NIS 支持。 您需要做的全部事情， 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. 这一行将在网络启动 (例如重新启动) 时， 把 NIS 域名配置为 test-domain。

   ```
   nis_server_enable="YES"
   ```

2. 这将要求 FreeBSD 在网络子系统启动之后立即启动 NIS 服务进程。

   ```
   nis_yppasswdd_enable="YES"
   ```

3. 这将启用 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， 可能还需要增加其他一些项目。 请参见 关于 NIS 服务器同时充当 NIS 客户机 这一节， 以了解进一步的情况。

设置好前面这些配置之后， 需要以超级用户身份运行 /etc/netstart 命令。 它会根据 /etc/rc.conf
的设置来配置系统中的其他部分。 最后， 在初始化 NIS 映射之前， 还需要手工启动 ypserv 服务程序：

```
# /etc/rc.d/ypserv start
```

30.4.4.2.2. 初始化 NIS 映射

NIS 映射 是一些数据库文件， 它们位于 /var/yp 目录中。 这些文件基本上都是根据 NIS 主服务器的 /etc 目录自动生成的， 唯一的例外是： /etc/master.passwd 文件。 一般来说， 您会有非常充分的理由不将 root 以及其他管理帐号的口令发到所有 NIS 域上的服务器上。 因此， 在开始初始化 NIS 映射之前， 我们应该：

```
# cp /etc/master.passwd /var/yp/master.passwd
# cd /var/yp
# vi master.passwd
```

这里， 删除掉和系统有关的帐号对应的项 (bin、 tty、 kmem、 games， 等等)， 以及其他不希望被扩散到 NIS 客户机的帐号 (例如 root 和任何其他 UID 0 (超级用户) 的帐号)。

确认 /var/yp/master.passwd 这个文件是同组用户， 以及其他用户不可读的 (模式 600)！ 如果需要的话， 用 chmod 命令来改它。

完成这些工作之后， 就可以初始化 NIS 映射了！ FreeBSD 提供了一个名为 ypinit 的脚本来帮助您完成这项工作 (详细信息， 请见其联机手册)。 请注意， 这个脚本在绝大多数 UNIX® 操作系统上都可以找到， 但并不是所有操作系统的都提供。 在 Digital UNIX/Compaq Tru64 UNIX 上它的名字是 ypsetup。 由于我们正在生成的是 NIS 主服务器的映射， 因此应该使用 ypinit 的 -m 参数。 如果已经完成了上述步骤， 要生成 NIS 映射， 只需执行：

```
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 应该会根据 /var/yp/Makefile.dist 来创建 /var/yp/Makefile 文件。 创建完之后，这个文件会假定您正在操作只有 FreeBSD 机器的单服务器 NIS 环境。 由于 test-domain 还有一个从服务器， 您必须编辑 /var/yp/Makefile：

```
ellington# vi /var/yp/Makefile
```

应该能够看到这样一行， 其内容是

```
NOPUSH = "True"
```

(如果还没有注释掉的话)。

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还要简单。 登录到从服务器上， 并按照前面的方法， 编辑 /etc/rc.conf 文件。 唯一的区别是， 在运行 ypinit 时需要使用 -s 参数。 这里的 -s 选项， 同时要求提供 NIS 主服务器的名字， 因此我们的命令行应该是：

```
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.
```

现在应该会有一个叫做 /var/yp/test-domain 的目录。 在这个目录中， 应该保存 NIS
主服务器上的映射的副本。 接下来需要确定这些文件都及时地同步更新了。 在从服务器上， 下面的
/etc/crontab 项将帮助您确保这一点：

```
20   *   *   *   *   root  /usr/libexec/ypxfr passwd.byname
21   *   *   *   *   root  /usr/libexec/ypxfr passwd.byuid
```

这两行将强制从服务器将映射与主服务器同步。 由于主服务器会尝试确保所有其 NIS
映射的变动都知会从服务器， 因此这些项并不是绝对必需的。 不过，
由于保持其他客户端的口令信息正确性十分重要， 而这则依赖于从服务器，
强烈推荐明确指定让系统时常强制更新口令映射。 对于繁忙的网络而言， 这一点尤其重要，
因为有时可能出现映射更新不完全的情况。

现在， 在从服务器上执行 /etc/netstart， 就可以启动 NIS 服务了。

30.4.4.3. NIS 客户机

NIS 客户机会通过 ypbind 服务程序来与特定的 NIS 服务器建立一种称作绑定的联系。 ypbind 会检查系统的默认域 (这是通过 domainname 命令来设置的)， 并开始在本地网络上广播 RPC 请求。 这些请求会指定 ypbind 尝试绑定的域名。 如果已经配置了服务器， 并且这些服务器接到了广播， 它将回应 ypbind， 后者则记录服务器的地址。 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 将使用第一个响应的地址。 从这一时刻开始， 客户机会把所有的 NIS 请求直接发给那个服务器。 ypbind 偶尔会 "ping" 服务器以确认其仍然在正常运行。 如果在合理的时间内没有得到响应， 则 ypbind 会把域标记为未绑定， 并再次发起广播， 以期找到另一台服务器。

30.4.4.3.1. 设置 NIS 客户机

配置一台 FreeBSD 机器作为 NIS 客户机是非常简单的。

1. 编辑 /etc/rc.conf 文件， 并在其中加上下面几行， 以设置 NIS 域名， 并在网络启动时启动 ypbind：

```
nisdomainname="test-domain"
nis_client_enable="YES"
```

2. 要从 NIS 服务器导入所有的口令项， 需要从您的 /etc/master.passwd 文件中删除所有用户， 并使用 vipw 在这个文件的最后一行加入：

```
+::::::::::
```

这一行将让 NFS 服务器的口令映射中的帐号能够登录。 也有很多修改这一行来配置 NIS 客户机的办法。 请参见稍后的 netgroups 小节 以了解进一步的情况。 要了解更多信息， 可以参阅 O’ Reilly 的 Managing NFS and NIS 这本书。

需要至少保留一个本地帐号 (也就是不通过 NIS 导入) 在您的 /etc/master.passwd 文件中， 而这个帐号应该是 wheel 组的成员。 如果 NIS 发生不测， 这个帐号可以用来远程登录， 成为 root， 并修正问题。

3. 要从 NIS 服务器上导入组信息， 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执行下列命令：

```
# /etc/netstart
# /etc/rc.d/ypbind start
```

完成这些步骤之后， 就应该可以通过运行 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， 任何远程用户都可以发起一个 RPC 到 ypserv(8) 并获得您的 NIS 映射的内容，

如果远程用户了解您的域名的话。 要避免这类未经授权的访问， ypserv(8) 支持一个称为 "securenets" 的特性， 用以将访问限制在一组特定的机器上。 在启动过程中， ypserv(8) 会尝试从 /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项，
每一项中包含了一个网络标识和子网掩码， 中间用空格分开。 以 "#" 开头的行会被认为是注释。 示范的 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规则的地址的请求， 则它会正常处理请求。 反之， 则请求将被忽略， 并记录一条警告信息。 如果 /var/yp/securenets 文件不存在， 则 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， 管理员就能够使用 TCP Wrapper 的配置文件来代替 /var/yp/securenets 完成访问控制。

尽管这两种访问控制机制都能够提供某种程度的安全， 但是， 和特权端口检查一样， 它们无法避免 "IP 伪造" 攻击。 您的防火墙应该阻止所有与 NIS 有关的访问。

使用 /var/yp/securenets 的服务器， 可能会无法为某些使用陈旧的 TCP/IP 实现的 NIS 客户机服务。 这些实现可能会在广播时， 将主机位都设置为 0，
或在计算广播地址时忽略子网掩码。 尽管这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。

在使用陈旧的 TCP/IP 实现的系统上， 使用 /var/yp/securenets 是一个非常糟糕的做法，
因为这将导致您的网络上的 NIS 丧失大部分功能。

使用 TCP Wrapper 软件包， 会导致您的 NIS 服务器的响应延迟增加。 而增加的延迟，
则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。
如果您的某个客户机因此而产生一些异常， 则应将这些客户机变为 NIS 从服务器，
并强制其绑定自己。

30.4.6. 不允许某些用户登录

在我们的实验室中， basie 这台机器， 是一台教员专用的工作站。 我们不希望将这台机器拿出 NIS 域，
而主 NIS 服务器上的 passwd 文件， 则同时包含了教员和学生的帐号。 这时应该怎么做?

有一种办法来禁止特定的用户登录机器， 即使他们身处 NIS 数据库之中。 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 这样的项， 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 会对您在 /etc/master.passwd 文件上所作的修改进行合法性检查， 并在编辑结束时重新构建口令数据库。 例如，
如果希望禁止用户 bill 登录 basie， 我们应该：

```
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，在您需要为非常少的用户和/或机器进行特殊的规则配置时还算凑合。
在更大的网络上，您 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每一台机器的配置，因而丢掉了 NIS 最重要的优越性：集中式 管理。

NIS 开发人员为这个问题提供的解决方案，被称作 netgroups。它们的作用和语义，基本上可以等同于 UNIX® 文件系统上使用的组。主要的区别是它们没有数字化的 ID，以及可以在 netgroup 中同时包含用户和其他 netgroup。

Netgroups 被设计用来处理大的、复杂的包含数百用户和机器的网络。一方面，
在您不得不处理这类情形时，这是一个很有用的东西。而另一方面，
它的复杂性又使得通过非常简单的例子很难解释 netgroup 到底是什么。
这一节的其余部分的例子将展示这个问题。

假设您在实验室中成功地部署 NIS 引起了上司的兴趣。您接下来的任务是将 NIS 域扩展，
以覆盖校园中的一些其他的机器。下面两个表格中包括了新用户和新机器，及其简要说明。

用户名	说明
alpha, beta	IT 部门的普通雇员
charlie, delta	IT 部门的学徒
echo, foxtrott, golf, …	普通雇员

用户名	说明
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	一台不包含关键数据的旧机器。 即使是实习生， 也允许登录它。

如果您尝试通过一个一个地阻止用户来实现这些限制， 就需要在每一个系统的 passwd 文件中，
为每一个不允许登录该系统的用户添加对应的 -user 行。 如果忘记了任何一个， 就可能会造成问题。
在进行初始配置时， 正确地配置也许不是什么问题， 但随着日复一日地添加新用户， 总有一天
您会忘记为新用户添加某个行。 毕竟， Murphy 是一个乐观的人。

使用 netgroups 来处理这一状况可以带来许多好处。 不需要单独地处理每一个用户；
您可以赋予用户一个或多个 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 "对每一个用户和机器执行 ……"。 如果您的 NIS 配置经过了谨慎的规划， 就只需要修改一个中央的配置文件，
就能够允许或禁止访问某台机器的权限了。

第一步是初始化 NIS 映射 netgroup。 FreeBSD 的 ypinit(8) 默认情况下并不创建这个映射， 但它的 NIS 实现能够在创建这个映射之后立即对其提供支持。 要创建空映射， 简单地输入

```
ellington# vi /var/yp/netgroup
```

并开始增加内容。 在我们的例子中， 至少需要四个 nergruop： 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。 每一个括号中的组中， 都有一些用户帐号。
组中的三个字段是：

1. 在哪些机器上能够使用这些项。 如果不指定主机名， 则项在所有机器上都有效。 如果指定了主机， 则很容易造成混淆。
2. 属于这个 netgroup 的帐号。
3. 帐号的 NIS 域。 您可以从其他 NIS 域中把帐号导入到您的 netgroup 中， 如果您管理多个 NIS 域的话。

每一个字段都可以包括通配符。 参见 netgroup(5) 了解更多细节。

Netgroup 的名字一般来说不应超过 8 个字符， 特别是当您的 NIS 域中有机器打算运行其它操作系统的时候。 名字是区分大小写的； 使用大写字母作为 netgroup 的名字， 能够让您更容易地区分用户、 机器和 netgroup 的名字。

某些 NIS 客户程序 (FreeBSD 以外的那些) 可能无法处理含有大量项的 netgroup。 例如，
某些早期版本的 SunOS™ 会在 netgroup 中包含多于 15 个 项 时出现问题。
要绕过这个问题， 可以创建多个 子netgroup，每一个中包含少于 15 个用户，
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
```

如果需要超过 225 个用户， 可以继续重复上面的过程。

激活并分发新的 NIS 映射非常简单：

```
ellington# cd /var/yp
ellington# make
```

这个操作会生成三个 NIS 映射， 即 netgroup、 netgroup.byhost 和 netgroup.byuser。 用 ypcat(1)
可以检查这些 NIS 映射是否可用了：

```
ellington% ypcat -k netgroup
ellington% ypcat -k netgroup.byhost
ellington% ypcat -k netgroup.byuser
```

第一个命令的输出， 应该与 /var/yp/netgroup 的内容相近。 第二个命令， 如果没有指定本机专有的
netgroup， 则应该没有输出。 第三个命令， 则用于显示某个用户对应的 netgroup 列表。

客户机的设置也很简单。 要配置服务器 war， 只需进入 vipw(8) 并把

```
+::::::::::
```

改为

```
+@IT_EMP::::::::::
```

现在， 只有 netgroup IT_EMP 中定义的用户会被导入到 war 的口令数据库中，
因此只有这些用户能够登录。

不过， 这个限制也会作用于 shell 的 ~， 以及所有在用户名和数字用户 ID 之间实施转换的函数的功能。
换言之， cd ~user 将不会正常工作， 而 ls -l 也将显示数字的 ID 而不是用户名， 并且 find . -user joe -print
将失败， 并给出 No such user 的错误信息。 要修正这个问题， 您需要导入所有的用户项， 而
不允许他们登录服务器。

这可以通过在 /etc/master.passwd 加入另一行来完成。 这行的内容是：

+::::::::::/sbin/nologin， 意思是 "导入所有的项， 但导入项的 shell 则替换为 /sbin/nologin"。 通过在
/etc/master.passwd 中增加默认值， 可以替换掉 passwd 中的任意字段。

务必确认 ::::::::::/sbin/nologin` 这一行出现在 `@IT_EMP:::::::::: 之后。 否则， 所有从

> NIS 导入的用户帐号将以 /sbin/nologin 作为登录 shell。

完成上面的修改之后， 在 IT 部门有了新员工时， 只需修改一个 NIS 映射就足够了。
您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 +:::::::::: 改为：

```
+@IT_EMP::::::::::
+@IT_APP::::::::::
+::::::::::/sbin/nologin
```

相关的用于普通工作站的配置则应是：

```
+@IT_EMP::::::::::
+@USERS::::::::::
+::::::::::/sbin/nologin
```

一切平安无事， 直到数周后， 有一天策略发生了变化： IT 部门也开始招收实习生了。 IT 实习生允许使用普通的终端， 以及不太重要的服务器； 而 IT 学徒， 则可以登录主服务器。 您增加了新的 netgroup IT_INTERN， 以及新的 IT 实习生到这个 netgroup 并开始修改每一台机器上的配置……
老话说得好："牵一发， 动全身"。

NIS 通过 netgroup 来建立 netgroup 的能力， 正可以避免这样的情形。 一种可能的方法是建立基于角色的 netgroup。 例如， 您可以创建称为 BIGSRV 的 netgroup， 用于定义最重要的服务器上的登录限制， 以及另一个成为 SMALLSRV 的 netgroup， 用以定义次重要的服务器， 以及第三个， 用于普通工作站的 netgroup USERBOX。 这三个 netgroup 中的每一个， 都包含了允许登录到这些机器上的所有 netgroup。
您的 NIS 映射中的新项如下所示：

```
BIGSRV   IT_EMP  IT_APP
SMALLSRV  IT_EMP  IT_APP  ITINTERN
USERBOX   IT_EMP  ITINTERN USERS
```

这种定义登录限制的方法， 在您能够将机器分组并加以限制的时候可以工作的相当好。 不幸的是， 这是种例外， 而非常规情况。 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， 是处理上述策略改动的另一种可能的方法。 此时， 每台机器的 /etc/master.passwd 中， 都包含两个 "+" 开头的行。 第一个用于添加允许登录的 netgroup 帐号， 而第二个则用于增加其它帐号， 并把 shell 设置为 /sbin/nologin。 使用 "全大写" 的机器名作为 netgroup 名是个好主意。 换言之， 这些行应该类似于：

```
+@BOXNAME::::::::::
+::::::::::/sbin/nologin
```

一旦在所有机器上都完成了这样的修改， 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 映射中进行。 这里是一个例子， 其中展示了在这一应用情景中所需要的 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]
```

如果您正使用某种数据库来管理帐号， 应该可以使用您的数据库的报告工具来创建映射的第一部分。 这样，新用户就自动地可以访问这些机器了。

最后的提醒： 使用基于机器的 netgroup 并不总是适用的。
如果正在为学生实验室部署数十台甚至上百台同样的机器， 您应该使用基于角色的 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. 需要牢记的事项

这里是一些其它在使用 NIS 环境时需要注意的地方。

- 每次需要在实验室中增加新用户时， 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。 如果您忘记了这样做， 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，如果要在实验室增加新用户 jsmith， 我们需要：

```
# pw useradd jsmith
# cd /var/yp
# make test-domain
```

也可以运行 adduser jsmith 而不是 pw useradd jsmith.

- 将管理用的帐号排除在 NIS 映射之外。 一般来说，
  您不希望这些管理帐号和口令被扩散到那些包含不应使用它们的用户的机器上。
- 确保 NIS 主和从服务器的安全， 并尽可能减少其停机时间。 如果有人攻入或简单地关闭这些机器，
  则整个实验室的任也就无法登录了。

  这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，
  您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 提供了某些为 NIS v1 客户提供服务的支持能力。 FreeBSD 的 NIS 实现， 只使用 NIS v2 协议， 但其它实现可能会包含 v1 协议， 以提供对旧系统的向下兼容能力。 随这些系统提供的 ypbind 服务将首先尝试绑定 NIS v1 服务器， 即使它们并不真的需要它 (有些甚至可能会一直广播搜索请求，
即使已经从某台 v2 服务器得到了回应也是如此)。 注意， 尽管支持一般的客户机调用， 这个版本的 ypserv 并不能处理 v1 的映射传送请求； 因而， 它就不能与较早的支持 v1 协议的 NIS 服务器配合使用，
无论是作为主服务器还是从服务器。 幸运的是， 现今应该已经没有仍然在用的这样的服务器了。

30.4.10. 同时作为 NIS 客户机的 NIS 服务器

在多服务器域的环境中， 如果服务器同时作为 NIS 客户， 在运行 ypserv 时要特别小心。 一般来说，
强制服务器绑定自己要比允许它们广播绑定请求要好， 因为这种情况下它们可能会相互绑定。
某些怪异的故障， 很可能是由于某一台服务器停机， 而其它服务器都依赖其服务所导致的。 最终，
所有的客户机都会超时并绑定到其它服务器， 但这个延迟可能会相当可观，
而且恢复之后仍然存在再次发生此类问题的隐患。

您可以强制一台机器绑定到特定的服务器， 这是通过 ypbind 的 -S 参数来完成的。 如果不希望每次启动 NIS 服务器时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"
```

参见 ypbind(8) 以了解更多情况。

30.4.11. 口令格式

在实现 NIS 时， 口令格式的兼容性问题是一种最为常见的问题。 假如您的 NIS 服务器使用 DES 加密口令，
则它只能支持使用 DES 的客户机。 例如， 如果您的网络上有 Solaris™ NIS 客户机， 则几乎肯定需要使用 DES 加密口令。

要检查您的服务器和客户机使用的口令格式， 需要查看 /etc/login.conf。 如果主机被配置为使用 DES 加密的口令， 则 default class 将包含类似这样的项：

```
default:\
  :passwd_format=des:\
  :copyright=/etc/COPYRIGHT:\
  [Further entries elided]
```

其他一些可能的 passwd_format 包括 blf 和 md5 (分别对应于 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库， 这是通过以 root 身份运行下面的程序来完成的：

```
# cap_mkdb /etc/login.conf
```

已经在 /etc/master.passwd 中的口令的格式不会被更新，
直到用户在登录性能数据库重建 之后 首次修改口令为止。

接下来， 为了确保所有的口令都按照您选择的格式加密了， 还需要检查 /etc/auth.conf 中 crypt_default 给出的优先选择的口令格式。 要完成此工作， 将您选择的格式放到列表的第一项。 例如， 当使用 DES 加密的口令时， 对应项应为：

```
crypt_default = des blf md5
```

在每一台基于 FreeBSD 的 NIS 服务器和客户机上完成上述工作之后，
就可以肯定您的网络上它们都在使用同样的口令格式了。 如果在 NIS 客户机上做身份验证时发生问题，
这也是第一个可能出现问题的地方。 注意： 如果您希望在混合的网络上部署 NIS 服务器，
可能就需要在所有系统上都使用 DES， 因为这是所有系统都能够支持的最低限度的公共标准。

30.5. 网络自动配置 (DHCP)

30.5.1. 什么是 DHCP?

DHCP， 动态主机配置协议， 是一种让系统得以连接到网络上， 并获取所需要的配置参数手段。 FreeBSD 使用来自 OpenBSD 3.7 的 OpenBSD dhclient。 这里提供的所有关于 dhclient 的信息， 都是以 ISC 或 OpenBSD DHCP 客户端程序为准的。 DHCP 服务器是 ISC 软件包的一部分。

30.5.2. 这一节都介绍哪些内容

这一节描述了 ISC 和 DHCP 系统中的客户端， 以及和 ISC DHCP 系统中的服务器端的组件。 客户端程序， dhclient， 是随 FreeBSD 作为它的一部分提供的； 而服务器部分， 则可以通过 net/isc-dhcp31-server port 得到。 dhclient(8)、 dhcp-options(5)、 以及 dhclient.conf(5) 联机手册，
加上下面所介绍的参考文献， 都是非常有用的资源。

30.5.3. 它如何工作

当 DHCP 客户程序， dhclient 在客户机上运行时， 它会开始广播请求配置信息的消息。 默认情况下，
这些请求是在 UDP 端口 68 上。 服务器通过 UDP 67 给出响应， 向客户机提供一个 IP 地址，
以及其他有关的配置参数， 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" 的形式给出， 并且只在一段特定的时间内有效 (这是由 DHCP 服务器的维护者配置的)。 这样，
那些已经断开网络的客户机使用的陈旧的 IP 地址就能被自动地回收了。

DHCP 客户程序可以从服务器端获取大量的信息。 关于能获得的信息的详细列表， 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， 这使得您不再需要去了解那些已经运行了 DHCP
服务器的网络的具体配置参数。

sysinstall 能够支持 DHCP。 在 sysinstall 中配置网络接口时， 它询问的第二个问题便是： "Do you want to try DHCP configuration of the interface? (您是否希望在此接口上尝试 DHCP 配置?)"。
如果做肯定的回答， 则将运行 dhclient， 一旦成功， 则将自动地填写网络配置信息。

要在系统启动时使用 DHCP， 您必须做两件事：

- 您的内核中， 必须包含 bpf 设备。 如果需要这样做， 需要将 device bpf 添加到内核的编译配置文件中， 并重新编译内核。 要了解关于编译内核的进一步信息， 请参见 配置FreeBSD的内核。

  bpf 设备已经是 FreeBSD 发行版中默认的 GENERIC 内核的一部分了， 因此如果您没有对内核进行定制， 则不用创建一份新的内核配置文件， DHCP 就能工作了。

  对于那些安全意识很强的人来说， 您应该知道 bpf 也是包侦听工具能够正确工作的条件之一 (当然， 它们还需要以 root 身份运行才行)。 bpf是 使用 DHCP 所必须的， 但如果您对安全非常敏感， 则很可能会有理由不把 bpf 加入到您的内核配置中， 直到您真的需要使用 DHCP 为止。

- 编辑您的 /etc/rc.conf 并加入下面的设置：

  ```
  ifconfig_fxp0="DHCP"
  ```

  务必将 fxp0 替换为您希望自动配置的网络接口的名字， 您可以在 设置网卡 找到更进一步的介绍。

  如果您希望使用另一位置的 dhclient， 或者需要给 dhclient 传递其他参数， 还可以添加下面的配置 (根据需要进行修改)：

  ```
  dhclient_program="/sbin/dhclient"
  dhclient_flags=""
  ```

DHCP 服务器， dhcpd， 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。 一般说来， 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) 联机手册中进行了进一步的阐述。

- /sbin/dhclient

  dhclient 是一个静态连编的， 它被安装到 /sbin 中。 dhclient(8) 联机手册给出了关于 dhclient 的进一步细节。

- /sbin/dhclient-script

  dhclient-script 是一个 FreeBSD 专用的 DHCP 客户端配置脚本。 在 dhclient-script(8) 中对它进行了描述， 但一般来说， 用户不需要对其进行任何修改， 就能够让一切正常运转了。

- /var/db/dhclient.leases

  DHCP 客户程序会维护一个数据库来保存有效的 lease， 它们被以日志的形式保存到这个文件中。 dhclient.leases(5) 给出了更为细致的介绍。

30.5.6. 进阶读物

DHCP 协议的完整描述是 RFC 2131。 关于它的其他信息资源的站点 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. 这一章包含哪些内容

这一章提供了关于如何在 FreeBSD 系统上使用 ISC (Internet 系统协会) 的 DHCP 实现套件来架设 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， 因此您需要安装 net/isc-dhcp31-server port 才能提供这个服务。 请参见 安装应用程序. Packages 和 Ports 以了解关于如何使用 Ports Collection 的进一步详情。

30.5.7.2. 安装 DHCP 服务器

为了在您的 FreeBSD 系统上进行配置以便作为 DHCP 服务器来使用， 需要把 bpf(4) 设备编译进内核。
要完成这项工作， 需要将 device bpf 加入到您的内核配置文件中， 并重新联编内核。
要得到关于如何联编内核的进一步信息， 请参见 配置FreeBSD的内核。

bpf 设备是 FreeBSD 所附带的 GENERIC 内核中已经联入的组件， 因此您并不需要为了让 DHCP 正常工作而特别地定制内核。

如果您有较强的安全意识， 应该注意 bpf 同时也是让听包程序能够正确工作的设备 (尽管这类程序仍然需要以特权用户身份运行)。 bpf是 使用 DHCP 所必需的， 但如果您对安全非常敏感， 您可能会不希望将 bpf 放进内核， 直到您真的认为 DHCP 是必需的为止。

接下来要做的是编辑示范的 dhcpd.conf， 它由 net/isc-dhcp31-server port 安装。 默认情况下， 它的名字应该是 /usr/local/etc/dhcpd.conf.sample， 在开始修改之前， 您需要把它复制为 /usr/local/etc/dhcpd.conf。

30.5.7.3. 配置 DHCP 服务器

dhcpd.conf 包含了一系列关于子网和主机的定义， 下面的例子可以帮助您理解它：

```
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 这个选项指定了提供给客户机作为默认搜索域的域名。 请参考 resolv.conf(5) 以了解关于这一概念的详情。

② 这个选项用于指定一组客户机使用的 DNS 服务器， 它们之间以逗号分隔。

③ 提供给客户机的子网掩码。

④ 客户机可以请求租约的有效期， 而如果没有， 则服务器将指定一个租约有效期， 也就是这个值 (单位是秒)。

⑤ 这是服务器允许租出地址的最大时长。 如果客户机请求了更长的租期， 则它将得到一个地址， 但其租期仅限于 max-lease-time 秒。

⑥ 这个选项用于指定 DHCP 服务器在一个地址被接受或释放时是否应对应尝试更新 DNS。 在 ISC 实现中， 这一选项是 必须指定的。

⑦ 指定地址池中可以用来分配给客户机的 IP 地址范围。 在这个范围之间， 以及其边界的 IP 地址将分配给客户机。

⑧ 定义客户机的默认网关。

⑨ 主机的硬件 MAC 地址 (这样 DHCP 服务器就能够在接到请求时知道请求的主机身份)。

⑩ 指定总是得到同一 IP 地址的主机。 请注意在此处使用主机名是对的， 因为 DHCP 服务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， 应在 /etc/rc.conf 中启用 DHCP 服务器， 也就是增加：

```
dhcpd_enable="YES"
dhcpd_ifaces="dc0"
```

此处的 dc0 接口名应改为 DHCP 服务器需要监听 DHCP 客户端请求的接口 (如果有多个， 则用空格分开)。

接下来， 可以用下面的命令来启动服务：

```
# /usr/local/etc/rc.d/isc-dhcpd start
```

如果未来您需要修改服务器的配置， 请务必牢记发送 SIGHUP 信号给 dhcpd 并 不会 导致配置文件的重新加载， 而这在其他服务程序中则是比较普遍的约定。 您需要发送 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 是静态连接的， 并安装到 /usr/local/sbin 中。 随 port 安装的 dhcpd(8) 联机手册提供了关于 dhcpd 更为详尽的信息。

- /usr/local/etc/dhcpd.conf

  dhcpd 需要配置文件， 即 /usr/local/etc/dhcpd.conf 才能够向客户机提供服务。 这个文件需要包括应提供给客户机的所有信息， 以及关于服务器运行的其他信息。 此配置文件的详细描述可以在随 port 安装的 dhcpd.conf(5) 联机手册上找到。

- /var/db/dhcpd.leases

  DHCP 服务器会维护一个它签发的租用地址数据库， 并保存在这个文件中， 这个文件是以日志的形式保存的。 随 port 安装的 dhcpd.leases(5) 联机手册提供了更详细的描述。

- /usr/local/sbin/dhcrelay

  dhcrelay 在更为复杂的环境中， 可以用来支持使用 DHCP 服务器转发请求给另一个独立网络上的 DHCP 服务器。 如果您需要这个功能， 需要安装 net/isc-dhcp31-relay port。 dhcrelay(8) 联机手册提供了更为详尽的介绍。

30.6. 域名系统 (DNS)

30.6.1. 纵览

FreeBSD 在默认情况下使用一个版本的 BIND (Berkeley Internet Name Domain)， 这是目前最为流行的 DNS 协议实现。 DNS 是一种协议， 可以通过它将域名同 IP 地址相互对应。 例如， 查询 www.FreeBSD.org 将得到 FreeBSD Project 的 web 服务器的 IP 地址， 而查询 ftp.FreeBSD.org 则将得到响应的 FTP 机器的 IP 地址。 类似地， 也可以做相反的事情。 查询 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 服务软件。
我们内建于系统中的版本提供了增强的安全特性、 新的文件目录结构， 以及自动的 chroot(8) 配置。

在 Internet 上的 DNS 是通过一套较为复杂的权威根域名系统， 顶级域名 (TLD)， 以及一系列小规模的， 提供少量域名解析服务并对域名信息进行缓存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

要理解这份文档， 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见叫法。
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进程
反向 DNS	将 IP 地址映射为主机名
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， 或者由同一机构管理的 DNS 的一部分。

域的例子：

- . 在本文档中通常指代根域。
- 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。

如您所见， 域名中越细节的部分会越靠左出现。 例如， example.org. 就比 org. 范围更小， 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类似： /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， 并对请求给出权威应答。
- 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 地址块需要反向 DNS 项 (IP 到主机名)。

- 备份服务器， 或常说的从 (slave) 服务器， 会在主服务器出现问题或无法访问时来应答查询请求。

下列情况需要有缓存域名服务器：

- 本地的 DNS 服务器能够缓存， 并比直接向外界的域名服务器请求更快地得到应答。

当有人查询 www.FreeBSD.org 时，解析器通常会向上级 ISP 的域名服务器发出请求， 并获得回应。
如果有本地的缓存 DNS 服务器， 查询只有在第一次被缓存 DNS 服务器发到外部世界。
其他的查询不会发向局域网外， 因为它们已经有在本地的缓存了。

30.6.4. DNS 如何运作

在 FreeBSD 中， BIND 服务程序被称为 named。

文件	描述
named(8)	BIND 服务程序
rndc(8)	域名服务控制程序
/etc/namedb	BIND 存放域名信息的位置。
/etc/namedb/named.conf	域名服务配置文件

随在服务器上配置的域的性质不同， 域的定义文件一般会存放到 /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响应查询时所需要的 DNS 信息。

30.6.5. 启动 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)。 如果希望启动这一配置， 可以使用下面的命令：

```
# /etc/rc.d/named onestart
```

如果希望 named 服务在每次启动的时候都能够启动， 需要在 /etc/rc.conf 中加入：

```
named_enable="YES"
```

当然， 除了这份文档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， 如果您需要了解 FreeBSD 中用于启动 named 的那些选项的话， 则可以查看 /etc/defaults/rc.conf
中的 named_* 参数， 并参考 rc.conf(5) 联机手册。 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， 在使用前应根据需要进行修改，
除非您只打算让它完成简单的域名解析服务。 这个目录同时也是您进行绝大多数配置的地方。

30.6.6.1. /etc/namedb/named.conf

```
// $FreeBSD$
//
// Refer to the named.conf(5) and named(8) man pages, and the documentation
// in /usr/shared/doc/bind9 for more details.
```

```
//
// If you are going to set up an authoritative server, make sure you
// understand the hairy details of how DNS works.  Even with
// simple mistakes, you can break connectivity for affected parties,
// or cause huge amounts of useless Internet traffic.

options {
   // Relative to the chroot directory, if any
   directory   "/etc/namedb";
   pid-file    "/var/run/named/pid";
   dump-file   "/var/dump/named_dump.db";
   statistics-file "/var/stats/named.stats";

// If named is being used only as a local resolver, this is a safe default.
// For named to be accessible to the network, comment this option, specify
// the proper IP address, or delete this option.
   listen-on   { 127.0.0.1; };

// If you have IPv6 enabled on this system, uncomment this option for
// use as a local resolver.  To give access to the network, specify
// an IPv6 address, or the keyword "any".
//  listen-on-v6    { ::1; };

// These zones are already covered by the empty zones listed below.
// If you remove the related empty zones below, comment these lines out.
   disable-empty-zone "255.255.255.255.IN-ADDR.ARPA";
   disable-empty-zone
"0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.IP6.ARPA";
   disable-empty-zone
"1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.IP6.ARPA";

// If you've got a DNS server around at your upstream provider, enter
// its IP address here, and enable the line below.  This will make you
// benefit from its cache, thus reduce overall DNS traffic in the Internet.
/*
   forwarders {
      127.0.0.1;
   };
*/

// If the 'forwarders' clause is not empty the default is to 'forward first'
// which will fall back to sending a query from your local server if the name
// servers in 'forwarders' do not have the answer.  Alternatively you can
```

```
// force your name server to never initiate queries of its own by enabling the
// following line:
//  forward only;

// If you wish to have forwarding configured automatically based on
// the entries in /etc/resolv.conf, uncomment the following line and
// set named_auto_forward=yes in /etc/rc.conf.  You can also enable
// named_auto_forward_only (the effect of which is described above).
//  include "/etc/namedb/auto_forward.conf";
```

正如注释所言， 如果希望从上级缓存中受益， 可以在此处启用 forwarders。 正常情况下，
域名服务器会逐级地查询 Internet 来找到特定的域名服务器， 直到得到答案为止。
这个选项将让它首先查询上级域名服务器 (或另外提供的域名服务器)， 从而从它们的缓存中得到结果。
如果上级域名服务器是一个繁忙的高速域名服务器， 则启用它将有助于改善服务品质。

127.0.0.1不会 正常工作。 一定要把地址改为您上级服务器的 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 "sf87HJqjkqh8ac87a02lla==";
};
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， 还给出了从域、转发域和反解析域的例子。

如果新增了域， 就必需在 named.conf 中加入对应的项目。

例如， 用于 example.org 的域文件的描述类似下面这样：

```
zone "example.org" {
  type master;
  file "master/example.org";
};
```

如 type 语句所标示的那样， 这是一个主域， 其信息保存在 /etc/namedb/master/example.org 中， 如 file 语句所示。

```
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传递过来， 并保存到对应的文件中。
当主域服务器发生问题或不可达时， 从域名服务器就有一份可用的域名信息， 从而能够对外提供服务。

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.
```

请注意以 "." 结尾的主机名是全称主机名， 而结尾没有 "." 的则是相对于原点的主机名。 例如， ns1 将被转换为 ns1.example.org.

域信息文件的格式如下:

```
记录名      IN 记录类型   值
```

最常用的 DNS 记录：

SOA
: 域权威开始

NS
: 权威域名服务器

A
: 主机地址

CNAME
: 别名对应的正规名称

MX
: 邮件传递服务器

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， 同时也是这个域信息文件的原点。

ns1.example.org.
: 该域的主/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 "@" 需要换掉 (admin@example.org 对应 admin.example.org)

2006051501
: 文件的序号。 每次修改域文件时都必须增加这个数字。 现今， 许多管理员会考虑使用 yyyymmddrr 这样的格式来表示序号。 2006051501 通常表示上次修改于 05/15/2006， 而后面的 01 则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.
```

这是一个 NS 项。 每个准备提供权威应答的服务器都必须有一个对应项。

```
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 记录指明了机器名。 正如在前面所看到的， ns1.example.org 将解析为 192.168.1.2。

```
          IN    A     192.168.1.1
```

这一行把当前原点 example.org 指定为使用 IP 地址 192.168.1.1。

```
www       IN CNAME     @
```

正规名 (CNAME) 记录通常用于为某台机器指定别名。 在这个例子中， 将 www 指定成了 "主"
机器的一个别名， 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记录并存。

```
          IN MX  10    mail.example.org.
```

MX 记录表示哪个邮件服务器负责接收发到这个域的邮件。 mail.example.org 是邮件服务器的主机名， 而
10 则是它的优先级。

可以有多台邮件服务器， 其优先级分别是 10、 20 等等。 尝试向 example.org 投递邮件的服务器，
会首先尝试优先级最高的 MX (优先级数值最小的记录)、 接着尝试次高的，
并重复这一过程直到邮件递达为止。

in-addr.arpa 域名信息文件 (反向 DNS)， 采用的格式是同样的， 只是 PTR 项代替了 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.
```

这个文件给出了上述假想域中 IP 地址到域名的映射关系。

需要说明的是， 在 PTR 记录右侧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种主要承担解析递归查询角色的域名服务器。 它简单地自行进行查询，
并将查询结果记住以备后续使用。

30.6.8. 安全

尽管 BIND 是最为常用的 DNS 实现， 但它总是有一些安全问题。
时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 会自动将 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您规避潜在的针对 DNS 服务的攻击。

阅读 CERT 的安全公告， 并订阅 the FreeBSD 安全问题通知邮件列表 是一个有助于帮助您了解最新 Internet 及 FreeBSD 安全问题的好习惯。

如果发现了问题， 确保源代码是最新的， 并重新联编一份 named 有可能会有所帮助。

30.6.9. 进一步阅读

BIND/named 联机手册： 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 - 实现及其标准

30.7. Apache HTTP 服务器

30.7.1. 纵览

FreeBSD 被用于运行许多全球最为繁忙的 web 站点。 大多数 Internet 上的 web 服务器， 都使用 Apache HTTP 服务器。 Apache 软件包可以在您的 FreeBSD 安装盘上找到。 如果没有在首次安装时附带安装 Apache， 则可以通过 www/apache13 或 www/apache22 port 来安装。

一旦成功地安装了 Apache， 就必须对其进行配置。

这一节介绍了 1.3.X 版本的 Apache HTTP 服务器 的配置， 因为它是随 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， 请参见 http://httpd.apache.org/。

30.7.2. 配置

主要的 Apache HTTP Server 配置文件， 在 FreeBSD 上会安装为 /usr/local/etc/apache/httpd.conf。
这是一个典型的 UNIX® 文本配置文件， 它使用 # 作为注释符。
关于全部配置选项的详尽介绍超出了本书的范围， 这里将只介绍最常被修改的那些。

ServerRoot "/usr/local"
: 这指定了 Apache 安装的顶级目录。 执行文件被放到服务器根目录 (server root) 的 bin 和 sbin 子目录中， 而配置文件则位于 etc/apache。

ServerAdmin you@your.address
: 这个地址是在服务器发生问题时应发送电子邮件的地址， 它会出现在服务器生成的页面上，

例如错误页面。

ServerName www.example.com
: ServerName 允许您配置发送回客户端的主机名， 如果您的服务器被用户以别的名字访问 (例如， 使用 www 而不是主机本身的真实名字)。

DocumentRoot "/usr/local/www/data"
: DocumentRoot： 这个目录是您的文档所在的目录。 默认情况下， 所有的请求都会从这个位置去获取， 但也可以通过符号连接和别名指定其它的位置。

在修改配置之前备份 Apache 的配置文件永远是一个好习惯。 一旦对初始配置满意了， 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 inetd 超级服务器来运行。
一般情况下会把它配置为一个独立的服务器， 以期在客户的 web 浏览器连入 HTTP 请求时，
能够获得更好的性能。 它提供了一个 shell 脚本来使启动、 停止和重新启动服务器变得尽可能地简单。
首次启动 Apache， 只需执行：

```
# /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
```

当由于某种原因修改了配置文件之后， 需要重启服务器：

```
# /usr/local/sbin/apachectl restart
```

要在重启 Apache 服务器时不中断当前的连接， 则应运行：

```
# /usr/local/sbin/apachectl graceful
```

更多的信息， 可以在 apachectl(8) 联机手册中找到。

要在系统启动时启动 Apache， 则应在 /etc/rc.conf 中加入：

```
apache_enable="YES"
```

或者对于Apache 2.2：

```
apache22_enable="YES"
```

如果您希望在系统引导时启动 Apache httpd 程序并指定其它一些选项， 则可以把下面的行加到 rc.conf：

```
apache_flags=""
```

现在 web 服务器就开始运行了， 您可以使用 web 浏览器打开 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机发来的 HTTP/1.1 头来辨别主机名。 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， 需要把类似下面的项加到您的 httpd.conf 中：

```
NameVirtualHost *
```

如果您的 web 服务器的名字是 www.domain.tld， 而您希望建立一个 www.someotherdomain.tld 的虚拟域， 则应在 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>
```

您需要把上面的地址和文档路径改为所使用的那些。

要了解关于虚拟主机的更多信息， 请参考官方的 Apache 文档， 这些文档可以在 http://httpd.apache.org/docs/vhosts/ 找到。

30.7.5. Apache 模块

有许多不同的 Apache 模块， 它们可以在基本的服务器基础上提供许多附加的功能。 FreeBSD 的 Ports Collection 为安装 Apache 和常用的附加模块提供了非常方便的方法。

30.7.5.1. mod_ssl

mod_ssl 这个模块使用 OpenSSL 库， 来提供通过 安全套接字层 (SSL v2/v3) 和 传输层安全 (TLS v1) 协议的强加密能力。 这个模块提供了从某一受信的证书签署机构申请签名证书所需的所有工具，
您可以藉此在 FreeBSD 上运行安全的 web 服务器。

如果您未曾安装 Apache， 也可以直接安装一份包含了 mod_ssl 的版本的 Apache 1.3.X， 其方法是通过 www/apache13-modssl port 来进行。 SSL 支持已经作为 Apache 2.X 的一部分提供， 您可以通过 www/apache22 port 来安装后者。

30.7.5.2. 语言绑定

Apache对于一些主要的脚本语言都有相应的模块。 这些模块使得完全使用某种脚本语言来写 Apache 模块成为可能。 他们通常也被嵌入到服务器作为一个常驻内存的解释器，
以避免启动一个外部解释器对于下一节将描述的动态网站所需时间和资源上的开销。

30.7.6. 动态网站

在过去的十年里，越来越多的企业为了增加收益和暴光率而转向了互联网。
这也同时增进了对于互动网页内容的需求。有些公司，比如 Microsoft®
推出了基于他们专有产品的解决方案，开源社区也做出了积极的回应。 比较时尚的选择包括 Django，Ruby on Rails， mod_perl, and mod_php.

