

FreeBSD 简明用户指南

Yanhui Shen • July 12, 2015

修订日期: 2018.05.18

注意:

1. 本文主要目的是介绍 FreeBSD 的正确使用姿势，确保用户在大方向上不出错。
2. 本文略去的操作细节都可以在《FreeBSD 使用手册》或者 `man` 帮助文档中找到答案。
3. 有能力的同学最好去阅读英文版“FreeBSD Handbook”，中文版有些落后。

目录:

- 与 Linux 发行版的区别
- 系统的大致结构
 - 基系统
 - 第三方应用
- 基系统的更新
 - 二进制更新
 - 编译更新
 - ZFS 的更新
- 第三方应用
 - 安装与更新
 - 科学卸载
 - 对 ports 用户的建议
 - 自制二进制安装包
- 配置文件
 - rc.conf
 - periodic.conf
 - 其他配置文件
- 文件系统
 - ZFS 使用建议
 - tmpfs 使用建议
- 相关资源与应用
 - 一些网站
 - 邮件列表
 - 新闻资讯
 - 书籍
 - 谁在使用 FreeBSD

1. 与 Linux 发行版的区别

Linux 发行版 (distribution) 本质上是一个**组装品**。发行版的制作者将 Linux 内核与其他开源软件整合起来, 例如 GRUB、GNU C Library、Bash、Coreutils^[1]、net-tools^[2]等, 再加上自己的包管理系统, 最后分发给了普通用户。

FreeBSD 是一套完整的操作系统。FreeBSD 团队不但开发自家的内核, 还开发 libc、POSIX shell、大部分系统命令、文档手册等。所有这些都在一棵源码树下, 有一套完整的构建脚本 (Makefile), 高级用户可以通过 `svn` 和 `make` 命令更新操作系统或者自制安装镜像。

其他区别如下:

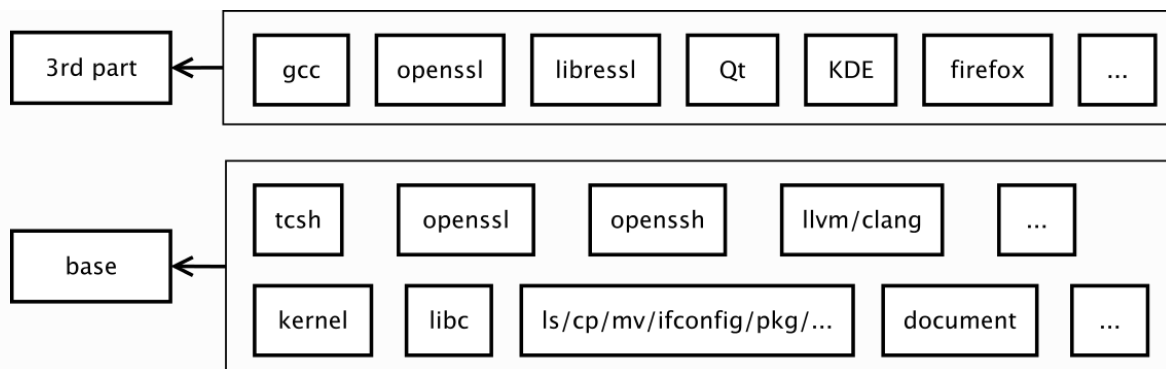
- FreeBSD root 用户的 shell 是 `tcsh`, 普通用户的 shell 是 `sh`。最好别改 root 用户的 shell。
- FreeBSD 的关机命令是 `shutdown -p now`。
- FreeBSD 的 `make/grep/sed/awk` 与 GNU 那套不同。
- FreeBSD 有个叫 `ee` 的文本编辑器, 风格与 `nano` 类似。
- FreeBSD 管理系统用户与组的命令是 `pw`。
- FreeBSD 使用 `gpart` 命令管理分区。
- FreeBSD 原生的防火墙是 `ipfw`。
- FreeBSD 原生的文件系统只有两种: UFS、ZFS。
- FreeBSD 有自己的引导方式, 不一定要用 GRUB。
- FreeBSD 不是 Unix[®], 然而历史告诉我们 FreeBSD 就是 Unix。