30.7.6.1. Django

Django 是一个以 BSD 许可证发布的 framework， 能让开发者快速写出高性能高品质的 web 应用程序。
它提供给一个对象关系映射组件，数据类型可以被当 Python 中的对象，和一组丰富的动态数据库访问 API， 使开发者避免了写 SQL 语句。它同时还提供了可扩展的模板系统， 让应用程序的逻辑部分与 HTML 的表现层分离。

Django 依赖与 mod_python， Apache, 和一个可选的 SQL 数据库引擎。
在设置了一些恰当的标志后，FreeBSD 的 Port 系统将会帮助你安装这些必需的依赖库。

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 和那些依赖的软件之后， 你需要创建一个 Django 项目的目录，然后配置 Apache，当有对于你网站上应用程序的某些指定的 URL 时调用内嵌的 Python 解释器。

例 39. Django/mod_python 有关 Apache 部分的配置

你需要在 Apache 的配置文件 httpd.conf 加入以下这几行， 把对某些 URL 的请求传给你的 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， 提供了一个全面的开发框架，能帮助 web 开发者工作更有成效和快速写出强大的应用。 它能非常容易的从 posts 系统安装。

```
# cd /usr/ports/www/rubygem-rails; make all install clean
```

30.7.6.3. mod_perl

Apache/Perl 集成计划， 将 Perl 程序设计语言的强大功能， 与 Apache HTTP 服务器紧密地结合到了一起。 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， 消除了由于启动外部的解释器为 Perl 脚本的启动所造成的性能损失。

mod_perl 通过多种方式提供。 要使用 mod_perl， 应该注意 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， 而以静态方式联编的版本， 则可以通过 www/apache13-modperl 来安装。 mod_perl 2.0 则可以通过 www/mod_perl2 安装。

30.7.6.4. mod_php

PHP， 也称为 "PHP: Hypertext Preprocessor"， 是一种特别适合于 Web 开发的通用脚本语言。 它能够很容易地嵌入到 HTML 之中， 其语法接近于 C、 Java™， 以及 Perl， 以期让 web 开发人员的一迅速撰写动态生成的页面。

要获得用于 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， 系统会自动显示可用的一系列 OPTIONS (配置选项)。 如果您没有看到菜单， 例如由于过去曾经安装过 lang/php5 port 等等， 可以用下面的命令再次显示配置菜单， 在 port 的目录中执行：

```
# make config
```

在配置选项对话框中， 选中 APACHE 这一项， 就可以联编出用于与 Apache web 服务器配合使用的可动态加载的 mod_php5 模块了。

由于各式各样的原因 (例如， 出于已经部署的 web 应用的兼容性考虑)， 许多网站仍在使用 PHP4。 如果您需要 mod_php4 而不是 mod_php5， 请使用 lang/php4 port。 lang/php4 port 也支持许多 lang/php5 port 提供的配置和编译时选项。

前面我们已经成功地安装并配置了用于支持动态 PHP 应用所需的模块。 请检查并确认您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>
```

这些工作完成之后， 还需要使用 apachectl 命令来完成一次 graceful restart 以便加载 PHP 模块：

```
# apachectl graceful
```

在未来您升级 PHP 时， make config 这步操作就不再是必需的了； 您所选择的 OPTIONS 会由 FreeBSD 的 Ports 框架自动保存。

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。

增加其他功能的支持非常简单，只需通过 lang/php5-extensions port 即可完成。这个 port 提供了一个菜单驱动的界面来帮助完成 PHP 扩展的安装。另外，也可以通过对应的 port 来单独安装扩展。

例如，要将对于 MySQL 数据库服务器的支持加入 PHP5，只需简单地安装 databases/php5-mysql。

安装完扩展之后，必须重新启动 Apache 服务器，来令其适应新的配置变更：

```
# apachectl graceful
```

30.8. 文件传输协议 (FTP)

30.8.1. 纵览

文件传输协议 (FTP) 为用户提供了一个简单的，与 FTP 服务器交换文件的方法。FreeBSD 系统中包含了 FTP 服务软件，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

最重要的配置步骤是决定允许哪些帐号访问 FTP 服务器。一般的 FreeBSD 系统包含了一系列系统帐号分别用于执行不同的服务程序，但未知的用户不应被允许登录并使用这些帐号。/etc/ftpusers 文件中，列出了不允许通过 FTP 访问的用户。默认情况下，这包含了前述的系统帐号，但也可以在这里加入其它不应通过 FTP 访问的用户。

您可能会希望限制通过 FTP 登录的某些用户，而不是完全阻止他们使用 FTP。这可以通过 /etc/ftpchroot 文件来完成。这一文件列出了希望对 FTP 访问进行限制的用户和组的表。而在 ftpchroot(5) 联机手册中，已经对此进行了详尽的介绍，故而不再赘述。

如果您想要在服务器上启用匿名的 FTP 访问，则必须建立一个名为 ftp 的 FreeBSD 用户。这样，用户就可以使用 ftp 或 anonymous 和任意的口令 (习惯上，应该是以那个用户的邮件地址作为口令) 来登录和访问您的 FTP 服务器。FTP 服务器将在匿名用户登录时调用 chroot(2)，以便将其访问限制在 ftp 用户的主目录中。

有两个文本文件可以用来指定显示在 FTP 客户程序中的欢迎文字。/etc/ftpwelcome 文件中的内容将在用户连接上之后，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。请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。这里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 重新加载 inetd 配置文件 所介绍的那样，修改这个文件之后，必须让 inetd 重新加载它，才能使新的设置生效。请参阅 设置 以获取更多有关如何在你系统上启用 inetd 的详细信息。

ftpd 也可以作为一个独立的服务启动。这样的话就需要在 /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启动，或者通过以 root 身份手动执行如下的命令启动：

```
# /etc/rc.d/ftpd start
```

现在可以通过输入下面的命令来登录您的 FTP 服务器了：

```
% ftp localhost
```

30.8.3. 维护

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
```

一定要小心对待在匿名 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应三思。 您可能发现自己的 FTP 站点成为了交易未经授权的商业软件的论坛， 或发生更糟糕的情况。 如果不需要匿名的 FTP 上传， 可以在文件上配置权限， 使得您能够在其它匿名用户能够下载这些文件之前复查它们。

30.9. 为 Microsoft® Windows® 客户机提供文件和打印服务 (Samba)

30.9.1. 纵览

Samba 是一个流行的开源软件包， 它提供了针对 Microsoft® Windows® 客户机的文件和打印服务。 这类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 软件包可以在您的 FreeBSD 安装盘上找到。 如果您没有在初次安装 FreeBSD 时安装 Samba， 则可以通过 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。这个文件必须复制为 /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 的运行时配置信息， 例如对于打印机的定义， 以及希望共享给 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 文件的简单方法。

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， 才能够使用 swat 来配置 Samba：

```
swat   stream  tcp    nowait/400    root   /usr/local/sbin/swat   swat
```

如 重新加载 inetd 配置文件 中所介绍的那样， 在修改了这个配置文件之后， 必须让 inetd 重新加载配置， 才能使其生效。

一旦在 inetd.conf 中启用了 swat， 就可以用浏览器访问 connect to http://localhost:901 了。 您将首先使用系统的 root 帐号登录。

只要成功地登录进了 Samba 配置页面， 就可以浏览系统的文档， 或从 Globals(全局) 选项卡开始配置了。 Globals 小节对应于 [global] 小节中的变量， 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 选项都是：

workgroup
NT 域名或工作组名， 其他计算机将通过这些名字来找到服务器。

netbios name
这个选项用于设置 Samba 服务器的 NetBIOS 名字。 默认情况下， 这是所在主机的 DNS 名字的第一部分。

server string
这个选项用于设置通过 net view 命令， 以及某些其他网络工具可以查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 中的两个最重要的配置， 是选定的安全模型，
以及客户机上用户的口令存放后端。 下面的语句控制这些选项：

security
最常见的选项形式是 security = share 和 security = user。 如果您的客户机使用用户名，
并且这些用户名与您的 FreeBSD 机器一致， 一般应选择用户级 (user) 安全。 这是默认的安全策略，
它要求客户机首先登录， 然后才能访问共享的资源。

如果采用共享级 (share) 安全， 则客户机不需要用有效的用户名和口令登录服务器，
就能够连接共享的资源。 这是较早版本的 Samba 中的默认值。

passdb backend
Samba 提供了若干种不同的验证后端模型。 您可以通过 LDAP、 NIS+、 SQL 数据库，
或经过修改的口令文件， 来完成客户端的身份验证。 默认的验证模式是 smbpasswd，
这也是本章将介绍的全部内容。

假设您使用的是默认的 smbpasswd 后端， 则必须首先创建一个 /usr/local/etc/samba/smbpasswd
文件， 来允许 Samba 对客户进行身份验证。 如果您打算让 UNIX® 用户帐号能够从 Windows®
客户机上登录， 可以使用下面的命令：

```
# smbpasswd -a username
```

目前推荐使用的后端是 tdbsam， 您应使用下面的命令来添加用户帐号：

```
# pdbedit -a -u username
```

请参考 官方的 Samba HOWTO 以了解关于配置选项的进一步信息。 按照前面给出的基本描述，
您应该已经可以启动 Samba 了。

30.9.3. 启动 Samba

net/samba34 port 会增加一个新的用于控制 Samba 的启动脚本。 要启用这个脚本，
以便用它来完成启动、 停止或重启 Samba 的任务， 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：

```
nmbd_enable="YES"
```

```
smbd_enable="YES"
```

这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 tdbs :
Starting nmbd.
Starting smbd.
```

请参见 在 FreeBSD 中使用 rc 以了解关于使用 rc 脚本的进一步信息。

Samba 事实上包含了三个相互独立的服务程序。 您应该能够看到 nmbd 和 smbd 两个服务程序都是通过 samba 脚本启动的。 如果在 smb.conf 中启用了 winbind 名字解析服务， 则应该可以看到 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 是一个复杂的软件包， 它提供了用于与 Microsoft® Windows® 网络进行集成的各式各样的功能。 要了解关于这里所介绍的基本安装以外的其它功能， 请访问 http://www.samba.org。

30.10. 通过 NTP 进行时钟同步

30.10.1. 纵览

随着时间的推移， 计算机的时钟会倾向于漂移。 网络时间协议 (NTP) 是一种确保您的时钟保持准确的方法。

许多 Internet 服务依赖、 或极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过才发送它。 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关重要， 因为这样才能使时间戳保持一致。 类似 cron(8) 这样的程序， 也依赖于正确的系统时钟， 才能够准确地执行操作。

FreeBSD 附带了 ntpd(8) NTP 服务器， 它可以用于查询其它的 NTP 服务器， 并配置本地计算机的时钟， 或者为其它机器提供服务。

30.10.2. 选择合适的 NTP 服务器

为了同步您的系统时钟， 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 都可能会提供用于这样目的的 NTP 服务器-请查看他们的文档以了解是否是这样。 另外， 也有一个在线的 公开的 NTP 服务器列表， 您可以从中选一个较近的 NTP 服务器。 请确认您选择的服务器的访问策略， 如果需要的话， 申请一下所需的许可。

选择多个相互不连接的 NTP 服务器是一个好主意， 这样在某个服务器不可达， 或者时钟不可靠时就可以有别的选择。 这是因为， ntpd(8) 会智能地选择它收到的响应- 它会更倾向于使用可靠的服务器。

30.10.3. 配置您的机器

30.10.3.1. 基本配置

如果只想在系统启动时同步时钟，则可以使用 ntpdate(8)。对于经常重新启动，并且不需要经常同步的桌面系统来说这比较适合，但绝大多数机器都应该运行 ntpd(8)。

在引导时使用 ntpdate(8) 来配合运行 ntpd(8) 也是一个好主意。ntpd(8) 渐进地修正时钟，而 ntpdate(8) 则直接设置时钟，无论机器的当前时间和正确时间有多大的偏差。

要启用引导时的 ntpdate(8)，需要把 ntpdate_enable="YES" 加到 /etc/rc.conf 中。此外，还需要通过 ntpdate_flags 来设置同步的服务器和选项，它们将传递给 ntpdate(8)。

30.10.3.2. 一般配置

NTP 是通过 /etc/ntp.conf 文件来进行配置的，其格式在 ntp.conf(5) 中进行了描述。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
```

这里，server 选项指定了使用哪一个服务器，每一个服务器都独立一行。如果某一台服务器上指定了 prefer (偏好) 参数，如上面的 ntplocal.example.com，则会优先选择这个服务器。
如果偏好的服务器和其他服务器的响应存在显著的差别，则丢弃它的响应，否则将使用来自它的响应，而不理会其他服务器。一般来说，prefer 参数应该标注在非常精确的 NTP 时源，例如那些包含特殊的时间监控硬件的服务器上。

而 driftfile 选项，则指定了用来保存系统时钟频率偏差的文件。ntpd(8) 程序使用它来自动地补偿时钟的自然漂移，从而使时钟即使在切断了外来时源的情况下，仍能保持相当的准确度。

另外，driftfile 选项也保存上一次响应所使用的 NTP 服务器的信息。这个文件包含了 NTP 的内部信息，它不应被任何其他进程修改。

30.10.3.3. 控制您的服务器的访问

默认情况下，NTP 服务器可以被整个 Internet 上的主机访问。如果在 /etc/ntp.conf 中指定 restrict 参数，则可以控制允许哪些机器访问您的服务器。

如果希望拒绝所有的机器访问您的 NTP 服务器，只需在 /etc/ntp.conf 中加入：

```
restrict default ignore
```

这样做会禁止您的服务器访问在本地配置中列出的服务器。如果您需要令 NTP 服务器与外界的 NTP 服务器同步时间，则应允许指定服务器。请参见联机手册 ntp.conf(5) 以了解进一步的细节。

如果只希望子网内的机器通过您的服务器同步时钟，而不允许它们配置为服务器，或作为同步时钟的节点来时用，则加入

```
restrict 192.168.1.0 mask 255.255.255.0 nomodify notrap
```

这里， 需要把 192.168.1.0 改为您网络上的 IP 地址， 并把 255.255.255.0 改为您的子网掩码。

/etc/ntp.conf 可能包含多个 restrict 选项。 要了解进一步的细节， 请参见 ntp.conf(5) 的 Access Control Support(访问控制支持) 小节。

30.10.4. 运行 NTP 服务器

要让 NTP 服务器在系统启动时随之开启， 需要把 ntpd_enable="YES" 加入到 /etc/rc.conf 中。
如果希望向 ntpd(8) 传递更多参数， 需要编辑 /etc/rc.conf 中的 ntpd_flags。

要在不重新启动机器的前提下启动服务器， 需要手工运行 ntpd， 并带上 /etc/rc.conf 中的 ntpd_flags 所指定的参数。 例如：

```
# ntpd -p /var/run/ntpd.pid
```

30.10.5. 在临时性的 Internet 连接上使用 ntpd

ntpd(8) 程序的正常工作并不需要永久性的 Internet 连接。 然而，
如果您的临时性连接是配置为按需拨号的， 那么防止 NTP 通讯频繁触发拨号， 或保持连接就有必要了。
如果您使用用户级 PPP， 可以使用 filter 语句， 在 /etc/ppp/ppp.conf 中进行必要的设置。 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
```

要了解进一步的信息， 请参考 ppp(8) 的 PACKET FILTERING(包过滤) 小节， 以及 /usr/shared/examples/ppp/ 中的例子。

某些 Internet 访问提供商会阻止低编号的端口， 这会导致 NTP 无法正常工作，
因为响应无法到达您的机器。

30.10.6. 进一步的信息

关于 NTP 服务器的文档， 可以在 /usr/shared/doc/ntp/ 找到 HTML 格式的版本。

30.11. 使用 syslogd 记录远程主机的日志

处理系统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，再或者是处于各种不同类型的网络中，
监视他们上面的日志文件则显得非常难以操作， 在这种情况下，
配置远程日志记录能使整个处理过程变得更加轻松。

集中记录日志到一台指定的机器能够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处配置， 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， 命名为 logserv.example.com， 将用来收集本地网络的日志信息。 主机 B， 命名为 logclient.example.com 将把日志信息传送给服务器。 在现实中，
这两个主机都需要配置正确的正向和反向的 DNS 或者在 /etc/hosts 中记录。 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。 在大多数的情况下这是为了方便配置，或者是为了更好的管理。 不论是何原因， 在继续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规则允许 514 端口上的 UDP 报文通过。
- syslogd 被配置成接受从远程客户发来的消息。
- syslogd 服务器和所有的客户端都必须有配有正确的正向和反向 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
```

更多关于各种被支持并可用的 facility 能在 syslog.conf(5) 手册页中找到。

一旦加入以后， 所有此类 facility 消息都会被记录到先前指定的文件 /var/log/logclient.log。

提供服务的机器还需要在其 /etc/rc.conf 中配置：

```
syslogd_enable="YES"
syslogd_flags="-a logclient.example.com -v -v"
```

第一个选项表示在系统启动时启用 syslogd 服务，
第二个选项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详细程度。 在调整 facility 配置的时候， 这个配置非常有用，
因为管理员能够看到哪些消息将作为哪个 facility 的内容来记录。

可以同时指定多个 -a 选项来允许多个客户机。 此外， 还可以指定 IP 地址或网段， 请参阅 syslog(3) 联机手册以了解可用配置的完整列表。

最后， 日志文件应该被创建。 不论你用何种方法创建， 比如 touch(1) 能很好的完成此类任务：

```
# touch /var/log/logclient.log
```

此时， 应该重启并确认一下 syslogd 守护进程：

```
# /etc/rc.d/syslogd restart
# pgrep syslog
```

如果返回了一个 PIC 的话， 服务端应该被成功重启了, 并继续开始配置客户端。 如果服务端没有重启的话，请在 /var/log/messages 日志中查阅相关输出。

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， 并在本地保存拷贝。

与日志服务器类似， 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们的日志服务器。
- 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 中有正确的记录。

相比服务器来说配置客户端更轻松一些。 客户端的机器在 /etc/rc.conf 中做如下的设置：

```
syslogd_enable="YES"
syslogd_flags="-s -v -v"
```

和前面类似， 这些选项会在系统启动过程中启用 syslogd 服务， 并增加日志消息的详细程度。 而 -s 选项则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哪部分生成的。 举例来说， ftp 和 ipfw 都是 facility。 当这两项服务生成日志消息时， 它们通常在日志消息中包含了这两种工具。 Facility 通常带有一个优先级或等级， 就是用来标记一个日志消息的重要程度。 最普通的为 warning 和 info。 请参阅 syslog(3) 手册页以获得一个完整可用的 facility 与优先级列表。

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器， 看上去差不多如下这样：

```
*.*    @logserv.example.com
```

添加后， 必须重启 syslogd 使得上述修改生效：

```
# /etc/rc.d/syslogd restart
```

测试日志消息是否能通过网络发送， 在准备发出消息的客户机上用 logger(1) 来向 syslogd 发出信息：

```
# logger "Test message from logclient"
```

这段消息现在应该同时出现在客户机的 /var/log/messages 以及日志服务器的 /var/log/logclient.log 中。

30.11.3. 调试日志服务器

在某些情况下， 如果日志服务器没有收到消息的话就需要调试一番了。 有几个可能的原因， 最常见的两个是网络连接的问题和 DNS 的问题。 为了测试这些问题， 请确认两边的机器都能使用 /etc/rc.conf 中所设定的主机名访问到对方。 如果这个能正常工作的话， 那么就需要对 /etc/rc.conf 中的 syslogd_flags 选项做些修改了。

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。 为了增加调试的输出， 修改 ayalogd_flags 选项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， 屏幕上将立刻闪现类似这样的调试数据：

```
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。 在一点一点的检查了配置文件之后， 发现了 /etc/rc.conf 中如下这行有输入错误：

```
syslogd_flags="-d -a logclien.example.com -v -v"
```

这行应该包涵有 logclient， 而不是 logclien。 在做了正确的修改并重启之后便能见到预期的效果了：

```
# /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

就像其他的网络服务一样， 在实现配置之前需要考虑安全性。 有时日志文件也包含了敏感信息，
比如本地主机上所启用的服务， 用户帐号和配置数据。 从客户端发出的数据经过网络到达服务器，
这期间既没有加密也没有密码保护。 如果有加密需要的话， 可以使用 security/stunnel，
它将在一个加密的隧道中传输数据。

本地安全也同样是个问题。 日志文件在使用中或循环转后都没有被加密。
本地用户可能读取这些文件以获得对系统更深入的了解。 对于这类情况，
给这些文件设置正确的权限是非常有必要的。 newsyslog(8) 工具支持给新创建和循环的日志设置权限。
把日志文件的权限设置为 600 能阻止本地用户不必要的窥探。

Chapter 31. 防火墙

31.1. 入门

防火墙的存在， 使得过滤出入系统的数据流成为可能。 防火墙可以使用一组或多组 "规则 (rules)"，
来检查出入您的网络连接的数据包， 并决定允许或阻止它们通过。
这些规则通常可以检查数据包的某个或某些特征， 这些特征包括， 但不必限于协议类型、
来源或目的主机地址， 以及来源或目的端口。

防火墙可以大幅度地改善主机或网络的安全。 它可以用来完成下面的任务：

- 保护和隔离应用程序、 服务程序， 以及您内部网络上的机器， 不受那些来自公共的 Internet 网络上您所不希望的数据流量的干扰。
- 限制或禁止从内部网访问公共的 Internet 上的服务。
- 支持网络地址转换 (NAT)， 它使得您的内部网络能够使用私有的 IP 地址， 并分享一条通往公共的 Internet 的连接 (使用一个 IP 地址， 或者一组公网地址)。

读完这章， 您将了解：

- 如何正确地定义包过滤规则。
- FreeBSD 中内建的几种防火墙之间的差异。
- 如何使用和配置 OpenBSD 的 PF 防火墙。
- 如何使用和配置 IPFILTER。
- 如何使用和配置 IPFW。

阅读这章之前， 您需要：

- 理解基本的 FreeBSD 和 Internet 概念。

31.2. 防火墙的概念

建立防火墙规则集的基本方法有两种： "明示允许 (inclusive)"型 或 "明示禁止 (exclusive)"型。
明示禁止的防火墙规则， 默认允许所有数据通过防火墙， 而这种规则集中定义的，
则是不允许通过防火墙的流量， 换言之， 与这些规则不匹配的数据， 全部是允许通过防火墙的。
明示允许的防火墙正好相反， 它只允许符合规则集中定义规则的流量通过， 而其他所有的流量都被阻止。

明示允许型防火墙能够提供对于传出流量更好的控制， 这使其更适合那些直接对 Internet 公网提供服务的系统的需要。 它也能够控制来自 Internet 公网到您的私有网络的访问类型。
所有和规则不匹配的流量都会被阻止并记录在案。 一般来说明示允许防火墙要比明示禁止防火墙更安全，
因为它们显著地减少了允许不希望的流量通过可能造成的风险。

除非特别说明， 这一章的配置和示范的规则集都是创建明示允许防火墙的。

使用了 "带状态功能的防火墙 (stateful firewall)"， 可以进一步地收紧安全机制。
这种防火墙能够记录通过防火墙的连接， 进而只允许与现有连接匹配的连接， 或创建新的连接。
带状态功能的防火墙的缺点是， 在很短时间内有大量的连接请求时， 它们可能会受到拒绝服务 (DoS) 攻击。 绝大多数防火墙都提供了同时启用两种防火墙的能力， 以便为站点提供更好的保护。

31.3. 防火墙软件包

FreeBSD 的基本系统内建了三种不同的防火墙软件包。 它们是 IPFILTER (也被称作 IPF)、 IPFIREWALL (也被称作 IPFW)， 以及 OpenBSD 的 PacketFilter (也被称为 PF)。 FreeBSD 也提供了两个内建的、用于流量整形 (基本上是控制带宽占用) 的软件包： altq(4) 和 dummynet(4)。 Dummynet 在过去一直和 IPFW 紧密集成， 而 ALTQ 则需要配合 PF 使用。 IPFILTER 的流量整形功能可以使用 IPFILTER 的 NAT 和过滤功能以及 IPFW 的 dummynet(4) 配合, 或者 使用 PF 跟 ALTQ 的组合。 IPFW， 以及 PF 都是用规则来控制是否允许数据包出入您的系统， 虽然它们采取了不同的实现方法和规则语法。

FreeBSD 包含多个内建的防火墙软件包的原因在于， 不同的人会有不同的需求和偏好。
任何一个防火墙软件包都很难说是最好的。

作者倾向于使用 IPFILTER， 因为它提供的状态式规则， 在 NAT 的环境中要简单许多， 而且它内建了 ftp 代理， 这简化了使用外部 FTP 服务时所需的配置。

由于所有的防火墙都基于检查所选定的包控制字段来实现功能， 撰写防火墙规则集时， 就必须了解 TCP/IP 是如何工作的， 以及包的控制字段在正常会话交互中的作用。 您可以在这个网站找到一份很好的解释文档：
http://www.ipprimer.com/overview.cfm.

31.4. OpenBSD Packet Filter (PF) 和 ALTQ

2003 年 7 月， OpenBSD 的防火墙， 也就是常说的 PF 被成功地移植到了 FreeBSD 上， 并可以通过 FreeBSD Ports Collection 来安装了； 第一个将 PF 集成到基本系统中的版本是 2004 年 11 月发行的 FreeBSD 5.3。 PF 是一个完整的提供了大量功能的防火墙软件， 并提供了可选的 ALTQ (交错队列， Alternate Queuing) 功能。 ALTQ 提供了服务品质 (QoS) 带宽整形功能。

OpenBSD 项目非常杰出的维护着一份 PF FAQ。 就其本身而言，这一节注重于 FreeBSD 的 PF 和提供一些关于使用方面的一般常识。更详细的使用信息请参阅 PF FAQ。

更多的详细信息， 可以在 FreeBSD 版本的 PF 网站上找到： http://pf4freebsd.love2party.net/。

31.4.1. 使用 PF 可加载的内核模块

要加载 PF 内核模块， 可以在 /etc/rc.conf 中加入下面的设置：

```
pf_enable="YES"
```

然后使用启动脚本来加载模块：

```
# /etc/rc.d/pf start
```

需要说明的是， 如果系统中没有规则集配置文件， 则上述操作不会加载 PF 模块。 配置文件的默认位置是 /etc/pf.conf。 如果 PF 规则集在其他位置， 可以用下面的 /etc/rc.conf 配置来告诉 PF：

```
pf_rules="/path/to/pf.conf"
```

pf.conf 的例子可以在 /usr/shared/examples/pf/ 找到。

PF 模块也可以手工从命令行加载：

```
# kldload pf.ko
```

PF 的日志记录功能是由 pflog.ko 提供的， 通过在 /etc/rc.conf 中加入下面的设置：

```
pflog_enable="YES"
```

然后使用启动脚本来加载模块：

```
# /etc/rc.d/pflog start
```

如果您需要其他 PF 特性， 则需要将 PF 支持联编进内核。

31.4.2. PF 内核选项

虽然你不必亲自把对 PF 的支持编译进 FreeBSD 内核，但是有时你仍然需要这么做来使用到 PF 的某些没有被收录进可加载模块的高级特性，比如 pfsync(4) 伪设备用来发送某些改变到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 的详细信息可以参阅本手册的 Common Address Redundancy Protocol (CARP， 共用地址冗余协议)。

The PF kernel options can be found in /usr/src/sys/conf/NOTES and are reproduced below:

有关 PF 的内核选项可以在 /usr/src/sys/conf/NOTES 中找到， 以下也略有阐述：

```
device pf
device pflog
device pfsync
```

device pf 选项用于启用 "Packet Filter" 防火墙的支持 （pf(4)）。

device pflog 启用可选的 pflog(4) 伪网络设备， 用以通过 bpf(4) 描述符来记录流量。 pflogd(8) 服务可以用来存储信息， 并把它们以日志形式记录到磁盘上。

device pfsync 选项启用可选的 pfsync(4) 支持，这是用于监视 "状态变更" 的伪网络设备。

31.4.3. 可用的 rc.conf 选项

The following rc.conf(5) statements configure PF and pflog(4) at boot:

以下 rc.conf(5) 中的语句用于启动时配置 PF 和 pflog(4)

```
pf_enable="YES"            # 启用 PF (如果需要的话， 自动加载内核模块)
pf_rules="/etc/pf.conf"      # pf 使用的规则定义文件
pf_flags=""                # 启动时传递给 pfctl 的其他选项
pflog_enable="YES"         # 启动 pflogd(8)
pflog_logfile="/var/log/pflog"  # pflogd 用于记录日志的文件名
pflog_flags=""             # 启动时传递给 pflogd 的其他选项
```

如果您的防火墙后面有一个 LAN， 而且需要通过它来转发 LAN 上的包， 或进行 NAT，还需要同时启用下述选项：

```
gateway_enable="YES"       # 启用为 LAN 网关
```

31.4.4. 建立过滤规则

PF 会从 pf.conf(5) (默认为 /etc/pf.conf) 文件中读取配置规则，并根据那里的规则修改、丢弃或让数据包通过。 默认安装的 FreeBSD 已经提供了一些简单的例子放在 /usr/shared/examples/pf/ 目录下。 请参阅 PF FAQ 获取完整的 PF 规则信息。

在浏览 PF FAQ 时， 请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 和之前的系统使用的是与 OpenBSD 4.1 相同版本的 PF。 FreeBSD 9.X 和之后的系统使用的是与 OpenBSD 4.5 相同版本的 PF。

FreeBSD packet filter 邮件列表 是一个提有关配置使用 PF 防火墙问题的好地方。请在提问之前查阅邮件列表的归档！

31.4.5. 使用 PF

使用 pfctl(8) 可以控制 PF。 以下是一些实用的命令 （请查阅 pfctl(8) 获得全部可用的选项）：

命令	作用
pfctl -e	启用 PF
pfctl -d	禁用 PF
pfctl -F all -f /etc/pf.conf	清除所有规则 (nat, filter, state, table, 等等。) 并读取 /etc/pf.conf
pfctl -s [rules | nat | state]	列出 filter 规则, nat 规则, 或状态表
pfctl -vnf /etc/pf.conf	检查 /etc/pf.conf 中的错误，但不加载相关的规则

31.4.6. 启用 ALTQ

ALTQ 只有在作为编译选项加入到 FreeBSD 内核时才能使用。ALTQ 目前还不是所有的可用网卡驱动都能够支持的。 请参见 altq(4) 联机手册了解您正使用的 FreeBSD 版本中的驱动支持情况。

下面这些选项将启用 ALTQ 以及一些附加的功能：

```
options     ALTQ
options     ALTQ_CBQ     # 基于分类的排列 (CBQ)
options     ALTQ_RED     # 随机先期检测 (RED)
options     ALTQ_RIO     # 对进入和发出的包进行 RED
options     ALTQ_HFSC    # 带等级的包调度器 (HFSC)
options     ALTQ_PRIQ    # 按优先级的排列 (PRIQ)
options     ALTQ_NOPCC   # 在联编 SMP 内核时必须使用，禁止读时钟
```

options ALTQ 将启用 ALTQ 框架的支持。

options ALTQ_CBQ 用于启用 基于分类的队列 (CBQ) 支持。 CBQ 允许您将连接分成不同的类别， 或者说，队列， 以便在规则中为它们指定不同的优先级。

options ALTQ_RED 将启用 随机预检测 (RED)。 RED 是一种用于防止网络拥塞的技术。 RED 度量队列的长度， 并将其与队列的最大和最小长度阈值进行比较。 如果队列过长， 则新的包将被丢弃。 如名所示， RED 从不同的连接中随机地丢弃数据包。

options ALTQ_RIO 将启用 出入的随机预检测。

options ALTQ_HFSC 启用 层次式公平服务平滑包调度器。 要了解关于 HFSC 进一步的信息， 请参见 http://www-2.cs.cmu.edu/~hzhang/HFSC/main.html。

options ALTQ_PRIQ 启用 优先队列 (PRIQ)。 PRIQ 首先允许高优先级队列中的包通过。

options ALTQ_NOPCC 启用 ALTQ 的 SMP 支持。 如果是 SMP 系统， 则必须使用它。

31.5. IPFILTER (IPF) 防火墙

IPFILTER 的作者是 Darren Reed。 IPFILTER 是独立于操作系统的： 它是一个开放源代码的应用，并且已经被移植到了 FreeBSD、 NetBSD、 OpenBSD、 SunOS、 HP/UX， 以及 Solaris 操作系统上。

IPFILTER 的支持和维护都相当活跃， 并且有规律地发布更新版本。

IPFILTER 提供了内核模式的防火墙和 NAT 机制，
这些机制可以通过用户模式运行的接口程序进行监视和控制。 防火墙规则可以使用 ipf(8)
工具来动态地设置和删除。 NAT 规则可以通过 ipnat(1) 工具来维护。 ipfstat(8) 工具则可以用来显示
IPFILTER 内核部分的统计数据。 最后， 使用 ipmon(8) 程序可以把 IPFILTER
的动作记录到系统日志文件中。

IPF 最初是使用一组 "以最后匹配的规则为准" 的策略来实现的， 这种方式只能支持无状态的规则。
随着时代的进步， IPF 被逐渐增强， 并加入了 "quick" 选项， 以及支持状态的 "keep state" 选项，
这使得规则处理逻辑变得更富有现代气息。 IPF 的官方文档只介绍了传统的规则编写方法和文件处理逻辑。
新增的功能只是作为一些附加的选项出现， 如果能完全理解这些功能，
则对于建立更安全的防火墙就很有好处。

这一节中主要是针对 "quick" 选项， 以及支持状态的 "keep state" 选项的介绍。
这是明示允许防火墙规则集最基本的编写要素。

要获得关于传统规则处理方式的详细信息， 请参考： 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 找到。

除此之外， 您还可以在 http://marc.theaimsgroup.com/?l=ipfilter 找到开放源代码的 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"， 系统就会自动地动态加载 IPF 内核模块。
这个内核模块在创建时启用了日志支持， 并加入了 default pass all 选项。
如果只是需要把默认的规则设置为 block all 的话， 就不需要把 IPF 编译到内核中。 简单地通过把 block all
这条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 内核编译选项并不是启用 IPF 所必需的。 这里只是作为背景知识来加以阐述。 如果将
IPF 编入了内核， 则对应的内核模块将不被使用。

关于 IPF 选项语句的内核编译配置的例子， 可以在内核源代码中的 /usr/src/sys/conf/NOTES 找到。
此处列举如下：

```
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 的规则的包， 都会被记录到 ipl
这个包记录伪-设备中。

options IPFILTER_DEFAULT_BLOCK 将改变防火墙的默认动作， 进而， 所有不匹配防火墙的 pass
规则的包都会被阻止。

这些选项只有在您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

要在启动时激活 IPF， 需要在 /etc/rc.conf 中增加下面的设置：

```
ipfilter_enable="YES"          # 启动 ipf 防火墙
ipfilter_rules="/etc/ipf.rules"  # 将被加载的规则定义， 这是一个文本文件
ipmon_enable="YES"            # 启动 IP 监视日志
ipmon_flags="-Ds"             # D = 作为服务程序启动
                              # s = 使用 syslog 记录
                              # v = 记录 tcp 窗口大小、 ack 和顺序号(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 地址范围的 LAN， 还需要增加下面的一些选项来启用 NAT 功能：

```
gateway_enable="YES"          # 启用作为 LAN 网关的功能
ipnat_enable="YES"            # 启动 ipnat 功能
ipnat_rules="/etc/ipnat.rules"  # 用于 ipnat 的规则定义文件
```

31.5.4. IPF

ipf(8) 命令可以用来加载您自己的规则文件。 一般情况下， 您可以建立一个包括您自定义的规则的文件，
并使用这个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 用于指定将要被读取的规则定义文件。

这个功能使得您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为使用全新的规则集， 而不需要重新启动系统。
这对于测试新的规则来说就很方便， 因为您可以任意执行上面的命令。

请参考 ipf(8) 联机手册以了解这个命令提供的其它选项。

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换的脚本。

也确实有办法利用脚本的非常强大的符号替换能力来构建 IPF 规则。 要了解进一步的细节， 请参考
构建采用符号替换的规则脚本。

31.5.5. IPFSTAT

默认情况下， ipfstat(8) 会获取并显示所有的累积统计，
这些统计是防火墙启动以来用户定义的规则匹配的出入流量， 您可以通过使用 ipf -Z
命令来将这些计数器清零。

请参见 ipfstat(8) 联机手册以了解进一步的细节。

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 (进入流量) 或者 -o (输出流量)，
这个命令就只获取并显示内核中所安装的对应过滤器规则的统计数据。

ipfstat -in 以规则号的形式显示进入的内部规则表。

ipfstat -on 以规则号的形式显示流出的内部规则表。

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 显示内部规则表中的进入流量， 每一个匹配规则前面会同时显示匹配的次数。

ipfstat -oh 显示内部规则表中的流出流量， 每一个匹配规则前面会同时显示匹配的次数。

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
正运行的进程表的方式来显示统计数据。 当您的防火墙正在受到攻击的时候， 这个功能让您得以识别、
试验， 并查看攻击的数据包。 这个选项提还提供了实时选择希望监视的目的或源 IP、 端口或协议的能力。
请参见 ipfstat(8) 联机手册以了解详细信息。

31.5.6. IPMON

为了使 ipmon 能够正确工作， 必须打开 IPFILTER_LOG 这个内核选项。
这个命令提供了两种不同的使用模式。 内建模式是默认的模式， 如果您不指定 -D 参数，
就会采用这种模式。

服务模式是持续地通过系统日志来记录的工作模式， 这样，
您就可以通过查看日志来了解过去曾经发生过的事情。 这种模式是 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动轮转功能， 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 = 作为服务程序启动
            # s = 使用 syslog 记录
            # v = 记录 tcp 窗口大小、 ack 和顺序号(seq)
            # n = 将 IP 和端口映射为名字
```

记录日志的好处是很明显的。 它提供了在事后重新审查相关信息， 例如哪些包被丢弃， 以及这些包的来源地址等等。 这将为查找攻击者提供非常有用的第一手资料。

即使启用了日志机制， IPF 仍然不会对其规则进行任何日志记录工作。 防火墙管理员可以决定规则集中的哪些应记录日志， 并在这些规则上加入 log 关键字。 一般来说， 只应记录拒绝性的规则。

作为惯例， 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， 作为您的规则集的最后一条。 这样就能够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" 的组。 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动作
LOG_NOTICE - 同时记录通过的那些数据包
LOG_WARNING - 同时记录阻止的数据包
LOG_ERR - 进一步记录含不完整的包头的数据包
```

要设置 IPFILTER 来将所有的数据记录到 /var/log/ipfilter.log， 需要首先建立这个文件。 下面的命令可以完成这个工作：

```
# 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 这样的用用程序所产生的系统消息的方法。

您需要将下列语句加到 /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。

要让 /etc/syslog.conf 中的修改立即生效， 可以重新启动计算机， 或者通过执行 /etc/rc.d/syslogd reload 来让它重新读取 /etc/syslog.conf。

不要忘了修改 /etc/newsyslog.conf 来让刚创建的日志进行轮转。

31.5.8. 记录消息的格式

由 ipmon 生成的消息由空格分隔的数据字段组成。 所有的消息都包含的字段是：

1. 接到数据包的日期。
2. 接到数据包的时间。 其格式为 HH:MM:SS.F， 分别是小时、 分钟、 秒， 以及分秒 (这个数字可能有许多位)。
3. 处理数据包的网络接口名字， 例如 dc0。
4. 组和规则的编号， 例如 @0:17。

可以通过 ipfstat -in 来查看这些信息。

1. 动作： p 表示通过， b 表示阻止， S 表示包头不全， n 表示没有匹配任何规则， L 表示 log 规则。 显示这些标志的顺序是： S, p, b, n, L。 大写的 P 或 B 表示记录包的原因是某个全局的日志配置， 而不是某个特定的规则。
2. 地址。 这实际上包括三部分： 源地址和端口 (以逗号分开)， 一个 → 符号， 以及目的地址和端口， 例如： 209.53.17.22,80 → 198.73.220.17,1722。
3. PR， 后跟协议名称或编号， 例如： PR tcp。
4. len， 后跟包头的长度， 以及包的总长度， 例如： len 20 40。

对于 TCP 包， 则还会包括一个附加的字段， 由一个连字号开始， 之后是表示所设置的标志的一个字母。 请参见 ipf(5) 联机手册， 以了解这些字母所对应的标志。

对于 ICMP 包， 则在最后会有两个字段。 前一个总是 "ICMP"， 而后一个则是 ICMP 消息和子消息的类型， 中间以斜线分靠， 例如 ICMP 3/3 表示端口不可达消息。

31.5.9. 构建采用符号替换的规则脚本

一些有经验的 IPF 会创建包含规则的文件， 并把它编写成能够与符号替换脚本兼容的方式。 这样做最大的好处是能够在修改时只修改符号名字所代表的值， 而在脚本执行时直接替换掉所有的名符。 作为脚本， 可以使用符号替换来把那些经常使用的值直接用于多个规则。 下面将给出一个例子。

这个脚本所使用的语法与 sh(1)、 csh(1)， 以及 tcsh(1) 脚本。

符号替换的前缀字段是美元符号： $。

符号字段不使用 $ 前缀。

希望替换符号字段的值， 必须使用双引号 (") 括起来。

您的规则文件的开头类似这样：

```
############## IPF 规则脚本的开头 ##########################
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 的静态 IP 地址
ks="keep state"
fks="flags S keep state"

# 可以使用这个脚本来建立 /etc/ipf.rules 文件，
# 也可以 "直接地" 运行它。
#
# 请删除两个注释号之一。
#
# 1) 保留下面一行， 则创建 /etc/ipf.rules：
#cat > /etc/ipf.rules << EOF
```

```
#
# 2) 保留下面一行， 则 "直接地" 运行脚本：
/sbin/ipf -Fa -f - << EOF

# 允许发出到我的 ISP 的域名服务器的访问
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

# 允许发出未加密的 www 访问请求
pass out quick on $oif proto tcp from $myip to any port = 80 $fks

# 允许发出使用 TLS SSL 加密的 https www 访问请求
pass out quick on $oif proto tcp from $myip to any port = 443 $fks
EOF
################### IPF 规则脚本的结束 ########################
```

这就是所需的全部内容。 这个规则本身并不重要， 它们主要是用于体现如何使用符号代换字段，
以及如何完成值的替换。 如果上面的例子的名字是 /etc/ipf.rules.script，
就可以通过输入下面的命令来重新加载规则：

```
# sh /etc/ipf.rules.script
```

在规则文件中嵌入符号有一个问题： IPF 无法识别符号替换， 因此它不能直接地读取这样的脚本。

这个脚本可以使用下面两种方法之一来使用：

- 去掉 cat 之前的注释， 并注释掉 /sbin/ipf 开头的那一行。 像其他配置一样， 将 ipfilter_enable="YES" 放到 /etc/rc.conf 文件中， 并在此后立刻执行脚本， 以创建或更新 /etc/ipf.rules。
- 通过把 ipfilter_enable="NO" (这是默认值) 加到 /etc/rc.conf 中， 来禁止系统启动脚本开启 IPFILTER。

  在 /usr/local/etc/rc.d/ 启动目录中增加一个类似下面的脚本。 应该给它起一个显而易见的名字， 例如 ipf.loadrules.sh。 请注意， .sh 扩展名是必需的。

  ```
  #!/bin/sh
  sh /etc/ipf.rules.script
  ```

  脚本文件必须设置为属于 root， 并且属主可读、 可写、 可执行。