开发者需要注意的:

- 不要在脚本中写 `#!/bin/bash`, 要写 `#!/usr/bin/env bash`。perl、python 等同理。
- FreeBSD 第三方库默认安装路径是 `/usr/local/`。
- FreeBSD 没有 `epoll`, 但是有 `kqueue`。
- FreeBSD 没有 `inotify`, 若要移植 Linux 相关代码可安装 `libinotify`。
- FreeBSD 没有 `sha256sum` 命令, 但是有 `sha256`。md5 等类似。
- FreeBSD 特有的 `jot` 命令能够生成数列, 某些场景下可以与 `xargs` 搭配使用。
- FreeBSD 的音频系统是 OSS, API 比 ALSA 简单好用。
- FreeBSD 的 filemon 内核模块可以监控子进程的所有文件操作。
- FreeBSD 的 libutil 库也许会派上用场, 相关文档: `/usr/src/lib/libutil/*.3`。
- FreeBSD 安装 Linux 二进制兼容层后, 能够直接运行 Linux 可执行程序。
- FreeBSD 项目大部分源代码遵循 BSD License, 对商业闭源应用十分友好。

2. 系统的大致结构

FreeBSD 大致由两个部分构成: 基系统 (base)、第三方应用 (3rd part), 如下图所示:



2.1. 基系统

基系统主要是 FreeBSD 团队开发的^[3]，会定期发布新版本，当前有 9.x、10.x、11.x 三个系列。我们也可以粗略地认为基系统就是 FreeBSD 安装盘给我们装上的那些东西，它们主要位于以下目录：

```
/boot
/etc
/bin
/sbin
/lib
/usr/bin
/usr/sbin
/usr/lib
/usr/share
/rescue
...
```

2.2. 第三方应用

第三方应用是通过 `pkg` 或 `ports` 安装的，默认的安装前缀是：`/usr/local`。例如 `tig-2.1.1` 这包安装的所有文件如下：

```
/usr/local/bin/tig
/usr/local/etc/tigrc
/usr/local/man/man1/tig.1.gz
/usr/local/man/man5/tigrc.5.gz
/usr/local/man/man7/tigmanual.7.gz
/usr/local/share/licenses/tig-2.1.1/GPLv2
/usr/local/share/licenses/tig-2.1.1/LICENSE
/usr/local/share/licenses/tig-2.1.1/catalog.mk
```

因此在 FreeBSD 中，系统自带的组件和用户安装的第三方组件有着清晰的界线，而不是像大多数 Linux 发行版那样全部混杂在 `/etc`、`/bin`、`/usr/bin` 下面。并且基系统和第三方应用也有着截然不同的更新方法，下文将分别讨论。

3. 基系统的更新

尽管 FreeBSD 名义上不是 Rolling Release，但是 FreeBSD 基系统完全具备滚动更新、无缝升级的能力。基系统的更新方法有两种，一种是二进制更新，比较适合于普通用户和服务用户；还有一种是编译更新，主要适合于有一定开发经验的高级用户。

3.1. 二进制更新

二进制更新是通过 `freebsd-update` 命令实现的。该命令其实只是一个 shell 脚本，它会借助 `sha256`、`diff`、`fetch` 等一系列基系统中的命令完成升级工作。它的工作流程大致是这样：首先从 FreeBSD 服务器上下载系统文件清单（含哈希值）、然后根据清单核对本地的基系统当前处于什么状态、最后从 FreeBSD 服务器上下载并安装所需的补丁（如果涉及配置文件，中途可能会让你手工合并）。所以从工作原理上来说，它总是可以把你机器上的**所有系统文件**同步到和 FreeBSD 官方发布的**完全一致**。只要 FreeBSD 官方发布的系统能跑，那你的系统也能跑，几乎不可能把系统升级挂了。

既然 `freebsd-update` 有能力比对所有系统文件的哈希值是否与官方发布的一致，显然它也可以用于检测系统中哪些文件被入侵的黑客篡改过。但是如果到了这一步，那么系统中的所有工具都已经不可信。所以最好是先关机，然后通过可靠的安装盘启动机器并挂载硬盘，最后使用安装盘中的 `freebsd-update` 进行检测与修复。

3.2. 编译更新

编译更新是通过 `svn`、`make` 以及 `mergemaster` 命令实现的。

首先要用 `svn co` 从 FreeBSD 官网把基系统的源代码下载下来。

如果你想追 stable/10 分支：

```
svn checkout svn://svn.freebsd.org/base/stable/10 /usr/src
```

如果你想追最新版（也称为 current）：

```
svn checkout svn://svn.freebsd.org/base/head /usr/src
```

此后，更新基系统的大致步骤如下：

1. 与 FreeBSD 官方服务器同步源码。
2. 阅读更新公告。
3. 编译安装基系统^[4]。
4. 编译安装内核。
5. 重启。
6. 删除废弃的系统文件。
7. 删除编译产生的临时文件。

为了简化工作，大家可以使用下面这些 tcsh alias：

```
alias base-sync      'svn update /usr/src'
alias base-notice    '$PAGER /usr/src/UPDATING'
alias make-world     'echo "`date "+%H:%M:%S"` making world..." \
```

```
make -C /usr/src -j3 buildworld > /tmp/make-world.log \  
echo "`date +%H:%M:%S`" build completed."  
alias install-world 'mergemaster -spF && make -C /usr/src installworld && mergemaster -s  
alias make-kernel 'echo "`date +%H:%M:%S`" making kernel..." \  
make -C /usr/src -j3 kernel > /tmp/make-kernel.log \  
echo "`date +%H:%M:%S`" build completed."  
alias make-rm-old 'make -C /usr/src -DBATCH_DELETE_OLD_FILES delete-old delete-old-libs'  
alias make-rm-obj 'chflags -R noschg /usr/obj/* && rm -rf /usr/obj/*'
```

若要搞懂上述 alias，建议阅读 `/usr/src/Makefile`，其中有详细的注释。

P.S. 想裁剪定制基系统的同学可以参考 `man src.conf` 和 `make make.conf`。

3.3. ZFS 的更新

不管基系统用二进制更新还是编译更新，如果想更新 ZFS，那么都需要阅读本节内容。

ZFS 的更新是通过 `zpool upgrade` 命令完成的。早年 ZFS 有版本号，每升级一个版本号就引入一些新特性。自 ZFSv28 之后，改成了“Feature Flags”模式，即版本号跳到1000，并且永远固定，所有新增特性由用户逐项决定是否启用^[5]。启用 ZFS 新特性并不需要重新格式化^[6]，只要按以下步骤操作：

1. 更新基系统。
2. 更新 bootcode。
3. 重新启动。
4. 执行 `zpool upgrade` 查看可用的新特性。
5. 执行 `zpool upgrade -a` 启用全部新特性，或者通过 `zpool set` 单独启用某一项特性。

注意，如果缺了步骤2，只做了其余4步，那么再次重启后很有可能导致系统无法启动。既然第2步如此重要，那怎样更新 bootcode 呢？

如果你使用 GPT，并且第一个分区 `ada0s1` 是 `freebsd-boot`，那么可以参考我这条 `tcsh` alias：

```
alias install-bootcode 'gpart bootcode -b /boot/pmbr -p /boot/gptzfsboot -i 1 ada0'
```