  ```
  # chmod 700 /usr/local/etc/rc.d/ipf.loadrules.sh
  ```

这样， 在系统启动时， 就会自动加载您的 IPF 规则了。

31.5.10. IPF 规则集

规则集是指一组编写好的依据包的值决策允许通过或阻止 IPF 规则。 包的双向交换组成了一个会话交互。
防火墙规则集会作用于来自于 Internet 公网的包以及由系统发出来回应这些包的数据包。 每一个 TCP/IP
服务 (例如 telnet, www, 邮件等等) 都由协议预先定义了其特权 (监听) 端口。

发到特定服务的包会从源地址使用非特权 (高编号) 端口发出， 并发到特定服务在目的地址的对应端口。
所有这些参数 (例如： 端口和地址） 都是可以为防火墙规则所利用的， 判别是否允许服务通过的标准。

IPF 最初被写成使用一组称作 "以最后匹配的规则为准" 的处理逻辑， 且只能处理无状态的规则。
随着时代的发展， IPF 进行了改进， 并提供了 "quick" 选项， 以及一个有状态的 "keep state" 选项。
后者使处理逻辑迅速地跟上了时代的步伐。

这一节中提供的一些指导， 是基于使用包含 "quick" 选项和有状态的 "keep state" 选项来进行阐述的。
这些是编写明示允许防火墙规则集的基本要素。

当对防火墙规则进行操作时， 应 谨慎行事。 某些配置可能会 将您反锁在 服务器外面。
保险起见， 您可以考虑在第一次进行防火墙配置时在本地控制台上， 而不是远程，
如通过 ssh 来进行。

31.5.11. 规则语法

这里给出的规则语法已经简化到只处理那些新式的带状态规则， 并且都是 "第一个匹配的规则获胜"
逻辑的。 要了解完整的传统规则语法描述， 请参见 ipf(8) 联机手册。

以 # 字符开头的内容会被认为是注释。 这些注释可以出现在一行规则的末尾， 或者独占一行。
空行会被忽略。

规则由关键字组成。 这些关键字必须以一定的顺序， 从左到右出现在一行上。
接下来的文字中关键字将使用粗体表示。 某些关键字可能提供了子选项， 这些子选项本身可能也是关键字，
而且可能会提供更多的子选项。 下面的文字中， 每种语法都使用粗体的小节标题呈现，
并介绍了其上下文。

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 网络接口的名字

SELECTION = proto 协议名称 | 源/目的 IP | port = 端口号 | flags 标志值

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)

动作对表示匹配规则的包应采取什么动作。 每一个规则 必须 包含一个动作。 可以使用下面两种动作之一：

block 表示如果规则与包匹配， 则丢弃包。

pass 表示如果规则与包匹配， 则允许包通过防火墙。

31.5.11.2. IN-OUT

每个过滤器规则都必须明确地指定是流入还是流出的规则。 下一个关键字必须要么是 in， 要么是 out，

否则将无法通过语法检查。

in 表示规则应被应用于刚刚从 Internet 公网上收到的数据包。

out 表示规则应被应用于即将发出到 Internet 的数据包。

31.5.11.3. OPTIONS

这些选项必须按下面指定的顺序出现。

log 表示包头应被写入到 ipl 日志 (如前面 LOGGING 小节所介绍的那样)， 如果它与规则匹配的话。

quick 表示如果给出的参数与包匹配， 则以这个规则为准， 这使得能够 "短路" 掉后面的规则。
这个选项对于使用新式的处理逻辑是必需的。

on 表示将网络接口的名称作为筛选参数的一部分。 接口的名字会在 ifconfig(8) 的输出中显示。
使用这个选项， 则规则只会应用到某一个网络接口上的出入数据包上。 要配置新式的处理逻辑，
必须使用这个选项。

当记录包时， 包的头会被写入到 IPL 包日志伪设备中。 紧跟 log 关键字， 可以使用下面几个修饰符
(按照下列顺序)：

body 表示应同时记录包的前 128 字节的内容。

first 如果 log 关键字和 keep state 选项同时使用， 则这个选项只在第一个包上触发，
这样就不用记录每一个 "keep state" 包信息了。

31.5.11.4. SELECTION

这一节所介绍的关键字可以用于所检察的包的属性。 有一个关键字主题， 以及一组子选项关键字，
您必须从他们中选择一个。 以下是一些通用的属性， 它们必须按下面的顺序使用：

31.5.11.5. PROTO

proto 是一个主题关键字， 它必须与某个相关的子选项关键字配合使用。
这个值的作用是匹配某个特定的协议。 要使用新式的规则处理逻辑， 就必须使用这个选项。

tcp/udp | udp | tcp | icmp 或其他在 /etc/protocols 中定义的协议。 特殊的协议关键字 tcp/udp
可以用于匹配 TCP 或 UDP 包， 引入这个关键字的作用是是避免大量的重复规则的麻烦。

31.5.11.6. SRC_ADDR/DST_ADDR

使用 all 关键词， 基本上相当于 "from any to any" 在没有配合其他关键字的情形。

from src to dst： from 和 to 关键字主要是用来匹配 IP 地址。 所有的规则都必须 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 地址的特殊关键字。 例如， 您可以使用 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码来表示 IP 的话， 表达地址就会很麻烦。 使用 net-mgmt/ipcalc port
可以帮助进行计算。 请参见下面的网页了解如何撰写长度掩码： http://jodies.de/ipcalc。

31.5.11.7. PORT

如果为源或目的指定了匹配端口， 规则就只能应用于 TCP 和 UDP 包了。 当编写端口比较规则时，
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对象一侧，
则被认为是源端口号； 反之， 则被认为是目的端口号。 要使用新式的规则处理逻辑， 就必须与 to
对象配合使用这个选项。 使用的例子： from any to any port = 80

对单个端口的比较可以多种方式进行， 并可使用不同的比较算符。 此外， 还可以指定端口的范围。

port "=" | "!=" | "<" | ">" | "<=" | ">=" | "eq" | "ne" | "lt" | "gt" | "le" | "ge".

要指定端口范围， 可以使用 "<>" | "><"。

在源和目的匹配参数之后， 需要使用下面两个参数， 才能够使用新式的规则处理逻辑。

31.5.11.8. TCP_FLAG

标志只对 TCP 过滤有用。 这些字母用来表达 TCP 包头的标志。

新式的规则处理逻辑使用 flags S 参数来识别 tcp 会话开始的请求。

31.5.11.9. STATEFUL

keep state 表示如果有一个包与规则匹配， 则其筛选参数应激活有状态的过滤机制。

如果使用新式的处理逻辑， 则这个选项是必需的。

31.5.12. 有状态过滤

有状态过滤将网络流量当作一种双向的包交换来处理。 如果激活它， keep-state
会动态地为每一个相关的包在双向会话交互过程中产生内部规则。
它能够确认发起者和包的目的地之间的会话是有效的双向包交换过程的一部分。 如果包与这些规则不符，
则将自动地拒绝。

状态保持也使得 ICMP 包能够与 TCP 或 UDP 会话相关。 因此，
如果您在浏览网站时收到允许的状态保持规则匹配的 ICMP 类型 3 代码 4 响应，
则这些响应会被自动地允许进入。 所有 IPF 能够处理的包， 都可以作为某种活跃会话的一部分，
即使它是另一种协议的， 也会被允许进入。

所发生的事情是：

将要通过连入 Internet 公网的网络接口发出的包， 首先会经过动态状态表的检查。
如果包与会话中预期的下一个包匹配， 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。
不属于活跃会话的包， 则简单地交给输出规则集去检查。

发到连入 Internet 公网接口的包， 也会先经过动态状态表的检查。 如果包与会话中预期的下一个包匹配，
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。 不属于活跃会话的包，
则简单地交给输入规则集去检查。

当会话结束时， 对应的项会在动态状态表中删除。

有状态过滤使得您能够集中于阻止/允许新的会话。 一旦新会话被允许通过，
则所有后续的包就都被自动地允许通过， 而伪造的包则被自动地拒绝。 如果新的会话被阻止，
则后续的包也都不会被允许通过。 有状态过滤从技术角度而言， 在阻止目前攻击者常用的洪水式攻击来说，
具有更好的抗御能力。

31.5.13. 明示允许规则集的例子

下面的规则集是如何编写非常安全的明示允许防火墙规则集的一个范例。 明示允许防火墙只让允许的服务
pass (通过)， 而所有其他的访问都会被默认地拒绝。 期望用来保护其他机器的防火墙， 通常也叫做
"网络防火墙"， 应使用至少两个网络接口， 并且通常只有一个接入到受信的一端 (LAN)，
而另一块则接入不受信的一端 (Internet 公网)。 另外， 防火墙也可以配置为只保护它所运行的那个系统 -
这种类型称作 "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 在内的所有类 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讯。 防火墙规则必须允许这些包无阻碍地通过。

接入 Internet 公网的网络接口， 是放置规则并允许将访问请求发到 Internet 以及接收响应的地方。
这有可能是用户模式的 PPP tun0 接口， 如果您的网卡同 DSL 或电缆调制解调器相联的话。

如果有网卡是直接接入私有网段的， 这些网络接口就可能需要配置允许来自这些 LAN 的包在彼此之间，以及到外界 (Internet) 上的对应的通过规则。

一般说来， 规则应被组织为三个主要的小节： 所有允许自由通过的接口规则， 发到公网接口的规则，以及进入公网接口的规则。

每一个公网接口规则中， 经常会匹配到的规则应该放置在尽可能靠前的位置。
而最后一个规则应该是阻止包通过， 并记录它们。

下面防火墙规则集中， Outbound 部分是一些使用 pass 的规则， 这些规则指定了允许访问的公网 Internet 服务， 并且指定了 quick、 on、 proto、 port， 以及 keep state 这些选项。 proto tcp 规则还指定了 flag 这个选项， 这样会话的第一个包将出发状态机制。

接收部分则首先阻止所有不希望的包， 这样做有两个不同的原因。
其一是恶意的包可能和某些允许的流量规则存在部分匹配， 而我们希望阻止， 而不是让这些包仅仅与 allow 规则部分匹配就允许它们进入。 其二是， 已经确信要阻止的包被拒绝这件事，
往往并不是我们需要关注的， 因此只要简单地予以阻止即可。
防火墙规则集中的每个部分的最后一条规则都是阻止并记录包，
这有助于为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确保系统对不希望的数据包不做回应。 无效的包应被丢弃和消失。 这样，攻击者便无法知道包是否到达了您的系统。 攻击者对系统了解的越少， 攻陷系统所需的时间也就越多。包含 log first 选项的规则只会记录它们第一次被触发时的包， 在例子中这个选项被用于记录 nmap OS 指纹探测 规则。 security/nmap 是攻击者常用的一种用于探测目标系统所用操作系统的工具。

如果您看到了 log first 规则的日志， 就应该用 ipfstat -hio 命令来看看那个规则被匹配的次数。
如果数目较大， 则表示系统正在受到洪水式攻击。

如果记录的包的端口号并不是您所知道的， 可以在 /etc/services 或 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， 了解木马使用的端口： 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 防火墙规则集。
直接使用这个规则集不会给您造成问题， 您所要做的只是注释掉那些您不需要 pass(允许通过) 的服务。

如果在日志中发现了希望 阻止 的记录， 只需在 inbound 小节中增加一条阻止规则集可。

您必须将每一个规则中的 dc0 替换为您系统上接入 Internet 的网络接口名称， 例如， 用户环境下的 PPP 应该是 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 网络地址转换(Network Address Translation) 的缩写。 对于那些熟悉 Linux® 的人来说，这个概念叫做 IP 伪装 (Masquerading)； NAT 和 IP 伪装是完全一样的概念。 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， 为什么需要这么做。 一般而言， ISP 会为非商业用户提供动态的 IP 地址。动态地址意味着每次登录到 ISP 都有可能得到不同的 IP 地址， 无论是采用电话拨号登录， 或使用 cable 以及 DSL 调制解调器的方式。 这个 IP 是您与 Internet 公网交互时使用的身份。

现在考虑家中有五台 PC 需要访问 Internet 的情形。 您可能需要向 ISP 为每一台 PC 所使用的独立的 Internet 账号付费， 并且拥有五根电话线。

有了 NAT， 您就只需要一个 ISP 账号， 然后将另外四台 PC 的网卡通过交换机连接起来， 并通过运行 FreeBSD 系统的那台机器作为网关连接出去。 NAT 会自动地将每一台 PC 在内网的 LAN IP 地址，在离开防火墙时转换为公网的 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， 有一些特殊的范围是保留供经过 NAT 的内网 LAN IP 地址使用的。 根据 RFC 1918，可以使用下面这些 IP 范围用于内网， 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 命令加载的。 默认情况下， NAT 规则会保存在 /etc/ipnat.rules 文件中。 请参见 ipnat(1) 了解更多的详情。

如果在 NAT 已经启动之后想要修改 NAT 规则， 可以修改保存 NAT 规则的那个文件， 然后在执行 ipnat 命令时加上 -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项。

要重新加载 NAT 规则， 可以使用类似下面的命令：

```
# ipnat -CF -f /etc/ipnat.rules
```

如果想要看看您系统上 NAT 的统计信息， 可以用下面的命令：

```
# ipnat -s
```

要列出当前的 NAT 表的映射关系， 使用下面的命令：

```
# ipnat -l
```

要显示详细的信息并显示与规则处理和当前的规则/表项：

```
# ipnat -v
```

31.5.16. IPNAT 规则

NAT 规则非常的灵活， 能够适应商业用户和家庭用户的各种不同的需求。

这里所介绍的规则语法已经被简化， 以适应非商用环境中的一般情况。 完整的规则语法描述， 请参考 ipnat(5) 联机手册中的介绍。

NAT 规则的写法与下面的例子类似：

```
map IF LAN_IP_RANGE -> PUBLIC_ADDRESS
```

关键词 map 出现在规则的最前面。

将 IF 替换为对外的网络接口名。

LAN_IP_RANGE 是内网中的客户机使用的地址范围。 通常情况下， 这应该是类似 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 这个特殊的关键字， 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 到达防火墙， 而目的地址是公网地址时， 它首先会通过 outbound 过滤规则。 接下来， NAT 会得到包， 并按自顶向下的顺序处理规则， 而第一个匹配的规则将生效。 NAT
接下来会根据包对应的接口名字和源 IP 地址检查所有的规则。 如果包和某个 NAT 规则匹配， 则会检查包的 (源 IP 地址， 例如， 内网的 IP 地址) 是否在 NAT 规则中箭头左侧指定的 IP 地址范围匹配。 如果匹配，
则包的原地址将被根据用 0/32 关键字指定的 IP 地址重写。 NAT 将向它的内部 NAT 表发送此地址， 这样，
当包从 Internet 公网中返回时， 就能够把地址映射回原先的内网 IP 地址，
并在随后使用过滤器规则来处理。

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。

使机器能够在不同的网络接口之间进行包的转发， 需要：

```
gateway_enable="YES"
```

每次开机时自动启动 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

对于在一个 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 上会导致资源不够的问题， 因为同一个端口可能在许多做了 NAT 的 LAN PC 上被多次使用， 并导致碰撞。 有两种方法来缓解这个难题。

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 时时不会发生变化的。 通过使用 portmap 关键字， 您可以要求 IPNAT 只使用指定范围内的端口地址。 比如说， 下面的规则将让 IPNAT 把源端口改为指定范围内的端口：

```
map dc0 192.168.1.0/24 -> 0/32 portmap tcp/udp 20000:60000
```

使用 auto 关键字可以让配置变得更简单一些， 它会要求 IPNAT 自动地检测可用的端口并使用：

```
map dc0 192.168.1.0/24 -> 0/32 portmap tcp/udp auto
```

31.5.19.2. 使用公网地址池

对很大的 LAN 而言， 总有一天会达到这样一个临界值， 此时的 LAN 地址已经多到了无法只用一个公网地址表现的程度。 如果有可用的一块公网 IP 地址， 则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 地址中挑选用于发包的地址， 并将其为这些包创建映射关系。

例如， 如果将下面这个把所有包都映射到同一公网 IP 地址的规则：

```
map dc0 192.168.1.0/24 -> 204.134.75.1
```

稍作修改， 就可以用子网掩码来表达 IP 地址范围：

```
map dc0 192.168.1.0/24 -> 204.134.75.0/255.255.255.0
```

或者用 CIDR 记法来指定的一组地址了：

```
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 分别放到 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应该被 NAT，
但必须有办法把进入的流量发到对应的局域网的 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， 您的 web 服务器的 LAN 地址是 10.0.10.25， 而您的唯一的公网 IP 地址是 20.20.20.5，
则可以编写这样的规则：

```
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， 也可以让 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 如今天这样为人所熟知之前就已经出现的恐龙， 那时，
研究机构和大学是通过租用的线路连到一起的， 而 FTP 则被用于在科研人员之间共享大文件。 那时，
数据的安全性并不是需要考虑的事情。 若干年之后， FTP 协议则被埋进了正在形成中的 Internet 骨干，
而它使用明文来交换用户名和口令的缺点， 并没有随着新出现的一些安全需求而得到改变。 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 进行了很好的阐述。

31.5.21.1. IPNAT 规则

IPNAT 提供了一个内建的 FTP 代理选项， 它可以在 NAT map 规则中指定。 它能够监视所有外发的 FTP
主动或被动模式的会话开始请求， 并动态地创建临时性的过滤器规则， 只打开用于数据通道的端口号。
这样， 就消除了 FTP 一般会给防火墙带来的， 需要大范围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 访问：

```
map dc0 10.0.10.0/29 -> 0/32 proxy port 21 ftp/tcp
```

这个规则能够处理来自网关的 FTP 访问：

```
map dc0 0.0.0.0/0 -> 0/32 proxy port 21 ftp/tcp
```

这个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 规则应该在普通的 map 规则之前出现。 所有的包会从最上面的第一个规则开始进行检查。
匹配的顺序是网卡名称， 内网源 IP 地址， 以及它是否是 FTP 包。 如果所有这些规则都匹配成功， 则 FTP
代理将建立一个临时的过滤规则， 以便让 FTP 会话的数据包能够正常出入， 同时对这些包进行 NAT。
所有的 LAN 数据包， 如果没有匹配第一条规则， 则会继续尝试匹配下面的规则， 并最终被 NAT。

31.5.21.2. IPNAT FTP 过滤规则

如果使用了 NAT FTP 代理， 则只需要为 FTP 创建一个规则。

如果不使用 FTP 代理， 就需要下面这三个规则：

```
# 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 发起的防火墙应用软件， 它由 FreeBSD
的志愿者成员编写和维护。 它使用了传统的无状态规则和规则编写方式，
以期达到简单状态逻辑所期望的目标。

标准的 FreeBSD 安装中， IPFW 所给出的规则集样例 (可以在 /etc/rc.firewall 和 /etc/rc.firewall6 中找到)
非常简单， 建议不要不加修改地直接使用。 该样例中没有使用状态过滤，
而该功能在大部分的配置中都是非常有用的， 因此这一节并不以系统自带的样例作为基础。

IPFW 的无状态规则语法， 是由一种提供复杂的选择能力的技术支持的，
这种技术远远超出了一般的防火墙安装人员的知识水平。 IPFW 是为满足专业用户，
以及掌握先进技术的电脑爱好者们对于高级的包选择需求而设计的。 要完全释放 IPFW
的规则所拥有的强大能力， 需要对不同的协议的细节有深入的了解，
并根据它们独特的包头信息来编写规则。 这一级别的详细阐述超出了这本手册的范围。

IPFW 由七个部分组成， 其主要组件是内核的防火墙过滤规则处理器， 及其集成的数据包记帐工具、
日志工具、 用以触发 NAT 工具的 divert (转发) 规则、 高级特殊用途工具、 dummynet 流量整形机制，
fwd rule 转发工具， 桥接工具， 以及 ipstealth 工具。 IPFW 支持 IPv4 和 IPv6。

31.6.1. 启用 IPFW

IPFW 是基本的 FreeBSD 安装的一部分， 以单独的可加载内核模块的形式提供。 如果在 rc.conf 中加入
firewall_enable="YES" 语句， 就会自动地加载对应的内核模块。 除非您打算使用由它提供的 NAT 功能，
一般情况下并不需要把 IPFW 编进 FreeBSD 的内核。

如果将 firewall_enable="YES" 加入到 rc.conf 中并重新启动系统， 则下列信息将在启动过程中，
以高亮的白色显示出来：

```
ipfw2 initialized, divert disabled, rule-based forwarding disabled, default to deny, logging
disabled
```

可加载内核模块在编译时加入了记录日志的能力。 要启用日志功能， 并配置详细日志记录的限制， 需要在
/etc/sysctl.conf 中加入一些配置。 这些设置将在重新启动之后生效：

```
net.inet.ip.fw.verbose=1
net.inet.ip.fw.verbose_limit=5
```

31.6.2. 内核选项

把下列选项在编译 FreeBSD 内核时就加入， 并不是启用 IPFW 所必需的， 除非您需要使用 NAT 功能。
这里只是将这些选项作为背景知识来介绍。

```
options   IPFIREWALL
```

这个选项将 IPFW 作为内核的一部分来启用。

```
options   IPFIREWALL_VERBOSE
```

这个选项将启用记录通过 IPFW 的匹配了包含 log 关键字规则的每一个包的功能。

```
options   IPFIREWALL_VERBOSE_LIMIT=5
```

以每项的方式， 限制通过 syslogd(8) 记录的包的个数。
如果在比较恶劣的环境下记录防火墙的活动可能会需要这个选项。 它能够避免潜在的针对 syslog
的洪水式拒绝服务攻击。

```
options   IPFIREWALL_DEFAULT_TO_ACCEPT
```

这个选项默认地允许所有的包通过防火墙， 如果您是第一次配置防火墙，
使用这个选项将是一个不错的主意。

```
options   IPDIVERT
```

这一选项启用 NAT 功能。

如果内核选项中没有加入 IPFIREWALL_DEFAULT_TO_ACCEPT，
而配置使用的规则集中也没有明确地指定允许连接进入的规则， 默认情况下，
发到本机和从本机发出的所有包都会被阻止。

31.6.3. /etc/rc.conf Options

启用防火墙：

```
firewall_enable="YES"
```

要选择由 FreeBSD 提供的几种防火墙类型中的一种来作为默认配置， 您需要阅读 /etc/rc.firewall 文件并选出合适的类型， 然后在 /etc/rc.conf 中加入类似下面的配置：

```
firewall_type="open"
```

您还可以指定下列配置规则之一：

- open - 允许所有流量通过。
- client - 只保护本机。
- simple - 保护整个网络。
- closed - 完全禁止除回环设备之外的全部 IP 流量。
- UNKNOWN - 禁止加载防火墙规则。
- filename - 到防火墙规则文件的绝对路径。

有两种加载自定义 ipfw 防火墙规则的方法。 其一是将变量 firewall_type 设为包含不带 ipfw(8) 命令行选项的 防火墙规则 文件的完整路径。 下面是一个简单的规则集例子：

```
add deny in
add deny out
```

除此之外， 也可以将 firewall_script 变量设为包含 ipfw 命令的可执行脚本， 这样这个脚本会在启动时自动执行。 与前面规则集文件等价的规则脚本如下：

ipfw 命令是在防火墙运行时， 用于在其内部规则表中手工逐条添加或删除防火墙规则的标准工具。 这一方法的问题在于， 一旦您的关闭计算机或停机， 则所有增加或删除或修改的规则也就丢掉了。 把所有的规则都写到一个文件中， 并在启动时使用这个文件来加载规则， 或一次大批量地替换防火墙规则， 那么推荐使用这里介绍的方法。

ipfw 的另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 的记账机制会为每一个规则动态地创建计数器， 用以记录与它们匹配的包的数量。 在测试规则的过程中， 列出规则及其计数器是了解它们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， 同时给出最后一次匹配的时间戳：

```
# ipfw -t list
```

列出所有的记账信息、 匹配规则的包的数量， 以及规则本身。 第一列是规则的编号， 随后是发出包匹配的数量， 进入包的匹配数量， 最后是规则本身。

```
# ipfw -a list
```

列出所有的动态规则和静态规则：

```
# ipfw -d list
```

同时显示已过期的动态规则：

```
# ipfw -d -e list
```

将计数器清零：

```
# ipfw zero
```

只把规则号为 NUM 的计数器清零：

```
# ipfw zero NUM
```

31.6.4. IPFW 规则集

规则集是指一组编写好的依据包的值决策允许通过或阻止 IPFW 规则。 包的双向交换组成了一个会话交互。
防火墙规则集会作用于来自于 Internet 公网的包以及由系统发出来回应这些包的数据包。 每一个 TCP/IP 服务 (例如 telnet, www, 邮件等等) 都由协议预先定义了其特权 (监听) 端口。
发到特定服务的包会从源地址使用非特权 (高编号) 端口发出， 并发到特定服务在目的地址的对应端口。
所有这些参数 (例如： 端口和地址） 都是可以为防火墙规则所利用的， 判别是否允许服务通过的标准。

当有数据包进入防火墙时， 会从规则集里的第一个规则开始进行比较， 并自顶向下地进行匹配。
当包与某个选择规则参数相匹配时， 将会执行规则所定义的动作， 并停止规则集搜索。 这种策略，
通常也被称作 "最先匹配者获胜" 的搜索方法。 如果没有任何与包相匹配的规则， 那么它就会根据强制的 IPFW 默认规则， 也就是 65535 号规则截获。 一般情况下这个规则是阻止包， 而且不给出任何回应。

如果规则定义的动作是 count、 skipto 或 tee 规则的话， 搜索会继续。

这里所介绍的规则， 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 选项的规则。 这是编写明示允许防火墙规则集所需的基本框架。

在操作防火墙规则时应谨慎行事， 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经经过了简化， 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说明， 请参见 ipfw(8) 联机手册。

规则是由关键字组成的： 这些关键字必须以特定的顺序从左到右书写。 下面的介绍中，
关键字使用粗体表示。 某些关键字还包括了子选项， 这些子选项本身可能也是关键字，
有些还可以包含更多的子选项。

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

每一个新的规则都应以 add 作为前缀， 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范围在 1 到 65535 之间的规则编号相关联。

31.6.4.1.3. ACTION

每一个规则可以与下列的动作之一相关联，
所指定的动作将在进入的数据包与规则所指定的选择标准相匹配时执行。

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， 并停止继续搜索规则。

check-state

根据动态规则表检查数据包。 如果匹配， 则执行规则所指定的动作， 亦即生成动态规则； 否则，
转移到下一个规则。 check-state 规则没有选择标准。 如果规则集中没有 check-state 规则， 则会在第一个 keep-state 或 limit 规则处， 对动态规则表实施检查。

deny | drop

这两个关键字都表示丢弃匹配规则的包。 同时， 停止继续搜索规则。

31.6.4.1.4. LOGGING

log or logamount

当数据包与带 log 关键字的规则匹配时， 将通过名为 SECURITY 的 facility 来把消息记录到 syslogd(8)。
只有在记录的次数没有超过 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。
如果将这两种限制值之一指定为零， 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计数或包计数清零来重新启用日志， 请参见 ipfw reset log 命令来了解细节。

日志是在所有其他匹配条件都验证成功之后， 在针对包实施最终动作 (accept, deny) 之前进行的。 您可以自行决定哪些规则应启用日志。

31.6.4.1.5. SELECTION

这一节所介绍的关键字主要用来描述检查包的哪些属性， 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 中所定义的协议。 这个值定义的是匹配的协议， 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， 用于匹配您的 FreeBSD 系统上所有网络接口上所配置的 IP 地址， 它可以用于表达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 带点的 IP 地址/掩码长度 (CIDR 记法)， 或者一个带点的 IP 地址的形式来指定。 这是编写规则时所必需的。 使用 net-mgmt/ipcalc port 可以用来简化计算。 关于这个工具的更多信息， 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协议，

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，
这样会比使用数字指定端口号直观一些。

in | out

相应地， 匹配进入和发出的包。 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进出的包。 这里的 via
关键字将使得接口名称成为匹配过程的一部分。

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。 在发生匹配时， 防火墙将创建一个动态规则， 其默认行为是，
匹配使用同一协议的、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时， 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. 状态规则选项

有状态过滤将网络流量当作一种双向的包交换来处理。 它提供了一种额外的检查能力，
用以检测会话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进行会话。
如果包与这些规则不符， 则将自动地拒绝它们。

check-state 用来识别在 IPFW 规则集中的包是否符合动态规则机制的规则。 如果匹配， 则允许包通过，
此时防火墙将创建一个新的动态规则来匹配双向交换中的下一个包。 如果不匹配，
则将继续尝试规则集中的下一个规则。

动态规则机制在 SYN-flood 攻击下是脆弱的， 因为这种情况会产生大量的动态规则， 从而耗尽资源。
为了抵抗这种攻击， 从 FreeBSD 中加入了一个叫做 limit 的新选项。
这个选项可以用来限制符合规则的会话允许的并发连接数。 如果动态规则表中的规则数超过 limit
的限制数量， 则包将被丢弃。

31.6.4.3. 记录防火墙消息

记录日志的好处是显而易见的： 它提供了在事后检查所发生的状况的方法， 例如哪些包被丢弃了，
这些包的来源和目的地， 从而为您提供找到攻击者所需的证据。

即使启用了日志机制， IPFW 也不会自行生成任何规则的日志。
防火墙管理员需要指定规则集中的哪些规则应该记录日志， 并在这些规则上增加 log 动作。 一般来说，
只有 deny 规则应记录日志， 例如对于进入的 ICMP ping 的 deny 规则。 另外， 复制 "默认的 ipfw 终极
deny 规则"， 并加入 log 动作来作为您的规则集的最后一条规则也是很常见的用法。 这样，
您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， 则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 攻击最为老套的手法之一。 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， 因此有过多的日志信息是很让人恼火的事情。

IPFIREWALL_VERBOSE_LIMIT=5 内核选项将限制同一个规则发到系统日志程序 syslogd(8)
的连续消息的数量。 当内核启用了这个选项时， 某一特定规则所产生的连续消息的数量将封顶为这个数字。
一般来说， 没有办法从连续 200 条一模一样的日志信息中获取更多有用的信息。 举例来说，
如果同一个规则产生了 5 次消息并被记录到 syslogd， 余下的相同的消息将被计数， 并像下面这样发给

syslogd：

```
last message repeated 45 times
```

所有记录的数据包包消息， 默认情况下会最终写到 /var/log/security 文件中， 后者在 /etc/syslog.conf 文件里进行了定义。

31.6.4.4. 编写规则脚本

绝大多数有经验的 IPFW 用户会创建一个包含规则的文件， 并且， 按能够以脚本形式运行的方式来书写。 这样做最大的一个好处是， 可以大批量地刷新防火墙规则， 而无须重新启动系统就能够激活它们。 这种方法在测试新规则时会非常方便， 因为同一过程在需要时可以多次执行。 作为脚本， 您可以使用符号替换来撰写那些经常需要使用的值， 并用同一个符号在多个规则中反复地表达它。 下面将给出一个例子。

这个脚本使用的语法同 sh(1)、 csh(1) 以及 tcsh(1) 脚本兼容。 符号替换字段使用美元符号 $ 作为前缀。 符号字段本身并不使用 $ 前缀。 符号替换字段的值必须使用 "双引号" 括起来。

可以使用类似下面的规则文件：

```
############### 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 ############
```

这就是所要做的全部事情了。 例子中的规则并不重要， 它们主要是用来表示如何使用符号替换。

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载规则。

```
# sh /etc/ipfw.rules
```

/etc/ipfw.rules 这个文件可以放到任何位置， 也可以命名为随便什么别的名字。

也可以手工执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. 带状态规则集

以下的这组非-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， 至少需要有两个网络接口， 并且其上必须配置规则，
以便让防火墙正常工作。

所有类 UNIX® 操作系统， 也包括 FreeBSD， 都设计为允许使用网络接口 lo0 和 IP 地址 127.0.0.1
来完成操作系统内部的通讯。 防火墙必须包含一组规则， 使这些数据包能够无障碍地收发。

接入 Internet 公网的那个网络接口上， 应该配置授权和访问控制， 来限制对外的访问， 以及来自 Internet
公网的访问。 这个接口很可能是您的用户态 PPP 接口， 例如 tun0， 或者您接在 DSL 或电缆 modem
上的网卡。

如果有至少一个网卡接入了防火墙后的内网 LAN， 则必须为这些接口配置规则，
以便让这些接口之间的包能够顺畅地通过。

所有的规则应被组织为三个部分， 所有应无阻碍地通过的规则， 公网的发出规则， 以及公网的接收规则。

公网接口相关的规则的顺序， 应该是最经常用到的放在尽可能靠前的位置， 而最后一个规则，
则应该是阻止那个接口在那一方向上的包。

发出部分的规则只包含一些 allow 规则， 允许选定的那些唯一区分协议的端口号所指定的协议通过，
以允许访问 Internet 公网上的这些服务。 所有的规则中都指定了 proto、 port、 in/out、 via 以及 keep
state 这些选项。 proto tcp 规则同时指定 setup 选项， 来区分开始协议会话的包， 以触发将包放入 keep
state 规则表中的动作。

接收部分则首先阻止所有不希望的包， 这样做有两个不同的原因。
其一是恶意的包可能和某些允许的流量规则存在部分匹配， 而我们希望阻止， 而不是让这些包仅仅与 allow
规则部分匹配就允许它们进入。 其二是， 已经确信要阻止的包被拒绝这件事，
往往并不是我们需要关注的， 因此只要简单地予以阻止即可。
防火墙规则集中的每个部分的最后一条规则都是阻止并记录包，
这有助于为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确保系统对不希望的数据包不做回应。 无效的包应被丢弃和消失。 这样，
攻击者便无法知道包是否到达了您的系统。 攻击者对系统了解的越少， 其攻击的难度也就越大。
如果不知道端口号， 可以查阅 /etc/services/ 或到
http://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers 并查找一下端口号，
以了解其用途。 另外， 您也可以在这个网页上了解常见木马所使用的端口：
http://www.sans.org/security-resources/idfaq/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。 使用它作为您的规则集不会有什么问题。
只需把那些不需要的服务对应的 pass 规则注释掉就可以了。 如果您在日志中看到消息，
而且不想再看到它们， 只需在接收部分增加一个一个 deny 规则。 您可能需要把 dc0
改为接入公网的接口的名字。 对于使用用户态 PPP 的用户而言， 应该是 tun0。

这些规则遵循一定的模式。

- 所有请求 Internet 公网上服务的会话开始包， 都使用了 keep-state。
- 所有来自 Internet 的授权服务请求， 都采用了 limit 选项来防止洪水式攻击。

- 所有的规则都使用了 in 或者 out 来说明方向。
- 所有的规则都使用了 via接口名 来指定应该匹配通过哪一个接口的包。

这些规则都应放到 /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 和带状态规则集的例子

要使用 IPFW 的 NAT 功能， 还需要进行一些额外的配置。 除了其他 IPFIREWALL 语句之外，
还需要在内核编译配置中加上 option IPDIVERT 语句。

在 /etc/rc.conf 中， 除了普通的 IPFW 配置之外， 还需要加入：

```
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
```

将带状态规则与 divert natd 规则 (网络地址转换) 会使规则集的编写变得非常复杂。 check-state 的位置，
以及 divert natd 规则将变得非常关键。 这样一来， 就不再有简单的顺序处理逻辑流程了。
提供了一种新的动作类型， 称为 skipto。 要使用 skipto 命令， 就必须给每一个规则进行编号， 以确定
skipto 规则号是您希望跳转到的位置。

下面给出了一些未加注释的例子来说明如何编写这样的规则， 用以帮助您理解包处理规则集的处理顺序。

处理流程从规则文件最上边的第一个规则开始处理， 并自顶向下地尝试每一个规则， 直到找到匹配的规则，
且数据包从防火墙中放出为止。 请注意规则号 100 101， 450， 500， 以及 510 的位置非常重要。

这些规则控制发出和接收的包的地址转换过程， 这样它们在 keep-state
动态表中的对应项中就能够与内网的 LAN IP 地址关联。 另一个需要注意的是， 所有的 allow 和 deny
规则都指定了包的方向 (也就是 outbound 或 inbound) 以及网络接口。 最后，
请注意所有发出的会话请求都会请求 skipto rule 500 以完成网络地址转换。

下面以 LAN 用户使用 web 浏览器访问一个 web 页面为例。 Web 页面使用 80 来完成通讯。
当包进入防火墙时， 规则 100 并不匹配， 因为它是发出而不是收到的包。 它能够通过规则 101，
因为这是第一个包， 因而它还没有进入动态状态保持表。 包最终到达规则 125， 并匹配该规则。 最终，
它会通过接入 Internet 公网的网卡发出。 这之前， 包的源地址仍然是内网 IP 地址。 一旦匹配这个规则，
就会触发两个动作。 keep-state 选项会把这个规则发到 keep-state 动态规则表中， 并执行所指定的动作。
动作是发到规则表中的信息的一部分。 在这个例子中， 这个动作是 skipto rule 500。 规则 500 NAT 包的 IP
地址， 并将其发出。 请务必牢记， 这一步非常重要。 接下来， 数据包将到达目的地，
之后返回并从规则集的第一条规则开始处理。 这一次， 它将与规则 100 匹配， 其目的 IP
地址将被映射回对应的内网 LAN IP 地址。 其后， 它会被 check-state 规则处理，
进而在暨存会话表中找到对应项， 并发到 LAN。 数据包接下来发到了内网 LAN PC 上，
而后者则会发送从远程服务器请求下一段数据的新数据包。 这个包会再次由 check-state 规则检查，
并找到发出的表项， 并执行其关联的动作， 即 skipto 500。 包跳转到规则 500 并被 NAT 后发出。

在接收一侧， 已经存在的会话的数据包会被 check-state 规则自动地处理， 并转到 divert nat 规则。
我们需要解决的问题是， 阻止所有的坏数据包， 而只允许授权的服务。 例如在防火墙上运行了 Apache
服务， 而我们希望人们在访问 Internet 公网的同时， 也能够访问本地的 web 站点。
新的接入开始请求包将匹配规则 100， 而 IP 地址则为防火墙所在的服务器而映射到了 LAN IP。 此后，
包会匹配所有我们希望检查的那些令人生厌的东西， 并最终匹配规则 425。 一旦发生匹配，
会发生两件事。 数据包会被发到 keep-state 动态表， 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。 这一做法能够挫败针对指定端口上服务的 DoS 攻击。 动作同时指定了
allow 包应被发到 LAN 上。 包返回时， check-state 规则会识别出包属于某一已经存在的会话交互，
并直接把它发到规则 500 做 NAT， 并发到发出接口。

示范规则集 #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  ###################
```

下面的这个规则集基本上和上面一样，但使用了易于读懂的编写方式，并给出了相当多的注解，以帮助经验较少的 IPFW 规则编写者更好地理解这些规则到底在做什么。

示范规则集 #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. 高级网络

32.1. 概述

本章将就一系列与网络有关的高级话题进行讨论。

读完这章，您将了解：

- 关于网关和路由的基础知识。
- 如何配置 IEEE® 802.11 和 Bluetooth® 设备。
- 如何用 FreeBSD 做网桥。
- 如何为无盘机上配置网络启动。
- 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 连接两台计算机。
- 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 (共用地址冗余协议)

在读这章之前， 您应：

- 理解 /etc/rc 脚本的基本知识。
- 熟悉基本的网络术语。
- 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. 网关和路由

要让网络上的两台计算机能够相互通讯， 就必须有一种能够描述如何从一台计算机到另一台计算机的机制，
这一机制称作 路由选择(routing)。 "路由项" 是一对预先定义的地址： "目的地(destination)" 和
"网关(gateway)"。 这个地址对所表达的意义是， 通过 网关 能够完成与 目的地 的通信。
有三种类型的目的地址： 单个主机、 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"， 有关默认路由的内容， 将在稍后的章节中进行讨论。 网关也有三种类型： 单个主机，
网络接口 (也叫 "链路 (links)") 和以太网硬件地址 (MAC 地址)。

32.2.1. 实例

为了说明路由选择的各个部分， 首先来看看下面的例子。 这是 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
```

头两行给出了当前配置中的默认路由 (将在 下一节 中进行介绍) 和 localhost (本机) 路由。

这里的路由表中给出的用于 localhost 的接口 (Netif 列) 是 lo0， 也就是大家熟知的 "回环设备"。
它表示所有以此为 "目的地" 的通信都留在本机， 而不通过 LAN 发出， 因为这些流量最终会回到起点。

接着出现的是以 0:e0: 开头的地址。这些是以太网硬件地址，也称为 MAC 地址。 FreeBSD
会自动识别在同一个以太网中的任何主机 (如 test0)， 并为其新增一个路由， 并通过那个以太网接口 - ed0
直接与它通讯 (译者注：那台主机)。与这类路由表相关的也有一个超时项 (
Expire列)，当我们在指定时间内没有收到从那个主机发来的信息，
这项就派上用场了。这种情况下，到这个主机的路由就会被自动删除。

这些主机被使用一种叫做RIP(路由信息协议—Routing Information
Protocol)的机制所识别，这种机制利用基于"最短路径选择 (shortest path
determination)"的办法计算出到本地主机的路由。

FreeBSD 也会为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
是这个子网相联的域名)。 名称 link#1 代表主机上的第一块以太网卡。
您会发现，对于它们没有指定另外的接口。

这两个组(本地网络主机和本地子网)的路由是由守护进程 routed 自动配置的。如果它没有运行，
那就只有被静态定义 (例如，明确输入的) 的路由才存在了。

host1 行代表我们的主机，它通过以太网地址来识别。 因为我们是发送端，FreeBSD知道使用回环接口 (
lo0) 而不是通过以太网接口来进行发送。

两个 host2 行是我们使用 ifconfig(8) 别名 (请看关于以太网的那部分就会知道我们为什么这么做)
时产生的一个实例。在 lo0 接口之后的 ⇒ 符号表明我们不仅使用了回环
(因为这个地址也涉及了本地主机)，而且明确指出它是个别名。
这类路由只有在支持别名的主机上才能显现出来。 所有本地网上的其它的主机对于这类路由只会简单拥有
link#1。

最后一行 (目标子网224) 用于处理多播——它会覆盖到其它的区域。

最后，每个路由的不同属性可以在 Flags 列中看到。下边是个关于这些标志和它们的含义的一个简表：

U	Up: 路由处于活动状态。
H	Host: 路由目标是单个主机。
G	Gateway: 所有发到目的地的网络传到这一远程系统上， 并由它决定最后发到哪里。
S	Static: 这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， 通过这个路由我们可以连接上这些机子。 这种类型的路由通常用于本地网络。
W	WasCloned: 指明一个路由——它是基于本地区域网络 (克隆) 路由自动配置的。
L	Link: 路由涉及到了以太网硬件。

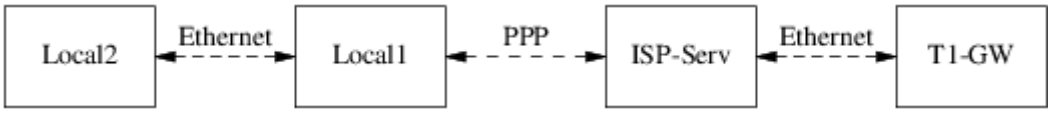
32.2.2. 默认路由

当本地系统需要与远程主机建立连接时，它会检查路由表以决定是否有已知的路径存在。
如果远程主机属于一个我们已知如何到达 (克隆的路由)
的子网内，那么系统会检查看沿着那个接口是否能够连接。

如果所有已知路径都失败，系统还有最后一个选择："默认"路由。这个路由是特殊类型的网关路由
(通常只有一个存在于系统里)，并且总是在标志栏使用一个 c来进行标识。对于本地区域网络里的主机，
这个网关被设置到任何与外界有直接连接的机子里 (无论是通过 PPP、DSL、cable modem、T1
或其它的网络接口连接)。

如果您正为某台本身就做为网关连接外界的机子配置默认路由的话，
那么该默认路由应该是您的"互联网服务商 (ISP)"那方的网关机子。

让我们来看一个关于默认路由的例子。这是个很普遍的配置：



主机 Local1 和 Local2 在您那边。Local1 通过 PPP 拨号连接到了 ISP。这个 PPP
服务器通过一个局域网连接到另一台网关机子——它又通过一个外部接口连接到 ISP 提供的互联网上。

您的每一台机子的默认路由应该是：

Host	Default Gateway	Interface
Local2	Local1	Ethernet
Local1	T1-GW	PPP

一个常见的问题是"我们为什么 (或怎样) 能将 T1-GW 设置成为 Local1 默认网关，而不是它所连接 ISP
服务器？"

记住，因为 PPP 接口使用的一个地址是在 ISP 的局域网里的，用于您那边的连接，对于 ISP
的局域网里的其它机子，其路由会自动产生。因此，您就已经知道了如何到达机子 T1-GW，
那么也就没必要中间那一步了——发送通信给 ISP 服务器。

通常使用地址 X.X.X.1 做为一个局域网的网关。因此 (使用相同的例子)，如果您本地的 C 类地址空间是
10.20.30，而您的 ISP 使用的是 10.9.9，那么默认路由表将是：

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)

您可以很轻易地通过 /etc/rc.conf 文件设定默认路由。在我们的实例里，在主机 Local2 里，我们在文件
/etc/rc.conf 里增加了下边内容：

```
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
```

要了解关于如何手工维护网络路由表的进一步细节，请参考 route(8) 联机手册。

32.2.3. 重宿主机(Dual Homed Hosts)

还有一种其它的类型的配置是我们要提及的， 这就是一个主机处于两个不同的网络。技术上，任何作为网关(上边的实例中，使用了 PPP 连接) 的机子就算作是重宿主机。
但这个词实际上仅用来指那种处于两个局域网之中的机子。

有一种情形，一台机子有两个网卡， 对于各个子网都有各自的一个地址。另一种情况，
这台机子仅有一张网卡，但使用 ifconfig(8) 做了别名。如果有两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，但逻辑上分成了两个独立的子网， 就使用后者。

每种情况都要设置路由表以便两子网都知道这台主机是到其它子网的网关——入站路由 (inbound route)。将一台主机配置成两个子网间的路由器，
这种配置经常在我们需要实现单向或双向的包过滤或防火墙时被用到。

如果想让主机在两个接口间转发数据包，您需要激活 FreeBSD
的这项功能。至于怎么做，请看下一部分了解更多。

32.2.4. 建立路由器

网络路由器只是一个将数据包从一个接口转发到另一个接口的系统。 互联网标准和良好的工程实践阻止了 FreeBSD 计划在 FreeBSD 中把它置成默认值。您在可以在 rc.conf(5) 中改变下列变量的值为 YES，使这个功能生效：

```
gateway_enable="YES"       # Set to YES if this host will be a gateway
```

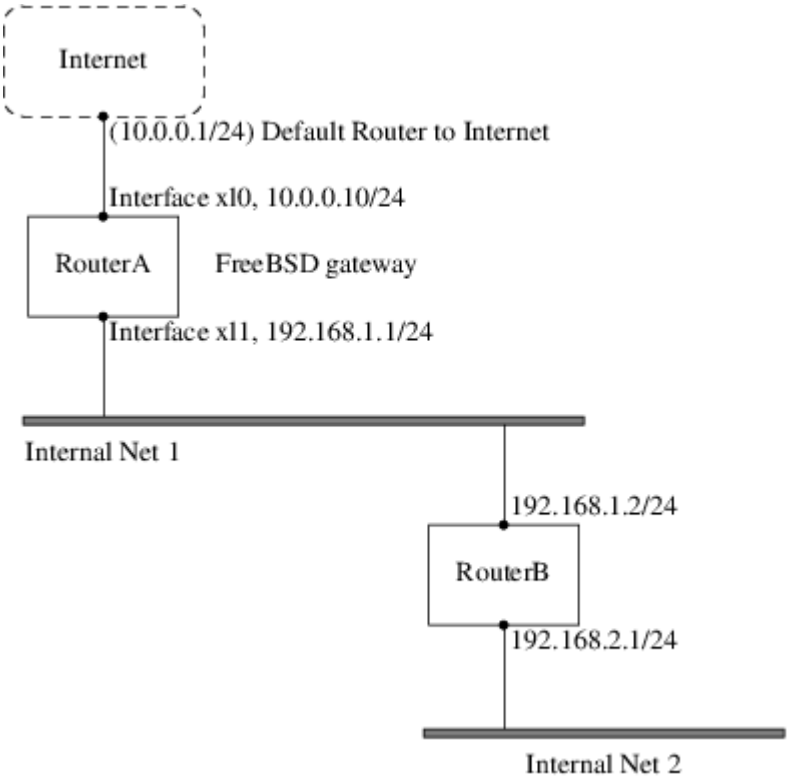
这个选项会把sysctl(8) 变量——net.inet.ip.forwarding 设置成 1。如果您要临时地停止路由，
您可以把它重设为 0。

新的路由器需要有路由才知道将数据传向何处。 如果网络够简单，您可以使用静态路由。FreeBSD 也自带一个标准的BSD路由选择守护进程 routed(8)， 称之为 RIP (version 1和 version 2) 和 IRDP。对 BGP v4，OSPF v2 和其它复杂路由选择协议的支持可以从 net/zebra 包中得到。 像 GateD® 一样的商业产品也提供了更复杂的网络路由解决方案。

32.2.5. 设置静态路由

32.2.5.1. 手动配置

假设如下这样一个网络：



在这里，RouterA 是我们的 FreeBSD 机子，它充当连接到互联网其它部分的路由器的角色。默认路由设置为10.0.0.1， 它就允许与外界连接。我们假定已经正确配置了 RouterB，并且知道如何连接到想去的任何地方。 (在这个图里很简单。只须在 RouterB 上增加默认路由，使用 192.168.1.1 做为网关。)

如果我们查看一下RouterA的路由表， 我们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 是不能到达我们的内网——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增加这条路由。以下的命令会把 Internal Net 2 网络加入到 RouterA 的路由表中，使用192.168.1.2 做为下一个跳跃：

```
# route add -net 192.168.2.0/24 192.168.1.2
```

现在 RouterA 就可以到达 192.168.2.0/24 网络上的任何主机了。

32.2.5.2. 永久配置

上面的实例对于运行着的系统来说配置静态路由是相当不错了。 只是，有一个问题——如果您重启您的 FreeBSD 机子，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您的 /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。每一串表示一个路由名字。
在上面的例子中我们中有一个串在 static_routes 里。这个字符串中 internalnet2。
然后我们新增一个配置变量 route_internalnet2， 这里我们把所有传给 route(8)命令的参数拿了过来。
在上面的实例中的我使用的命令是：

```
# route add -net 192.168.2.0/24 192.168.1.2
```

因此，我们需要的是 "-net 192.168.2.0/24 192.168.1.2"。

前边已经提到， 可以把多个静态路由的名称， 放到 static_routes 里边。 接着我们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增加 192.168.0.0/24 和 192.168.1.0/24
之间静态路由的例子：

```
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. 路由传播

我们已经讨论了如何定义通向外界的路由， 但未谈及外界是如何找到我们的。

我们已经知道可以设置路由表， 这样任何指向特定地址空间 (在我们的例子中是一个 C 类子网)
的数据都会被送往网络上特定的主机， 然后由这台主机向地址空间内部转发数据。

当您得到一个分配给您的网络的地址空间时， ISP(网络服务商)会设置它们的路由表，
这样指向您子网的数据就会通过 PPP 连接下传到您的网络。
但是其它跨越国界的网络是如何知道将数据传给您的 ISP 的呢?

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，
并说明它们连接到互联网骨干(Internet backbone)的点。 "骨干(Backbone)"
指的是负责全世界和跨国的传输的主要干线。 每一台骨干主机(backbone
machine)有一份主要表集的副本， 它将发送给特定网络的数据导向相应的骨干载体上(backbone
carrier)， 从结点往下遍历服务提供商链，直到数据到达您的网络。

服务提供商的任务是向骨干网络广播，以声明它们就是通向您的网点的连接结点
(以及进入的路径)。这就是路由传播。

32.2.7. 问题解答

有时候，路由传播会有一个问题，一些网络无法与您连接。
或许能帮您找出路由是在哪里中断的最有用的命令就是 traceroute(8)了。当您无法与远程主机连接时，
这个命令一样有用(例如 ping(8) 失败)。

traceroute(8) 命令将以您想连接的主机的名字作为参数执行。
不管是到达了目标，还是因为没有连接而终止， 它都会显示所经过的所有网关主机。

想了解更多的信息，查看 traceroute(8) 的手册。

32.2.8. 多播路由

FreeBSD 一开始就支持多播应用软件和多播路由选择。 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编译入内核：

```
options MROUTING
```

另外，多播路由守护进程——mrouted(8) 必须通过 /etc/mrouted.conf 配置来开启通道和 DVMRP。更多关于多播路由配置的信息可以在 mrouted(8) 的手册里找到。

多播路由服务 mrouted(8) 实现了 DVMRP 多播路由协议， 在许多采用多播的场合，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 中找到。

32.3. 无线网络

32.3.1. 无线网络基础

绝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 频段的无线电波广播的站点 (不过， 随所处地域的不同， 或者为了能够更好地进行通讯， 具体的频率会在 2.3GHz 和 4.9GHz 的范围内变化)。

802.11 网络有两种组织方式： 在 infrastructure 模式 中， 一个通讯站作为主站， 其他通讯站都与其关联；这种网络称为 BSS， 而主站则成为无线访问点 (AP)。 在 BSS 中， 所有的通讯都是通过 AP 来完成的；即使通讯站之间要相互通讯， 也必须将消息发给 AP。 在第二种形式的网络中， 并不存在主站，通讯站之间是直接通讯的。 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 标准所定义的协议。这些标准定义了采用的操作频率、 包括分帧和传输速率 (通讯过程中可以使用不同的速率) 在内的 MAC 层特性等。 稍后的 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传输速率。 其后定义的 802.11g 标准启用了在 2.4GHz 上如何使用 802.11a 信号和传输机制， 以提供对较早的 802.11b 网络的向前兼容。

802.11 网络中采用的各类底层传输机制提供了不同类型的安全机制。 最初的 802.11 标准定义了一种称为 WEP 的简单安全协议。 这个协议采用固定的预发布密钥， 并使用 RC4 加密算法来对在网络上传输的数据进行编码。 全部通讯站都必须采用同样的固定密钥才能通讯。这一格局已经被证明很容易被攻破， 因此目前已经很少使用了，采用这种方法只能让那些接入网络的用户迅速断开。 最新的安全实践是由 IEEE® 802.11i 标准给出的，它定义了新的加密算法， 并通过一种附加的协议来让通讯站向无线访问点验证身份，并交换用于进行数据通讯的密钥。 更进一步， 用于加密的密钥会定期地刷新，而且有机制能够监测入侵的尝试 (并阻止这种尝试)。 无线网络中另一种常用的安全协议标准是 WPA。这是在 802.11i 之前由业界组织定义的一种过渡性标准。 WPA 定义了在 802.11i 中所规定的要求的子集，并被设计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 则不但允许使用 TKIP， 而且还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这种算法时所需的计算复杂性太高。)

除了前面介绍的那些协议标准之外， 还有一种需要介绍的标准是 802.11e。 它定义了用于在 802.11 网络上运行多媒体应用， 如视频流和使用 IP 传送的语音 (VoIP) 的协议。 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)， 它也是由业界组织定义的 802.11e 的子集，以便能够在旧式硬件中使用多媒体应用。 关于 802.11e 与 WME/WMM 之间的另一项重要区别是，前者允许对流量通过服务品质 (QoS) 协议和增强媒体访问协议来安排优先级。 对于这些协议的正确实现，能够实现高速突发数据和流量分级。

FreeBSD 支持采用 802.11a, 802.11b 和 802.11g 的网络。 类似地， 它也支持 WPA 和 802.11i 安全协议 (与 11a、 11b 和 11g 配合)， 而 WME/WMM 所需要的 QoS 和流量分级， 则在部分无线设备上提供了支持。

32.3.2. 基本安装

32.3.2.1. 内核配置

要使用无线网络， 您需要一块无线网卡， 并适当地配置内核令其提供无线网络支持。
后者被分成了多个模块， 因此您只需配置使用您所需要的软件就可以了。

首先您需要的是一个无线设备。 最为常用的一种无线配件是 Atheros 生产的。 这些设备由 ath(4)
驱动程序提供支持， 您需要把下面的配置加入到 /boot/loader.conf 文件中：

```
if_ath_load="YES"
```

Atheros 驱动分为三个部分： 驱动部分 (ath(4))、 用于处理芯片专有功能的支持层 (ath_hal(4))，
以及一组用以选择传输帧速率的算法 (ath_rate_sample here)。 当以模块方式加载这一支持时，
所需的其它模块会自动加载。 如果您使用的不是 Atheros 设备， 则应选择对应的模块； 例如：

```
if_wi_load="YES"
```

表示使用基于 Intersil Prism 产品的无线设备 (wi(4) 驱动)。

在这篇文档余下的部分中， 我们将以 ath(4) 卡来进行示范， 如果要套用这些配置的话，
可能需要根据您实际的配置情况来修改示例中的设备名称。 在 FreeBSD
兼容硬件说明中提供了目前可用的无线网络驱动， 以及兼容硬件的列表。
针对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。 如果您的无线设备没有与之对应的 FreeBSD 专用驱动程序， 也可以尝试使用
NDIS 驱动封装机制来直接使用 Windows® 驱动。

对于 FreeBSD 7.X， 在配置好设备驱动之后， 您还需要引入驱动程序所需要的 802.11 网络支持。 对于
ath(4) 驱动而言， 至少需要 wlan(4) wlan_scan_ap 和 wlan_scan_sta 模块； wlan(4)
模块会自动随无线设备驱动一同加载， 剩下的模块必须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) 驱动的基础组件， 并会随适配器驱动一起动态加载。

除此之外， 您还需要提供您希望使用的安全协议所需的加密支持模块。 这些模块是设计来让 wlan(4)
模块根据需要自动加载的， 但目前还必须手工进行配置。 您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) 这两个驱动都只有在您希望采用 WPA
和/或 802.11i 安全协议时才需要。 如果您的网络不采用加密， 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"
```

通过系统引导配置文件 (也就是 /boot/loader.conf) 中的这些信息生效， 您必须重新启动运行 FreeBSD
的计算机。 如果不想立刻重新启动， 也可以使用 kldload(8) 来手工加载。

如果不想加载模块， 也可以将这些驱动编译到内核中，
方法是在内核的编译配置文件中加入下面的配置：

```
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 时， 还需要配置下面这两行； FreeBSD 的其他版本不需要它们。

```
device wlan_scan_ap     # 802.11 AP mode scanning
device wlan_scan_sta    # 802.11 STA mode scanning
```

将这些信息写到内核编译配置文件中之后， 您需要重新编译内核， 并重新启动运行 FreeBSD 的计算机。

在系统启动之后， 您会在引导时给出的信息中， 找到类似下面这样的关于无线设备的信息：

```
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。 在这种模式中，
有一系列无线访问点接入了有线网络。 每个无线网都会有自己的名字， 这个名字称作网络的 SSID。
无线客户端都通过无线访问点来完成接入。

32.3.3.1. FreeBSD 客户机

32.3.3.1.1. 如何查找无线访问点

您可以通过使用 ifconfig 命令来扫描网络。
由于系统需要在操作过程中切换不同的无线频率并探测可用的无线访问点，
这种请求可能需要数分钟才能完成。 只有超级用户才能启动这种扫描：

```
# 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， 必须将网络接口设为 up。 后续的扫描请求就不需要再将网络接口设为 up 了。

在 FreeBSD 7.X 中， 会直接适配器设备， 例如 ath0， 而不是 wlan0 设备。
因此您需要把前面的命令行改为：

```
# ifconfig ath0 up scan
```

在这份文档余下的部分中， 您也需要注意 FreeBSD 7.X 上的这些差异，
并对命令行示例进行类似的改动。

扫描会列出所请求到的所有 BSS/IBSS 网络列表。 除了网络的名字 SSID 之外， 我们还会看到 BSSID
即无线访问点的 MAC 地址。 而 CAPS 字段则给出了网络类型及其提供的功能， 其中包括：

表 13. 通讯站功能代码

功能代码	含义
E	Extended Service Set (ESS)。 表示通讯站是 infrastructure 网络 (相对于 IBSS/ad-hoc 网络) 的成员。
I	IBSS/ad-hoc 网络。 表示通讯站是 ad-hoc 网络 (相对于 ESS 网络) 的成员。
P	私密。 在 BSS 中交换的全部数据帧均需保证数据保密性。 这表示 BSS 需要通讯站使用加密算法， 例如 WEP、 TKIP 或 AES-CCMP 来加密/解密与其他通讯站交换的数据帧。
S	短前导码 (Short Preamble)。 表示网络采用的是短前导码 (由 802.11b High Rate/DSSS PHY 定义， 短前导码采用 56-位 同步字段， 而不是在长前导码模式中所采用的 128-位 字段)。
s	短碰撞槽时间 (Short slot time)。 表示由于不存在旧式 (802.11b) 通讯站， 802.11g 网络正使用短碰撞槽时间。

要显示目前已知的网络， 可以使用下面的命令：

```
# ifconfig wlan0 list scan
```

这些信息可能会由无线适配器自动更新， 也可使用 scan 手动更新。 快取缓存中的旧数据会自动删除，
因此除非进行更多扫描， 这个列表会逐渐缩小。

32.3.3.1.2. 基本配置

在这一节中我们将展示一个简单的例子来介绍如何让无线网络适配器在 FreeBSD 中以不加密的方式工作。
在您熟悉了这些概念之后， 我们强烈建议您在实际的使用中采用 WPA 来配置网络。

配置无线网络的过程可分为三个基本步骤： 选择无线访问点、 验证您的通讯站身份， 以及配置 IP 地址。
下面的几节中将分步骤地介绍它们。

32.3.3.1.2.1. 选择无线访问点

多数时候让系统以内建的探测方式选择无线访问点就可以了。 这是在您将网络接口置为 up 或在
/etc/rc.conf 中配置 IP 地址时的默认方式， 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"
```

如果存在多个无线访问点， 而您希望从中选择具体的一个， 则可以通过指定 SSID 来实现：

```
wlans_ath0="wlan0"
ifconfig_wlan0="ssid your_ssid_here DHCP"
```

在某些环境中， 多个访问点可能会使用同样的 SSID (通常， 这样做的目的是简化漫游)，
这时可能就需要与某个具体的设备关联了。 这种情况下， 您还应指定无线访问点的 BSSID (这时可以不指定 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够约束查找无线访问点的范围， 例如限制系统扫描的频段， 等等。
如果您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"
```

就会强制卡使用采用 2.4GHz 的 802.11g， 这样在扫描的时候， 就不会考虑那些 5GHz 的频段了。
除此之外， 还可以通过 channel 参数来将操作锁定在特定频率， 以及通过 chanlist
参数来指定扫描的频段列表。 关于这些参数的进一步信息， 可以在联机手册 ifconfig(8) 中找到。

32.3.3.1.2.2. 验证身份

一旦您选定了无线访问点， 您的通讯站就需要完成身份验证， 以便开始发送和接收数据。
身份验证可以通过许多方式进行， 最常用的一种方式称为开放式验证，
它允许任意通讯站加入网络并相互通信。 这种验证方式只应在您第一次配置无线网络进行测试时使用。
其它的验证方式则需要在进行数据通讯之前， 首先进行密钥协商握手；
这些方式要么使用预先分发的密钥或密码， 要么是用更复杂一些的后台服务， 如 RADIUS。
绝大多数用户会使用默认的开放式验证， 而第二多的则是 WPA-PSK， 它也称为个人 WPA， 在 下面
的章节中将进行介绍。

如果您使用 Apple® AirPort® Extreme 基站作为无线访问点， 则可能需要同时在两端配置
WEP 共享密钥验证。 这可以通过在 /etc/rc.conf 文件中进行设置， 或使用
wpa_supplicant(8) 程序来手工完成。 如果您只有一个 AirPort® 基站，
则可以用类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， 应尽量避免使用共享密钥这种验证方法， 因为它以非常受限的方式使用 WEP

> 密钥，使得攻击者能够很容易地破解密钥。如果必须使用 WEP (例如，为了兼容旧式的设备) 最好使用 WEP 配合 open 验证方式。关于 WEP 的更多资料请参见 WEP。

32.3.3.1.2.3. 通过 DHCP 获取 IP 地址

在您选定了无线访问点，并配置了验证参数之后，还必须获得 IP 地址才能真正开始通讯。多数时候，您会通过 DHCP 来获得无线 IP 地址。要达到这个目的，需要编辑 /etc/rc.conf 并在配置中加入 DHCP：

```
wlans_ath0="wlan0"
ifconfig_wlan0="DHCP"
```

现在您已经完成了启用无线网络接口的全部准备工作了，下面的操作将启用它：

```
# /etc/rc.d/netif start
```

一旦网络接口开始运行，就可以使用 ifconfig 来查看网络接口 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 0xffffff00 broadcast 192.168.1.255
    media: IEEE 802.11 Wireless Ethernet OFDM/54Mbps mode 11g
    status: associated
    ssid dlinkap channel 11 (2462 Mhz 11g) bssid 00:13:46:49:41:76
    country US ecm authmode OPEN privacy OFF txpower 21.5 bmiss 7
    scanvalid 60 bgscan bgscanintvl 300 bgscanidle 250 roam:rssi 7
    roam:rate 5 protmode CTS wme burst
```

这里的 status: associated 表示您已经连接到了无线网络 (在这个例子中，这个网络的名字是 dlinkap)。bssid 00:13:46:49:41:76 是指您所用无线访问点的 MAC 地址； authmode OPEN 表示您通讯的内容将将不加密。

32.3.3.1.2.4. 静态 IP 地址

如果无法从某个 DHCP 服务器获得 IP 地址，则可以配置一个静态 IP 地址，方法是将前面的 DHCP 关键字替换为地址信息。请务必保持其他用于连接无线访问点的参数：

```
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 保护访问) 是一种与 802.11 网络配合使用的安全协议，其目的是消除 WEP 中缺少身份验证能力的问题，以及一些其它的安全弱点。WPA 采用了 802.1X 认证协议，并采用从多种与 WEP 不同的加密算法中选择一种来保证数据保密性。WPA 支持的唯一一种加密算法是 TKIP (临时密钥完整性协议)，TKIP 是一种对 WEP 所采用的基本 RC4 加密算法的扩展，除此之外还提供了对检测到的入侵的响应机制。TKIP 被设计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， 在可能时倾向于使用这种加密； 表达这一规范的常用术语是 WPA2 (或 RSN)。

WPA 定义了验证和加密协议。 验证通常是使用两种方法之一来完成的： 通过 802.1X 或类似 RADIUS 这样的后端验证服务， 或通过在通讯站和无线访问点之间通过事先分发的密码来进行最小握手。 前一种通常称作企业 WPA， 而后者通常也叫做个人 WPA。 因为多数人不会为无线网络配置 RADIUS 后端服务器， 因此 WPA-PSK 是在 WPA 中最为常见的一种。

对无线连接的控制和身份验证工作 (密钥协商或通过服务器验证) 是通过 wpa_supplicant(8) 工具来完成的。 这个程序运行时需要一个配置文件， /etc/wpa_supplicant.conf。 关于这个文件的更多信息， 请参考联机手册 wpa_supplicant.conf(5)。

32.3.3.1.3.1. WPA-PSK

WPA-PSK 也称作 个人-WPA， 它基于预先分发的密钥 (PSK)， 这个密钥是根据作为无线网络上使用的主密钥的密码生成的。 这表示每个无线用户都会使用同样的密钥。 WPA-PSK 主要用于小型网络， 在这种网络中， 通常不需要或没有办法架设验证服务器。

无论何时， 都应使用足够长， 且包括尽可能多字母和数字的强口令， 以免被猜出和/或攻击。

第一步是修改配置文件 /etc/wpa_supplicant.conf， 并在其中加入在您网络上使用的 SSID 和事先分发的密钥：

```
network={
 ssid="freebsdap"
 psk="freebsdmall"
}
```

接下来， 在 /etc/rc.conf 中， 我们将指定无线设备的配置， 令其采用 WPA， 并通过 DHCP 来获取 IP 地址：

```
wlans_ath0="wlan0"
ifconfig_wlan0="WPA DHCP"
```

下面启用无线网络接口：

```
# /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 0xffffff00 broadcast 192.168.0.255
    media: IEEE 802.11 Wireless Ethernet OFDM/36Mbps mode 11g
```

```
    status: associated
    ssid freebsdap channel 1 (2412 Mhz 11g) bssid 00:11:95:c3:0d:ac
    country US ecm authmode WPA2/802.11i privacy ON deftxkey UNDEF
    AES-CCM 3:128-bit txpower 21.5 bmiss 7 scanvalid 450 bgscan
    bgscanintvl 300 bgscanidle 250 roam:rssi 7 roam:rate 5 protmode CTS
    wme burst roaming MANUAL
```

除此之外， 您也可以手动地使用 above 中那份 /etc/wpa_supplicant.conf 来配置， 方法是执行：

```
# 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 服务器获取 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 0xffffff00 broadcast 192.168.0.255
    media: IEEE 802.11 Wireless Ethernet OFDM/36Mbps mode 11g
    status: associated
    ssid freebsdap channel 1 (2412 Mhz 11g) bssid 00:11:95:c3:0d:ac
    country US ecm authmode WPA2/802.11i privacy ON deftxkey UNDEF
    AES-CCM 3:128-bit txpower 21.5 bmiss 7 scanvalid 450 bgscan
    bgscanintvl 300 bgscanidle 250 roam:rssi 7 roam:rate 5 protmode CTS
    wme burst roaming MANUAL
```

如果在 /etc/rc.conf 中把 ifconfig_wlan0 设置成了 DHCP (像 ifconfig_wlan0="DHCP" 这样)， 那么在 wpa_supplicant 连上了无线接入点 (AP) 之后，则会自动运行 dhclient。

如果不打算使用 DHCP 或者 DHCP 不可用， 您可以在 wpa_supplicant 为通讯站完成了身份认证之后，指定静态 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 0xffffff00 broadcast 192.168.0.255
    media: IEEE 802.11 Wireless Ethernet OFDM/36Mbps mode 11g
    status: associated
    ssid freebsdap channel 1 (2412 Mhz 11g) bssid 00:11:95:c3:0d:ac
    country US ecm authmode WPA2/802.11i privacy ON deftxkey UNDEF
    AES-CCM 3:128-bit txpower 21.5 bmiss 7 scanvalid 450 bgscan
    bgscanintvl 300 bgscanidle 250 roam:rssi 7 roam:rate 5 protmode CTS
    wme burst roaming MANUAL
```

如果没有使用 DHCP， 还需要手工配置默认网关， 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 的第二种方式是使用 802.1X 后端验证服务器。 在这个例子中， WPA 也称作 企业-WPA，
以便与安全性较差、 采用事先分发密钥的 个人-WPA 区分开来。 在 企业-WPA 中， 验证操作是采用 EAP 完成的 (可扩展认证协议)。

EAP 并未附带加密方法。 因此设计者决定将 EAP 放在加密信道中进行传送。 目前有许多 EAP 验证方法，
最常用的方法是 EAP-TLS、 EAP-TTLS 和 EAP-PEAP。

EAP-TLS (带 传输层安全 的 EAP) 是一种在无线世界中得到了广泛支持的验证协议， 因为它是 Wi-Fi 联盟
核准的第一个 EAP 方法。 EAP-TLS 需要使用三个证书： CA 证书 (在所有计算机上安装)、
用以向您证明服务器身份的服务器证书， 以及每个无线客户端用于证明身份的客户机证书。 在这种 EAP 方式中， 验证服务器和无线客户端均通过自己的证书向对方证明身份，
它们均验证对方的证书是本机构的证书发证机构 (CA) 签发的。

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="freebsdap" 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="freebsdmallclient" ⑨
}
```

① 这个字段表示网络名 (SSID)。

② 这里， 我们使用 RSN (IEEE® 802.11i) 协议， 也就是 WPA2。

③ key_mgmt 这行表示所用的密钥管理协议。 在我们的例子中， 它是使用 EAP 验证的 WPA： WPA-EAP。

④ 这个字段中， 提到了我们的连接采用 EAP 方式。

⑤ identity 字段包含了 EAP 的实体串。

⑥ ca_cert 字段给出了 CA 证书文件的路径名。 在验证服务器证书时， 这个文件是必需的。

⑦ client_cert 这行给出了客户机证书的路径名。 对每个无线客户端而言，
这个证书都是在全网范围内唯一的。

⑧ private_key 字段是客户机证书私钥文件的路径名。

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 0xffffff00 broadcast 192.168.0.255
    media: IEEE 802.11 Wireless Ethernet DS/11Mbps mode 11g
    status: associated
    ssid freebsdap channel 1 (2412 Mhz 11g) bssid 00:11:95:c3:0d:ac
    country US ecm authmode WPA2/802.11i privacy ON deftxkey UNDEF
    AES-CCM 3:128-bit txpower 21.5 bmiss 7 scanvalid 450 bgscan
    bgscanintvl 300 bgscanidle 250 roam:rssi 7 roam:rate 5 protmode CTS
    wme burst roaming MANUAL
```

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， 参与验证过程的服务器和客户机都需要证书， 而在使用 EAP-TTLS
(带传输层安全隧道的 EAP) 时， 客户机证书则是可选的。 这种方式与某些安全 web 站点更为接近，
即使访问者没有客户端证书， 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
隧道来传送验证信息。

对于它的配置， 同样是通过 /etc/wpa_supplicant.conf 文件来进行的：

```
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 这个字段是我们的连接所采用的 EAP 方式。

② identity 字段中是在加密 TLS 隧道中用于 EAP 验证的身份串。

③ password 字段中是用于 EAP 验证的口令字。

④ ca_cert 字段给出了 CA 证书文件的路径名。 在验证服务器证书时， 这个文件是必需的。

⑤ 这个字段中给出了加密 TLS 隧道中使用的验证方式。 在这个例子中， 我们使用的是带 MD5-加密口令 的 EAP。 "inner authentication" (译注：内部鉴定) 通常也叫 "phase2"。

您还必须把下面的配置写入 /etc/rc.conf：

```
wlans_ath0="wlan0"
ifconfig_wlan0="WPA DHCP"
```

下一步是启用网络接口：

```
# /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
DHCPREQUEST on wlan0 to 255.255.255.255 port 67 interval 21
DHCPACK from 192.168.0.20
bound to 192.168.0.254 -- renewal in 300 seconds.
wlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    ether 00:11:95:d5:43:62
    inet 192.168.0.254 netmask 0xffffff00 broadcast 192.168.0.255
    media: IEEE 802.11 Wireless Ethernet DS/11Mbps mode 11g
    status: associated
    ssid freebsdap channel 1 (2412 Mhz 11g) bssid 00:11:95:c3:0d:ac
    country US ecm authmode WPA2/802.11i privacy ON deftxkey UNDEF
    AES-CCM 3:128-bit txpower 21.5 bmiss 7 scanvalid 450 bgscan
    bgscanintvl 300 bgscanidle 250 roam:rssi 7 roam:rate 5 protmode CTS
    wme burst roaming MANUAL
```

32.3.3.1.3.4. 使用 EAP-PEAP 的 WPA

PEAPv0/EAP-MSCHAPv2 是最常见的 PEAP 方法。 此文档的以下部分将使用 PEAP 指代这些方法。

PEAP (受保护的 EAP) 被设计用以替代 EAP-TTLS， 并且是在 EAP-TLS 之后最为常用的 EAP 标准。

换言之， 如果您的网络中有多种不同的操作系统， PEAP 将是仅次于 EAP-TLS 的支持最广的标准。

PEAP 与 EAP-TTLS 很像： 它使用服务器端证书， 通过在客户端与验证服务器之间建立加密的 TLS 隧道来向用户验证身份， 这保护了验证信息的交换过程。 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 隧道传送的。 而 EAP-TTLS 在传送用户名和口令时， 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 这个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 验证的身份串。

③ password 字段中是用于 EAP 验证的口令字。

④ ca_cert 字段给出了 CA 证书文件的路径名。 在验证服务器证书时， 这个文件是必需的。

⑤ 这个字段包含了第一阶段验证 (TLS 隧道) 的参数。 随您使用的验证服务器的不同， 您需要指定验证的标签。 多数时候， 标签应该是 "客户端 EAP 加密"， 这可以通过使用 peaplabel=0 来指定。 更多信息可以在联机手册 wpa_supplicant.conf(5) 中找到。

⑥ 这个字段的内容是验证协议在加密的 TLS 隧道中使用的信息。 对 PEAP 而言， 这是 auth=MSCHAPV2。

您还必须把下面的配置加入到 /etc/rc.conf：

```
wlans_ath0="wlan0"
ifconfig_wlan0="WPA DHCP"
```

下一步是启用网络接口：

```
# /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
DHCPREQUEST on wlan0 to 255.255.255.255 port 67 interval 21
DHCPACK from 192.168.0.20
bound to 192.168.0.254 -- renewal in 300 seconds.
wlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    ether 00:11:95:d5:43:62
```

```
    inet 192.168.0.254 netmask 0xffffff00 broadcast 192.168.0.255
    media: IEEE 802.11 Wireless Ethernet DS/11Mbps mode 11g
    status: associated
    ssid freebsdap channel 1 (2412 Mhz 11g) bssid 00:11:95:c3:0d:ac
    country US ecm authmode WPA2/802.11i privacy ON deftxkey UNDEF
    AES-CCM 3:128-bit txpower 21.5 bmiss 7 scanvalid 450 bgscan
    bgscanintvl 300 bgscanidle 250 roam:rssi 7 roam:rate 5 protmode CTS
    wme burst roaming MANUAL
```

32.3.3.1.4. WEP

WEP (有线等效协议) 是最初 802.11 标准的一部分。 其中没有提供身份验证机制， 只提供了弱访问控制，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
```

- weptxkey 指明了使用哪个 WEP 密钥来进行数据传输。 这里我们使用第三个密钥。它必须与无线接入点的配置一致。 如果你不清楚你的无线接入点， 尝试用 1（就是说第一个密钥）来设置这个变量。
- wepkey 用于选择 WEP 密钥。 其格式应为 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。

您需要将 0x3456789012 改为在无线接入点上配置的那个。

我们建议您阅读联机手册 ifconfig(8) 来了解进一步的信息。

wpa_supplicant 机制也可以用来配置您的无线网卡使用 WEP。 前面的例子也可以通过在 /etc/wpa_supplicant.conf 中加入下述设置来实现：

```
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， 也称为 ad-hoc 模式， 是为点对点连接设计的。 例如， 如果希望在计算机 A 和 B 之间建立 ad-hoc 网络， 我们只需选择两个 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 0xffffff00 broadcast 192.168.0.255
    media: IEEE 802.11 Wireless Ethernet autoselect mode 11g <adhoc>
    status: running
    ssid freebsdap channel 2 (2417 Mhz 11g) bssid 02:11:95:c3:0d:ac
    country US ecm authmode OPEN privacy OFF txpower 21.5 scanvalid 60
    protmode CTS wme burst
```

此处的 adhoc 参数表示无线网络接口应以 IBSS 模式运转。

此时， 在 B 上应该能够检测到 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 再次确认了 A 机是以 ad-hoc 模式运行的。 我们只需给 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 0xffffff00 broadcast 192.168.0.255
    media: IEEE 802.11 Wireless Ethernet autoselect mode 11g <adhoc>
    status: running
    ssid freebsdap channel 2 (2417 Mhz 11g) bssid 02:11:95:c3:0d:ac
    country US ecm authmode OPEN privacy OFF txpower 21.5 scanvalid 60
    protmode CTS wme burst
```

这样， A 和 B 就可以交换信息了。

32.3.5. FreeBSD 基于主机的（无线）访问接入点

FreeBSD 可以作为一个（无线）访问接入点（AP）， 这样可以不必再去买一个硬件 AP 或者使用 ad-hoc

模式的网络。 当你的 FreeBSD 机器作为网关连接到另外一个网络的时候将非常有用。

32.3.5.1. 基本配置

在把你的 FreeBSD 机器配置成一个 AP 以前， 你首先需要先在内核配置好对你的无线网卡的无线网络支持。 当然你还需要加上你想用的安全协议。想获得更详细的信息， 请参阅 基本安装。

目前还不支持使用 Windows® 驱动和 NDIS 驱动包装的网卡做为 AP 使用。只有 FreeBSD 原生的无线驱动能够支持 AP 模式。

一旦装载了无线网络的支持， 你就可以检查一下看看你的无线设备是否支持基于主机的无线访问接入模式（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,SHPREAM
BLE,MONITOR,MBSS,WPA1,WPA2,BURST,WME,WDS,BGSCAN,TXFRAG>
cryptocaps=1f<WEP,TKIP,AES,AES_CCM,TKIPMIC>
```

这段输出显示了网卡所支持的各种功能； 其中的关键字 HOSTAP 表示这块网卡可以作为无线网络接入点来使用。 此外， 这里还会给出所支持的加密算法： WEP、 TKIP、 AES， 等等。 这些信息对于知道在访问接入点上使用何种安全协议非常重要。

只有创建网络伪设备时能够配置无线设备是否以 hostap 模式运行， 如果之前已经存在了相应的设备， 则需要首先将其销毁：

```
# ifconfig wlan0 destroy
```

接着， 在配置其它参数前， 以正确的选项重新生成设备：

```
# 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 检查 wlan0 网络接口的状态：

```
# ifconfig wlan0
 wlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
    ether 00:11:95:c3:0d:ac
    inet 192.168.0.1 netmask 0xffffff00 broadcast 192.168.0.255
    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。

通过在 /etc/rc.conf 中加入下面的配置， 也可以在系统引导的过程中自动完成对于网络接口的配置：

```
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. 不使用认证或加密的（无线）访问接入点

尽管我们不推荐运行一个不使用任何认证或加密的 AP， 但这是一个非常简单的检测 AP
是否正常工作的方法。 这样配置对于调试客户端问题也非常重要。

一旦 AP 被配置成了我们前面所展示的那样， 就可以在另外一台无线机器上初始化一次扫描来找到这个 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
```

在客户机上能看到已经连接上了（无线）访问接入点：

```
# 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 0xffffff00 broadcast 192.168.0.255
    media: IEEE 802.11 Wireless Ethernet OFDM/54Mbps mode 11g
    status: associated
    ssid freebsdap channel 1 (2412 Mhz 11g) bssid 00:11:95:c3:0d:ac
    country US ecm authmode OPEN privacy OFF txpower 21.5 bmiss 7
    scanvalid 60 bgscan bgscanintvl 300 bgscanidle 250 roam:rssi 7
    roam:rate 5 protmode CTS wme burst
```

32.3.5.3. 使用 WPA 的（无线）访问接入点

这一段将注重介绍在 FreeBSD （无线）访问接入点上配置使用 WPA 安全协议。 更多有关 WPA 和配置基于
WPA 无线客户端的细节 请参阅 WPA。

hostapd 守护进程将被用于处理与客户端的认证和在启用 WPA （无线）访问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
能够正确的工作了，便把如下这行加入 /etc/rc.conf 使得 hostapd 能在机器启动的时候自动运行：

```
hostapd_enable="YES"
```

在配置 hostapd 以前， 请确保你已经完成了基本配置中所介绍的步骤 基本配置。

32.3.5.3.1. WPA-PSK

WPA-PSK 旨在为没有认证服务器的小型网络而设计的。

配置文件为 /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=freebsdmall ⑦
wpa_key_mgmt=WPA-PSK ⑧
wpa_pairwise=CCMP TKIP ⑨
```

① 这一项标明了访问接入点所使用的无线接口。

② 这一项设置了执行 hostapd 时候显示相关信息的详细程度。 1 表示最小的级别。

③ ctrl_interface 这项给出了 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。这里使用了默认值。

④ ctrl_interface_group 这行设置了允许访问控制界面文件的组属性 （这里我们使用了 wheel 组）。

⑤ 这一项是设置网络的名称。

⑥ wpa 这项表示启用了 WPA 而且指明要使用何种 WPA 认证协议。 值 1 表示 AP 将使用 WPA-PSK。

⑦ wpa_passphrase 这项包含用于 WPA 认证的 ASCII 密码。

⑧ wpa_key_mgmt 这行表明了我们所使用的密钥管理协议。 在这个例子中是 WPA-PSK。

⑨ wpa_pairwise 这项表示（无线）访问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 外的另一种选择， CCMP 一般作为首选密码；仅有在 CCMP 不能被使用的环境中选择 TKIP。

接下来的一步就是运行 hostapd：

```
# /etc/rc.d/hostapd forcestart
```

```
# ifconfig wlan0
 wlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 2290
    inet 192.168.0.1 netmask 0xffffff00 broadcast 192.168.0.255
    inet6 fe80::211:95ff:fec3:dac%ath0 prefixlen 64 scopeid 0x4
    ether 00:11:95:c3:0d:ac
    media: IEEE 802.11 Wireless Ethernet autoselect mode 11g <hostap>
    status: associated
    ssid freebsdap channel 1 bssid 00:11:95:c3:0d:ac
    authmode WPA2/802.11i privacy MIXED deftxkey 2 TKIP 2:128-bit txpowmax 36
protmode CTS dtimperiod 1 bintval 100
```

现在客户端能够连接上运行的（无线）访问接入点了， 更多细节可以参阅 WPA。 查看有哪些客户连接上了 AP 可以运行命令 ifconfig wlan0 list sta。

32.3.5.4. 使用 WEP 的（无线）访问接入点

我们不推荐使用 WEP 来设置一个（无线）访问接入点，因为没有认证的机制并容易被破解。
一些历史遗留下的无线网卡仅支持 WEP 作为安全协议，这些网卡仅允许搭建不含认证或 WEP 协议的 AP。

在设置了正确的 SSID 和 IP 地址后，无线设备就可以进入 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
```

- weptxkey 表示传输中使用哪一个 WEP 密钥。这个例子中用了第3把密钥（请注意密钥的编号从1开始）。这个参数必须设置以用来加密数据。
- wepkey 表示设置所使用的 WEP 密钥。它应该符合 index:key 这样的格式。如果没有指定 index，那么默认值为 1。这就是说如果我们使用了除第一把以外的密钥，那么就需要指定 index。

再使用一次 ifconfig 命令查看 wlan0 接口的状态：

```
# ifconfig wlan0
 wlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
    ether 00:11:95:c3:0d:ac
    inet 192.168.0.1 netmask 0xffffff00 broadcast 192.168.0.255
    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
```

现在可以从另外一台无线机器上初始化一次扫描来找到这个 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
```

现在客户机能够使用正确的参数（密钥等）找到并连上（无线）访问接入点了，更多细节请参阅WEP。

32.3.6. 同时使用有线和无线连接

一般而言，有线网络的速度更快而且更可靠，而无线网络则提供更好的灵活及机动性，
使用笔记本的用户，往往会希望结合两者的优点，并能够在两种连接之间无缝切换。

在 FreeBSD 上可以将多个网络接口合并到一起，并以 "故障转移" 的方式自动切换，也就是说，
这一组网络接口有一定的优先顺序，而操作系统在链路状态发生变化时则自动进行切换，
例如当同时存在有线和无线连接的时候优先使用有线网络，而当有线网络断开时，则自动切换到无线网络。

我们将在稍后的 链路聚合与故障转移 中介绍链路聚合和故障转移，并在
有线网络和无线网络接口间的自动切换 中对这种配置方式进行示范。

32.3.7. 故障排除

如果您在使用无线网络时遇到了麻烦， 此处提供了一系列用以帮助排除故障的步骤。

- 如果您在列表中找不到无线访问点， 请确认您没有将无线设备配置为使用有限的一组频段。
- 如果您无法关联到无线访问点， 请确认您的通讯站配置与无线访问点的配置一致。
  这包括认证模式以及安全协议。 尽可能简化您的配置。 如果您正使用类似 WPA 或 WEP
  这样的安全协议， 请将无线访问点配置为开放验证和不采用安全措施， 并检查是否数据能够通过。
- 一旦您能够关联到无线访问点之后， 就可以使用简单的工具如 ping(8) 来诊断安全配置了。

  wpa_supplicant 提供了许多调试支持； 尝试手工运行它， 在启动时指定 -dd 选项， 并察看输出结果。

- 除此之外还有许多其它的底层调试工具。 您可以使用 /usr/src/tools/tools/net80211 中的 wlandebug
  命令来启用 802.11 协议支持层的调试功能。 例如：

  ```
  # wlandebug -i ath0 +scan+auth+debug+assoc
   net.wlan.0.debug: 0 => 0xc80000<assoc,auth,scan>
  ```

  可以用来启用与扫描无线访问点和 802.11 协议在安排通讯时与握手有关的控制台信息。

  还有许多有用的统计信息是由 802.11 层维护的； wlanstats 工具可以显示这些信息。
  这些统计数据能够指出由 802.11 层识别出来的错误。 请注意某些错误可能是由设备驱动在 802.11
  层之下识别出来的， 因此这些错误可能并不显示。 要诊断与设备有关的问题，
  您需要参考设备驱动程序的文档。

如果上述信息没能帮助您找到具体的问题所在， 请提交问题报告， 并在其中附上这些工具的输出。

32.4. 蓝牙

32.4.1. 简介

Bluetooth (蓝牙) 是一项无线技术， 用于建立带宽为 2.4GHZ，波长为 10 米的私有网络。
网络一般是由便携式设备，比加手机 (cellular phone)， 掌上电脑 (handhelds) 和膝上电脑 (laptops)) 以
ad-hoc 形式组成。不象其它流行的无线技术——Wi-Fi，Bluetooth 提供了更高级的服务层面，像类 FTP
的文件服务、文件推送 (file pushing)、语音传送、串行线模拟等等。

在 FreeBSD 里，蓝牙栈 (Bluetooth stack) 通过使用 Netgraph 框架 (请看 netgraph(4)) 来的实现。
大量的"Bluetooth USB dongle"由 ng_ubt(4) 驱动程序支持。 基于 Broadcom BCM2033 芯片组的
Bluetooth 设备可以通过 ubtbcmfw(4) 和 ng_ubt(4) 驱动程序支持。 3Com Bluetooth PC 卡 3CRWB60-A
由 ng_bt3c(4) 驱动程序支持。 基于 Serial 和 UART 的蓝牙设备由 sio(4)、ng_h4(4) 和
hcseriald(8)。本节介绍 USB Bluetooth dongle 的使用。

32.4.2. 插入设备

默认的 Bluetooth 设备驱动程序已存在于内核模块里。 接入设备前，您需要将驱动程序加载入内核：

```
# kldload ng_ubt
```

如果系统启动时 Bluetooth 设备已经存在于系统里， 那么从 /boot/loader.conf 里加载这个模块：

```
ng_ubt_load="YES"
```

插入USB dongle。控制台(console)(或syslog中)会出现类似如下的信息：

```
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 (蓝牙栈)的。 最好在拔出设备前停止 stack(stack)，当然也不是非做不可。 启动 stack (栈) 时，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连接管理器的命令接口及访问硬件状态字和控制寄存器的通道。
这个接口提供了访问蓝牙基带 (Bluetooth baseband) 功能的统一方式。 主机上的 HCI 层与蓝牙硬件上的 HCI 固件交换数据和命令。 主控制器的传输层 (如物理总线) 驱动程序提供两个 HCI 层交换信息的能力。

为每个蓝牙 (Bluetooth) 设备创建一个 hci 类型的 Netgraph 结点。 HCI 结点一般连接蓝牙设备的驱动结点 (下行流) 和 L2CAP 结点 (上行流)。 所有的HCI操作必须在 HCI 结点上进行而不是设备驱动结点。HCI 结点的默认名是 "devicehci"。更多细节请参考 ng_hci(4) 的联机手册。

最常见的任务是发现在 RF proximity 中的蓝牙 (Bluetooth) 设备。这个就叫做 质询(inquiry)。质询及 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找出范围内的蓝牙设备。
在几秒钟内您应该得到一张设备列表。 注意远程主机只有被置于 discoverable(可发现) 模式才能答应质询。

```
% hccontrol -n ubt0hci inquiry
Inquiry result, num_responses=1
Inquiry result #0
    BD_ADDR: 00:80:37:29:19:a4
    Page Scan Rep. Mode: 0x1
    Page Scan Period Mode: 00
    Page Scan Mode: 00
    Class: 52:02:04
    Clock offset: 0x78ef
```

```
Inquiry complete. Status: No error [00]
```

BD_ADDR 是蓝牙设备的特定地址， 类似于网卡的 MAC 地址。需要用此地址与某个设备进一步地通信。
可以为 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获得分配给远程设备的可读名。

```
% hccontrol -n ubt0hci remote_name_request 00:80:37:29:19:a4
BD_ADDR: 00:80:37:29:19:a4
Name: Pav's T39
```

如果在远程蓝牙上运行质询，您会发现您的计算机是 "your.host.name (ubt0)"。
分配给本地设备的名字可随时改变。

蓝牙系统提供点对点连接 (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动基带 (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)
会自动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 命令不需要超级用户权限。

32.4.4. 逻辑连接控制和适配协议(L2CAP)

逻辑连接控制和适配协议 (L2CAP) 为上层协议提供面向连接和无连接的数据服务，
并提供多协议功能和分割重组操作。L2CAP 允许上层协议和应用软件传输和接收最大长度为 64K 的 L2CAP
数据包。

L2CAP 基于 通道(channel) 的概念。 通道 (Channel) 是位于基带 (baseband) 连接之上的逻辑连接。
每个通道以多对一的方式绑定一个单一协议 (single protocol)。
多个通道可以绑定同一个协议，但一个通道不可以绑定多个协议。 每个在通道里接收到的 L2CAP
数据包被传到相应的上层协议。 多个通道可共享同一个基带连接。

为每个蓝牙 (Bluetooth) 设备创建一个 l2cap 类型的 Netgraph 结点。 L2CAP 结点一般连接 HCI
结点(下行流)和蓝牙设备的驱动结点(上行流)。 L2CAP 结点的默认名是 "devicel2cap"。 更多细节请参考
ng_l2cap(4) 的联机手册。

一个有用的命令是 l2ping(8)， 它可以用来 ping 其它设备。
一些蓝牙实现可能不会返回所有发送给它们的数据， 所以下例中的 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 上进行多种操作。 以下这个例子展示如何取得本地设备的逻辑连接 (通道) 和基带连接的列表：

```
% 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 O        0 OPEN
```

另一个诊断工具是 btsockstat(1)。 它完成与 netstat(1) 类似的操作， 只是用了蓝牙网络相关的数据结构。 以下这个例子显示与 l2control(8) 相同的逻辑连接。

```
% 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 协议

RFCOMM 协议提供基于 L2CAP 协议的串行端口模拟。 该协议基于 ETSI TS 07.10 标准。RFCOMM 是一个简单的传输协议， 附加了摸拟 9 针 RS-232(EIATIA-232-E) 串行端口的定义。 RFCOMM 协议最多支持 60 个并发连接 (RFCOMM通道)。

为了实现 RFCOMM， 运行于不同设备上的应用程序建立起一条关于它们之间通信段的通信路径。 RFCOMM实际上适用于使用串行端口的应用软件。 通信段是一个设备到另一个设备的蓝牙连接 (直接连接)。

RFCOMM 关心的只是直接连接设备之间的连接， 或在网络里一个设备与 modem 之间的连接。RFCOMM 能支持其它的配置， 比如在一端通过蓝牙无线技术通讯而在另一端使用有线接口。

在FreeBSD，RFCOMM 协议在蓝牙套接字层 (Bluetooth sockets layer) 实现。

32.4.6. 设备的结对(Pairing of Devices)

默认情况下，蓝牙通信是不需要验证的， 任何设备可与其它任何设备对话。一个蓝牙设备 (比如手机) 可以选择通过验证以提供某种特殊服务 (比如拨号服务)。 蓝牙验证一般使用 PIN码(PIN codes)。 一个 PIN 码是最长为 16 个字符的 ASCII 字符串。 用户需要在两个设备中输入相同的PIN码。用户输入了 PIN 码后， 两个设备会生成一个 连接密匙(link key)。 接着连接密钥可以存储在设备或存储器中。

连接时两个设备会使用先前生成的连接密钥。 以上介绍的过程被称为 结对(pairing)。
注意如果任何一方丢失了连接密钥，必须重新进行结对。

守护进程 hcsecd(8) 负责处理所有蓝牙验证请求。 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 码被预设为"1234"：