然后步骤2就等于 `install-bootcode`。

如果你没有使用 GPT，或者第一个分区不是 `freebsd-boot`，那就只能多读一下 `gpart` 帮助手册了^[7]。

4. 第三方应用

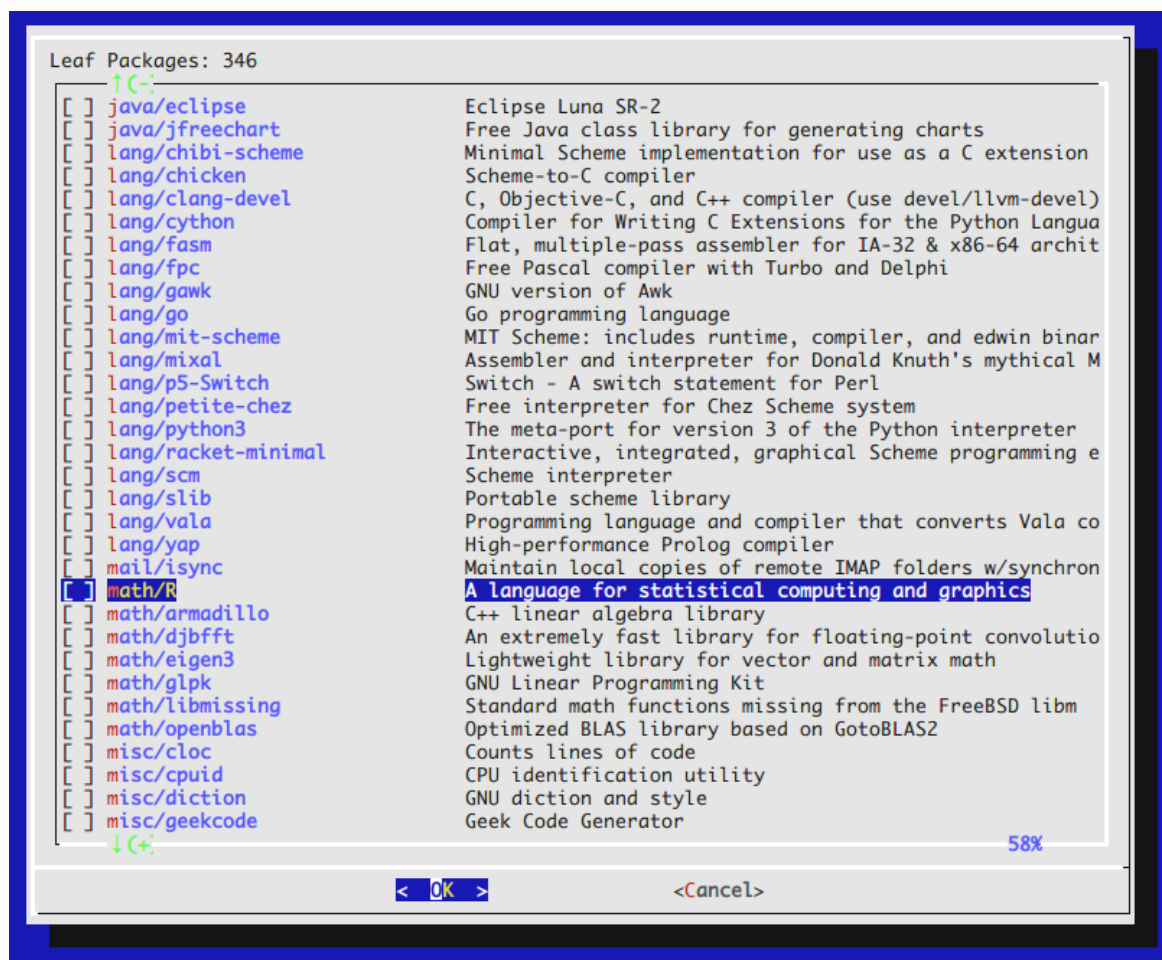
4.1. 安装与更新

第三方应用是通过 `pkg` 或 `ports` 安装的。前者是二进制安装，类似 Arch Linux 的 `pacman`，适合于普通用户和服务器用户；后者是编译安装，类似于 Gentoo Linux 的 `portage`，适合于机器性能比较好的、需要深度定制的用户。注意，这两种安装方式最好不要混用。