```
device {
    bdaddr  00:80:37:29:19:a4;
    name    "Pav's T39";
    key     nokey;
    pin     "1234";
   }
```

PIN 码没有限制(除了长度)。有些设备 (例如蓝牙耳机) 会有一个预置的 PIN 码。-d 开关强制 hcsecd(8)
守护进程处于前台，因此很容易看清发生了什么。 设置远端设备准备接收结对
(pairing)，然后启动蓝牙连接到远端设备。 远端设备应该回应接收了结对并请求PIN码。输入与
hcsecd.conf 中一样的 PIN 码。 现在您的个人计算机已经与远程设备结对了。
另外您也可以在远程设备上初始结点。

可以通过在 /etc/rc.conf 文件中增加下面的行， 以便让 hcsecd 在系统启动时自动运行：

```
hcsecd_enable="YES"
```

以下是简单的 hcsecd 服务输出样本：

```
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. 服务发现协议 (SDP)

服务发现协议 (SDP) 提供给客户端软件一种方法， 它能发现由服务器软件提供的服务及属性。
服务的属性包括所提供服务的类型或类别， 使用该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。
服务器维护一张服务记录列表，它介绍服务器上服务的特性。
每个服务记录包含关于单个服务的信息。通过发出 SDP 请求，
客户端会得到服务记录列表的信息。如果客户端 (或者客户端上的应用软件)
决定使用一个服务，为了使用这个服务它必须与服务提供都建立一个独立的连接。 SDP
提供了发现服务及其属性的机制，但它并不提供使用这些服务的机制。

一般地，SDP客户端按照服务的某种期望特征来搜索服务。 但是，即使没有任何关于由 SDP
服务端提供的服务的预设信息， 有时也能令人满意地发现它的服务记录里所描述的是哪种服务类型。

这种发现所提供服务的过程称为 浏览(browsing)。

蓝牙 SDP 服务端 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 浏览查询。

```
% 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。注意每个服务有一个属性 (比如 RFCOMM 通道)列表。 根据服务您可能需要为一些属性做个注释。
有些"蓝牙实现 (Bluetooth implementation)"不支持服务浏览，
可能会返回一个空列表。这种情况，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 里为蓝牙客户端提供服务，可以使用 sdpd(8) 服务。 您可以通过在 /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
```

需要为远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发出 SDP 浏览查询获得：

```
# sdpcontrol -l browse
```

32.4.8. 拨号网络 (DUN) 和使用 PPP(LAN) 层面的网络接入

拨号网络 (DUN) 配置通常与 modem 和手机一起使用。 如下是这一配置所涉及的内容：

- 计算机使用手机或 modem 作为无线 modem 来连接拨号因特网连入服务器，或者使用其它的拨号服务；
- 计算机使用手机或 modem 接收数据请求。

使用 PPP(LAN) 层面的网络接入常使用在如下情形：

- 单个蓝牙设备的局域网连入；
- 多个蓝牙设备的局域网接入；
- PC 到 PC (使用基于串行线模拟的 PPP 网络)。

在 FreeBSD 中，两个层面使用 ppp(8) 和 rfcomm_pppd(8) (一种封装器，可以将 RFCOMM 蓝牙连接转换为 PPP 可操作的东西) 来实现。 在使用任何层面之前，一个新的 PPP 标识必须在 /etc/ppp/ppp.conf 中建立。 想要实例请参考 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 的设备的 RFCOMM 连接。具体的 RFCOMM 通道号要通过 SDP 从远端设备获得。也可以手动指定通 RFCOMM，这种情况下 rfcomm_pppd(8) 将不能执行 SDP 查询。使用 sdpcontrol(8) 来查找远端设备上的 RFCOMM 通道。

```
# rfcomm_pppd -a 00:80:37:29:19:a4 -c -C dun -l rfcomm-dialup
```

为了提供 PPP(LAN) 网络接入服务，必须运行 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。 想要实例请参考 rfcomm_pppd(8)。 最后，在有效地通道号上开始 RFCOMM PPP 服务。 RFCOMM PPP 服务会使用本地 SDP 进程自动注册蓝牙 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) 层面

OBEX协议被广泛地用于移动设备之间简单的文件传输。 它的主要用处是在红外线通信领域，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实现，它可以从 comms/obexapp port 安装。

OBEX 客户端用于向 OBEX 服务器推入或接出对象。 一个对像可以是(举个例子)商业卡片或约会。 OBEX 客户能通过 SDP 从远程设备取得 RFCOMM 通道号。这可以通过指定服务名代替 RFCOMM 通道号来完成。支持的服务名是有：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，一个设备信息对像从手机中被拉出，一个新的对像被推入手机的目录。

```
% 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。必须创建一个根目录用于存放所有进入的对象。根文件夹的默认路径是 /var/spool/obex。 最后，在有效的 RFCOMM 通道号上开始 OBEX 服务。OBEX 服务会使用 SDP 进程自动注册 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)层面

串口(SP)层面允许蓝牙设备完成 RS232 (或类似) 串口线的仿真。 这个层面所涉及到情形是，通过虚拟串口使用蓝牙代替线缆来处理以前的程序。

工具 rfcomm_sppd(1) 来实现串口层。 "Pseudo tty" 用来作为虚拟的串口。下面的例子展示如何连接远程设备的串口服务。 注意您不必指定 RFCOMM 通道——rfcomm_sppd(1) 能够通过 SDP 从远端设备那里获得。 如果您想代替它的话，可以在命令行里指定 RFCOMM 通道来实现：

```
# rfcomm_sppd -a 00:07:E0:00:0B:CA -t /dev/ttyp6
rfcomm_sppd[94692]: Starting on /dev/ttyp6...
```

一旦连接上，"pseudo tty"就可以充当串口了：

```
# cu -l ttyp6
```

32.4.11. 问题解答

32.4.11.1. 不能连接远端设备

一些较老的蓝牙设备并不支持角色转换 (role switching)。默认情况下，FreeBSD 接受一个新的连接时，它会尝试进行角色转换并成为主控端 (master)。 不支持角色转换的设备将无法连接。
注意角色转换是在新连接建立时运行的， 因此如果远程设备不支持角色转换，就不可能向它发出请求。 一个 HCI 选项用来在本地端禁用角色转换。

```
# hccontrol -n ubt0hci write_node_role_switch 0
```

32.4.11.2. 如果有错， 能否知道到底正在发生什么?

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。 它可以用来显示蓝牙数据包的内容， 并将其记录到文件中。

32.5. 桥接

32.5.1. 简介

有时， 会有需要将一个物理网络分成两个独立的网段， 而不是创建新的 IP 子网， 并将其通过路由器相连。
以这种方式连接两个网络的设备称为 "网桥 (bridge)"。 有两个网络接口的 FreeBSD
系统可以作为网桥来使用。

网桥通过学习每个网络接口上的 MAC 层地址 (以太网地址) 工作。
只当数据包的源地址和目标地址处于不同的网络时， 网桥才进行转发。

在很多方面，网桥就像一个带有很少端口的以太网交换机。

32.5.2. 适合桥接的情况

适合使用网桥的， 有许多种不同的情况。

32.5.2.1. 使多个网络相互联通

网桥的基本操作是将两个或多个网段连接在一起。 由于各式各样的原因，
人们会希望使用一台真正的计算机， 而不是网络设备来充任网桥的角色， 常见的原因包括线缆的限制、
需要进行防火墙， 或为虚拟机网络接口连接虚拟网络。 网桥也可以将无线网卡以 hostap
模式接入有线网络。

32.5.2.2. 过滤/数据整形防火墙

使用防火墙的常见情形是无需进行路由或网络地址转换的情况 (NAT)。

举例来说， 一家通过 DSL 或 ISDN 连接到 ISP 的小公司， 拥有 13 个 ISP 分配的全局 IP 地址和 10 台 PC。
在这种情况下， 由于划分子网的问题， 采用路由来实现防火墙会比较困难。

基于网桥的防火墙可以串接在 DSL/ISDN 路由器的后面， 而无需考虑 IP 编制的问题。

32.5.2.3. 网络监视

网桥可以用于连接两个不同的网段， 并用于监视往返的以太网帧。 这可以通过在网桥接口上使用 bpf(4)
/tcpdump(1)， 或通过将全部以太网帧复制到另一个网络接口 (span 口) 来实现。

32.5.2.4. 2层 VPN

通过 IP 连接的网桥， 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层 冗余

网络可以通过多条链路连接在一起， 并使用生成树协议 (Spanning Tree Protocol) 来阻止多余的通路。
为使以太网能够正确工作， 两个设备之间应该只有一条激活通路， 而生成树能够检测环路，
并将多余的链路置为阻断状态。 当激活通路断开时， 协议能够计算另外一棵树， 并重新激活阻断的通路，
以恢复到网络各点的连通性。

32.5.3. 内核配置

这一节主要介绍 if_bridge(4) 网桥实现。 除此之外， 还有一个基于 netgraph 的网桥实现，
如欲了解进一步细节， 请参见联机手册 ng_bridge(4)。

网桥驱动是一个内核模块， 并会随使用 ifconfig(8) 创建网桥接口时自动加载。 您也可以将 device
if_bridge 加入到内核配置文件中， 以便将其静态联编进内核。

包过滤可以通过使用了 pfil(9) 框架的任意一种防火墙软件包来完成。 这些防火墙可以以模块形式加载，
也可以静态联编进内核。

通过配合 altq(4) 和 dummynet(4)， 网桥也可以用于流量控制。

32.5.4. 启用网桥

网桥是通过接口复制来创建的。 您可以使用 ifconfig(8) 来创建网桥接口， 如果内核不包括网桥驱动，则它会自动将其载入。

```
# 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桥接口， 并为其随机分配了以太网地址。 maxaddr 和 timeout 参数能够控制网桥在转发表中保存多少个 MAC 地址， 以及表项中主机的过期时间。 其他参数控制生成树的运转方式。

将成员网络接口加入网桥。 为了让网桥能够为所有网桥成员接口转发包，网桥接口和所有成员接口都需要处于启用状态：

```
# ifconfig bridge0 addm fxp0 addm fxp1 up
# ifconfig fxp0 up
# ifconfig fxp1 up
```

网桥现在会在 fxp0 和 fxp1 之间转发以太网帧。 等效的 /etc/rc.conf 配置如下，如此配置将在系统启动时创建同样的网桥。

```
cloned_interfaces="bridge0"
ifconfig_bridge0="addm fxp0 addm fxp1 up"
ifconfig_fxp0="up"
ifconfig_fxp1="up"
```

如果网桥主机需要 IP 地址， 则应将其绑在网桥设备本身， 而不是某个成员设备上。 这可以通过静态设置或 DHCP 来完成：

```
# ifconfig bridge0 inet 192.168.0.1/24
```

除此之外， 也可以为网桥接口指定 IPv6 地址。

32.5.5. 防火墙

当启用包过滤时， 通过网桥的包可以分别在进入的网络接口、 网桥接口和发出的网络接口上进行过滤。 这些阶段均可禁用。 当包的流向很重要时， 最好在成员接口而非网桥接口上配置防火墙。

网桥上可以进行许多配置以决定非 IP 及 ARP 包能否通过， 以及通过 IPFW 实现二层防火墙。 请参见 if_bridge(4) 联机手册以了解进一步的细节。

32.5.6. 生成树

网桥驱动实现了快速生成树协议 (RSTP 或 802.1w)， 并与较早的生成树协议 (STP) 兼容。
生成树可以用来在网络拓扑中检测并消除环路。 RSTP 提供了比传统 STP 更快的生成树覆盖速度，
这种协议会在相邻的交换机之间交换信息， 以迅速进入转发状态， 并避免产生环路。 FreeBSD 支持以
RSTP 和 STP 模式运行， 而 RSTP 是默认模式。

使用 stp 命令可以在成员接口上启用生成树。 对包含 fxp0 和 fxp1 的网桥， 可以用下列命令启用 STP：

```
# ifconfig bridge0 stp fxp0 stp fxp1
bridge0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
    ether d6:cf:d5:a0:94:6d
    id 00:01:02:4b:d4:50 priority 32768 hellotime 2 fwddelay 15
    maxage 20 holdcnt 6 proto rstp maxaddr 100 timeout 1200
    root id 00:01:02:4b:d4:50 priority 32768 ifcost 0 port 0
    member: fxp0 flags=1c7<LEARNING,DISCOVER,STP,AUTOEDGE,PTP,AUTOPTP>
        port 3 priority 128 path cost 200000 proto rstp
        role designated state forwarding
    member: fxp1 flags=1c7<LEARNING,DISCOVER,STP,AUTOEDGE,PTP,AUTOPTP>
        port 4 priority 128 path cost 200000 proto rstp
        role designated state forwarding
```

网桥的生成树 ID 为 00:01:02:4b:d4:50 而优先级为 32768。 其中 root id 与生成树相同，
表示这是作为生成树根的网桥。

另一个网桥也启用了生成树：

```
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
```

这里的 root id 00:01:02:4b:d4:50 priority 32768 ifcost 400000 port 4 表示根网桥是前面的
00:01:02:4b:d4:50， 而从此网桥出发的通路代价为 400000, 此通路到根网桥是通过 port 4 即 fxp0
连接的。

32.5.7. 网桥的高级用法

32.5.7.1. 重建流量流

网桥支持监视模式， 在 bpf(4) 处理之后会将包丢弃， 而不是继续处理或转发。
这可以用于将两个或多个接口上的输入转化为一个 bpf(4) 流。 在将两个独立的接口上的传输的 RX/TX

信号重整为一个时， 这会非常有用。

如果希望将四个网络接口上的输入转成一个流：

```
# ifconfig bridge0 addm fxp0 addm fxp1 addm fxp2 addm fxp3 monitor up
# tcpdump -i bridge0
```

32.5.7.2. 镜像口 (Span port)

网桥收到的每个以太网帧都可以发到镜像口上。 网桥上的镜像口数量没有限制，
如果一个接口已经被配置为镜像口， 则它就不能再作为网桥的成员口来使用。
这种用法主要是为与网桥镜像口相连的监听机配合使用。

如果希望将所有帧发到名为 fxp4 的接口上：

```
# ifconfig bridge0 span fxp4
```

32.5.7.3. 专用接口 (Private interface)

专用接口不会转发流量到除专用接口之外的其他端口。 这些流量会无条件地阻断， 因此包括 ARP
在内的以太网帧均不会被转发。 如果需要选择性地阻断流量， 则应使用防火墙。

32.5.7.4. 自学习接口 (Sticky Interfaces)

如果网桥的成员接口标记为自学习， 则动态学习的地址项一旦进入转发快取缓存， 即被认为是静态项。
自学习项不会从快取缓存中过期或替换掉， 即使地址在另一接口上出现也是如此。
这使得不必事先发布转发表， 也能根据学习结果得到静态项的有点， 但在这些网段被网桥看到的客户机，
就不能漫游至另一网段了。

另一种用法是将网桥与 VLAN 功能连用， 这样客户网络会被隔离在一边， 而不会浪费 IP 地址空间。 考虑
CustomerA 在 vlan100 上， 而 CustomerB 则在 vlan101 上。 网桥地址为 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
```

两台客户机均将 192.168.0.1 作为默认网关， 由于网桥快取缓存是自学习的， 因而它们无法伪造 MAC
地址来截取其他客户机的网络流量。

在 VLAN 之间的通讯可以通过专用接口 (或防火墙) 来阻断：

```
# ifconfig bridge0 private vlan100 private vlan101
```

这样这些客户机就完全相互隔离了。 可以使用整个的 /24 地址空间， 而无需划分子网。

32.5.7.5. 地址限制

接口后的源 MAC 地址数量是可以控制的。 一旦到达了限制未知源地址的包将会被丢弃，
直至现有缓存中的一项过期或被移除。

下面的例子是设置 CustomerA 在 vlan100 上可连接的以太网设备最大值为 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 守护进程进行管理。导出的网桥 MIB 符和 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 文件中去掉以下这行的注释 begemotSnmpdModulePath."bridge" = "/usr/lib/snmp_bridge.so" 并启动 bsnmpd 守护进程。 其他的配置选项诸如 community names 和 access lists 可能也许也需要修改。 参阅 bsnmpd(1) 和 snmp_bridge(3) 获取更多信息。

以下的例子中使用了 Net-SNMP 软件 (net-mgmt/net-snmp) 来查询一个网桥，当然同样也能够使用port net-mgmt/bsnmptools。 在 SNMP 客户端 Net-SNMP 的配置文件 $HOME/.snmp/snmp.conf 中 加入以下几行来导入网桥的 MIB 定义：

```
mibdirs +/usr/shared/snmp/mibs
mibs +BRIDGE-MIB:RSTP-MIB:BEGEMOT-MIB:BEGEMOT-BRIDGE-MIB
```

通过 IETF BRIDGE-MIB(RFC4188) 监测一个单独的网桥

```
% 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的值为2 意味着 STP 网桥拓扑改变了2次，拓扑的改变表示1个或多个网络中的连接改变或失效并且有一个新树生成。 dot1dStpTimeSinceTopologyChange.0 的值则能够显示这是何时改变的。

监测多个网桥接口可以使用 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 子树改变正在被监测的网桥接口：

```
% 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. 介绍

使用 lagg(4) 接口， 能够将多个网络接口聚合为一个虚拟接口， 以提供容灾和高速连接的能力。

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。
您在这里加入的第一个网口便会被视为主网口； 此后加入的其他网口，
则会被视为故障转移的备用网口。 如果发生故障转移之后， 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，
并不进行节点间协商或交换以太网帧来监控链路情况。 如果交换机支持 LACP，
则应使用后者而非这种配置。

FEC 将输出流量在激活的网口之间以协议头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) 和标记协议。 LACP
能够在节点与若干链路聚合组之间协商链路。 每一个链路聚合组 (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， 会以总速度最大的原则进行分摊。
当物理链路发生变化时， 链路聚合会迅速适应变动形成新的配置。

LACP 也是将输出流量在激活的网口之间以协议头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)
这是 FEC 模式的别名。

Round-robin (轮转)
将输出流量以轮转方式在所有激活端口之间调度， 并从任意激活端口接收进入流量。
这种模式违反了以太网帧排序规则， 因此应小心使用。

32.6.3. 例子

例 40. 与 Cisco® 交换机配合完成 LACP 链路聚合

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组接到交换机上。
在此基础上， 还可以增加更多的网口， 以提高吞吐量和故障容灾能力。
由于以太网链路上两节点间的帧序是强制性的， 因此两个节点之间的连接速度，
会取决于一块网卡的最大速度。 传输算法会尽量采用更多的信息，
以便将不同的网络流量分摊到不同的网络接口上， 并平衡不同网口的负载。

在 Cisco® 交换机上将 FastEthernet0/1 和 FastEthernet0/2 这两个网口添加到 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， 启用这个接口并配置 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
```

标记为 ACTIVE 的接口是激活据合组的部分， 这表示它们已经完成了与远程交换机的协商， 同时，
流量将通过这些接口来收发。 在 ifconfig(8) 的详细输出中会给出 LAG 的标识。

```
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， 则应使用 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:

          LACP port                Oper   Port    Port
Port      Flags  Priority  Dev ID      Age    Key    Number  State
Fa0/1     SA     32768     0005.5d71.8db8 29s    0x146  0x3     0x3D
Fa0/2     SA     32768     0005.5d71.8db8 29s    0x146  0x4     0x3D
```

如欲查看进一步的详情， 则需要使用 show lacp neighbor detail 命令。

如果希望在系统重启时保持这些设置， 应在 /etc/rc.conf 中增加如下配置：

```
ifconfig_fxp0="up"
ifconfig_fxp1="up"
cloned_interfaces="lagg0"
ifconfig_lagg0="laggproto lacp laggport fxp0 laggport fxp1 10.0.0.3/24"
```

例 41. 故障转移模式

故障转移模式中， 当首选链路发生问题时， 会自动切换到备用端口。 首先启用成员接口， 接着是配置 lagg(4) 接口， 其中， 使用 fxp0 作为首选接口， fxp1 作为备用接口， 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， 接口状态会是类似下面这样， 主要的区别是 MAC 地址和设备名：

```
# ifconfig lagg0
```

```
lagg0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
        options=8<VLAN_MTU>
        ether 00:05:5d:71:8d:b8
        inet 10.0.0.15 netmask 0xffffff00 broadcast 10.0.0.255
        media: Ethernet autoselect
        status: active
        laggproto failover
        laggport: fxp1 flags=0<>
        laggport: fxp0 flags=5<MASTER,ACTIVE>
```

系统将在 fxp0 上进行流量的收发。 如果 fxp0 的连接中断， 则 fxp1 会自动成为激活连接。
如果主端口的连接恢复， 则它又会成为激活连接。

如果希望在系统重启时保持这些设置， 应在 /etc/rc.conf 中增加如下配置：

```
ifconfig_fxp0="up"
ifconfig_fxp1="up"
cloned_interfaces="lagg0"
ifconfig_lagg0="laggproto failover laggport fxp0 laggport fxp1 10.0.0.15/24"
```

例 42. 有线网络和无线网络接口间的自动切换

对于使用笔记本的用户来说， 通常会希望使用无线网络接口作为备用接口，
以便在有线网络不可用时继续保持网络连接。 通过使用 lagg(4)， 我们可以只使用一个 IP
地址的情况下， 优先使用性能和安全性都更好的有线网络，
同时保持通过无线网络连接来传输数据的能力。

要实现这样的目的， 就需要将用于连接无线网络的物理接口的 MAC 地址修改为与所配置的 lagg(4)
一致， 后者是从主网络接口， 也就是有线网络接口， 继承而来。

在这个配置中， 我们将优先使用有线网络接口 bge0 作为主网络接口， 而将无线网络接口 wlan0
作为备用网络接口。 这里的 wlan0 使用的物理设备是 iwn0， 我们需要将它的 MAC
地址修改为与有线网络接口一致。 为了达到这个目的首先要得到有线网络接口上的 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
```

您可能需要将 bge0 改为您系统上实际使用的接口， 并从输出结果中的 ether 这行找出有线网络的
MAC 地址。 接着是修改物理的无线网络接口， iwn0：

```
# ifconfig iwn0 ether 00:21:70:da:ae:37
```

启用无线网络接口， 但不在其上配置 IP 地址：

```
# ifconfig wlan0 create wlandev iwn0 ssid my_router up
```

启用 bge0 接口。 创建 lagg(4) 接口， 其中 bge0 作为主网络接口， 而以 wlan0 作为备选接口：

```
# ifconfig bge0 up
# ifconfig lagg0 create
# ifconfig lagg0 up laggproto failover laggport bge0 laggport wlan0
```

新创建的接口的状态如下， 您系统上的 MAC 地址和设备名等可能会有所不同：

```
# 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 客户端来获取 IP 地址：

```
# dhclient lagg0
```

如果希望在系统重启时保持这些设置， 应在 /etc/rc.conf 中增加如下配置：

```
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. 无盘操作

FreeBSD 主机可以从网络启动而无需本地磁盘就可操作， 使用的是从 NFS 服务器装载的文件系统。
除了标准的配置文件，无需任何的系统修改。 很容易设置这样的系统因为所有必要的元素都很容易得到：

- 至少有两种可能的方法从网络加载内核：
  - PXE：Intel® 的先启动执行环境 (Preboot eXecution Environment) 系统是一种灵活的引导 ROM 模式，这个 ROM 内建在一些网卡或主板的中。查看 pxeboot(8) 以获取更多细节。
  - Etherboot port (net/etherboot) 产生通过网络加载内核的可 ROM 代码。这些代码可以烧入网卡上的 PROM 上，或从本地软盘 (或硬盘) 驱动器加载，或从运行着的 MS-DOS® 系统加载。它支持多种网卡。
- 一个样板脚本 (/usr/shared/examples/diskless/clone_root) 简化了对服务器上的工作站根文件系统的创建和维护。 这个脚本需要少量的自定义，但您能很快的熟悉它。
- /etc 存在标准的系统启动文件用于侦测和支持无盘的系统启动。
- 可以向 NFS 文件或本地磁盘进行交换(如果需要的话)。

设置无盘工作站有许多方法。 有很多相关的元素大部分可以自定义以适合本地情况。 以下将介绍一个完整系统的安装，强调的是简单性和与标准 FreeBSD 启动脚本的兼容。介绍的系统有以下特性：

- 无盘工作站使用一个共享的只读 / 文件系统和一个共享的只读/usr。

  root 文件系统是一份标准的 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盖。

  root 文件系统必须可写的部分被 md(4) 文件系统覆盖。 任何的改写在重启后都会丢失。

- 内核由 etherboot 或 PXE 传送和加载， 有些情况可能会指定使用其中之一。

如上所述，这个系统是不安全的。 它应该处于网络的受保护区域并不被其它主机信任。

这部分所有的信息均在 5.2.1-RELEASE 上测试过。

32.7.1. 背景信息

设置无盘工作站相对要简单而又易出错。 有时分析一些原因是很难的。例如：

- 编译时选项在运行时可能产生不同的行为。
- 出错信息经常是加密了的或根本就没有。

在这里， 涉及到的一些背景知识对于可能出现的问题的解决是很有帮助的。

要成功地引导系统还有些操作需要做。

- 机子需要获取初始的参数，如它的 IP 地址、执行文件、服务器名、根路径。这个可以使用 或 BOOTP 协议来完成。 DHCP 是 BOOTP 的兼容扩展， 并使用相同的端口和基本包格式。

  只使用 BOOTP 来配置系统也是可行的。 bootpd(8) 服务程序被包含在基本的 FreeBSD 系统里。

  不过，DHCP 相比 BOOTP 有几个好处 (更好的配置文件，使用 PXE 的可能性，以及许多其它并不直接相关的无盘操作)， 接着我们会要描述一个 DHCP 配置， 可能的话会利用与使用 bootpd(8) 相同的例子。这个样板配置会使用ISC DHCP 软件包 (3.0.1.r12 发行版安装在测试服务器上)。

- 机子需要传送一个或多个程序到本地内存。 TFTP 或 NFS 会被使用。选择TFTP 还是 NFS 需要在几个地方的"编译时间"选项里设置。 通常的错误源是为文件名指定了错误的协议：TFTP 通常从服务器里的一个单一目录传送所有文件，并需要相对这个目录的文件名。 NFS 需要的是绝对文件路径。
- 介于启动程序和内核之间的可能的部分需要被初始化并执行。 在这部分有几个重要的变量：
  - PXE 会装入 pxeboot(8)——它是 FreeBSD 第三阶段装载器的修改版。 loader(8) 会获得许多参数用于系统启动， 并在传送控制之前把它们留在内核环境里。 在这种情况下，使用 GENERIC 内核就可能了。

- Etherboot 会做很少的准备直接装载内核。 您要使用指定的选项建立 (build) 内核。

  PXE 和 Etherboot 工作得一样的好。 不过， 因为一般情况下内核希望 loader(8) 做了更多的事情，PXE 是推荐的方法。

  如果您的 BIOS 和网卡都支持 PXE， 就应该使用它。

- 最后，机子需要访问它的文件系统。 NFS 使用在所有的情况下。

查看 diskless(8) 手册页。

32.7.2. 安装说明

32.7.2.1. 配置使用ISC DHCP

ISC DHCP 服务器可以回应 BOOTP 和 DHCP 的请求。

ISC DHCP 4.2 并不属于基本系统。首先您需要安装 net/isc-dhcp42-server port 或相应的"包"。

一旦安装了 ISC DHCP， 还需要一个配置文件才能运行 (通常名叫 /usr/local/etc/dhcpd.conf)。
这里有个注释过的例子，里边主机 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值。 另外可能会增加一个 option host-name margaux 到 host 声明里。

② next-server 正式指定 TFTP 或 NFS 服务用于载入装载器或内核文件 (默认使用的是相同的主机作为DHCP 服务器)。

③ filename 正式定义这样的文件——etherboot 或 PXE 为执行下一步将装载它。 根据使用的传输方式，它必须要指定。 Etherboot 可以被编译来使用 NFS 或 TFTP。 FreeBSD port 默认配置了NFS。 PXE 使用 TFTP， 这就是为什么在这里使用相对文件名 (这可能依赖于 TFTP 服务器配置，不过会相当典型)。 同样，PXE 会装载 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 目录装载 pxeboot (因为 pxeboot(8) 能够装载 GENERIC 内核，这就使得可以使用 PXE 从远程的 CD-ROM 里启动)。

④ root-path 选项定义到根 (root) 文件系统的路径，通常是 NFS 符号。当使用 PXE 时，只要您不启用内核里的 BOOTP 选项，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

这里紧跟的是一个等效的 bootpd 配置 (减少到一个客户端)。这个可以在 /etc/bootptab 里找到。

请注意：为了使用BOOTP，etherboot 必须使用非默认选项 NO_DHCP_SUPPORT 来进行编译，而且 PXE 需要 DHCP。bootpd 的唯一可见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准备启动程序

Etherboot 的网站 包含有更多的文档 ——主要瞄准的是 Linux 系统，但无疑包含有有用的信息。 如下列出的是关于在 FreeBSD 系统里使用 Etherboot。

首先您必须安装net/etherboot 包或 port。

您可以改变 Etherboot 的配置 (如使用 TFTP 来代替 NFS)， 方法是修改 Config 文件——在 Etherboot 源目录里。

对于我们的设置，我们要使用一张启动软盘。 对于其它的方法(PROM，或 MS-DOS®程序)， 请参考 Etherboot 文档。

想要使用启动软盘，先插入一张软盘到安装有 Etherboot 的机器的驱动器里， 然后把当前路径改到 src 目录——在 Etherboot 树下， 接着输入：

```
# gmake bin32/devicetype.fd0
```

devicetype 依赖于无盘工作站上的以太网卡的类型。 参考在同一个目录下的 NIC 文件确认正确的 devicetype。

32.7.2.4. 使用PXE启动

默认地，pxeboot(8) 装载器通过 NFS 装载内核。它可以编译来使用 TFTP——通过在文件 /etc/make.conf 里指定 LOADER_TFTP_SUPPORT 选项来代替。 请参见 /usr/shared/examples/etc/make.conf 里的注释了解如何配置。

除此之外还有两个未说明的 make.conf 选项——它可能对于设置一系列控制台无盘机器会有用：BOOT_PXELDR_PROBE_KEYBOARD和 BOOT_PXELDR_ALWAYS_SERIAL。

当机器启动里，要使用 PXE， 通常需要选择 Boot from network 选项——在 BIOS 设置里， 或者在 PC 初始化的时候输入一个功能键 (function key)。

32.7.2.5. 配置 TFTP 和 NFS 服务器

如果您正在使用 PXE 或 Etherboot——配置使用了 TFTP，那么您需要在文件服务器上启用 tftpd：

1. 建立一个目录——从那里 tftpd 可以提供文件服务，如 /tftpboot。
2. 把这一行加入到 /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。 要正确执行这个命令， 在 /etc/rc.conf 文件中必须加入 inetd_enable="YES"：

```
# /etc/rc.d/inetd restart
```

您可把 tftpboot 目录放到服务器上的什何地方。 确定这个位置设置在 inetd.conf 和 dhcpd.conf 里。

在所有的情况下，您都需要启用 NFS， 并且 NFS 服务器上导出相应的文件系统。

1. 把这一行加入到/etc/rc.conf里：

```
nfs_server_enable="YES"
```

2. 通过往 /etc/exports 里加入下面几行(调整"载入点"列， 并且使用无盘工作站的名字替换 margaux corbieres)， 导出文件系统——无盘根目录存在于此：

```
/data/misc -alldirs -ro margaux corbieres
```

3. 让 mountd 重读它的配置文件。如果您真的需要启用第一步的 /etc/rc.conf 里 NFS，您可能就要重启系统了。

```
# /etc/rc.d/mountd restart
```

32.7.2.6. 建立无盘内核

如果您在使用 Etherboot， 您需要为无盘客户端建立内核配置文件， 使用如下选项(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
```

您可能也想使用 BOOTP_NFSV3， BOOT_COMPAT 和 BOOTP_WIRED_TO (参考 NOTES 文件)。

这些名字具有历史性，并且有些有些误导， 因为它们实际上启用了内核里 (它可能强制限制 BOOTP 或 DHCP 的使用)，与 DHCP 和 BOOTP 的无关的应用。

编译内核(参考配置FreeBSD的内核)， 然后将它复制到 dhcpd.conf 里指定的地方。

当使用 PXE 里， 使用以上选项建立内核并不做严格要求(尽管建议这样做)。
启用它们会在内核启动时引起更多的 DHCP
提及过的请求，带来的小小的风险是在有些特殊情况下新值和由 pxeboot(8)
取回的值之间的不一致性。 使用它们的好处是主机名会被附带设置。否则，
您就需要使用其它的方法来设置主机名，如在客户端指定的 rc.conf 文件里。

为了使带有 Etherboot 的内核可引导，就需要把设备提示 (device hint)
编译进去。通常要在配置文件(查看 NOTES 配置注释文件) 里设置下列选项：

```
hints    "GENERIC.hints"
```

32.7.2.7. 准备根(root)文件系统

您需要为无盘工作站建立根文件系统， 它就是 dhcpd.conf 里的 root-path 所指定的目录。

32.7.2.7.1. 使用 make world 来复制根文件系统

这种方法可以迅速安装一个彻底干净的系统 (不仅仅是根文件系统) 到 DESTDIR。
您要做的就是简单地执行下面的脚本：

```
#!/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，您可能需要定制 /etc/rc.conf 和 /etc/fstab——根据您的需要放到 DESTDIR里。

32.7.2.8. 配置 swap(交换)

如果需要，位于服务器上的交换文件可以通过 NFS 来访问。

32.7.2.8.1. NFS 交换区

内核并不支持在引导时启用 NFS 交换区。 交换区必须通过启动脚本启用，
其过程是挂接一个可写的文件系统， 并在其上创建并启用交换文件。 要建立尺寸合适的交换文件，
可以这样做：

```
# dd if=/dev/zero of=/path/to/swapfile bs=1k count=1 oseek=100000
```

要启用它，您须要把下面几行加到 rc.conf里：

```
swapfile=/path/to/swapfile
```

32.7.2.9. 杂项问题

32.7.2.9.1. 运行时 /usr 是只读在

如果无盘工作站是配置来支持 X， 那么您就必须调整 XDM 配置文件，因为它默认把错误信息写到 /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运行的是不 FreeBSD，您须要在 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，有时对于 /dev 里的一些特殊的文件会有问题，原因就是不同的 "最大/最小"整数大小。
一种解决的方法就是从非 FreeBSD 服务里导出一个目录， 并把它载入 FreeBSD 到机子上， 并使用
devfs(5) 来为用户透明地分派设备节点。

32.8. 从 PXE 启动一个 NFS 根文件系统

Intel® 预启动执行环境 （PXE） 能让操作系统从网络启动。 通常由近代主板的 BIOS 提供 PXE
支持，它可以通过在 BIOS 设置里选择从网络启动开启。 一个功能完整的 PXE 配置还需要正确地设置 DHCP
和 TFTP 服务。

当计算机启动的时候， 通过 DHCP 获取关于 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 下载并执行引导加载器。 这些记载于 预启动执行环境 (PXE) 规范
的 2.2.1 章节中。 在 FreeBSD 中， 在 PXE 过程中获取的引导加载器为 /boot/pxeboot。 在
/boot/pxeboot 执行之后， FreeBSD 的内核被加载， 接着是其他的 FreeBSD 相关引导部分依次被执行。
更多关于 FreeBSD 启动过程的详细信息请参阅 FreeBSD 引导过程。

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.

   选择一个可被用户 NFS 挂载并安装有 FreeBSD 的目录。 比如可以使用像
   /b/tftpboot/FreeBSD/install 这样的一个目录。

   ```
   # export NFSROOTDIR=/b/tftpboot/FreeBSD/install
   # mkdir -p ${NFSROOTDIR}
   ```

2. 使用如下的命令开启 NFS 服务 配置NFS.
3. 将下面这行加入 /etc/exports 用以通过 NFS 导出此目录：

```
/b -ro -alldirs
```

4. 重起 NFS 服务：

```
# /etc/rc.d/nfsd restart
```

5. 按照 设置 中标明的步骤启用 inetd(8)。
6. 将如下这行加入到 /etc/inetd.conf：

```
tftp dgram udp wait root /usr/libexec/tftpd tftpd -l -s /b/tftpboot
```

7. 重启 inetd：

```
# /etc/rc.d/inetd restart
```

8. 重新编译 FreeBSD 内核和用户态：

```
# cd /usr/src
# make buildworld
# make buildkernel
```

9. 把 FreeBSD 安装到 NFS 挂载目录：

```
# make installworld DESTDIR=${NFSROOTDIR}
# make installkernel DESTDIR=${NFSROOTDIR}
# make distribution DESTDIR=${NFSROOTDIR}
```

10. 测试 TFTP 服务是否能下载将从 PXE 获取的引导加载器：

```
# tftp localhost
tftp> get FreeBSD/install/boot/pxeboot
Received 264951 bytes in 0.1 seconds
```

11. 编辑 ${NFSROOTDIR}/etc/fstab 并加入以下这行挂载 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
```

此为设置从 PXE 启动的客户机的 root 密码。

13. 允许 ssh root 登录从 PXE 启动的客户机， 编辑 ${NFSROOTDIR}/etc/ssh/sshd_config 并开启 PermitRootLogin 选项。 关于此选项的说明请参阅 sshd_config(5)。
14. 对 ${NFSROOTDIR} 的 chroot(8) 环境做些其他的定制。 这可以是像使用 pkg_add(1) 安装二进制包， 使用 vipw(8) 修改密码， 或者编辑 amd.conf(8) 映射自动挂载等。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 如果检测到是从 NFS 启动便会运行 /etc/rc.initdiskless 脚本。 请阅读此脚本中的注释部分以便了解到底发生了什么。 我们需要把 /etc 和 /var 做成内存文件系统的原因是这些目录需要能被写入， 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 就会被复制进去。

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， 但是必须能够从你的网络访问到它。

1. 按照此文档处 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 选项指向你的 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。 这里一份 dhcpd.conf 实例：

```
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 客户端与调试连接问题

1. 当客户端启动的时候， 进入 BIOS 配置菜单。 设置 BIOS 从网络启动。
   如果之前你所有的配置步骤都正确的话， 那么所有部分应该能 "正常工作"。
2. 使用 net/wireshark port 查看 DHCP 和 TFTP 的网络流量来调试各种问题。
3. 确保 pxeboot 能从 TFTP 获取。 在你的 TFTP 服务器上检查 /var/log/xferlog 日志确保 pxeboot 被从正确的位置获取。 可以这样测试上面例子 dhcpd.conf 中所设置的：

   ```
   # tftp 192.168.0.1
   tftp> get FreeBSD/install/boot/pxeboot
   Received 264951 bytes in 0.1 seconds
   ```

   请阅读 tftpd(8) 和 tftp(1)。 其中的 BUGS 列出了 TFTP 的一些限制。
4. 确保根文件系统能够从 NFS 挂载。 可以这样测试上面例子 dhcpd.conf 中所设置的：

   ```
   # mount -t nfs 192.168.0.1:/b/tftpboot/FreeBSD/install /mnt
   ```
5. 阅读 src/sys/boot/i386/libi386/pxe.c 中的代码以了解 pxeboot 加载器如何设置诸如 boot.nfsroot.server 和 boot.nfsroot.path 之类的变量。 这些变量被用在了 src/sys/nfsclient/nfs_diskless.c 的 NFS 无盘根挂载代码中。
6. Read pxeboot(8) and loader(8).

32.9. ISDN

关于 ISDN 技术和硬件的一个好的资源是Dan Kegel 的 ISDN 主页。

一个快速简单的到 ISDN 的路线图如下：

- 如果您住在欧洲，您可能要查看一下 ISDN 卡部分。
- 如果您正计划首要地使用 ISDN 基于拨号非专用线路连接到带有提供商的互联网，
  您可能要了解一下终端适配器。如果您更改提供商的话， 这会给您带来最大的灵活性、最小的麻烦。
- 如果您连接了两个局域网 (LAN)，或使用了专用的 ISDN
  连线连接到互联网，您可能要考虑选择单独的路由器/网桥。

在决定选择哪一种方案的时候，价格是个很关键的因素。 下面列有从不算贵到最贵的选择：

32.9.1. ISDN 卡

FreeBSD 的 ISDN 工具通过被动卡 (passive card) 仅支持 DSS1/Q.931(或 Euro-ISDN) 标准。
此外也支持一些 active card， 它们的固件也支持其它信号协议， 这其中包括最先得到支持的 "Primary Rate (PRI) ISDN"卡。

isdn4bsd 软件允许连接到其它 ISDN 路由器，使用的是原始的 HDLC 上的 IP 或利用同步 PPP：使用带有 isppp (一个修改过的 sppp(4) 驱动程序)的 PPP 内核，或使用用户区 (userland) ppp(8)。通过使用 userland ppp(8)，两个或更多 ISDN 的 B 通道联结变得可能。 除了许多如 300 波特 (Baud) 的软 modem 一样的工具外， 还可以实现电话应答机应用。

在 FreeBSD 里，正有更多的 PC ISDN 卡被支持；
报告显示在整个欧洲及世界的其它许多地区可以成功使用。

被支持的主动型 ISDN 卡主要是带有 Infineon (以前的 Siemens) ISAC/HSCX/IPAC ISDN 芯片组，另外还有带有 Cologne (只有 ISA 总线) 芯片的 ISDN 卡、带有 Winbond W6692 芯片的 PCI 卡、一部分带有 Tiger300/320/ISAC 芯片组的卡以及带有一些商家专有的芯片组的卡 (如 AVM FritzCard PCI V.1.0 和 the AVM FritzCard PnP)。

当前积极的支持的 ISDN 卡有 AVM B1 (ISA 和 PCI) BRI 卡和 AVM T1 PCI PRI 卡。

关于 isdn4bsd 的文档，请查看 isdn4bsd的主页， 那里也有提示、勘误表以及更多的文档 (如 isdn4bsd手册)。

要是您有兴趣增加对不同 ISDN 协议的支持，对当前还不支持的 ISDN PC 卡的支持或想增强 isdn4bsd 的性能，请联系 Hellmuth Michaelis <hm@FreeBSD.org>。

对于安装、配置以及 isdn4bsd 故障排除的问题，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) 对于 ISDN 就好比 modem 对于常规电话线。

许多 TA 使用标准的 Hayes modem AT 命令集，并且可以降级来代替 modem。

TA 基本的运作同 modem 一样，不同之处是连接和整个速度更比老 modem 更快。同 modem 的安装一样，您也需要配置 PPP。确认您的串口速度已足够高。

使用 TA 连接互联网提供商的主要好处是您可以做动态的 PPP。 由于 IP 地址空间变得越来越紧张，许多提供商都不愿再提供静态 IP。许多的独立的路由器是不支持动态 IP 分配的。

TA 完全依赖于您在运行的 PPP 进程， 以完成它们的功能和稳定的连接。这可以让您在 FreeBSD 机子里轻易地从使用 modem 升级到 ISDN，要是您已经安装了 PPP 的话。只是，在您使用 PPP 程序时所体验到任何问题同时也存在。

如果您想要最大的稳定性，请使用 PPP 内核选项，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 提供商试图让他们的产品可以接受大部分的标准 modem AT 命令集。

对于外置 TA 的实际问题是：象 modem 要一样，您机子需要有一个好的串行卡。

想要更深入地理解串行设备以及异步和同步串口这间的不同点， 您就要读读 FreeBSD 串行硬件教程了。

TA 将标准的 PC 串口 (同步的) 限制到了 115.2 Kbs，即使您有 128 Kbs 的连接。 想要完全利用 ISDN 有能力达到的 128 Kbs，您就需要把 TA 移到同步串行卡上。

当心被骗去买一个内置的 TA 以及自认为可以避免同步/异步问题。内置的 TA 只是简单地将一张标准 PC 串口芯片内建在里边。 所做的这些只是让您省去买另一根串行线以及省去寻找另一个空的插孔。

带有 TA 的同步卡至少和一个独立的路由器同一样快地， 而且仅使用一个简单的 386 FreeBSD 盒驱动它。

选择同步卡/TA 还是独立的路由器，是个要高度谨慎的问题。
在邮件列表里有些相关的讨论。我们建议您去搜索一下关于完整讨论的记录。

32.9.3. 单独的 ISDN 桥/路由器

ISDN 桥或路由器根本就没有指定要 FreeBSD
或其它任何的操作系统。更多完整的关于路由和桥接技术的描述， 请参考网络指南的书籍。

这部分的内容里，路由器和桥接这两个词汇将会交替地使用。

随着 ISDN 路由器/桥的价格下滑，对它们的选择也会变得越来越流行。 ISDN
路由器是一个小盒子，可以直接地接入您的本地以太网，
并且自我管理到其它桥/路由器的连接。它有个内建的软件用于与通信——通过 PPP 和其它流行的协议。

路由器有比标准 TA 更快的吞吐量，因为它会使用完全同步的 ISDN 连接。

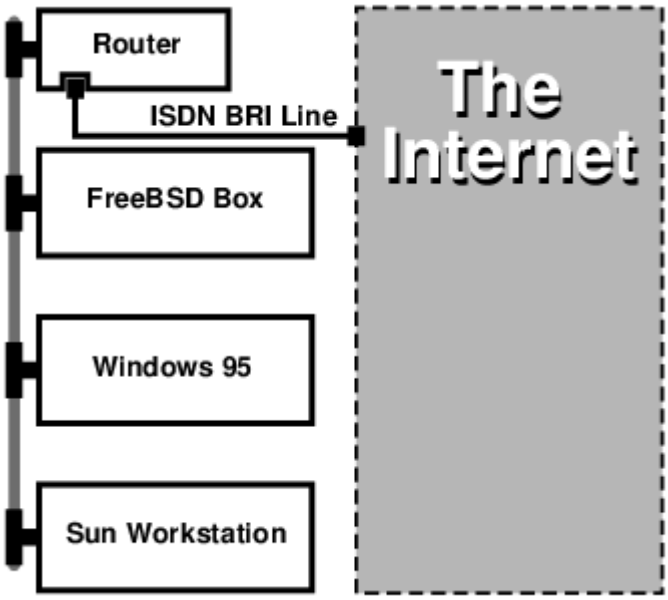
使用 ISDN 路由器和桥的主要问题是两个生产商之间的协同性仍存在问题。
如果您计划连接到互联网提供商，您应该跟他们进行交涉。

如果您计划连接两个局域网网段，如您的家庭网和办公网，
这将是最简单最低维护的解决方案。因为您买的设备是用于连接两边的， 可以保证这种连接一定会成功。

例如连接到家里的计算机，或者是办公网里的一个分支连接到办公主网， 那么下面的设置就可能用到：

例 43. 办公室局部或家庭网

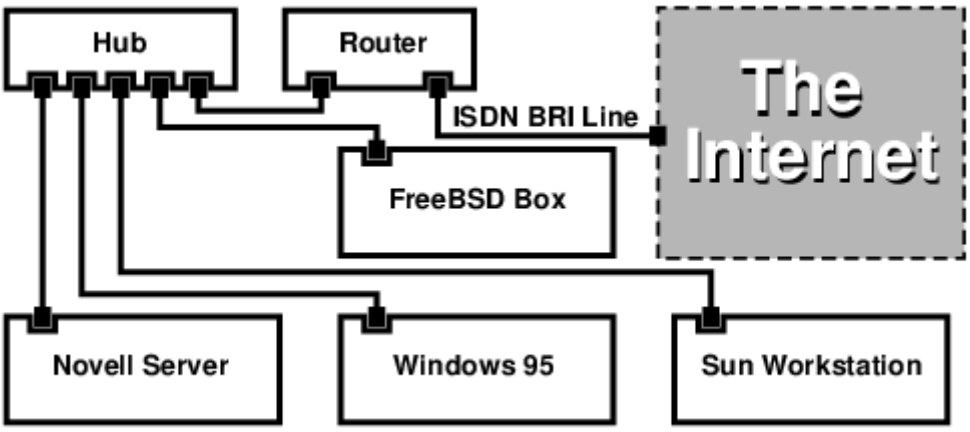
网络使用基于总线拓扑的 10 base 2 以太网 ("瘦网(thinnet)")。如果有必要，用网线连接路由器和 AUI/10BT 收发器。



如果您的家里或办公室支部里只有一台计算机，
您可以使用一根交叉的双绞线直接连接那台独立路由器。

例 44. 主办公室或其它网络

网络使用的是星形拓扑的 10 base T 以太网("双绞线")。



大部分路由器/网桥有一大好处就是，它们允许您在 同一 时间，有两个 分开独立的 PPP 连接到两个分开的点上。这点在许多的 TA 上是不支持的，
除非带有两个串口的特定模式(通常都很贵)。请不要把它与通道连接、MPP 等相混淆。

这是个非常有用的功能，例如，如果在您的办公室里您有个专有的 ISDN 连接，而且您想接入到里边，但休想让另一根 ISDN 线也能工作。
办公室里的路由器能够管理专有的B通道连接到互联网 (64 Kbps) 以及使用另一个通道 B 来完成单独的数据连接。 第二个 B 通道可以用于拨进、拨出或动态与第一个B通道进行连接 (MPP等)，以获取更大宽带。

以太网桥也允许您传输的不仅仅是 IP 通信。您也可以发送 IPX/SPX 或其它任何您所使用的协议。

32.10. 网络地址转换

32.10.1. 概要

FreeBSD 的网络地址转换服务， 通常也被叫做 natd(8)， 是一个能够接收连入的未处理 IP 包，
将源地址修改为本级地址然后重新将这些包注入到发出 IP 包流中。 natd(8) 同时修改源地址和端口，
当接收到响应数据时，它作逆向转换以便把数据发回原先的请求者。

NAT 最常见的用途是为人们所熟知的 Internet 连接共享。

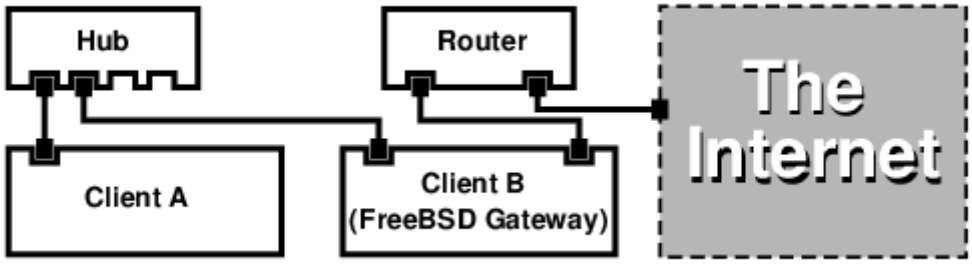
32.10.2. 安装

随着 IPv4 的 IP 地址空间的日益枯竭， 以及使用如 DSL 和电缆等高速连接的用户的逐渐增多，
越来越多的人开始需要 Internet 连接共享这样的解决方案。 由于能够将许多计算机通过一个对外的 IP 地址进行接入， natd(8) 成为了一个理想的选择。

更为常见的情况， 一个用户通过电缆或者 DSL 线路 接入，并拥有一个 IP 地址，同时，希望通过这台接入 Internet 的计算机来为 LAN 上更多的计算机提供接入服务。

为了完成这一任务， 接入 Internet 的 FreeBSD 机器必须扮演网关的角色。 这台网关必须有两块网卡 - 一块用于连接 Internet 路由器， 另一块用来连接 LAN。 所有 LAN 上的机器通过 Hub 或交换机进行连接。

有多种方法能够通过 FreeBSD 网关将 LAN 接入 Internet。
这个例子只介绍了有至少两块网卡的网关。



上述配置被广泛地用于共享 Internet 连接。 LAN 中的一台机器连接到 Internet 中。
其余的计算机则通过那台 "网关" 机来连接 Internet。

32.10.3. 引导加载器配置

在默认的 GENERIC 内核中， 并没有启用通过 natd(8) 进行网址翻译的功能， 不过， 这一功能可以通过在 /boot/loader.conf 中添加两项配置来在引导时自动予以加载：

```
ipfw_load="YES"
ipdivert_load="YES"
```

此外， 还可以将引导加载器变量 net.inet.ip.fw.default_to_accept 设为 1：

```
net.inet.ip.fw.default_to_accept="1"
```

在刚开始配置防火墙和 NAT 网关时， 增加这个配置是个好主意。 默认的 ipfw(8) 规则将是 allow ip from any to any 而不是默认的 deny ip from any to any， 这样， 在系统重启时， 也就不太容易被反锁在外面。

32.10.4. 内核配置

当不能使用内核模块， 或更希望将全部需要的功能联编进内核时，
可以在内核配置中添加下面的设置来实现：

```
options IPFIREWALL
options IPDIVERT
```

此外，下列是一些可选的选项：

```
options IPFIREWALL_DEFAULT_TO_ACCEPT
options IPFIREWALL_VERBOSE
```

32.10.5. 系统引导时的配置

如果希望在系统引导过程中启用防火墙和 NAT 支持， 应在 /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 将机器配置为网关。 执行 sysctl net.inet.ip.forwarding=1 效果相同。

② 在启动时启用 /etc/rc.firewall 中的防火墙规则。

③ 指定一个预定义的允许所有包进入的防火墙规则集。 参见 /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转发包 (接入 Internet 的那一个)。

⑤ 其他希望在启动时传递给 natd(8) 的参数。

在 /etc/rc.conf 中加入上述选项将在系统启动时运行 natd -interface fxp0。 这一工作也可以手工完成。

当有太多选项要传递时，也可以使用一个 natd(8) 的配置文件来完成。这种情况下，这个配置文件必须通过在 /etc/rc.conf 里增加下面内容来定义：

```
natd_flags="-f /etc/natd.conf"
```

/etc/natd.conf 文件会包含一个配置选项列表，每行一个。在紧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
```

关于配置文件的更多信息，参考 natd(8) 手册页中关于 -f 选项那一部分。

在LAN后面的每一台机子和接口应该被分配私有地址空间(由RFC 1918定义) 里的 IP 地址，并且默认网关设成 natd 机子的内连 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，同时 natd 机子的 LAN 接口上的 IP 地址是 192.168.0.1。客户端 A 和 B 的默认网关必须要设成 natd 机子的 IP——192.168.0.1。natd 机子外连，或互联网接口不需要为了 natd(8) 而做任何特别的修改就可工作。

32.10.6. 端口重定向

使用 natd(8) 的缺点就是 LAN 客户不能从互联网访问。LAN 上的客户可以进行到外面的连接，而不能接收进来的连接。如果想在 LAN 的客户端机子上运行互联网服务，这就会有问题。 对此的一种简单方法是在 natd 机子上重定向选定的互联网端口到 LAN 客户端。

例如：在客户端 A 上运行 IRC 服务，而在客户端 B 上运行 web 服务。 想要正确的工作，在端口 6667 (IRC) 和 80 (web) 上接收到的连接就必须重定向到相应的机子上。

-redirect_port 需要使用适当的选项传送给 natd(8)。语法如下：

```
-redirect_port proto targetIP:targetPORT[-targetPORT]
        [aliasIP:]aliasPORT[-aliasPORT]
        [remoteIP[:remotePORT[-remotePORT]]]
```

在上面的例子中，参数应该是：

```
-redirect_port tcp 192.168.0.2:6667 6667
-redirect_port tcp 192.168.0.3:80 80
```

这就会重定向适当的 tcp 端口到 LAN 上的客户端机子。

-redirect_port 参数可以用来指出端口范围来代替单个端口。例如， tcp 192.168.0.2:2000-3000 2000-3000 就会把所有在端口 2000 到 3000 上接收到的连接重定向到主机 A 上的端口 2000 到 3000。

当直接运行 natd(8) 时，就可以使用这些选项， 把它们放到 /etc/rc.conf 里的 natd_flags="" 选项上，或通过一个配置文件进行传送。

想要更多配置选项，请参考 natd(8)。

32.10.7. 地址重定向

如果有几个 IP 地址提供，那么地址重定向就会很有用， 然而他们必须在一个机子上。使用它，natd(8) 就可以分配给每一个 LAN 客户端它们自己的外部 IP 地址。natd(8) 然后会使用适当的处部 IP 地址重写从 LAN 客户端外出的数据包， 以及重定向所有进来的数据包——一定的 IP 地址回到特定的 LAN 客户端。这也叫做静态 NAT。例如，IP 地址 128.1.1.1、 128.1.1.2 和 128.1.1.3 属于 natd 网关机子。128.1.1.1 可以用来作 natd 网关机子的外连 IP 地址，而 128.1.1.2 和 128.1.1.3 用来转发回 LAN 客户端 A 和 B。

-redirect_address 语法如下：

```
-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。

在这个例子里，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 一样，这些参数也是放在 /etc/rc.conf 里的 natd_flags="" 选项上，或通过一个配置文件传送给它。使用地址重定向， 就没有必要用端口重定向了，因为所有在某个 IP 地址上收到的数据都被重定向了。

在 natd 机子上的外部 IP 地址必须激活并且别名到 (aliased) 外连接口。要这做就看看 rc.conf(5)。

32.11. 并口电缆 IP (PLIP)

PLIP 允许我们在两个并口间运行 TCP/IP。 在使用笔记本电脑， 或没有网卡的计算机时， 这会非常有用。这一节中， 我们将讨论：

- 制作用于并口的 (laplink) 线缆。
- 使用 PLIP 连接两台计算机。

32.11.1. 制作并口电缆。

您可以在许多计算机供应店里买到并口电缆。 如果买不到， 或者希望自行制作， 则可以参阅下面的表格，它介绍了如何利用普通的打印机并口电缆来改制：

表 14. 用于网络连接的并口电缆接线方式

A-name	A 端	B 端	描述	Post/Bit
.... DATA0 -ERROR	.... 2 15	.... 15 2	数据	.... 0/0x01 1/0x08 ....
.... DATA1 +SLCT	.... 3 13	.... 13 3	数据	.... 0/0x02 1/0x10 ....
.... DATA2 +PE	.... 4 12	.... 12 4	数据	.... 0/0x04 1/0x20 ....
.... DATA3 -ACK	.... 5 10	.... 10 5	脉冲 (Strobe)	.... 0/0x08 1/0x40 ....
.... DATA4 BUSY	.... 6 11	.... 11 6	数据	.... 0/0x10 1/0x80 ....
GND	18-25	18-25	GND	-

32.11.2. 设置 PLIP

首先，您需要一根 laplink 线。然后， 确认两台计算机的内核都有对 lpt(4) 驱动程序的支持：

```
# grep lp /var/run/dmesg.boot
lpt0: Printer on ppbus0
lpt0: Interrupt-driven port
```

并口必须是一个中断驱动的端口， 您应在 /boot/device.hints 文件中配置：

```
hint.ppc.0.at="isa"
hint.ppc.0.irq="7"
```

然后检查内核配置文件中是否有一行 device plip 或加载了 plip.ko 内核模块。 这两种情况下， 在使用 ifconfig(8) 命令时都会显示并口对应的网络接口， 类似这样：

```
# ifconfig plip0
plip0: flags=8810<POINTOPOINT,SIMPLEX,MULTICAST> mtu 1500
```

用 laplink 线接通两台计算机的并口。

在两边以 root 身份配置通讯参数。 例如， 如果你希望将 host1 通过另一台机器 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
```

您现在应该有个工作的连接了。想要更详细的信息， 请阅读 lp(4) 和 lpt(4) 手册页。

您还应该增加两个主机到 /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
```

要确认连接是否工作，可以到每一台机子上，然后 ping 另外一台。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。 和其他现代的 *BSD 系统一样， FreeBSD 包含了 KAME 的 IPv6 参考实现。 因此， 您的 FreeBSD 系统包含了尝试 IPv6 所需要的所有工具。 这一节主要集中讨论如何配置和使用 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，

主要的两个担心是：

- 用尽所有的地址。 当然现在这个问题已经不再那样尖锐， 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) 技术已经被广泛采用。
- 路由表条目变得太大。这点今天仍然是焦点。

IPv6 解决这些和其它许多的问题：

- 128 位地址空间。换句话，理论上有 340,282,366,920,938,463,463,374,607,431,768,211,456 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。
- 路由器仅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 还有其它许多有用的功能，如：

- 地址自动配置 (RFC2462)
- Anycast (任意播) 地址("一对多")
- 强制的多播地址
- IPsec (IP 安全)
- 简单的头结构
- 移动的 (Mobile) IP
- IPv6 到 IPv4 的转换机制

要更多信息，请查看：

- IPv6 概观，在playground.sun.com
- KAME.net

32.12.1. 关于 IPv6 地址的背景知识

有几种不同类型的 IPv6 地址：Unicast，Anycast 和 Multicast。

Unicast 地址是为人们所熟知的地址。一个被发送到 unicast 地址的包实际上会到达属于这个地址的接口。

Anycast 地址语义上与 unicast 地址没有差别， 只是它们强调一组接口。指定为 anycast 地址的包会到达最近的 (以路由为单位) 接口。Anycast 地址可能只被路由器使用。

Multicast 地址标识一组接口。指定为 multicast 地址的包会到达属于 multicast 组的所有的接口。

IPv4 广播地址 (通常为 xxx.xxx.xxx.255) 由 IPv6 的 multicast 地址来表示。

表 15. 保留的 IPv6 地址

IPv6 地址	预定长度 (bits)	描述	备注
::	128 bits	未指定	类似 IPv4 中的 0.0.0.0
::1	128 bits	环回地址	类似 IPv4 中的 127.0.0.1
::00:xx:xx:xx:xx	96 bits	嵌入的 IPv4	低 32 bits 是 IPv4 地址。这也称作 "IPv4 兼容 IPv6 地址"
::ff:xx:xx:xx:xx	96 bits	IPv4 影射的 IPv6 地址	低的 32 bits 是 IPv4 地址。 用于那些不支持 IPv6 的主机。
fe80:: - feb::	10 bits	链路环回	类似 IPv4 的环回地址。
fec0:: - fef::	10 bits	站点环回	
ff::	8 bits	多播	

IPv6 地址	预定长度 (bits)	描述	备注
001 (base 2)	3 bits	全球多播	所有的全球多播地址都指定到这个地址池中。前三个二进制位是 "001"。

32.12.2. IPv6 地址的读法

规范形式被描述为：x:x:x:x:x:x:x:x， 每一个"x"就是一个 16 位的 16 进制值。当然，
每个十六进制块以三个"0"开始头的也可以省略。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， 因此每个地址的这种子串常被简写为"::"。 例如：fe80::1
对应的规范形式是 fe80:0000:0000:0000:0000:0000:0000:0001。

第三种形式是以众所周知的用点"."作为分隔符的十进制 IPv4 形式，写出最后 32 Bit 的部分。例如
2002::10.0.0.1 对应的十进制正规表达方式是 2002:0000:0000:0000:0000:0000:0a00:0001
它也相当于写成 2002::a00:1.

到现在，读者应该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 0xffffff00 broadcast 10.0.0.255
        inet6 fe80::200:21ff:fe03:8e1%rl0 prefixlen 64 scopeid 0x1
        ether 00:00:21:03:08:e1
        media: Ethernet autoselect (100baseTX )
        status: active
```

fe80::200:21ff:fe03:8e1%rl0 是一个自动配置的链路环回地址。它作为自动配置的一部分由 MAC 生成。

关于 IPv6 地址的结构的更多信息，请参看 RFC3513。

32.12.3. 进行连接

目前，有四种方式可以连接到其它 IPv6 主机和网络：

- 咨询你的互联网服务提供商是否提供 IPv6。
- SixXS 向全球范围提供通道。
- 使用 6-to-4 通道 (RFC3068)
- 如果您使用的是拨号连接， 则可以使用 net/freenet6 port。

32.12.4. IPv6 世界里的 DNS

对于 IPv6 有两种类型的 DNS 记录：IETF 已经宣布 A6 是过时标准；现行的标准是 AAAA 记录。

使用AAAA记录是很简单的。通过增加下面内容， 给您的主机分配置您刚才接收到的新的 IPv6 地址：

```
MYHOSTNAME      AAAA   MYIPv6ADDR
```

到您的主域 DNS 文件里，就可以完成。要是您自已没有 DNS 域服务，您可以询问您的 DNS 提供商。目前的 bind 版本 (version 8.3 与 9) 和 dns/djbdns(含IPv6补丁) 支持 AAAA 记录。

32.12.5. 在 /etc/rc.conf 中进行所需的修改

32.12.5.1. IPv6 客户机设置

这些设置将帮助您把一台您 LAN 上的机器配置为一台客户机， 而不是路由器。 要让 rtsol(8) 在启动时自动配置您的网卡， 只需添加：

```
ipv6_enable="YES"
```

要自动地静态指定 IP 地址， 例如 2001:471:1f11:251:290:27ff:fee0:2093， 到 fxp0 上， 则写上：

```
ipv6_ifconfig_fxp0="2001:471:1f11:251:290:27ff:fee0:2093"
```

要指定 2001:471:1f11:251::1 作为默认路由， 需要在 /etc/rc.conf 中加入：

```
ipv6_defaultrouter="2001:471:1f11:251::1"
```

32.12.5.2. IPv6 路由器/网关配置

这将帮助您从隧道提供商那里取得必要的资料， 并将这些资料转化为在重启时能够保持住的设置。 要在启动时恢复您的隧道， 需要在 /etc/rc.conf 中增加：

列出要配置的通用隧道接口， 例如 gif0：

```
gif_interfaces="gif0"
```

配置该接口使用本地端地址 MY_IPv4_ADDR 和远程端地址 REMOTE_IPv4_ADDR：

```
gifconfig_gif0="MY_IPv4_ADDR REMOTE_IPv4_ADDR"
```

应用分配给您用于 IPv6 隧道远端的 IPv6 地址， 需要增加：

```
ipv6_ifconfig_gif0="MY_ASSIGNED_IPv6_TUNNEL_ENDPOINT_ADDR"
```

此后十设置 IPv6 的默认路由。 这是 IPv6 隧道的另一端：

```
ipv6_defaultrouter="MY_IPv6_REMOTE_TUNNEL_ENDPOINT_ADDR"
```

32.12.5.3. IPv6 隧道配置

如果服务器将您的网络通过 IPv6 路由到世界的其他角落， 您需要在 /etc/rc.conf 中添加下面的配置：

```
ipv6_gateway_enable="YES"
```

32.12.6. 路由宣告和主机自动配置

这节将帮助您配置 rtadvd(8) 来宣示默认的 IPv6 路由。

要启用 rtadvd(8) 您需要在 /etc/rc.conf 中添加：

```
rtadvd_enable="YES"
```

指定由哪个网络接口来完成 IPv6 路由请求非常重要。 举例来说， 让 rtadvd(8) 使用 fxp0：

```
rtadvd_interfaces="fxp0"
```

接下来我们需要创建配置文件， /etc/rtadvd.conf。 示例如下：

```
fxp0:\
  :addrs#1:addr="2001:471:1f11:246::":prefixlen#64:tc=ether:
```

将 fxp0 改为您打算使用的接口名。

接下来， 将 2001:471:1f11:246:: 改为分配给您的地址前缀。

如果您拥有专用的 /64 子网， 则不需要修改其他设置。 反之， 您需要把 prefixlen# 改为正确的值。

32.13. 异步传输模式 (ATM)

32.13.1. 配置 classical IP over ATM (PVCs)

Classical IP over ATM (CLIP) 是一种最简单的使用带 IP 的 ATM 的方法。 这种方法可以用在交换式连接 (SVC) 和永久连接 (PVC) 上。这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

第一种使用PVC来设置 CLIP 的方式就是通过专用的 PVC 让网络里的每一台机子都互连在一起。
尽管这样配置起来很简单，但对于数量更多一点的机子来说就有些不切实际了。
例如我们有四台机子在网络里，每一台都使用一张 ATM 适配器卡连接到 ATM 网络。第一步就是规划 IP 地址和机子间的 ATM 连接。我们使用下面的：

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

为了建造完全交错的网络，我们需要在第一对机子间有一个 ATM 连接：

机器	VPI.VCI 对
hostA - hostB	0.100
hostA - hostC	0.101
hostA - hostD	0.102
hostB - hostC	0.103

机器	VPI.VCI 对
hostB - hostD	0.104
hostC - hostD	0.105

在每一个连接端 VPI 和 VCI 的值都可能会不同， 只是为了简单起见，我们假定它们是一样的。
下一步我们需要配置每一个主机上的 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。 现在 PVC 需要配置到 hostA 上 (我们假定它们都已经配置在了 ATM 交换机上，至于怎么做的， 您就需要参考一下该交换机的手册了)。

```
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 外其它的通信协定也可让 ATM 适配器支持这些。
此种情况下，通信协定的名字要跟人通信参数后边。工具 atmconfig(8) 的帮助可以这样得到：

```
# atmconfig help natm add
```

或者在 atmconfig(8) 手册页里得到。

相同的配置也可以通过 /etc/rc.conf 来完成。对于 hostA，看起来就象这样：

```
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 路由的当前状态可以使用如下命令获得：

```
hostA# atmconfig natm show
```

32.14. Common Address Redundancy Protocol (CARP，共用地址冗余协议)

Common Address Redundancy Protocol， 或简称 CARP 能够使多台主机共享同一 IP 地址。
在某些配置中， 这样做可以提高可用性， 或实现负载均衡。 下面的例子中，
这些主机也可以同时使用其他的不同的 IP 地址。

要启用 CARP 支持， 必须在 FreeBSD 内核配置中增加下列选项， 并按照 配置FreeBSD的内核
章节介绍的方法重新联编内核：

```
device  carp
```

另外的一个方法是在启动时加载 if_carp.ko 模块。 把如下的这行加入到 /boot/loader.conf：

```
if_carp_load="YES"
```

这样就可以使用 CARP 功能了， 一些具体的参数， 可以通过一系列 sysctlOID 来调整。

OID	描述
net.inet.carp.allow	接受进来的 CARP 包。 默认启用。
net.inet.carp.preempt	当主机中有一个 CARP 网络接口失去响应时， 这个选项将停止这台主机上所有的 CARP 接口。 默认禁用。
net.inet.carp.log	当值为 0 表示禁止记录所有日志。 值为 1 表示记录损坏的 CARP 包。任何大于 1 表示记录 CARP 网络接口的状态变化。默认值为 1。
net.inet.carp.arpbalance	使用 ARP 均衡本地网络流量。 默认禁用。
net.inet.carp.suppress_preempt	此只读 OID 显示抑制抢占的状态。 如果一个接口上的连接失去响应, 则抢占会被抑制。 当这个变量的值为 0 时，表示抢占未被抑制。 任何问题都会使 OID 递增。

CARP 设备可以通过 ifconfig 命令来创建。

```
# ifconfig carp0 create
```

在真实环境中， 这些接口需要一个称作 VHID 的标识编号。 这个 VHID 或 Virtual Host Identification
(虚拟主机标识) 用于在网络上区分主机。

32.14.1. 使用 CARP 来改善服务的可用性 (CARP)

如前面提到的那样， CARP 的作用之一是改善服务的可用性。 这个例子中，
将为三台主机提供故障转移服务， 这三台服务器各自有独立的 IP 地址， 并提供完全一样的 web 内容。
三台机器以 DNS 轮询的方式提供服务。 用于故障转移的机器有两个 CARP 接口，
分别配置另外两台服务器的 IP 地址。 当有服务器发生故障时， 这台机器会自动得到故障机的 IP 地址。
这样以来， 用户就完全感觉不到发生了故障。 故障转移的服务器提供的内容和服务，
应与其为之提供热备份的服务器一致。

两台机器的配置， 除了主机名和 VHID 之外应完全一致。 在我们的例子中， 这两台机器的主机名分别是 hosta.example.org 和 hostb.example.org。 首先， 需要将 CARP 配置加入到 rc.conf。 对于 hosta.example.org 而言， rc.conf 文件中应包含下列配置：

```
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， 对应的 rc.conf 配置则是：

```
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"
```

在两台机器上由 ifconfig 的 pass 选项指定的密码必须是一致的， 这一点非常重要。 carp 设备只会监听和接受来自持有正确密码的机器的公告。 此外， 不同虚拟主机的 VHID 必须不同。

第三台机器， provider.example.org 需要进行配置， 以便在另外两台机器出现问题时接管。
这台机器需要两个 carp 设备， 分别处理两个机器。 对应的 rc.conf 配置类似下面这样：

```
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"
```

配置两个 carp 设备， 能够让 provider.example.org 在两台机器中的任何一个停止响应时， 立即接管其 IP 地址。

默认的 FreeBSD 内核 可能 启用了主机间抢占。 如果是这样的话，
provider.example.org 可能在正式的内容服务器恢复时不释放 IP 地址。 此时，
管理员必须手工强制 IP 回到原来内容服务器。 具体做法是在 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现问题的主机对应的那个 carp 接口上进行。

现在您已经完成了 CARP 的配置， 并可以开始测试了。 测试过程中，
可以随时重启或切断两台机器的网络。

如欲了解更多细节， 请参见 carp(4) 联机手册。

Part V: 附录

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买到 FreeBSD 的盒装产品 (FreeBSD CD， 附加软件，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 光盘可以从许多在线零售商那里买到：

- FreeBSD Mall, Inc.
  700 Harvest Park Ste F
  Brentwood, CA 94513
  USA
  Phone: +1 925 240-6652
  Fax: +1 925 674-0821
  Email: <info@freebsdmall.com>
  WWW: http://www.freebsdmall.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>
  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>
  WWW: http://linuxcenter.ru/shop/freebsd

A.1.3. 发行人

如果您是销售商并且想销售 FreeBSD CDROM 产品， 请和发行人联系：

- Cylogistics
  809B Cuesta Dr., #2149
  Mountain View, CA 94040
  USA
  Phone: +1 650 694-4949
  Fax: +1 650 694-4953
  Email: <sales@cylogistics.com>
  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@kudzuenterprises.com>

- LinuxCenter.Kz
  Ust-Kamenogorsk
  Kazakhstan
  Phone: +7-705-501-6001
  Email: <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>
  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 源代码可以从遍布全球的镜像站点 通过匿名 FTP 下载。 站点
ftp://ftp.FreeBSD.org/pub/FreeBSD/ 有着良好的网络连接并且允许大量的并发连接， 但是
您或许更想找一个 "更近的" 镜像站点 (特别是当您想进行某种形式的镜像的时候)。

FreeBSD 可以从下面这些镜像站点通过匿名 FTP 下载。如果您选择了通过匿名 FTP 获取 FreeBSD，
请尽量使用离您比较近的站点。被列为 "主镜像站点" 的镜像站点一般都有完整的 FreeBSD 文件
(针对每种体系结构的所有当前可用的版本)，
您或许从您所在的国家或地区的站点下载会得到更快的下载速度。
每个站点提供了最流行的体系结构的最近的版本而有可能不提供完整的 FreeBSD
存档。所有的站点都提供匿名 FTP 访问而有些站点也提供其他的访问方式。对每个站点可用的访问方式
在其主机名后有所说明。

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> 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> 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> for this domain.

- ftp://ftp.au.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp2.au.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp3.au.FreeBSD.org/pub/FreeBSD/ (ftp)

Austria

In case of problems, please contact the hostmaster <hostmaster@at.FreeBSD.org> for this domain.

- ftp://ftp.at.FreeBSD.org/pub/FreeBSD/ (ftp / ftpv6 / http://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> 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> 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/ / 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> 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> for this domain.

- ftp://ftp.ee.FreeBSD.org/pub/FreeBSD/ (ftp)

Finland

In case of problems, please contact the hostmaster <hostmaster@fi.FreeBSD.org> for this domain.

- ftp://ftp.fi.FreeBSD.org/pub/FreeBSD/ (ftp)

France

In case of problems, please contact the hostmaster <hostmaster@fr.FreeBSD.org> for this domain.

- ftp://ftp.fr.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp1.fr.FreeBSD.org/pub/FreeBSD/ (ftp / http://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> 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> 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> 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> for this domain.

- ftp://ftp.jp.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp2.jp.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp3.jp.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp4.jp.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp5.jp.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp6.jp.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp7.jp.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp8.jp.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp9.jp.FreeBSD.org/pub/FreeBSD/ (ftp)

Korea

In case of problems, please contact the hostmaster <hostmaster@kr.FreeBSD.org> for this domain.

- ftp://ftp.kr.FreeBSD.org/pub/FreeBSD/ (ftp / rsync)
- ftp://ftp2.kr.FreeBSD.org/pub/FreeBSD/ (ftp / http://ftp2.kr.FreeBSD.org/pub/FreeBSD/)

Latvia

In case of problems, please contact the hostmaster <hostmaster@lv.FreeBSD.org> for this domain.

- ftp://ftp.lv.FreeBSD.org/pub/FreeBSD/ (ftp / http://ftp.lv.FreeBSD.org/pub/FreeBSD/)

Lithuania

In case of problems, please contact the hostmaster <hostmaster@lt.FreeBSD.org> for this domain.

- ftp://ftp.lt.FreeBSD.org/pub/FreeBSD/ (ftp / http://ftp.lt.FreeBSD.org/pub/FreeBSD/)

Netherlands

In case of problems, please contact the hostmaster <hostmaster@nl.FreeBSD.org> for this domain.

- ftp://ftp.nl.FreeBSD.org/pub/FreeBSD/ (ftp / http://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> 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> for this domain.

- ftp://ftp.pl.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp2.pl.FreeBSD.org

Russia

In case of problems, please contact the hostmaster <hostmaster@ru.FreeBSD.org> for this domain.

- ftp://ftp.ru.FreeBSD.org/pub/FreeBSD/ (ftp / http://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> 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> 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> 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> 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> 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> 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> for this domain.

- ftp://ftp.ch.FreeBSD.org/pub/FreeBSD/ (ftp / ftp://ftp.tw.FreeBSD.org/pub/FreeBSD/ / rsync / rsyncv6)
- 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)
- 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 / 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> for this domain.

- ftp://ftp.uk.FreeBSD.org/pub/FreeBSD/ (ftp)
- ftp://ftp2.uk.FreeBSD.org/pub/FreeBSD/ (ftp / 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> 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(或人们常说的 anoncvs)是由和 FreeBSD 附带的 CVS 实用工具提供的用于和远程的 CVS 代码库同步的一种特性。 尤其是， 它允许 FreeBSD 用户不需要特殊的权限对任何一台 FreeBSD 项目的官方 anoncvs 服务器执行只读的 CVS 操作。 要使用它，简单的设置 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仓库一样来访问它。

cvs login 命令把用来登录 CVS 服务器的密码储存在您的 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。 请创建一个空的 .cvspass 文件，然后试试重新登录。

也可以这么说 CVSup 和 anoncvs 服务本质上提供了同样的功能，但是有各种各样 不同的场合可以影响用户对同步方式的选择。简单来说， CVSup 在网络资源利用方面 更加有效，而且是到目前为止在两者之中技术上更成熟的 除了成本方面。要使用 CVSup，在下载任何东西之前 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块的 CVSup 称作

collections。

相比之下，anoncvs 可以通过 CVS 模块名来从单个文件里检出任何东西并赋给特定的程序 (比如 ls 或者 grep)。 当然，anoncvs 也只适用于对 CVS 仓库的只读操作，所以如果您是想用和 FreeBSD 项目共享的仓库提供本地开发的话， CVSup 几乎是您唯一的选择。

A.3.2. 使用匿名 CVS

配置 cvs(1) 使用匿名 CVS 仓库可以简单的设定 CVSROOT 环境变量指向 FreeBSD 项目的 anoncvs 服务器之一。 到此书写作为止， 下面的服务器都是可用的：

- 法国: :pserver:anoncvs@anoncvs.fr.FreeBSD.org:/home/ncvs (使用 pserver 模式时， 用 cvs login 配合口令 "anoncvs" 来登录。 如果使用的是 ssh， 则不需要口令。)
- 台湾地区: :pserver:anoncvs@anoncvs.tw.FreeBSD.org:/home/ncvs (使用 pserver 模式时， 用 cvs login 配合口令 "anoncvs" 来登录。 如果使用的是 ssh， 则不需要口令。)

  ```
  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 时， 请使用协议版本 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
  ```

因为 CVS 实际上允许 "检出" 曾经存在的 (或者，某种情况下将会存在) FreeBSD 源代码的任意版本，您需要熟悉 cvs(1) 的版本 (-r) 参数， 以及在 FreeBSD 代码库中可用的值。

有两种标签，修订标签和分支标签。 修订标签特指一个特定的修订版本。含义始终是不变的。分支标签，另一方面， 指代给定时间给定开发分支的最新修订， 因为分支标签不涉及特定的修订版本，它明天所代表的含义就可能和今天的不同。

CVS 标签 包括了用户可能感兴趣的 修订标签。 请注意， 这些标签并不适用于 Ports Collection，因为它并不包含多个开发分支。

当您指定一个分支标签，您通常会得到那个开发分支的文件的最新版本。如果您希望得到一些旧的版本，您可以用 -D date 标记制定一个日期。 察看 cvs(1) 手册页了解更多细节。

A.3.3. 示例

在这之前强烈建议您通读 cvs(1) 的手册页， 这里有一些简单的例子来展示如何使用匿名 CVS：

例 45. 从 -CURRENT 检出些东西 (ls(1))：

```
% setenv CVSROOT :pserver:anoncvs@anoncvs.tw.FreeBSD.org:/home/ncvs
% cvs login
>在提示符处，输入任意密码 “password”.
% cvs co ls
```

例 46. 通过 SSH 检出整个 src/ 代码树：

```
% 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. 检出 8-STABLE 分支中的 ls(1) 版本：

```
% setenv CVSROOT :pserver:anoncvs@anoncvs.tw.FreeBSD.org:/home/ncvs
% cvs login
在提示符处，输入任意密码 “password”。
% cvs co -rRELENG_8 ls
```

例 48. 创建 ls(1) 的变化列表(用标准的 diff)

```
% setenv CVSROOT :pserver:anoncvs@anoncvs.tw.FreeBSD.org:/home/ncvs
% cvs login
在提示符处，输入任意密码 “password”。
% cvs rdiff -u -rRELENG_8_0_0_RELEASE -rRELENG_8_1_0_RELEASE ls
```

例 49. 找出可以使用的其它的模块名：

```
% setenv CVSROOT :pserver:anoncvs@anoncvs.tw.FreeBSD.org:/home/ncvs
% cvs login
在提示符处，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对学习 CVS 有帮助：

- CVS 教程，来自加州州立理工大学。
- CVS 主页，CVS 开发和支持社区。
- CVSweb 是 FreeBSD 项目的 CVS web 界面。

A.4. 使用 CTM

CTM 是保持远程目录树和中央服务器目录树同步的一种方法。 它被开发用于 FreeBSD

的源代码树，虽然其他人随着时间推移会发现它可以用于其他目的。
当前几乎没有，也或者只有很少的文档讲述创建 deltas 的步骤， 所以如果您希望使用 CTM 去做其它事情，
请联系 ctm-users-desc 邮件列表了解更多信息。

A.4.1. 为什么我该使用 CTM?

CTM 会给您一份 FreeBSD 源代码树的本地副本。 代码树有很多的 "flavors"
可用。不管您是希望跟踪完整的 CVS 树还是只是一个分支， CTM 都会给您提供信息。 如果您是 FreeBSD
上的一个活跃的开发者，但是缺乏或者不存在 TCP/IP 连接，或者只是希望把变化自动发送给您， CTM
就是适合您的。对于最积极的分支 您将会每天获得三个以上的 deltas。
然而，您应该考虑通过邮件来自动发送。 升级的大小总是保证尽可能的小。 通常小于
5K，也偶然(十分之一可能)会有 10-50K，也不时地有个大的 100K+ 甚至更大的。

您也需要让自己了解直接和开发代码而不是预发行版本打交道的各种警告。这种情况会很显著，
如果您选择了 "current" 代码的话。强烈建议您阅读和 FreeBSD 保持同步。

A.4.2. 使用 CTM 我需要做什么?

您需要两样东西：CTM 程序，还有初始的 deltas 来 feed it(达到 "current" 级别)。

CTM 程序从版本 2.0 发布以来 已经是 FreeBSD 的一部分了，如果您安装了源代码副本的话， 它位于
/usr/src/usr.sbin/ctm。

您喂给 CTM 的 "deltas" 可以有两种方式，FTP 或者 email。 如果您有普通的访问 Internet 的 FTP 权限，
那么下面的 FTP 站点支持访问 CTM：

ftp://ftp.FreeBSD.org/pub/FreeBSD/CTM/

或者看看这一小节镜像。

FTP 访问相关的目录并取得 README 文件，从那里开始。

如果您希望通过 email 得到您的 deltas：

订阅一个 CTM 分发列表。 ctm-src-cur-desc 支持最新的开发分支。 ctm-src-7-desc 支持 7.X 发行分支，
等等。。(如果您不知道如何订阅邮件列表， 点击上面的列表名或者到 https://lists.freebsd.org
点击您希望订阅的列表。 列表页包含了所有必要的订阅指导。)

当您开始接收到您邮件中的 CTM 升级时，您可以使用 ctm_rmail 程序来解压并应用它们。
事实上如果您想要让进程以全自动的形式运行的话，您可以通过在 /etc/aliases 中设置直接使用 ctm_rmail
程序。 查看 ctm_rmail 手册页了解更多细节。

不管您使用什么方法得到 CTM deltas，您都应该订阅 ctm-announce-desc 邮件列表。
以后会有单独的地方提交有关 CTM 系统的操作的公告。
点击上面的邮件列表名并按照指示订阅邮件列表。

A.4.3. 第一次使用 CTM

在您开始使用 CTM delta 之前，您需要获得一个起始点。

首先您应该确定您已经有了什么。每个人都可以从一个"空"目录开始。 您必须用一个初始的 "空的" delta
来开始您的 CTM 支持树。曾经为了您的便利这些 "起始" deltas 被有意的通过 CD 来发行，
然而现在已经不这样做了。

因为代码树有数十兆字节，您应该更喜欢从手头上已经有的东西开始。如果您有一张 -RELEASE CD
光盘，您可以从里面复制或者解压缩一份初始代码出来。 这会节省非常多的数据传输量。

您会发现这些"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
的设计符合您的初始 "seed" 的由来。 Empty 是一个空目录。通常一个基本的从 Empty 开始的转换由 100
个 deltas 构成。顺便说一下，他们都很大！ 70 到 80 兆字节的 gzip 压缩的数据对于 XEmpty deltas
是很平常的。

一旦您已经选定了一个基本的 delta 开始，您就需要比这个数高的所有的 delta。

A.4.4. 在您的日常生活中使用 CTM

要应用 deltas，简单的键入：

```
# cd /where/ever/you/want/the/stuff
# ctm -v -v /where/you/store/your/deltas/src-xxx.*
```

CTM 能够理解被 gzip 压缩的 deltas，所以您不需要先 gunzip 他们，这可以节省磁盘空间。

除非觉得整个过程非常可靠， CTM 不会涉及到您的代码树的。您也可以使用 -c 标记来校验 delta， 这样 CTM 就不会涉及代码树； 它会只校验 delta 的完整性看看是否可以安全的用于您的当前代码树。

CTM 还有其他的一些参数， 查看手册页或者源代码了解更多信息。

这真的就是全部的事情了。每次得到一个新的 delta，就通过 CTM 运行它来保证您的代码是最新的。

如果这些 deltas 很难重新下载的话不要删除它们。 有些东西坏掉的时候您会想到保留它们的。
即使您只有软盘，也请考虑使用 fdwrite 来做一份副本。

A.4.5. 维持您本地的变动

作为一名开发者喜欢实验，改动代码树中的文件。 CTM 用一种受限的方式支持本地修改：再检查文件 foo 存在之前，首先查找 foo.ctm。如果这个文件存在， CTM 会对它操作而不是 foo。

这种行为给我们提供了一种简单的方式来维持本地的改动： 只要复制您计划修改的文件并用 .ctm 的后缀重新命名。 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 选项

A.4.6.1. 正确的找出哪些将被更新

您可以确定变动列表， CTM 可以做到，在您的代码库上使用 CTM 的 -l 选项。

这很有用如果您想要保存改动的日志， pre- 或者 post- 用各种风格处理修改的文件的纪录，
或者仅仅是想感受一下孩子般的疯狂。

A.4.6.2. 在升级前制作备份

有时您可能想备份将要被 CTM 升级所改动的所有文件。

指定 -B backup-file 选项会导致 CTM 备份将要被给定的 CTM delta 改动的所有文件到 backup-file。

A.4.6.3. 限定受升级影响的文件

有时您可能对限定一个给定的 CTM 升级的范围感兴趣，也有可能想知道怎样从一列 deltas
中解压缩一部分文件。

您可以通过使用 -e 和 -x 选项指定过滤规则表达式来控制 CTM 即将对之操作的文件列表。

例如，要从您保存的CTM deltas 集里解压缩出一个最新的 lib/libc/Makefile 文件，运行这个命令：