关于更新，如果用 `pkg` 安装应用，那么更新还是用 `pkg`。如果用 `ports` 安装应用，那么建议使用 `portmaster`，注意该命令并不是基系统的一部分，需要用 `ports` 安装^[8]。

4.2. 科学卸载

不管用 `pkg` 还是 `ports`，卸载都是通过 `pkg delete` 这个命令完成的^[9]。但是一般不建议直接那么干，因为这样很容易把依赖关系搞坏。通常来说，只有当一个应用不被其余任何应用依赖时，我们才能卸载它，这样的应用也称为「孤儿」或「叶子」。卸载孤儿应用是件容易的事，只要一个命令就行。但想要「卸载孤儿应用，并卸载它依赖所有的不被其余孤儿应用依赖的应用」，就不是那么容易手工操作了。因此最好是借助专门的卸载工具来完成，推荐大家使用我编写的 `pkg-rmleaf` 命令^[10]：



4.3. 对 ports 用户的建议

如果你决定使用 `ports` 编译安装应用，那么可以参考下面几条 `tcsh` alias：

```
alias mininstall 'make install clean'

alias pkg-msg      "pkg info -D"
alias pkg-fresh    'pkg version -vl "<"'

alias port-init     'portsnap fetch extract'
alias port-sync     'portsnap fetch update'
alias port-rebuild  'portmaster -d'
alias port-update   'portmaster -w -D -a'
```

```
alias port-clean 'portmaster --clean-distfiles -y'
alias port-notice '$PAGER /usr/ports/UPDATING'
alias port-dep "echo '^[[1m^[[32mBuild depends:^[[0m' && make build-depends-list|sort \
                echo '^[[1m^[[32mRun depends:^[[0m' && make run-depends-list|sort \
                echo '^[[1m^[[31mMissing:^[[0m' && make missing|sort"
```

基于上述 alias，更新 ports 的步骤就成了：

1. 用 `port-sync` 同步 ports。
2. 用 `pkg-fresh` 查看有哪些可用更新。
3. 用 `port-notice` 查看公告，如果提到了你安装了的应用，遵照它给出的更新步骤。
4. 用 `port-rebuild` 更新指定的几个应用，或者用 `port-update` 全部更新。

此外还可以这样用：

- 在安装某个 port 之前，先看看它依赖哪些东西：`cd /usr/ports/www/nginx && port-dep`。
- 安装某个 port，并清除编译时产生的临时文件：`cd /usr/ports/www/nginx && mininstall`。
- 清理不必要的 distfile：`port-clean`。

有时候，如果更新 ports 前没注意看 `/usr/ports/UPDATING`，把一些重要应用的 ABI 破坏了，例如在命令行执行程序时出现类似下面这种提示：

```
% fvwm
Shared object "libpng16.so.16" not found, required by "fvwm"
```

这时就需要手工修复。

修复的思路很简单，对每个已安装的应用，检查其名下所有文件，如果是 ELF 可执行文件，那么用 `grep` 检查其 `ldd` 输出，若有“not found”字样，那就重新编译这个应用。然而这种脚本写起来比较麻烦，建议安装 `bsdadminsceipts`，它提供的 `pkg_libchk` 命令可以很好地完成上述工作。而我们只要等它跑完，将给出的所有坏掉的应用交给 `port-rebuild` 就好了^[11]。

4.4. 自制二进制安装包

`pkg` 提供的应用定制选项一般很保守，例如 `nginx` 默认只开了这些：

```
Options :
    HTTP           : on
    HTTP_CACHE     : on
    HTTP_REWRITE   : on
    HTTP_SSL       : on
    HTTP_STATUS    : on
    IPV6           : on
    WWW            : on
```