```
# cd /where/ever/you/want/to/extract/it/
# ctm -e '^lib/libc/Makefile' ~ctm/src-xxx.*
```

对于每一个在 CTM delta 中指定的文件，-e 和 -x 选项按照命令行给定的顺序应用。 文件只有在所有的 -e 和 -x 被应用之后标记为合格之后 才能被 CTM 操作。

A.4.7. CTM 未来的计划

其中几项：

- 在 CTM 中使用一些认证方式， 这样来允许察觉冒充的 CTM 补丁。
- 整理 CTM 的选项， 它们变得杂乱而违反直觉了。

A.4.8. 杂项

也有一系列的 ports collection 的 deltas，但是人们对它的兴致还没有那么高。

A.4.9. CTM 镜像

CTM/FreeBSD 可以在下面的镜像站点通过匿名 FTP 下载。如果您选择通过匿名 FTP 获取 CTM，请试着使用一个离您较近的站点。

如果有问题，请联系 ctm-users-desc 邮件列表。

加利福尼亚州，旧金山湾区，官方源代码

- ftp://ftp.FreeBSD.org/pub/FreeBSD/development/CTM/

南非，旧的 deltas 的备份服务器

- 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/

如果您在您附近找不到镜像或者镜像不完整， 试着使用搜索引擎比如 alltheweb.

A.5. 使用 CVSup

A.5.1. 概述

CVSup 是一个用于从远程服务器主机上的主 CVS 仓库发布和升级源代码树的软件包。 FreeBSD 的源代码维护在加利福尼亚州一台主开发服务器的 CVS 仓库里。 有了 CVSup， FreeBSD 用户可以很容易的保持他们自己的源代码树更新。

CVSup 使用所谓的升级 pull 模式。在 pull 模式下，客户端在需要的时候向服务器端请求更新。服务器被动的等待客户端的升级请求。 因此所有的升级都是客户端发起的。服务器决不会发送未请求的升级。用户必须手动运行 CVSup 客户端获取更新， 或者设置一个 cron 作业来让它以固定的规律自动运行。

术语 CVSup用大写字母写正是表示， 代表了完整的软件包。它的主要组件是运行在每个用户机器上的客户端 cvsup， 和运行在每个 FreeBSD 镜像站点上的服务器端 cvsupd。

当您阅读 FreeBSD 文档和邮件列表时，您可能会看见 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， 它还是用使用和 sup 的兼容的配置文件。 Sup 已经不再被 FreeBSD 项目使用了， 因为 CVSup 既快又有更好的灵活性。

csup 是用 C 语言对 CVSup 软件的重写。 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，您可以直接使用它， 因为它是基本系统的一部分。 假如您决定使用 csup，

则可以跳过安装 CVSup 这一步， 并在文章中余下部分提到的 CVSup 改为 csup。

A.5.2. 安装

安装 CVSup 最简单的方式就是使用 FreeBSD packages collection 中预编译的 net/cvsup 包。
如果您想从源代码构建 CVSup， 您可以使用 net/cvsup port。但是预先警告一下： net/cvsup port 依赖于 Modula-3 系统，会花费相当的时间和磁盘空间来下载编译。

如果想在没有安装 Xorg 的计算机， 例如服务器上使用 CVSup， 则只能使用不包含 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：

- 您想接收 哪些文件?
- 您想要它们的 哪个版本?
- 您想从哪里 获取它们?
- 您想把它们 放在您自己机器的什么地方?
- 您想把 您的状态文件放在哪?

在下面的章节里，我们通过依次回答这些问题来创建一个典型的 supfile 文件。首先，我们描述一下 supfile 的整体构成。

supfile 是个文本文件。注释用 # 开头，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户希望接收的文件。 每行以 "collection"，
由服务器端定义的合理的文件分组，的名字开头。 collection 的名字告诉服务器您想要的文件。 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。
字段类型有两种：标记字段和值字段。 标记字段由独立的关键字组成，比如， delete 或者 compress。值字段也用关键字开头， 关键字后面跟 = 和第二个词而没有空格。 例如，release=cvs 是一个值字段。

一个典型的 supfile 往往接收多于一个的 collection。创建 supfile 的一种方式是明确的为每一个 collection 指定相关的字段。然而，这样使得 supfile 的行变得特别长，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。用特定的伪 collection 名 *default 开头的行可以被用来设置标记和值为 supfile 中随后的 collection 中的默认值。
默认值可以通过为这个 collection 自身指定不同的值来对单个的 collection 覆盖设置， 也可以在 mid-supfile 中通过附加的 *default 行改变或扩充。

知道了这些，我们现在就可以开始创建一个 用于接收和升级 FreeBSD-CURRENT 主源代码树的 supfile 文件了。

- 您想接收哪些文件?

  通过 CVSup 可用的文件组织成叫做 "collections" 的名称组。 这些可用的 collection 在 随后的章节 中描述。 在这个例子里， 我们希望接收 FreeBSD 系统的完整的主代码树。 有一个单独的大的 collection src-all 让我们完成这个。 创建我们的 supfile 的第一步， 我们简单的列出这些 collection，每个一行(在这个例子里， 只有一行)：

  ```
  src-all
  ```

- 您想要他们的 哪个版本?

通过 CVSup，您实际上可以接收 曾经存在的源代码的任何版本。 这是有可能的，因为 cvsupd 服务器直接通过 CVS 仓库工作，那包含了所有的版本。您可以 用 tag= 和 date= 值字段 指定一个您想要的版本。

仔细的正确指定任何 tag= 字段。有一些 tag 只对特定的 collection 文件合法。 如果您指定了一个不正确的或者 拼写错误的 tag，CVSup 会删除您可能不想删除的文件。 特别地，对 ports-* collection 只使用 tag=.。

tag= 字段在仓库中表示为一个符号标签。 有两种标签，修订标签和分支标签。 修订标签代表一个特定的修订版本。 它的含义是一成不变的。 分支标签，另一方面，代表给定开发线上给定时间的最新修订。 因为分支标签不代表一个特定的修订版本， 它明天的含义就可能和今天的有所不同。

CVS 标签 包含了用户可能感兴趣的分支标签。 当在 CVSup 的配置文件中指定标签的时候，必须用 tag= 开头 (RELENG_8 会变成 tag=RELENG_8)。 记住只有 tag=. 可以用于 Ports Collection。

注意像看到的那样正确的输入标签名。 CVSup 不能辨别合法和不合法标签。 如果您拼写错了标签名， CVSup 会像您指定了一个没有任何文件的合法标签一样工作， 那会删除您已经存在的代码。

当您指定一个分支标签的时候，您通常会收到开发线上文件的最新版本。 如果您希望接收一些过时的版本，您可以通过用 date= 值字段指定一个日期来做到。 cvsup(1) 手册页解释了如何来做。

对于我们的示例来说，我们希望接收 FreeBSD-CURRENT。 我们在我们的 supfile 的开头添加这行：

```
*default tag=.
```

有一个重要的特例， 如果您既没指定 tag= 字段也没指定 date= 字段的情况。这种情况下， 您会收到直接来自于服务器 CVS 仓库的真实的 RCS 文件， 而不是某一特定版本。 开发人员一般喜欢这种操作模式。 通过在他们的系统上维护一份仓库自身的副本， 他们可以浏览修订历史以及检查文件过去的版本。 然而，这个好处是以大量的磁盘空间为代价的。

- 您想从哪里获取他们?

  我们使用 host= 字段来告诉 cvsup 从哪里获取更新。 任何一个 CVSup 镜像站点都可以， 虽然您应该选择一个离您比较近的站点。 在这个例子里我们将使用一个虚拟的 FreeBSD 发布站点， cvsup99.FreeBSD.org：

  ```
  *default host=cvsup99.FreeBSD.org
  ```

  您需要在运行 CVSup 之前把这个改成一个实际存在的站点。 在任何 cvsup 运行的特定时刻， 您都可以在命令行上使用 -h hostname 选项来覆盖主机设置。

- 您想把它们放在 您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。 在这个例子里，我们把源代码文件直接放进我们的主源代码树， /usr/src。 src 目录已经隐含在我们选择接收的 collection 里了， 所以正确的写法是：

  ```
  *default prefix=/usr
  ```

- cvsup 在哪里维护它的状态文件?

CVSup 客户端在被叫做 "base" 的目录里维护了几个状态文件。 这些文件帮助 CVSup 更有效的工作，通过跟踪您已经接收到哪些更新的方式。 我们将使用标准的 base 目录， /var/db：

```
*default base=/var/db
```

如果您的 base 目录还不存在，现在最好创建它。 如果 base 目录不存在，cvsup 客户端会拒绝工作。

- 其他的 supfile 设置：

  在 supfile 中有一些其他选项需要介绍一下：

  ```
  *default release=cvs delete use-rel-suffix compress
  ```

  release=cvs 显示服务器应该从 FreeBSD 的主 CVS 仓库中获取信息。
  事实上总是这样的，但是也有可能会超出这个讨论的范围。

  delete 给 CVSup 权限删除文件。 您应该总是指定这个，这样 CVSup
  可以保证您的源代码树完全更新。CVSup 很小心的只删除那些不再依赖的文件。
  您拥有的任何额外的文件会被严格的保留。

  use-rel-suffix 是 … 不可思议的。 如果您真的想了解它，查看 cvsup(1) 手册页。
  否则，就指定而不用担心这个。

  compress 启用 gzip 风格的信道压缩。 如果您的网络连接是 T1 或者更快， 您可能不想使用压缩。
  否则，它非常有帮助。

- 把它们放在一起：

  这是我们的示例的完整 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，这意味着您要连接到 CVSup 服务器，服务器说，"这有些您能下载的东西 …"，然后您的客户端反应"好，我要这个，
这个，这个，还有这个。"在默认的配置中， CVSup 客户端会取回您在配置文件中选定的 collection
和标签的每个文件。 然而，并不总是您想要的， 尤其是您在同步 doc，ports，或者 www 树 -
大部分人都不能阅读四种或者五种 语言，因此他们不需要下载特定语言的文件。 如果您在 CVSup Ports Collection，您 可以通过单独指定每个 collection 来避免这个 (比如，ports-astrology， ports-biology，等等取代简单的说明 ports-all)。然而，因为 doc 和 www 树没有特定语言的 collection，您必须使用 CVSup 许多极好的特性之一： refuse 文件。

refuse 文件本质上是告诉 CVSup 它不应该从 collection 中取得某些文件；换句话说，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，如果您没有，应该创建一个)。base 在您的 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； 它仅仅包含您不希望下载的文件和目录名。
例如，如果您除了英语和德语之外不会讲其他语言， 而且也不打算阅读德文的文档翻译版本，
则可以把下面这些放在您的 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(您可以通过浏览 FreeBSD CVS 仓库找到完整的列表)。

有这个非常有用的特性，那些慢速连接或者要为他们的 Internet
连接按时付费的用户就可以节省宝贵的时间因为他们不再需要 下载那些从来不用的文件。要了解 refuse
文件的更多信息以及其它 CVSup 的优雅的特性，请浏览它的 手册页。

A.5.4. 运行 CVSup

您现在准备尝试升级了。命令很简单：

```
# cvsup supfile
```

supfile 的位置当然就是您刚刚创建的 supfile 文件名啦。 如果您在 X11 下面运行，cvsup
会显示一个有一些可以做平常事情的按钮的 GUI 窗口。 按 [go] 按钮，然后看着它运行。

在这个例子里您将要升级您目前的 /usr/src 树，您将需要 用 root 来运行程序，这样 cvsup
有需要的权限来更新您的文件。 刚刚创建了您的配置文件，又从来没有使用过这个程序，
紧张不安是可以理解的。有一个简单的方法不改变您当前的文件
来做一次试验性的运行。只要在方便的地方创建一个 空目录，并在命令行上作为一个额外的参数说明：

```
# mkdir /var/tmp/dest
# cvsup supfile /var/tmp/dest
```

您指定的目录会作为所有文件更新的目的路径。 CVSup 会检查您在 /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。 在这种方式下运行 CVSup 也会把它的 base

目录状态文件保持原样。这些文件的新版本 会被写到指定的目录。 因为您有 /usr/src 目录的读权限，所以执行这种试验性的运行 甚至不需要使用 root 用户。

如果您没有运行 X11 或者不喜欢 GUI， 当您运行 cvsup 的时候需要在命令行添加 两个选项：

```
# cvsup -g -L 2 supfile
```

-g 告诉 CVSup 不要使用 GUI。如果您 没在运行 X11 这个是自动的，否则您必须指定它。

-L 2 告诉 CVSup 输出所有正在升级的文件的细节。 有三个等级可以选择，从 -L 0 到 -L 2。默认是 0，意味着除了错误消息 什么都不输出。

还有许多其它的选项可用。想要一个简短的列表， 输入 cvsup -H。要查看更详细的描述， 请查看手册页。

一旦您对升级工作的方式满意了，您就 可以使用 cron(8) 来安排规则的运行 CVSup。很显然的，您不应该让 CVSup 通过 cron(8) 运行的时候使用它的 GUI。

A.5.5. CVSup 文件 collection

CVSup 可用的文件 collection 是分级组织的。 有几个大的 collection，然后它们有分成更小的子 collection。接收一个大的 collection 等同于 接收它的每一个子 collection。 collection 的等级关系在下面列表中通过缩进的使用 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。

cvs-all release=cvs
: FreeBSD 主 CVS 仓库，包含 密码系统的代码。

distrib release=cvs
: FreeBSD 发行版本和镜像相关的 文件。

doc-all release=cvs
: FreeBSD 使用手册和其它文档的源代码。 其中不包含 FreeBSD web 站点的文件。

ports-all release=cvs
: FreeBSD Ports Collection。

如果您不想升级全部的 ports-all(整个 ports 树)，而只是使用下面列出的一个子集， 请确保您总是升级了 ports-base 子 collection！ 无论何时在 ports 构建下层构造有所改变的时候都会通过 ports-base 表现出来，事实上某些 改变会很快的被"实际的" ports 使用，因此，如果您只升级了 "实际的" ports 而他们使用了一些新的特性，就有极大的可能编译会因一些神秘的错误信息而失败。 这种情况下 非常快速的要做的事情 就是确保您的 ports-base 子 collection 更新到 最新。

要自行构建 ports/INDEX， 您 必须 接受 ports-all (完整的 ports tree)。 在部分 ports tree 上构建 ports/INDEX 是不被支持的。 请参见 FAQ。

ports-accessibility release=cvs
: 用以帮助残疾用户的软件。

ports-arabic release=cvs
: 阿拉伯语支持。

ports-archivers release=cvs
: 存档工具。

ports-astro release=cvs
: 天文相关的 ports。

ports-audio release=cvs
: 声音支持。

ports-base release=cvs
: Ports Collection 构建下部构造 - 位于 /usr/ports 的 Mk/ 和 Tools/ 子目录的 各种各样的文件。

请查看重要警告：您应该 总是更新这个 子 collection，无论您更新 FreeBSD Ports Collection 的任何部分的时候！

ports-benchmarks release=cvs
: 基准。

ports-biology release=cvs
: 生物学。

ports-cad release=cvs
: 计算机辅助设计工具。

ports-chinese release=cvs
: 中文语言支持。

ports-comms release=cvs
: 通信软件。

ports-converters release=cvs
: 字符编码转换。

ports-databases release=cvs
: 数据库

ports-deskutils release=cvs
: 计算机发明前常出现在桌面上的东西。

ports-devel release=cvs
: 开发工具。

ports-dns release=cvs
: DNS 相关软件。

ports-editors release=cvs
: 编辑器

ports-emulators release=cvs
: 其它操作系统的模拟器

ports-finance release=cvs
: 货币，金融相关应用程序。

ports-ftp release=cvs
: FTP 客户端和服务器端工具。

ports-games release=cvs
: 游戏

ports-german release=cvs
: 德语支持。

ports-graphics release=cvs
: 图形图像工具。

ports-hebrew release=cvs
: 希伯来语支持。

ports-hungarian release=cvs
: 匈牙利语言支持。

ports-irc release=cvs
: Internet 多线交谈(IRC)工具。

ports-japanese release=cvs
: 日语支持。

ports-java release=cvs
: Java™ 工具。

ports-korean release=cvs
: 韩国语言支持。

ports-lang release=cvs
: 编程语言。

ports-mail release=cvs
: 邮件软件。

ports-math release=cvs
: 数值计算软件。

ports-misc release=cvs
: 杂样工具。

ports-multimedia release=cvs
: 多媒体软件。

ports-net release=cvs
: 网络软件。

ports-net-im release=cvs
: 即时消息软件。

ports-net-mgmt release=cvs
: 网管软件。

ports-net-p2p release=cvs
: 对等网 (peer to peer network) 应用。

ports-news release=cvs
: 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 和预编译包的工具。

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 包含桌面出版)。

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 驱动程序。

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 内部项目的代码库。

src-all release=cvs
: FreeBSD 主代码，包含密码系统的代码。

src-base release=cvs
: /usr/src 顶层的各式各样的文件。

src-bin release=cvs
: 单用户模式下可能用到的用户工具 (/usr/src/bin)。

src-cddl release=cvs
: 采用了 CDDL 授权的实用工具和函数库 (/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 公共许可协议的工具 (/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联编的程序； 请参见 rescue(8) (/usr/src/rescue)。

src-sbin release=cvs
: 单用户模式的系统工具 (/usr/src/sbin)。

src-secure release=cvs
: 密码相关库和命令 (/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
: 维护 FreeBSD 的各种各样的工具 (/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 跟踪数据库。

mail-archive release=current
: FreeBSD 邮件列表存档。

www release=current
: 预处理过的 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， 请查看 CVSup 主页。

多数与 FreeBSD 有关的 CVSup 讨论会在 FreeBSD 技术讨论邮件列表 进行。 这个软件的新版本会在那里和 FreeBSD 公告邮件列表 公布。

如果对于 CVSup 有任何问题， 或希望提交 bug 报告， 请参阅 CVSup FAQ。

A.5.7. CVSup 站点

FreeBSD 的 CVSup 服务器运行于 下列站点：

A.6. CVS 标签

当使用 cvs 或者 CVSup 获取和升级源代码的时候，必须指定一个修订标签。 修订标签代表 FreeBSD 开发的一个特定分支， 或者一个特定的时间点。第一种叫做 "分支标签"，第二种叫做 "版本标签"。

A.6.1. 分支标签

所有这些，除了 HEAD (这个总是 合法标签)以外，只适用于 src/ 树。ports/，doc/，和 www/ 树没有分支。

HEAD
: 主线的符号名，或者说 FreeBSD-CURRENT。 当没有指定修订版本的时候也是默认的。

  在 CVSup 里，这个标签通过 一个 . 来反映出来(不是标点，而是一个 . 字符)。

在 CVS 里，当没有修订标签指定时这是默认的。 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，除非这是您的本意。

RELENG_8
: 这是 FreeBSD-8.X 的开发分支， 也被称作 FreeBSD 8-STABLE。

RELENG_8_2
: 这是 FreeBSD-8.2 发行版分支，只用于安全公告， 以及其他重要更新。

RELENG_8_1
: FreeBSD-8.1 的发行版分支， 只用于安全公告， 以及其他重要更新。

RELENG_8_0
: FreeBSD-8.0 的发行版分支， 只用于安全公告， 以及其他重要更新。

RELENG_7
: 这是 FreeBSD-7.X 的开发分支， 也被称作 FreeBSD 7-STABLE。

RELENG_7_4
: FreeBSD-7.4 的发行版分支， 只用于安全公告， 以及其他重要更新。

RELENG_7_3
: FreeBSD-7.3 的发行版分支， 只用于安全公告， 以及其他重要更新。

RELENG_7_2
: FreeBSD-7.2 的发行版分支， 只用于安全公告， 以及其他重要更新。

RELENG_7_1
: FreeBSD-7.1 的发行版分支， 只用于安全公告， 以及其他重要更新。

RELENG_7_0
: FreeBSD-7.0 的发行版分支， 只用于安全公告， 以及其他重要更新。

RELENG_6
: 这是 FreeBSD-6.X 的开发分支， 也被称作 FreeBSD 6-STABLE。

RELENG_6_4
: FreeBSD-6.4 的发行版分支， 只用于安全公告， 以及其他重要更新。

RELENG_6_3
: FreeBSD-6.3 的发行版分支， 只用于安全公告， 以及其他重要更新。

RELENG_6_2
: FreeBSD-6.2 的发行版分支， 只用于安全公告， 以及其他重要更新。

RELENG_6_1
: FreeBSD-6.1 的发行版分支， 只用于安全公告， 以及其他重要更新。

RELENG_6_0
: FreeBSD-6.0 的发行版分支， 只用于安全公告， 以及其他重要更新。

RELENG_5
: 这是 FreeBSD-5.X 的开发分支， 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
: 针对 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
: 针对 FreeBSD-5.1 的发行版本分支， 只被安全公告和其它重要更新使用。

RELENG_5_0
: 针对 FreeBSD-5.0 的发行版本分支， 只被安全公告和其它重要更新使用。

RELENG_4
: FreeBSD-4.X 开发线，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 的开发线，也被叫做 3.X-STABLE。

RELENG_2_2
: FreeBSD-2.2.X 的开发线， 也被叫做 2.2-STABLE。 这个分支基本上已经过时了。

A.6.2. 版本标签

当一个特定的 FreeBSD 版本发行时， 这些标签代表了一个指定的时间点。发布工程进程在 Release Engineering Information 和 Release Process 文档中被详细描述。 src 树使用以 RELENG_ 开头的标签。 ports 和 doc 树使用以 RELEASE 开头的标签。 最后， www 树上不会有任何特定发行版的标签。

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/

```
stacken.kth.se       # Stacken Computer Club, KTH, Sweden
130.237.234.43       #hot.stacken.kth.se
130.237.237.230      #fishburger.stacken.kth.se
```

```
130.237.234.3       #milko.stacken.kth.se
```

维护者 ftp@stacken.kth.se

A.8. rsync 站点

下面的站点让 FreeBSD 可以通过 rsync 协议下载。 rsync 实用程序和 rcp(1) 的工作方式很相像，但是有更多的选项，使用 rsync 远程更新协议只传输 两份文件的不同之处，因此能够大幅度的提高网络同步速率。 如果您是 FreeBSD FTP 服务器或者 CVS 仓库的镜像站点，这一点非常有用。 rsync 套件可以工作在许多种 操作系统上，在 FreeBSD 上，查看 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: 对于 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 问题追踪数据库。
- 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 镜像。

附录 B: 参考文献

尽管手册页能够提供对于 FreeBSD 操作系统最为权威的参考资料，
它们有时却不能告诉我们如何让整个系统很好地运转起来。 因此， 一本关于 UNIX® 系统管理的好书，
以及一份好的用户手册是不可或缺的。

B.1. 关于 FreeBSD 的专业书籍与杂志

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 (简体中文译本)， 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 (第二版简体中文译本)， 人民邮电出版社。ISBN 7-115-10541-3。
- FreeBSD 3.x Internet 高级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 集成组网实务 （简体中文）， 中国铁道出版社。 ISBN 7-113-03845-X。
- FreeBSD 网站架设实务 （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 （日文译本）, 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 Modere 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. 用户指南

- 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 编写了一份 UNIX 入门教程 并提供了在线的 HTML 和 PostScript 格式的版本。
- 这份文档的意大利文 翻译 是 FreeBSD Italian Documentation Project 的一部分。
- Jpman Project, Japan FreeBSD Users Group. FreeBSD User’ s Reference Manual (日文译本). Mainichi Communications Inc., 1998. ISBN4-8399-0088-4 P3800E.
- Edinburgh University has written an Online Guide for newcomers to the UNIX environment.

B.3. 管理员指南

- 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, Æleen. 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 (日文译本). 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. 开发指南

- 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. Dobb’ s Journal. 19(15), December 1994. pp68-71, 97-99.

B.5. 操作系统原理

- Andleigh, Prabhat K. UNIX System Architecture. Prentice-Hall, Inc., 1990. ISBN 0-13-949843-5
- Jolitz, William. "Porting UNIX to the 386". Dr. Dobb’ 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

  (这本书的第二章的 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 公司在他们的 开发人员网站上, 提供了关于他们的 CPU，芯片组，以及标准的文档。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-Wesley Pub. Co., 2002. ISBN 0-201-59616-4

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 族谱. 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. 各种期刊

- 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使得现有的印刷、平面媒体跟不上它的发展进度!
而电子版的也许是最好的，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，
用户社区通常也扮演着是"技术支持部门"的角色。通过电子邮， Web 论坛件和 USENET
新闻组可以很快的找到他们。

以下列出了尽量多的联系FreeBSD用户社区的方式。如果您发现有其他的资源没有被包括在这儿，
请告诉FreeBSD 文档计划邮件列表，以便将它们加入到这里。

C.1. 邮件列表

邮件列表通常是提问或是发起有关 FreeBSD 某一方面的专项技术讨论最直接的途径。 有多种针对于不同 FreeBSD 话题的邮件列表。 把你的问题发送到最合适的邮件列表通常能获得更加快速准确的回复。

本文的最后给出了各个不同的邮件列表的使用规则。 在订阅其中任何一个列表之前，请先阅读使用条文。
现在订阅这些邮件列表的人每天都会收到上百封关于FreeBSD的信件。
设立列表的使用条文有助于维护讨论质量。否则这些讨论计划的列表将失去其意义。

如果你想要尝试发送一封邮件到 FreeBSD 邮件列表，你可以把邮件发往 FreeBSD 测试邮件列表。 请不要往其他的列表发送测试邮件。

如果不知道哪个邮件列表适合于发送您的问题， 请参见 如何从 FreeBSD-questions 邮件列表中更快地得到答案。

在列表中发送任何问题之前， 请首先学习使用邮件列表的最佳方式， 例如如何通过阅读 邮件列表常见问题回答集 (FAQ) 文档， 来避免经常重复的讨论。

全部的邮件列表记录都可以在FreeBSD World Wide Web服务器上找到。此服务器提供了很棒的关键词搜寻功能，可让您找到FAQ的解答。
而在邮件列表上提问之前，请先搜寻是否已有答案。 请注意这意味着所有发往 FreeBSD 邮件列表的消息都会被永久归档保存。 当涉及到隐私保护的话，
可以考虑使用一个可使用后丢弃的电子邮件地址并只发送公开的信息。

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，鼓励大家加入他们:

目录	用途
freebsd-advocacy	FreeBSD鼓吹者
FreeBSD 公告邮件列表	重要的事件和里程碑
freebsd-arch	架构和设计的讨论
freebsd-bugbusters	与FreeBSD问题报告数据库和有关工具维护相关的讨论
freebsd-bugs	报告FreeBSD的Bug
freebsd-chat	和技术无关的FreeBSD讨论区
freebsd-chromium	FreeBSD Chromium 相关的讨论
FreeBSD-CURRENT 邮件列表	讨论使用 FreeBSD-CURRENT 有关的一些问题
freebsd-isp	ISP使用FreeBSD的讨论
freebsd-jobs	与FreeBSD有关的工作机会
freebsd-questions	用户问题和技术支持
FreeBSD 安全问题通知邮件列表	安全通知
FreeBSD-STABLE; 邮件列表	讨论使用 FreeBSD-STABLE 有关的一些问题

目录	用途
FreeBSD 测试邮件列表	在真正发送一个邮件到邮件列表之前可以先发送到这里测试

技术性的邮件列表: 以下的邮件列表是用来讨论技术性问题的。
在加入订阅及讨论之前请务必认真阅读每个列表主题，因为他们讨论的内容都是严格地被限制着的。

目录	用途
FreeBSD ACPI 邮件列表	ACPI 和电源管理的开发
freebsd-afs	将 AFS 移植到 FreeBSD
freebsd-aic7xxx	为 Adaptec® AIC 7xxx 开发驱动
freebsd-amd64	将 FreeBSD 移植到 AMD64 系统
freebsd-apache	关于与 Apache 有关的 ports 的讨论
freebsd-arm	将 FreeBSD 移植到 ARM® 处理器
freebsd-atm	在 FreeBSD 上使用 ATM 网络
freebsd-bluetooth	在 FreeBSD 上使用 Bluetooth® 技术
freebsd-cluster	在集群环境中使用 FreeBSD
freebsd-cvsweb	CVSweb 维护
freebsd-database	讨论 FreeBSD 下开发和使用数据库
freebsd-doc	创建 FreeBSD 相关文档
freebsd-drivers	为 FreeBSD 撰写驱动
freebsd-eclipse	FreeBSD 上的 Eclipse IDE、工具、 富客户应用，以及 ports 的用户讨论。
freebsd-embedded	在嵌入式应用中使用 FreeBSD
freebsd-eol	关于与 FreeBSD 有关， 但已不再为 FreeBSD Project 所维护的软件的互助支持。
freebsd-emulation	在 FreeBSD 上模拟其它系统， 如 Linux/MS-DOS®/Windows®
freebsd-firewire	FreeBSD 的 FireWire® (iLink, IEEE 1394) 技术讨论
freebsd-fs	文件系统
freebsd-gecko	Gecko 渲染引擎 issues
freebsd-geom	针对 GEOM 的讨论和实现
freebsd-gnome	移植 GNOME 和 GNOME应用程序
freebsd-hackers	一般性的技术讨论
freebsd-hardware	一般性的支持 FreeBSD 的硬件的讨论
freebsd-i18n	FreeBSD 的国际化
freebsd-ia32	在 IA-32 (Intel® x86) 平台上运行 FreeBSD
freebsd-ia64	将 FreeBSD 移植到 Intel® 即将推出的 IA64 系统
freebsd-ipfw	关于 IP 防火墙代码再设计的技术性讨论
freebsd-isdn	ISDN开发人员
freebsd-jail	关于 jail(8) 机制的讨论
freebsd-java	Java™ 开发人员以及移植 JDK™s 到 FreeBSD 的人们
freebsd-kde	移植 KDE 和 KDE 应用程序
freebsd-lfs	移植 LFS 到FreeBSD上

目录	用途
freebsd-mips	移植 FreeBSD 到 MIPS®
freebsd-mobile	关于便携式计算机的讨论
freebsd-mono	FreeBSD 上的 Mono 和 C# 应用
FreeBSD 多媒体应用邮件列表	多媒体应用程序
freebsd-new-bus	技术讨论关于总线架构
freebsd-net	网络子系统和 TCP/IP 源代码的讨论
freebsd-office	FreeBSD 上的办公套件
freebsd-performance	高性能、负载下安装后的性能调整问题
freebsd-perl	许多与 perl 相关的 ports 的维护
freebsd-pf	关于 packet filter 防火墙系统的讨论
freebsd-platforms	关于向非 Intel® 架构的平台上移植的讨论
freebsd-ports	关于 Ports Collection 的讨论
freebsd-ports-bugs	ports bugs/PRs讨论
freebsd-ppc	移植 FreeBSD 到 PowerPC®
freebsd-proliant	关于 FreeBSD 在 HP ProLiant 服务器平台上的技术讨论
freebsd-python	FreeBSD 专属的 Python 问题
freebsd-rc	关于 rc.d 系统及其开发的讨论
freebsd-realtime	FreeBSD 实时扩展的开发
freebsd-ruby	关于 FreeBSD 上 Ruby 的讨论
freebsd-scsi	SCSI 子系统
FreeBSD 安全问题邮件列表	系统安全
freebsd-small	在嵌入式系统上使用 FreeBSD (已过时； 请使用 freebsd-embedded 代替)
freebsd-sparc64	移植 FreeBSD 到 SPARC® 系统
freebsd-standards	让 FreeBSD 顺应 C99 以及 POSIX® 标准
freebsd-sysinstall	sysinstall(8) 的开发
freebsd-threads	线程
freebsd-testing	FreeBSD 性能和稳定性测试
freebsd-tilera	讨论将 FreeBSD 移植到 Tilera 系列 CPU
freebsd-tokenring	在 FreeBSD 中支持 Token Ring
freebsd-toolchain	维护在 FreeBSD 中集成的联编工具集
freebsd-usb	关于 FreeBSD 的 USB 支持的讨论
freebsd-virtualization	讨论各种 FreeBSD 支持的虚拟化技术
freebsd-vuxml	关于 VuXML 的问题讨论
freebsd-x11	维护和支持在 FreeBSD 上运行的 X11
freebsd-xen	讨论 FreeBSD Xen™ 上的移植 - 实现和使用
freebsd-xfce	FreeBSD 上 XFCE 的移植和维护

限制订阅的列表: 以下的列表是针对某些特定的读者而设的，
而且并不适合被当成是一般公开讨论区。您最好在某一技术讨论区参与讨论后再选择订阅这些
限制订阅的邮件列表，因为这样您可以了解到在这些讨论区发言所需要的礼仪。

目录	用途
freebsd-hubs	运行镜象站点的成员(支持基本服务)
freebsd-user-groups	用户组调整
freebsd-vendors	商家在发布之前的调整
freebsd-wip-status	FreeBSD 项目进度状态
freebsd-wireless	讨论 802.11 栈， 工具和设备驱动开发
freebsd-www	www.FreeBSD.org的维护

分类列表: 所有以上的列表在一个分类格式里面是可利用的。
一旦订阅了一个列表，您可以在您的账号选项里面设置您的分类选项。

CVS 和 SVN 列表: 以下的邮件是给对FreeBSD源代码的变更记录有兴趣的人看的， 而且它们是只读的邮件列表，您不能发Email给他们。

列表	源位置	描述
cvs-all	/usr/(CVSROOT|doc|ports)	所有对源代码的改变纪录 (其他 CVS commit 列表的超集)
cvs-doc	/usr/(doc|www)	所有对 doc 和 www 源代码的改变记录
cvs-ports	/usr/ports	所有对 ports 源代码的改变记录
cvs-projects	/usr/projects	所有对 projects 源代码的改变记录
cvs-src	/usr/src	所有对 src 源代码的改变记录 （由 svn-to-cvs 提交导入程序生成）
SVN 整个 src 树的修订讯息 (除了 "user" 与 "projects")	/usr/src	所有对 Subversion 仓库的改变记录 (除了 user 和 projects)
SVN src 树 head/-current 分支的修订讯息	/usr/src	所有对 Subversion 仓库 "head" 分支的改变记录 (FreeBSD-CURRENT 分支)
svn-src-projects	/usr/projects	所有对 Subversion 源码仓库中有关 projects 部分的改变记录
svn-src-release	/usr/src	所有对 Subversion 源码仓库中有关 releases 部分的改变记录
svn-src-releng	/usr/src	所有对 Subversion 源码仓库中有关 releng 部分的改变记录 (security / release engineering 分支)
svn-src-stable	/usr/src	所有对 Subversion 源码仓库中有关 stable 分支的改变记录
svn-src-stable-6	/usr/src	所有对 Subversion 源码仓库中有关 stable/6 分支的改变记录
svn-src-stable-7	/usr/src	所有对 Subversion 源码仓库中有关 stable/7 分支的改变记录
svn-src-stable-8	/usr/src	所有对 Subversion 源码仓库中有关 stable/8 分支的改变记录

列表	源位置	描述
SVN commit messages for only the 9-stable src tree	/usr/src	所有对 Subversion 源码仓库中有关 stable/9 分支的改变记录
svn-src-stable-other	/usr/src	所有对 Subversion 源码仓库中早期 stable 分支的改变记录
svn-src-svnadmin	/usr/src	所有对 Subversion 源码仓库中管理用脚本, hook 和其他配置数据的改变记录
svn-src-user	/usr/src	所有对 Subversion 源码仓库中有关 user 部分的改变记录
svn-src-vendor	/usr/src	所有对 Subversion 源码仓库中有关 vender 部分的改变记录

C.1.2. 如何订阅

订阅一个列表，点击上面的列表名字或到 https://lists.freebsd.org
并点击进入您感兴趣的列表，这个列表的页面包含了所必需的订阅操作指南。

其实您只需发送邮件到 列表名@FreeBSD.org。 它将被再次转发到全世界的这个邮件列表的成员。

点击上面的 URL，在列表的底部可以从订阅的列表中退出。 也可以发送一个电子邮件到 列表名-unsubscribe@FreeBSD.org 来退订。

此外，我们要求您必须保持在技术性的邮件列表中只是讨论技术。
如果您只是对一些重要的公告感兴趣，建议您加入 FreeBSD 公告邮件列表， 它的通信量比较低。

C.1.3. 列表规章

所有 FreeBSD 的邮件列表都有同样的基本规则， 所有人必须按照规则来做。 违反这些规则时， FreeBSD Postmaster postmaster@FreeBSD.org 会在前两次发送警告， 如果第三次违反， FreeBSD Postmaster 将从所有 FreeBSD 的邮件列表中删除这样的人， 并过滤来自发信人之后的所有邮件。
我们很遗憾必须要遵守这样的规则， 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。

具体规则:

- 任何发表的主题都应当附合基本的列表概况。例如，如果列表是有关技术问题的，
  那您发表的文章包含技术讨论。不要把不相关的讨论放在一起。
  对于没有主题的自由形式的讨论，可以使用 FreeBSD-chat

  freebsd-chat@FreeBSD.org。

- 不要将同一个问题发送到超过两个的邮件列表上，当有一个清晰和明显的必须要
  发表到两个列表的要求时，也只能是两个。对于大多数的列表，已经有相当多的订户了，
  除了一些比较深奥的问题(如"-stable & -scsi")，没有必要同时将一个问题发到多个列表上。
  如果一个信息以这种方式（多个邮件列表在Cc行出现）被发送给您， 那
  Cc行在把它再发送出去之前也将被整理。 无论谁是最初发表者，都会导致您自己的交叉发送。
- 不容许进行人身攻击和亵渎（在前后的争论中），包括用户和开发人员。
  应当遵守最起码的网络礼节，象需要征得同意才可以引用或张贴私人邮件等。
  然而，也有非常少的情况下，这样的内容会符合列表规章， 因此，它会在最初给予警告（或禁止）。
- 严格的禁止非FreeBSD相关产品或服务的广告，一旦发现将马上取缔。

单独的列表规章:

FreeBSD ACPI 邮件列表
: ACPI和电源管理开发

freebsd-afs
: Andrew文件系统

  这个列表是用来讨论porting和从CMU/Transarc使用AFS。

FreeBSD 公告邮件列表
: 重要事件/里程碑

  这是一个发布FreeBSD重大事件的邮件列表。这包括有关snapshots和其他版本的公告，
  新的FreeBSD的性能的公告，还可以用于指派志愿者等等。这个列表比较小。

freebsd-arch
: 架构和设计讨论

  这个列表是讨论FreeBSD的架构。本质上应保证内容的纯技术性。例如主题是：

  - 如何重新创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层工作。
  - 我们怎么改变设备驱动程序接口以便能够在多种总线和体系结构上使用同样的驱动程序。
  - 如何写一个网络驱动。

freebsd-bluetooth
: FreeBSD 上的 Bluetooth®

  这是一个 FreeBSD 的 Bluetooth® 用户聚集的讨论区。 这里欢迎关于设计问题、 实现细节、 补丁、
  问题报告、 开发进度报告， 功能需求以及其他与 Bluetooth® 相关的讨论。

freebsd-bugbusters
: 同等问题报告处理结果

  这个列表的目的是作为一个调整和讨论论坛来服务于Bug列表的成员，Bugbuster列表成员
  和其他任何的对PR数据库真正的有兴趣的成员。这个列表不是为了讨论关于Bug细节，补丁或PRs。

freebsd-bugs
: Bug报告

  这是一个报告FreeBSD的Bug的邮件列表。可以随时通过 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关非技术、社会信息的内容。
  包括谈论Jordan看起来是否像一个机敏的侦探，是否句首的字母要大写， 谁喝了很多咖啡，
  哪儿的啤酒酿造的最好，谁在他们的地下室里酿造了啤酒等等。对于偶然宣布重大的事件
  （例如：将要举行的聚会，婚礼，生日，新工作等等）也能使用这种技术列表，除上述列举之外
  任何事情都可以发布在-chat列表上。

freebsd-chromium
: FreeBSD 上的 Chromium

  这是一个讨论 FreeBSD 上 Chromium 相关问题的邮件列表。 这是一个讨论开发和安装 Chromium
  的技术类列表。

Core Team
: FreeBSD核心团队

这是一个只供核心成员内部使用的邮件列表，只有当一个与FreeBSD相关的严重的事情需要裁决或严格审核时， 才能发送消息到这个邮件列表。

FreeBSD-CURRENT 邮件列表

关于使用FreeBSD-CURRENT版的讨论

这是一个针对FreeBSD-CURRENT用户的邮件列表。
它包括一些可能影响用户的新特性的警告，使用FreeBSD-current的一些指导。
任何运行"CURRENT"的人必须同意这个列表，这是一个纯技术的邮件列表。

freebsd-cvsweb

FreeBSD CVSweb计划

关于FreeBSD-CVSweb的使用，开发和维护的技术性讨论。

freebsd-doc

文档计划

这个邮件列表是与FreeBSD创建的文档的出版和计划的讨论。 这个邮件列表的成员都会提交到"The FreeBSD Documentation Project"。 它是一个开放的列表，可以自由地加入和做贡献!

freebsd-drivers

为 FreeBSD 撰写设备驱动

这是关于 FreeBSD 上的设备驱动的技术论坛。 它主要供编写设备驱动的开发人员提出关于如何使用 FreeBSD 内核提供的 API 来编写设备驱动程序的问题。

freebsd-eclipse

FreeBSD 上的 Eclipse IDE、工具、 富客户应用， 以及 ports 的用户讨论。

这个邮件列表的目的， 是为在 FreeBSD 平台上选择、 安装、 使用、 开发和维护 Eclipse IDE、 工具、 富客户应用的用户， 提供互助式支持， 以及为将 Eclipse IDE 和插件移植到 FreeBSD 环境中提供帮助。

另一个目的是建立一个在 Eclipse 社区和 FreeBSD 社区之间的交流管道， 以达到互惠互利。

尽管这个列表主要关注的是 Eclipse 用户的诉求， 它也为使用 Eclipse 框架开发 FreeBSD 专用的应用提供了论坛。

freebsd-embedded

在嵌入式应用中使用 FreeBSD

这个列表讨论关于在嵌入式系统中如何使用 FreeBSD 的话题。 这是一个技术性的邮件列表， 其主要内容是技术讨论。 针对这一邮件列表， 我们将嵌入式系统定义为那些不作为桌面系统、 只完成某些单一任务的计算设备。 这些实例包括路由器交换机和 PBX 这样的网络设备、 远程测量设备、 PDA、 PoS 系统，等等。

freebsd-emulation

模拟其他系统， 例如 Linux/MS-DOS®/Windows®

这是一个讨论关于如何在 FreeBSD 上运行为其他操作系统所撰写的程序的论坛。

freebsd-eol

关于与 FreeBSD 有关， 但已不再为 FreeBSD Project 所维护的软件的互助支持。

这个邮件列表主要用于那些有兴趣提供或使用针对已不再为 FreeBSD Project 官方所支持 (例如， 以安全更新或补丁的形式) 的 FreeBSD 相关软件的用户或公司讨论。

freebsd-firewire

FireWire® (iLink, IEEE 1394)

这个邮件列表是关于FreeBSD子系统FireWire® (aka IEEE 1394 aka iLink)的设计和执行。相关特定的主题包括标准，总线设计和他们的协议，适配器板/卡/芯片设置，及他们的正确的代码的结构和实施。

freebsd-fs
文件系统

关于FreeBSD文件系统的讨论。这是一个纯技术的邮件列表。

freebsd-gecko
Gecko 渲染引擎

这是一个讨论 FreeBSD 上 Gecko 有关的应用程序的邮件列表。

围绕 FreeBSD 上 Gecko Ports 应用程序的讨论， 以及它们的安装， 开发和支持。

freebsd-geom
GEOM

针对GEOM和相关执行的讨论。这是一个纯技术的邮件列表。

freebsd-gnome
GNOME

讨论关于在FreeBSD系统上的GNOME桌面环境 这是一个纯技术的邮件列表。

freebsd-ipfw
IP防火墙

这是关于在FreeBSD里重新设计IP防火墙代码的技术讨论论坛。

freebsd-ia64
移植FreeBSD到IA64

这是一个有关将FreeBSD移植到Intel® IA64架构上的技术讨论列表，讨论一些相关的问题与解决方案。也欢迎对这些问题感兴趣的个别讨论者。

freebsd-isdn
ISDN通信

这是一个FreeBSD支持的ISDN系统开发的邮件列表。

freebsd-java
Java™开发

这是一个讨论Java™ 应用开发和 JDK™s的porting与维护的邮件列表。

freebsd-jobs
工作的提供和寻找

这个论坛是针对与 FreeBSD 相关的雇佣信息和个人简历， 比如： 如果您想找一个与 FreeBSD 相关的工作或有一个工作需要 FreeBSD 这是一个让您来广告的好地方。 这 不是对一般性雇佣问题的邮件列表， 对这个问题已经有了足够多的论坛。

注意这个列表，像其他的 FreeBSD.org 邮件列表一样是会分发给全世界的订阅者的。因此，您需要明白关于位置和地域问题，确定之间是容易联系和可合作的。

Email最好应该使用 -纯文本格式，不过基本的PDF,HTML和很少其他的能被更多读者接受的格式也是可以的。Microsoft® Word (.doc)格式是被邮件列表服务器拒绝的。

freebsd-kde
KDE

讨论关于在FreeBSD系统上使用KDE。 这是一个纯技术的邮件列表。

freebsd-hackers
技术讨论

这是一个与FreeBSD相关的技术讨论论坛，是一个主要的技术性邮件列表。
他是针对个别的工作在FreeBSD上的人来提出问题或讨论相关的解决方案，
也欢迎对这些问题感兴趣的个别的讨论者。这是一个纯技术的邮件列表。

freebsd-hardware
FreeBSD硬件的普通讨论

有关FreeBSD运行的硬件类型的普通讨论，包括是否该买的一些问题和建议。

freebsd-hubs
镜象站点

人们运行FreeBSD的镜象站点的公告和讨论。

freebsd-isp
ISP供应商问题

这是一个讨论使用FreeBSD的ISP供应商的邮件列表。这是一个纯技术的邮件列表。

freebsd-mono
FreeBSD 上的 Mono 和 C# 应用

这是一个讨论 FreeBSD 上的 Mono 开发框架的邮件列表。 这是一个纯技术的邮件列表。 它是为将 Mono
或 C# 应用移植到 FreeBSD， 以及提出问题及讨论其他解决方案的人准备的。 此外，
也欢迎有兴趣参与讨论的其他人。

freebsd-office
FreeBSD 上的办公套件应用

关于办公套件应用， 它们的安装、 开发和 FreeBSD 支持的讨论中心。

freebsd-performance
讨论关于调整及高速运行FreeBSD

这个邮件列表提供了一个为黑客，管理员和有关的团体去讨论与FreeBSD性能相关的主题的空间。
可以在这里进行讨论的包括在任意高负载下，体验版下或者是有限制的条件下安装FreeBSD。
非常鼓励自愿地为了改进FreeBSD性能的相关团体去订阅这个列表。
这是个高技术含量的列表理论上说适合有丰富经验的FreeBSD用户，黑客，或对FreeBSD的速度、性能
、升级感兴趣的管理员。这不是一个问答式的列表，关于这些应该去读相关文档，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题。

freebsd-pf
关于 packet filter 防火墙系统的问题和讨论

关于 FreeBSD 环境下 packet filter (pf) 防火墙系统的讨论。 这里欢迎技术讨论，
以及一般的应用问题。 此外， 这里也是讨论 ALTQ QoS 框架的合适场所。

freebsd-platforms
移植到非 Intel® 平台上

跨平台的 FreeBSD 问题， 关于非 Intel® FreeBSD 移植版本的讨论和提议。
这是一个纯技术性的邮件列表， 其讨论内容严格限制为技术。

freebsd-ports
"ports"的讨论

关于FreeBSD的"ports collection" (/usr/ports)的讨论,
ports的基础构造和调整过的ports结构。这是一个纯技术的邮件列表。

freebsd-ports-bugs
"ports" bugs的讨论

讨论关于FreeBSD的"ports collection" (/usr/ports),问题报告
ports建议，或者ports的修正。这是一个纯技术的邮件列表。

freebsd-proliant
关于 FreeBSD 在 HP ProLiant 服务器平台上的技术讨论

这个邮件列表用来讨论在 HP ProLiant 服务器上使用 FreeBSD， 包括讨论 ProLiant 专用的驱动、管理软件、 配置工具， 以及 BIOS 更新等。 同样地， 这里也是讨论 hpasmd、 hpasmcli， 以及 hpacucli 模块的主要场所。

freebsd-python
FreeBSD 上的 Python

这是一个讨论关于如何在 FreeBSD 上改善 Python 支持的邮件列表。 这是一个纯技术的邮件列表。
它是为那些移植 Python、 其第三方模块， 以及 Zope 相关软件到 FreeBSD 上的人准备的。
这里也欢迎参与技术讨论的人。

freebsd-questions
用户问题

这是一个有关FreeBSD问题的邮件列表。您不应当发送"how to"
问题给技术列表，除非您认为这个问题是非常可爱的技术问题。

freebsd-ruby
有关 FreeBSD 上 Ruby 的讨论

这是一个讨论关于 Ruby 在 FreeBSD 上支持的邮件列表。 这是一个纯技术的邮件列表。它是为那些移植 Ruby、第三方库以及 各种 framework 准备的。

这里也欢迎参与技术讨论的人。

freebsd-scsi
SCSI子系统

这是一个讨论FreeBSD的SCSI子系统的邮件列表。这是一个纯技术的列表。

FreeBSD 安全问题邮件列表
安全问题

FreeBSD的计算机安全问题（DES,Kerberos,已知的安全漏洞和修复等）。
这是一个纯技术的邮件列表。注意：这不是一个问和答的列表，但是同时给出
问题和答案到FAQ是欢迎的。

FreeBSD 安全问题通知邮件列表
安全通知

FreeBSD安全问题和修复的通知。这不是一个讨论列表，讨论的列表应当是FreeBSD-security

freebsd-small
在嵌入式应用程序中使用FreeBSD

这个列表讨论了与极小的和嵌入的FreeBSD安装的讨论主题。 这是一个纯技术的列表。

这一列表已被 freebsd-embedded 代替。

FreeBSD-STABLE; 邮件列表
: 讨论关于FreeBSD-STABLE版的使用

这是一个FreeBSD-STABLE用户的邮件列表。它包括-STABLE的新特性可能会影响用户的警告。
任何运行"STABLE"的人应当经常关注这个列表。这是一个纯技术的列表。

freebsd-standards
: C99 & POSIX一致

这是关于FreeBSD顺应C99和POSIX标准的技术讨论论坛。

freebsd-toolchain
: 维护 FreeBSD 中集成的联编工具集

这是关于维护 FreeBSD 中集成的联编工具集的论坛。 这里有包括 Clang 和 GCC，
以及其他类似汇编器、 连接器和调试器等软件的讨论。

freebsd-usb
: 讨论 FreeBSD 的 USB 支持

这个邮件列表是关于 FreeBSD 上的 USB 支持的技术性讨论。

freebsd-user-groups
: 用户组调整列表

这个邮件列表为协调从各地的使用群体到彼此相互讨论问题和
从核心团队中指定个人。这个邮件列表应被限制到大纲和协调用户组 计划的范围之内。

freebsd-vendors
: 商家

讨论FreeBSD计划和FreeBSD软硬件商家的协调。

freebsd-virtualization
: 讨论各种 FreeBSD 支持的虚拟化技术

讨论 FreeBSD 所支持的各种虚拟化技术的邮件列表。 在注重实现基本功能，加入新特性的同时，
也为用户提供了一个寻求帮助和讨论他们的使用经验的场所。

freebsd-wip-status
: FreeBSD 项目进度状态

这个邮件列表是用来发布 FreeBSD 相关项目的创建和工作进度的。
发至这个列表的消息将会先被审核。通常建议把消息用 "To:" 发给一个更典型的 FreeBSD
列表，而只仅仅 "BCC:" 给这个列表。 这样你的工作进度就能在典型的列表上讨论，
因为这个列表是不允许讨论问题的。

查看一下归档中合适的消息作为例子。

可能每隔几个月， 会从这个列表中的消息中提取出一个评论性的消息摘要发到 FreeBSD
网站做为状态报告的一部分 。 你也能从那里找到更多的例子和以往的报告。

freebsd-wireless
: 讨论 802.11 栈， 工具驱动开发

FreeBSD-wireless 邮件列表集中讨论 802.11 栈 (sys/net80211)， 驱动程序和工具的开发。

freebsd-xen
: 讨论 FreeBSD 有关 Xen™ 上的移植 - 实现和使用

  这个邮件列表集中讨论 FreeBSD 的 Xen™ 移植。 预期的流量会很小，所以这个列表旨在同时为设计与实现细节的技术讨论和管理部属问题 提供一个讨论的场所。

freebsd-xfce
: XFCE

  这是讨论关于向 FreeBSD 移植 XFCE 的论坛。 这是一个技术性的邮件列表。 其成员是目前正活跃地进行 FreeBSD XFCE 移植的开发人员， 主要用于提出问题或讨论其他解决方法。 此外，也欢迎希望关注相关技术讨论的其他人士。

C.1.4. 过滤邮件列表

FreeBSD邮件列表是使用了多种过滤方法去消除垃圾邮件、病毒和其他没用的电子邮件。
这部分所描述的并不包括所有常用的保护邮件列表的消除方法。

邮件列表只包含一些允许的附件类型。所有在列表中有MIME类型的附件的电子邮件在邮件列表中被转发之前将被过滤掉。

- 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

一些邮件列表可以允许附件为其他MIME类型，但是以上列出的应该被多数的邮件列表所采用。

如果一个电子邮件包含HTML和纯文本形式，HTML的形式将被删除。
如果一个电子邮件内容只是HTML形式，他将被转换为纯文本格式。

C.2. Usenet新闻组

除了FreeBSD两个特殊的新闻组，还有很多讨论FreeBSD或与FreeBSD用户相关的其他讨论组。
一些新闻组的关键词搜索档案是可以使用的， 有什么问题可以与Warren Toomey wkt@cs.adfa.edu.au联系。

C.2.1. BSD特殊的新闻组

- comp.unix.bsd.freebsd.announce
- comp.unix.bsd.freebsd.misc
- de.comp.os.unix.bsd (德语)
- fr.comp.os.bsd (法语)

- it.comp.os.freebsd (意大利语)
- tw.bbs.comp.386bsd (繁体中文)

C.2.2. Internet上其他的UNIX®新闻组

- 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系统

- 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， 社会性网络

- The FreeBSD Forums 提供了一个基于 web 的论坛用以讨论 FreeBSD 相关问题与技术。
- Planet FreeBSD 提供了众多由 FreeBSD 开发者部落格摘要的集合。
  很多的开发者都在上面发表有关他们工作简要的笔记， 新的补丁和工作进度。
- The BSDConferences YouTube Channel 提供了一组世界各地 BSD 峰会的高质量视频。
  这个是一个不错的观看重要开发者展示最新 FreeBSD 有关成果的方法。

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://www3.uk.FreeBSD.org/

United States of America

- http://www5.us.FreeBSD.org/ (IPv6)

C.4. Email地址

下面的用户组提供了与FreeBSD相关的邮件地址。如果他被滥用的话， 这个列表的管理员有收回的权利。

域	工具	用户组	管理员
ukug.uk.FreeBSD.org	Forwarding only	ukfreebsd@uk.FreeBSD.org	Lee Johnston lee@uk.FreeBSD.org

附录 D: PGP公钥

有些时候，您可能需要校验签名或者发送加密的邮件给官员或者开发者，
这里为了方便您而提供了一些密钥。完整的 FreeBSD.org 用户密钥可以在 pgpkeyring.txt 下载。

D.1. Officers

D.1.1. Security Officer 团队 <security-officer@FreeBSD.org>