虽然 ports 能够自由定制更多选项，但是如果你管理了 n 台服务器，希望给它们都装上带有 Lua 加持的 nginx，那么一台台都用 ports 定制安装也很麻烦。这个时候有两种解决方案，第一种是在一台机器上使用 ports 编译安装带 Lua 支持的 nginx，然后使用 `pkg create nginx` 创建二进制安装包，再把安装包分发到其余机器上，让它们通过 `pkg add` 安装。第二种方案是使用 poudriere 搭建自己的 pkg 源，具体可参考这篇官方文档。

5. 配置文件

5.1. rc.conf

rc.conf 掌管着所有系统服务。与之相关的文件和路径如下：

1. 默认的配置位于 `/etc/defaults/rc.conf`。最好不要修改它，但是建议阅读一下它，看看 FreeBSD 默认做了哪些设置，开机启动哪些服务。
2. 用户自定义的配置位于 `/etc/rc.conf`。例如，如果想让系统自动启动 ssh、ipfw、nginx 等服务，就要修改本文件。注意，如果某项配置与默认的配置有冲突，则以本文件为准。
3. 基系统的服务脚本位于 `/etc/rc.d/`。第三方应用的服务脚本位于 `/usr/local/etc/rc.d/`。当遇到问题时，建议阅读这些脚本，找出问题所在。

其次，`service` 命令可用于控制各种服务项。对于 `rc.conf` 中已启用的服务，我们可以这样操作：

- 让正在运行中的 nginx 重载配置文件：`service nginx reload`。
- 停止 nginx 服务：`service nginx stop`。
- 对 em0 接口重启 DHCP 服务：`service dhclient restart em0`。

当然也可以直接调用 `/etc/rc.d/` 和 `/usr/local/etc/rc.d/` 下的那些脚本，只不过要多打一些字：

- `/usr/local/etc/rc.d/nginx reload`
- `/usr/local/etc/rc.d/nginx stop`
- `/etc/rc.d/dhclient restart em0`

如果 rc.conf 中并没有启用某项服务，但我们想临时启动它，那么可以这样（以 tomcat7 为例）：

- `service tomcat7 onestart`
- `service tomcat7 onestop`

最后，建议笔记本用户启用 `powerd` 服务，因为它可以根据系统负载调节 CPU 主频，节电降温：

```
echo 'powerd_enable="YES"' >> /etc/rc.conf
service powerd start
```

5.2. periodic.conf

FreeBSD 默认有一些周期执行的任务，它们是通过 `periodic` 命令执行的，由 `cron` 自动调用。与 `periodic` 有关的配置和路径如下：

1. 默认的配置位于 `/etc/defaults/periodic.conf`。
2. 用户自定义的配置位于 `/etc/periodic.conf`。

3. 基系统的任务脚本位于 `/etc/periodic/`。
4. 第三方应用的任务脚本位于 `/usr/local/etc/periodic/`。

以 `locate` 命令的所依赖的路径数据库 `/var/db/locate.database` 为例，该数据库由 `/etc/periodic/weekly/310.locate` 这个脚本每周更新一次。如果你要立即更新，也可以直接执行这个脚本。

5.3. 其他配置文件

- `crontab`: `cron` 配置，位于 `/etc/crontab`，参考 `man crontab`。
- `syslog.conf`: 系统日志配置，位于 `/etc/syslog.conf`，参考 `man syslog.conf`。
- `loader.conf`: 系统启动配置，位于 `/boot/loader.conf`，参考 `man loader.conf`。
- `sysctl.conf`: 内核参数配置，位于 `/etc/sysctl.conf`，参考 `man sysctl.conf`。

6. 文件系统

6.1. ZFS 使用建议

- 最好不要在内存少于 1G 的机器上使用 ZFS。
- 最好不要在非 64 位系统上使用 ZFS。
- 最好不要把 swap 放到 ZFS 上。
- 要么彻底用 ZFS、要么彻底用 UFS，两者混搭会造成不必要的内存消耗。
- 为了提高机械硬盘随机读能力，可设置 `vfs.zfs.prefetch_disable=1`。
- 为了避免 ZFS 吃掉太多内存，最好要设置 `vfs.zfs.arc_max="?"`，例如：1024M。
- 如果要复制某个文件系统，可以用 `zfs send/recv`，这样还能通过 ssh 跨网络传输。
- 使用 `zfs-periodic` 或 `zfsnap` 自动对 `/home` 每小时做快照，保留最近 n 个小时的快照。
- 使用 SSD 硬盘可以改善 ZFS 随机读能力，并且 ZFS 这种写时复制的文件系统也有益于 SSD 寿命。