```
pub   rsa4096/D9AD2A18057474CB 2022-12-11 [C] [expires: 2026-01-24]
    Key fingerprint = 0BE3 3275 D74C 953C 79F8  1107 D9AD 2A18 0574 74CB
uid                 FreeBSD Security Officer <security-officer@freebsd.org>
sub   rsa4096/6E58DE901F001AEF 2022-12-11 [S] [expires: 2025-01-15]
sub   rsa4096/46DB26D62F6039B7 2022-12-11 [E] [expires: 2025-01-15]
```

```
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBGOVdeUBEADHF5VGg1iPbACB+7lomX6aDytUf0k2k2Yc/Kp6lfYv7JKU+1nr
TcNF7Gt1YkajPSeWRKNZw/X94g4w5TEOHbJ6QQWx9g+N7RjEq75actQ/r2N5zY4S
ujfFTepbvgR55mLTxlxGKFBmNrfNbpHRyh4GwFRgPlxf5Jy9SB+0m54yFS4QlSd0
pIzO0CLkjHUFy/8S93oSK2zUkgok5gLWruBXom+8VC3OtBElkWswPkE1pKZvMQCv
VyM+7BS+MCFXSdZczDZZoEzpQJGhUYFsdg0KqlLv6z1rP+HsgUYKTkRpcrumDQV0
MMuCE4ECU6nFDDTnbR8Wn3LF5oTt0GtwS0nWf+nZ1SFTDURcSPR4Lp/PKjuDAkOS
P8BaruCNx1ItHSwcnXw0gS4+h8FjtWNZpsawtzjjgApcl+m9KP6dkBcbN+i1DHm6
NG6YQVtVWyN8aOKmoC/FEm1CWh1bv+ri9XOkF2EqT/ktbjbT1hFoFGBkS9/35y1G
3KKyWtwKcyF4OXcArl6sQwGgiYnZEG3sUMaGrwQovRtMf7le3cAYsMkXyiAnEufa
deuabYLD8qp9L/eNo+9aZmhJqQg4EQb+ePH7bGPNDZ+M5oGUwReX857FoWaPhs4L
dAKQ1YwASxdKKh8wnaamjIeZSGP5TCjurH7pADAIaB3/D+ZNl2a7od+C1wARAQAB
tDdGcmVlQlNEIFNlY3VyaXR5IE9mZmljZXIgPHNlY3VyaXR5LW9mZmljZXJAZnJl
ZWJzZC5vcmc+iQJSBBMBCgA8AhsBBAsJCAcEFQoJCAUWAgMBAAIeBQIXgBYhBAvj
MnXXTJU8efgRB9mtKhgFdHTLBQJjlXeQBQkF3u+rAAoJENmtKhgFdHTLOVoQALS3
cj7rqYkHiV4zDYrgPEp9O1kAyGI8VdfGAMkDVTqr+wP4v/o7LIUrgwZl5qxesVFB
VknFr0Wp5g9h0iAjasoI5sDd6tH2SmumhBHXFVdftzDQhrugxH6fWRhHs0SaFYCk
Qt5nFbcpUfWgtQ35XTbsL8iENdYpjKXsSFQrJneGSwxIjWYTFn6ps/AI3gwR8+Bn
OffEFdYugJ049O6Vu6YBFJHrnMO7NbF4v95dVYuLtpMIaXWM+V9KITmhaBzFz5fM
Q7UOzcLlbxOYKNIWcp8QQk429mayKW5VUeUExUD1ZzBHn+P6ZG7QTMDu/RmBqiHo
ewCMVz4n9uXT5BiOngE4CvS0WQwHzK+k9MLpG2u/Bo9+LT0Ceh9Ou1rfU5+0tRwl
GyOFFjf3INS7I7gkcAwxQ7dzDItN/UQPZpg8y9mABU2x4enz0AvTnb61d/1dnTEr
tdNgU433he0ZnD1HurZCjBEWC656wv6iMdWcD8gjhMbmEpPmjvXcYlTO6zhEygSM
DiwdQCWK2W4++YJerA6ULBi3niNWBpofOFH8XylV56ruhjtHCo7+/3carcMoPOJv
lVZ1zCKxLro3TRBT15JTFBGqblRyTopFK3PuxW//GTnZOtpQEOV6yL4RAXcWeC1d
1hb5k/YxUmRF6XsDNEH4b08T8ZO8dV3dAV43Wh1oiQEzBBABCAAdFiEEuyjUCzYO
7pNq7RVv5fe8y6O93fgFAmObXVYACgkQ5fe8y6O93fiBlwf/W8y1XXJIx1ZA3n6u
```

f7aS70rbP9KFPr4U0dixwKE/gbtIQ9ckeNXrDDWz0v0NCz4qS+33IPiJg1WcY3vR
W90e7QgAueCo5TdZPImPbCs42vadpa5byMXS4Pw+xyT+d/yp2oLKYbj3En4bg1GM
w71DezIjvV+e01UR++u1t9yZ8LOWM5Kumz1zyQLZDZ8qIKt1bBfpa+E0cEqtNQWu
iGhQE3AHI8eWV+jBkg5y2zHRIevbWb1UPsj43lgkFtAGHk9rrM8Rmgr4AXr531iD
srBwauKZ/MElcF3MINuLH+gkPPaFHw/YIpLRLaZXZVsw3Xi1RNXI2n2ea29dvs/C
Lcf1vYkCMwQQAQgAHRYhBPwOh4rlr+eIAo1jVdOXkvSep+XCBQJjm14FAAoJENOX
kvSep+XC0DcP/1ZB7k9p1T+9QbbZZE1PJiHby3815ccH3XKexbNmmakHIn3L6Cet
F891Kqt9ssbhFRMNtyZ/k/8y8Hv5bKxVep5/HMyK+8aqfDFN0WMrqZh0/CiR6DJh
gnAmPNw/hAVHMHaYGII9kCrFfPFJ02FKoc81g9F08odb7TV+UlvRjkErhRxF+dGS
wQoO0RCbf0Z1cs7nd0Vb2z4IJh4XMxBjWc/uQ2Q9dH/0uRzwpAnR4YX+MG5YrX7Z
zBvDyR0r76iQwRSDKgioNgkr6R3rq1NZGdaj+8b0LzdOqtzKJ/eupDe3+H67e/EN
qymtreGjrubpiU9bKvYArisUqhE5KtguryvR6Qz9bj87nPg33DT3WWGVrwFRxBox
dbWzjQFv0wug8m4GAwVF7fPR5/eW7IHw8zvgn0vSPcZz7MZ4e6Y5jN4kA5/xWJYZ
Sps54qQWB+FA30unIXN68KqdIzONIbtaY3W4/JjJUCm4T+wEjKaH+wJX8w1DMjlg
mkTmGh/UrTyC1vXbPgk9Sy3cRTICR1T9z7W8UlmTtnKrUklrjlFR7SXzrEXzLGOX
Fm+NEHpHNXqzcm6c3QfzY/yQ9HSAQ/t7SUQ9caRePbDz3/msyPxtGFor9roQv6VN
wRXCyRgkH4Y5tPhJAQ8G/FxX+VXFb93QL0lfelb23/BBu6cUwW63SRn5uQINBGOV
dskBEADqo8z6TFAhrvHhJV5wHdj67guoYvpXP8gvdCqos8SLluqi0AWgJEwlqu7L
mKQ6qMoJ+2DN6y+dEtvOVgBAgF63LLf3FQKq9FB/3uqeIiQlCIl3H43f8KttEZzf
/lbry4Y6QhS2OXM31Ut9Q+1IfTGwvs1E8/J1U4jQrAGqNKknXyQyMweJ0jvvcSLJ
nv3S7COUJVOT3cTgVeh3RIQlFzqK2rSQmygDpS8bT8MjCsZr+KGezKpbddKXio4a
QW/e6nCMyYR8bo0GQ9DpsyAOsaENnkghncQhA7GdPZK9xLMNQMCp0OdcZlqRVjRZ
OutuzNW6PPoczs/NQq02YWK4BPtSV7+ldS9gPZTLIpnRNQRzcnA0vnQTqSAfasVw
sAGm+MpH7zcaMf2Tw1K08u7+5gyObgzUzQmGLCgo9VIncnDis0s4gfTmtrr5jCeV
7LYDQX+2fApMtXbVXeKJem1PS+Z6LPbW2HklxYuG5nFgewCYlQjKujfiwW1Clhi4
JQeE1Naobbaar99V/VeoHrOYAEWP0bkUyrFcocLJ+0g3KpjSkctIptgGGpMBKe4U
9O7pWoTki8Yz/uYQn/p0iZcG8SfKM8I4283jdsi5SUiNNJJZCBQTVA7d8MxUVv5+
qpX/v5XqYM3pHza2DLXzwfAE9O2dgN1OMZYIld+OnWcpm2PxIwARAQABiQRyBBgB
CgAmFiEEC+MydddMlTx5+BEH2a0qGAV0dMsFAmOVdskCGwIFCQICKQACQAkQ2a0q
GAV0dMvBdCAEGQEKAB0WIQS2FSd+gQh991yBgztuWN6QHwAa7wUCY5V2yQAKCRBu
WN6QHwAa77gbEADpUBT14cesITuMsOWYsyEtNmB4UlTFWCktk/YzyCotasZxIhMP
Xih9G1tDo9ExIWT8jNjSSA+w0Viua/PirDLvI8JtX1JiK3nwMenwlXwlkRAk9TJW
y944YegHF/5ytntwZ/L4BMYc3MztyZbw+sDwnNBZKYmO8gwfYobtfoGxOR4Onb37
bbUVw62xHQIn2zafSmMQ4oMXZTm9EteIYwgcrC1h+Urv5IXCJZHrqmXCPE5g5XZ1
G9jqkwlaRYWjcLD0qxwc5m9LNrF6OBS9N6S7DncIYt9VupI5OCr1uRSqzqaBMFDC
lTTH+dAx3b6J1KFB0UiHP3FeTalFh8L3NE+dN9apNAgkUWv/v4oo/6dkRu3NZse2
RAo/o2X5r40qk/lhydQRZTSTFsiuH3VUWVsgmqAHnHW7pMMw8FAlKhyRSFnhbW7r
e0jj8XMIO7G5yjQKQCnYuPdXbx++bP1PzsEWDv9j/sph5arcosdo6tEXklWHED17
MEPIton1+NRfsU0peEVggQXlwdTcZN/h7FeCZ56dcwCWdCpSlv6CcWzRXSNUyJpK
a9qfIqBX/monjy7w5IHmhvLwAYI6IoT11h1QDEfGfhrwWPwOjnXsaYm5E7wv8w69
PxMbOJbMpWSg8L7xW3LXKR1VwXggUC1+b3y67E5Ggi1hf0lfTnTMpL2ClO2QD/oC
hMIafhzxbjh2WzgYahVHZH3gpHc1/0Bnc07s9+Pa6EYYM9r0XzezLW7bswOjVloR

FreQ3FIF/2OSN0OGdm7dyYl0OliTIDDDlwK/l8bcckUcpHNR1dw0P3KvDlmLmzZy
G4HmzzSBa9jiFirEfcg2rnGc6Zi382jGVALuYVplPXyMOUiChp0AAQZzTlYpXw/g
pBE6em2k740yuK6WqG4yXXgk67FoH10TQvMd4Q73K4zw+9DMpThlUHcfBmAoViZw
il7C0xl+ysHX8ZI3JU8s1r3XAnpqdHi4Wpixm/ctXbVnTSA3FQr2SctJYqR1VHRW
GMW+Ii2SQDS+t9bZTzOgAPLDtfy+JqhBpwCB1a1EHftkJEojpfZipaYGkf3yc+vN
wUeUHp/csF9CT7Qbqaj1t7fVWzv7jcVKpRwngIT4vTSzqbo6WC34FuUAH0t7tJ5K
eZ625AqEFLmtqtDo+ydJhZrVrXBNXPfkx5hSVW/I9hvckMNwA3t0KfQC2sz+Z1Q1
a4vDWQYRytfyrgZkWGbXMn6l1JyqIolgJZuax2kYs7Vu3t8KptqCbv0ZBAGoMm7r
RLgVodhI9voA8YxCirSChrueJYn+JKk8MIyk3DdXpBoocMIAjFJAUgXjV5NQpZMy
xR8BEiQnBcHRIKVWEEyhbLtHpmCEsnKNyKVGoxs31IkEcgQYAQoAJgIbAhYhBAvj
MnXXTJU8efgRB9mtKhgFdHTLBQJlhctvBQkD8n2mAkDBdCAEGQEKAB0WIQS2FSd+
gQh991yBgztuWN6QHwAa7wUCY5V2yQAKCRBuWN6QHwAa77gbEADpUBT14cesITuM
sOWYsyEtNmB4UlTFWCktk/YzyCotasZxIhMPXih9G1tDo9ExIWT8jNjSSA+w0Viu
a/PirDLvI8JtX1JiK3nwMenwlXwlkRAk9TJWy944YegHF/5ytntwZ/L4BMYc3Mzt
yZbw+sDwnNBZKYmO8gwfYobtfoGxOR4Onb37bbUVw62xHQIn2zafSmMQ4oMXZTm9
EteIYwgcrC1h+Urv5IXCJZHrqmXCPE5g5XZ1G9jqkwlaRYWjcLD0qxwc5m9LNrF6
OBS9N6S7DncIYt9VupI5OCr1uRSqzqaBMFDClTTH+dAx3b6J1KFB0UiHP3FeTalF
h8L3NE+dN9apNAgkUWv/v4oo/6dkRu3NZse2RAo/o2X5r40qk/lhydQRZTSTFsiu
H3VUWVsgmqAHnHW7pMMw8FAlKhyRSFnhbW7re0jj8XMIO7G5yjQKQCnYuPdXbx++
bP1PzsEWDv9j/sph5arcosdo6tEXklWHED17MEPIton1+NRfsU0peEVggQXlwdTc
ZN/h7FeCZ56dcwCWdCpSlv6CcWzRXSNUyJpKa9qfIqBX/monjy7w5IHmhvLwAYI6
IoT11h1QDEfGfhrwWPwOjnXsaYm5E7wv8w69PxMbOJbMpWSg8L7xW3LXKR1VwXgg
UC1+b3y67E5Ggi1hf0lfTnTMpL2ClAkQ2a0qGAV0dMsjqhAAorQ725G342raJ+os
6+E/EFNsr4SR5H+AeinlQ2ymNSeO/ODsV6dmyYD3hed0mAXvIJt2B46fFC4eAP9f
VOIbMMhPMpnJuZyLPDi8gXcZLgWSRhJ88R98KIsmKlh+/fdZM4RI1JLjICi7kyNR
4jtKCzLj0DYVBzp1mn0lTwtFzv7SC9djpqFLnO5YoGPWfQHHhY02Trh2posRwAHO
oacXSFvsoQv6k6XNlStJ4lnrkH6t+Od4kU3/TJ0eQXs7Zd2WEVnMe1IhbihsGcAY
mzZzZlLOhskHCeVE2taHiXC6h4tC3/69I16N8ICauxGY41clPhiNmVaAzmkunOPz
ry5utl6HkpZ5/3UMVHI1JIvsfJW+vSMUhdcQILAv6DbRWWHeax3ZZ6iAVGCtJS7U
glwZM1Xor0okGtIS+aJ/Cw7tZ8Nm18lutcrf2MVW+BWpzMQKnWFQYTn1NEWjzYnx
9Na22+E8AvW02TdS0NSiP0sG/0q7lBNEck9vH4WEbbEXktj51Dg4ISUhQyW8BWwW
X+kSiNeqtcaikUb8SFj5vpTDotTSzikfT/jisvR5goTMNFCVHFZdXCdsbUZd8Iub
egAOh6Db/06y3mFYDEfcGJipab400OY03a2xw9Vz+YxrKfELCTBo2tZv+3K8kXgq
XFcbyJnkXmjnYM/sw5kKqtzuc7i5Ag0EY5V3BwEQAMpFVczZo9ZPNsgW791UW5o6
wnrnd1nIO+S4rc37q2TEz8KGHCuxo5NwffZ2t6Ln04BI54pbapg17b7a0hPka37H
FkL28n4VyMdx0CsAm3QEfUsdK6xwKV2SucYeVcrV1upcN4PdXD7su1I7/A4CWXFJ
G047zJ0Z89lJZiQEiAq7ghvEoinC0sm+0a6ao/ocqCgWCKM1yCPOyzJXleRrv29S
RnYziMR+q2U0x9xg9Xl6GMwUmFwbJc9nORVvLH7fbU6/du8EgoAYrglFOFZG/TSo
lSGWRSMiavz0JSD/i+rEN4aIT4WfBe+L9Wy1AmrNxiAO+zKmzHQu3JSxDncr+y+h
cd+W0gqw10FoI9jWLcL7kR+6a0iOjuJSXSopq2l3DafiPxtCFmr4CGQhzBHM6e4/
v/NNd3F0XpVbJ6RQph7lkfvfz8q2lvUlHhezJ0p1xXmhff9CHjdVMhmAmz5+imBA
Xk2mottNfKb0pFEen1xY3K/UPA4g+oPsSj495MsvIg9eIMCcC3/z0SEUMWH/styy

```
JzPqfpyfGwZeTcIj9vg2o+RnGvmcLVYA/EGToPk905kv/cK73oy8bZyOB0zMg7T9
PaWgLUO0sqjqo0Mw3knFySg3oRXlcilPQvfPdX0JvwLpc9DWlr1+1GkCXJ08lWug
Jc96CJQupKRb1IbC0oUXABEBAAGJAjwEGAEKACYWIQQL4zJ110yVPHn4EQfZrSoY
BXR0ywUCY5V3BwIbDAUJAgIpAAAKCRDZrSoYBXR0ywwtD/wIDmEcHdFlyFRTomUB
jbeK2uzcZIhkkgL58lc63UPle5iJ2FBvmYS+0rQS53sVEsccn5KfkOwTryKllvWb
l0IzuiqfawxALcfWpfZJHzTMSnDHfgXvOOyFMQruqRDAHAr7PNC0CnbT0sEF2ZFz
ad8M9fLqtkXUx4mgECNGJ4CVqg75KY8uUzv/BmRwEf587FT5/iAIed5MjFB2VFDX
9GABcvTTbHxCZIxnxl3cs15SxT0lAofZ2ueU6kWYWZSXFeaEM/4ymPJws2mmV0Ak
bJghLXCn9Mx3nX6NTZZ9Harbru+RzW3/Hg3DZd0J9vko8PafP0l1NWtgyX74CqvT
gjzTxXTnqrRXzcczK7fhcC2u4i0prPtXXcyyi7SwpoLikaZCLFFhUmOx+mS5Tjtg
FyFZBNxnO7iAwkzfcTcC9sPoWaFmiQf6q5EIYzG+WQpncj80mxl3HWOP6oFj/hZJ
RYseKeMkvJzLTo87rFdM6CsMrLwETR6e+aWM0btPFil1rXVACNOjsy0bxTV8OJEf
yxnYmyjvnBvB0kdiaVEDdVhxgSqzLAX4mgXa49/V6M/uzMr+n3/A1Jdk4V6fVm8S
5cFIXxoUat3cB4xGaT9OWD3o1NPr6eS9Vo0EsJlRl81SG68fS+Qtk2fX27T68YG4
Aa3zMfZxUsVuFLtTuQbRC+fJpIkCPAQYAQoAJgIbDBYhBAvjMnXXTJU8efgRB9mt
KhgFdHTLBQJlhcuqBQkD8n1oAAoJENmtKhgFdHTLoO0QAJsTE9fkleb7YzPEuP9G
J3jx8PGdWm7n+8UNdr24kS6gOXVUfPZrWa5So21hcIwZb4PZDqHSVSQnRciKhSnG
7gplYPNGZ4+FWbLr/mBRYarjkVFLUuCPexSIjxV1KSGJnWs9YTVAKZAz75GpCML6
jD6biCOQCQ86wqOdWvZIZR8YvurrxR64ABB0rjbsaG8cNOUX1cwAfdLwthf64dS+
2m3lqNGDHkP5eNL0RIxC5gXYEp0lvmlMH3ZuO5WrfH73PTDg89bxXeuhrFmSEwf4
xWm603oi8/2qQvR9/7jb0o+t71NQuWrWIFONZWWgZBUGso+uyT3XgY4YqKGR3z2Q
zKHYnJ6M7SvSYpqS7RtcxcCXF0HGNfES8cAgtKVpFtbtSwXXp8O8oLyjmVIO/NjU
pbLOGdFIsarsezLFV9f2fqZ63J34hyUSg8LrYVV1fA5DJUpebbX4hLpdk0MMtgG4
3BwKIGlJTpL5RkQ/uQU3YW2kairy7o+1imDD0TRzQxtdjVOI5vnlTNcfJZIIfLx4
drABA12OvpX3dfPV62R+8BAlJFT430CG6AISJIBqJRFvuikmnZGUvEHmOUs/FLbb
aXTPkKc7tR2WIwljRvMV+Qk84cWcX6YchMslMuiDM1mtlQZig34WHGSE+zCWnXAs
lIHlSwox7qfdO0Kz2XncSbIA
=QvUh
-----END PGP PUBLIC KEY BLOCK-----
```

D.1.2. 核心团队秘书 <core-secretary@FreeBSD.org>

```
pub   rsa4096/4D632518C3546B05 2024-02-17 [SC] [expires: 2025-02-16]
    Key fingerprint = 1A23 6A92 528D 00DD 7965  76FE 4D63 2518 C354 6B05
uid                 FreeBSD Core Team Secretary <core-secretary@FreeBSD.org>
sub   rsa4096/CABFDE12CA516ED2 2024-02-17 [E] [expires: 2025-02-16]
```

```
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBGXQ1o8BEAC+Rcg8cmVxuP17Vu+q5KgCx/XiulQuqKXAqqBlYCH2jqk6DINP
yFrREGBhzd/qNmlAYEahQ4Zgl0bUZNTrZVDyzicOvPP0jH+KSTQwRs7NOawEdlVO
cyHrwDCPEqf5ZzD4NhfTriEOw+j0pEH/onitUGvoQRtx15xWyaJQxDEBMTYMLewE
```

86D1bltwnTNczE3UZb7oQLJXkAX5hcLtou70XJGgZITvJkK+kp/xot2eFjnqRz/u
WeXnKhYAmC07EKwZ1uw047eHKwMMRBYqzApLwoQtfE430Kxf2q8de64x8zDbi6YM
1J4r8OAxOtHVyfJ0j7Q23DEZz0VVb4b1Tx5OG2Re/KSNvqI0awJO4TcRmOR88OyY
dzyXgnX6Sa7GVQY1FXvn7vtFuDAt7egZOzeomSHL9bdX07LTQ4UtM88EV9wm3q4q
smoatV9jsvPQ1zxCU3aQD/5eWTJH2/kz1LIuBL/Qi5XQpJn91lBtUWJrCgkHWPGu
f//rnnXmsG7DAChW+yZ7cFO8lfNa8sFhPqSxCYphWmJTrvadyQtDngB8JakWdnmK
pfGS6y5lel+181vw38ZZKt04AKM+nDY8051lBM7Q9Q6kTLI33UZeImndx5xYukVD
kV6aQ31HYfEark15c7iEz+OAcwFnM2ntXMt7kKGd40CqzusiPcQkPqPbAQARAQAB
tDhGcmVlQlNEIENvcmUgVGVhbSBTZWNyZXRhcnkgPGNvcmUtc2VjcmV0YXJ5QEZy
ZWVCU0Qub3JnPokCVwQTAQoAQRYhBBojapJSjQDdeWV2/k1jJRjDVGsFBQJl0NaP
AhsDBQkB4TOACAsJDQgMBwsDBRUKCQgLBRYDAgEAAh4FAheAAAoJEE1jJRjDVGsF
nacP/3PSg8JPmWoBfWrgT287NZ7OAU16/uGpDxlBUoVeEtkEDqZVW8yBFzrMhbwj
bJs3CZ+L85HMUDLZoxSwVnPM8PLVRzHTybYV7agYYzMox5C/jp2aeAgy9KYVd0Tk
07GMTYrSh4fhHWpxXz7IBOxk0RXvQxTHlg1u0DASkibYb2UTDcUNG5Q9kP/8jaIZ
kVDX8a5LDdOCgWaYdKPg4blv/UMjkegJz+Ayp7gXTcux6koW5F6ysSw9sgLBWb2D
b/KNIi4MBMe46xyXB/dqGAR4ibrUXTcQ4OAZNq1L6uWG1A49XuSgykdIwr00MzQw
wfVpKT31ww4ayVHLgj7NuqPlab9S5/fPfJ4MAvGE4GqWQFgsPKgKImUMgnnxTGpv
l7Dqk2MnWqn+wEi0bRES0PVBG96G+sZJQeaxBhoB+HwUSFqoZQg166AJIl//4t2w
bx0a1aWQSS0DZt3wsQW3NW9AE6L+FnFfIc2pQVoLjmvcGalDuvnemRmEOgotiZmt
32bi2aWxg0/Qio2rjLS2LpV+fhwDSN3Agvtnu53yUdD1TFFjTSMloM4SKhiXoPbI
XgfCLiBLNMsZL0AvO7wQfSePzPYxDLyEcwsfPJ8be+eGG1L62RUyad+MdfyXMH/S
m0sgqW/MW6Nv10RyPQOq3Jbgmp2laRMzKTOvQt5WwQf2FEl9uQINBGXQ1o8BEAC9
1cBYn6Z0QmM0OFWdXQI6fMOeNokaa6ngPgt7bzW5NjryqTdwyHOPZdm4DWf1SO/0
+fJRCqxbICyuMAFrb9fDle8bodALjm5ZquTL3D61HpZD4+RwOzOjYP6wLm7h38HT
/yIyK882Ovlw4Xz/TeSiL/VUSWE9twW7yz3oreCeLUBAfzacS9y+syO+aquEd0/x
JBz+mPQbrqfS64rCZXMZEivgsjkQoE6RM+n1rF4kw4Eu3E2kPevVwsoAaY+MEUM8
JAXaJMaNcLIhbeMy7dO/z6z2I3h5bUw5KxfVwzYSzSeRpYh53dNaB4NY+f5/vTrL
4dZmqBcLgcV0zZ02dj/u0SiwWlUFUpFGuSiW16DN7+2zG1zOWi7Nl44JawM62Tlf
mO8zruVGEHaV3e8fFwBLRKM0Sc7e3aLECISsfYeC5ZbRRbpQ1KX+VQr3FBKAMzG4
l9Go7vZ+UcLKpQx2rVPTJt1vDnRV49X6CF2Q/lV9iafQ4MTy6ACdAloT1yfH/lhU
iWQo1qDyRCSlmNBDsYl8gLrwMp4gGQAv3imZHxnJF5ru3nUYGG0U08D5mf2sWv5P
Wh7By8Jm8bmaP8cUF86lO9BJXh2d9QN5jqrAtXqYzenZ+ABSoL1XrD/yv3270rH7
H4gAUtgP+vJ3uMyRu9055OC+ie/b613NojCW5nYN2QARAQABiQI8BBgBCgAmFiEE
GiNqklKNAN15ZXb+TWMlGMNUawUFAmXQ1o8CGwwFCQHhM4AACgkQTWMlGMNUawXh
7w/+KjbEWTwAhjm2HJ3w4tXtPC5URg+A+BzYYVH/q0+956c1QeD0LYafHBw4LEMI
lhRvHQnmzwjtY8v/DgmLOVDMiMWVHo0Q2iQyMvOT1WyEPcgOTJLhvyVzDqRZx7AS
B4G8uNVkKAdBZ7OSXAP27lR/2SEoG05esw8b7Y39pVtucC3aeiua+19PLJWadBjj
XuvXuSCho0km+nk4IgadYmxIDyiMeyKZ8wCl7CJkzECm83q2OtNsMe3k8lgEXybt
KlQxnYApZmhqLMV5ob8WOk3AgAVsif1m332CiEIb1Sfx6wt3nXy/410CXdDDucuj
ndJVfJ6Un33tn0irZ5scPA2HmzK1PGMfgOGtkM8B3LE/x8kEKeWKb3l9boB32Unm
iTfKgEna+JISEab3bzOPWdCQFB8LyGXuWlhtvqmRoX8GtiMRy/F4mzh+l3lYHjj3
4EvPVyippO5zwU+S9HElJ2G37K6zrOmd5cGBrw4aBDoO70QVrMN4O86uvC9kChDb

```
qyFF5UgXg29QxJjiSCv98ksDMqpJ5AFYrmrsBtwU64OANrxxJ4AZLQ1apYmG9RWD
VHZgfeI60FNBLfKwix9UffFT7piQ/MLrjSde8gPH5S6ezBMrYpfGEopaI9A5qXe3
LnHz88gfdmaBM77YDZM/p23nmCrUxlE3kkbgjTY8NRjYyF8=
=MkAH
-----END PGP PUBLIC KEY BLOCK-----
```

D.1.3. Ports 管理团队秘书 <portmgr-secretary@FreeBSD.org>

```
pub   ed25519/E3C401F60D709D59 2023-03-06 [SC] [expires: 2027-03-05]
    Key fingerprint = BED4 A1D3 6555 B681 2E9F  ABDA E3C4 01F6 0D70 9D59
uid                 FreeBSD Ports Management Team Secretary <portmgr-
secretary@FreeBSD.org>
sub   cv25519/2C92B55E27A641C3 2023-03-06 [E] [expires: 2027-03-05]
```

```
-----BEGIN PGP PUBLIC KEY BLOCK-----

mDMEZAXJvxYJKwYBBAHaRw8BAQdASFAC20WL3R1T6uNyGMZbfJCxDkcP4C5vi3Op
tcZ2fbq0R0ZyZWVCU0QgUG9ydHMgTWFuYWdlbWVudCBUZWFtIFNlY3JldGFyeSA8
cG9ydG1nci1zZWNyZXRhcnlARnJlZUJTRC5vcmc+iJYEExYKAD4WIQS+1KHTZVW2
gS6fq9rjxAH2DXCdWQUCZAXJvwIbAwUJB4TOAAULCQgHAwUVCgkICwUWAwIBAAIe
BQIXgAAKCRDjxAH2DXCdWYN1AP43TjyfZtZ3DLYT++g0+SuPsoO/3yWVybA+UmFL
zb8MngEA+LLNUfvEwCuXS/soh+ww5bpfmi3UUmeGiQEAXug3iA+JATMEEAEKAB0W
IQT7N0XIbxXo7ayBMvzYKU7Du8TX1QUCZAXLkwAKCRDYKU7Du8TX1XHMB/9R1MX4
6zMgpKqPPt76GOI+eGEdBK6bY8aJZjQGdqTh9f6VtXVoTGIG7cvhc9X8tDBoB0PT
2KZWheF51AV1+NHU4HwLAQ1BMebrFvWSfkw4xg4fBGwDhz9/GN85No+Js772V5ey
8lRiL6meRVWxMlLyWcxGd8JjcC5yX/iAUQ3SBGCLqW7unWjjg7CTd+AMBwcqPGrv
ax8q6eFVguJcHJAjMnKf6HAy4cpK3s+uMoUBCGnszSN12B3ysKfyC4pNO/pix5tA
Q5v8aRqTeFPh5zmNhWo0KGPzplTPqRQSHDl7GDQC8Ru3MhzFkeWzHsexjZVwS6W2
DPcYpuuAsA0XOZIZiQIzBBABCgAdFiEEEBpxaxYrAOVb7eoFrbv4YQo3ibcFAmQF
0u0ACgkQrbv4YQo3ibccwg/9F2Xuic3nhKxRbB3mJeDo6SYQETa/Gh1qQ34+8zlt
8UMazOx67gnYQfy+pXjro6eQ2up0a4eUYezcNOudqAQD21nRz3HA6EQVNcE/TzEA
xl5CJntTaLOt7S+EDXFW5BuQIvhhoMGgm8+WNVgA0EJ7tfL0OcYBSvr19fqwChEn
9c14cSk6mgHSsleP5NvskYN053pxHwy0LTSb8YBBv52th37t/CRFC1363rS5q+D7
JixFopd1O5pKpA5ipvE4gGgRjPtwjx0SjjepwK/3fuhEJQQyKzTIKlMfu2Dj/iR2
Li1Sfccau5LQXOj9fUITU3u1YG7yrm8VGzT7ao4d+KRwgMLjd2pLqiGIbbJwGBiP
FRmtilWQoeIlmSlFX4obAA517DOK0pW1mH8+eEn4EJd3SekT3yzFyKTASv0J48Z8
3F928xg+eZvHxVC0t1J+J5IG0gt3EEncuWKIPQGR7PiQbti6R3FQVTz6WfMWOebP
Qi0E9F/Aqakr6Vj2sKGrDq+ebpaF5G8Yw1YrUl2IDiPzkCegp3ZbI0wh11Xvzhi8
LXPQGK4jBQas4G8cegfitzmtdGRHYrbMv0R9I4mvaL+WlOuD2AvyVG28lguqVhnN
AZP+ohdquYyX2CNCVvbKWAtXo6Ur0vWG8BL8m6defAtEkIwVBALaOHQOSI3aNUz4
lwy4OARkBcm/EgorBgEEAZdVAQUBAQdAsefmSfxEOdOr02+K/6noYCuJ1FeAWVz6
jFYQ+9w6jggDAQgHiH4EGBYKACYWIQS+1KHTZVW2gS6fq9rjxAH2DXCdWQUCZAXJ
```

```
vwIbDAUJB4TOAAAKCRDjxAH2DXCdWRl4AP9h5ot212BK29S6ZcMBhHvmtF5PG1oD
c7LnZycSRmbFiwEAndCMpAGOhDW8iVgDd0wLQq/ZMPe+xccfG1b3zFH2EgE=
=iiAT
-----END PGP PUBLIC KEY BLOCK-----
```

D.1.4. <doceng-secretary@FreeBSD.org>

```
pub   rsa2048/E1C03580AEB45E58 2019-10-31 [SC] [expires: 2022-10-30]
    Key fingerprint = F24D 7B32 B864 625E 5541  A0E4 E1C0 3580 AEB4 5E58
uid                 FreeBSD Doceng Team Secretary <doceng-secretary@freebsd.org>
sub   rsa2048/9EA8D713509472FC 2019-10-31 [E] [expires: 2022-10-30]
```

```
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQENBF27FFcBCADeoSsIgyQUY8vREwkTikwFFlNg31MVy5s/Nq1cNK1PRfRMnprS
yfB62KqbYuz16bmQKaA9zHN4FGfiTvR6tl66LVHm1s/5HPiLv8sP14GsruLro9zN
v72dO7a9i68bMw+jarPOnu9dGiDFEI0dACOkdCGEYKEUapQeNpmWRrQ46BeXyFwF
JcNx76bJJUkwk6fWC0W63D762e6lCEX6ndoaPjjLBnFvtx13heNGUc8RukBwe2mA
U5pSGHj47J05bdWiRSwZaXa8PcW+20zTWaP755w7zWe4h60GANY7OsT9nuOqsioJ
QonxTrJuZweKRV8fNQ1EfDws3HZr7/7iXvO3ABEBAAG0PEZyZWVCU0QgRG9jZW5n
IFRlYW0gU2VjcmV0YXJ5IDxkb2Nlbmctc2VjcmV0YXJ5QGZyZWVic2Qub3JnPokB
VAQTAQoAPhYhBPJNezK4ZGJeVUGg5OHANYCutF5YBQJduxRXAhsDBQkFo5qABQsJ
CAcDBRUKCQgLBRYDAgEAAh4BAheAAAoJEOHANYCutF5YB2IIALw+EPYmOz9qlqIn
oTFmk/5MrcdzC5iLEfxubbF6TopDWsWPiOh5mAuvfEmROSGf6ctvdYe9UtQV3VNY
KeeyskeFrIBOFo2KG/dFqKPAWef6IfhbW3HWDWo5uOBg01jHzQ/pB1n6SMKiXfsM
idL9wN+UQKxF3Y7S/bVrZTV0isRUolO9+8kQeSYT/NMojVM0H2fWrTP/TaNEW4fY
JBDAl5hsktzdl8sdbNqdC0GiX3xb4GvgVzGGQELagsxjfuXk6PfOyn6Wx2d+yRcI
FrKojmhihBp5VGFQkntBIXQkaW0xhW+WBGxwXdaAl0drQlZ3W+edgdOl705x73kf
Uw3Fh2a5AQ0EXbsUVwEIANEPAsltM4vFj2pi5xEuHEcZIrIX/ZJhoaBtZkqvkB+H
4pu3/eQHK5hg0Dw12ugffPMz8mi57iGNI9TXd8ZYMJxAdvEZSDHCKZTX9G+FcxWa
/AzKNiG25uSISzz7rMB/lV1gofCdGtpHFRFTiNxFcoacugTdlYDiscgJZMJSg/hC
GXBdEKXR5WRAgAGandcL8llCToOt1lZEOkd5vJM861w6evgDhAZ2HGhRuG8/NDxG
r4UtlnYGUCFof/Q4oPNbDJzmZXF+8OQyTNcEpVD3leEOWG1Uv5XWS2XKVHcHZZ++
ISo/B5Q6Oi3SJFCVV9f+g09YF+PgfP/mVMBgif2fT20AEQEAAYkBPAQYAQoAJhYh
BPJNezK4ZGJeVUGg5OHANYCutF5YBQJduxRXAhsMBQkFo5qAAAoJEOHANYCutF5Y
kecIAMTh2VHQqjXHTszQMsy3NjiTVVITI3z+pzY0u2EYmLytXQ2pZMzLHMcklmub
5po0X4EvL6bZiJcLMI2mSrOs0Gp8P3hyMI40IkqoLMp7VA2LFlPgIJ7K5W4oVwf8
khY6lw7qg2l69APm/MM3xAyiL4p6MU8tpvWg5AncZ6lxyy27rxVflzEtCrKQuG/a
oVaOlMjH3uxvOK6IIxlhvWD0nKs/e2h2HIAZ+ILE6ytS5ZEg2GXuigoQZdEnv71L
xyvE9JANwGZLkDxnS5pgN2ikfkQYlFpJEkrNTQleCOHIIIp8vgJngEaP51xOIbQM
CiG/y3cmKQ/ZfH7BBvlZVtZKQsI=
=MQKT
```

```
-----END PGP PUBLIC KEY BLOCK-----
```