6.2. tmpfs 使用建议

tmpfs 是一种高效的内存文件系统。如果你的内存足够大，那么就可以把一些不是很重要的数据丢到 `tmpfs` 上，从而提高文件读写速度，降低硬盘功耗，延长 SSD 寿命。

为了启用 `tmpfs` 内核模块，使系统在开机时自动将其挂载到 `/tmp`，可执行以下命令并重启：

```
echo 'tmpfs_load="YES"' >> /etc/loader.conf
echo 'tmpfs /tmp tmpfs rw 0 0' >> /etc/fstab
```

从此 `/tmp` 里的东西就在内存上了。

如果你想在内存中编译 ports，那么修改 `tcsh` 环境变量即可：

```
echo 'setenv WRKDIRPREFIX "/tmp"' >> /etc/csh.cshrc
source /etc/csh.cshrc
```

如果你想把 `~/.cache` 也丢到内存上^[12]，那么可以在 `.xinitrc` 中加入以下配置：

```
CacheDir=/tmp/cache-`whoami`  
mkdir -p $CacheDir  
ln -sfh $CacheDir $HOME/.cache
```

7. 相关资源与应用

7.1. 一些网站

- 官方论坛: 发帖有格式要求, 活跃程度一般。
- 报告 bug: 基于 bugzilla, 比老的 send-pr 好用。
- 安全公告: 可以用 python 和 beautifulsoup 定期抓取解析。
- FreeBSDChina.org: 中文论坛, 只是主题看起来有点旧.....
- 水木社区 - 红色小魔鬼 FreeBSD: 好像不支持用 https:// 打开?
- FreeBSD/ARM on Raspberry Pi: 如果想折腾树莓派的话。

7.2. 邮件列表

通常在邮件列表中反映问题能够得到更快的回复, 以下是推荐订阅的邮件列表:

- freebsd-security: 安全公告什么的。
- freebsd-hackers: 技术讨论, 话题宽泛。
- freebsd-current: 有关 current 分支的话题可以发到这里。
- freebsd-stable: 有关 stable 分支的话题可以发到这里。
- freebsd-ports: 有关第三方应用的话题可以发到这里。

上述邮件列表可以在这里找到。

7.3. 新闻资讯

- BSD Talk: 音频播客, 有很多跟 BSD 核心开发者的访谈。
- BSD Now: 三个 BSD 爱好者创办视频播客, 用 BT 下载比较快。
- DragonFly BSD Digest: 会定期发一些 Reading List, 内容广泛。













7.4. 书籍

- 2014 The Design and Implementation of the FreeBSD Operating System (Second Edition)
- 2013 深入理解 FreeBSD 设备驱动程序开发
- 2012 FreeBSD Device Drivers
- 2007 Absolute FreeBSD
- 2006 FreeBSD 操作系统设计与实现 (影印版, 第一版)
- 2004 The Design and Implementation of the FreeBSD Operating System (First Edition)
- 2003 The Complete FreeBSD

7.5. 谁在使用 FreeBSD

- Apple: MacOS, iOS
- Sony: PlayStation 4

- WhatsApp 和 Netflix 的服务器
- 更多应用案例

-
- [1] 很多 Linux 新手并不知道 `ls`、`cp`、`chmod` 之类的命令其实是出自这个软件包。 
- [2] 与上一条情况类似，提供了 `arp`、`hostname`、`ifconfig` 等基本的网络工具。 
- [3] 当然也包含了一些「钦点的」第三方组件。 
- [4] 一般来说，基系统编译好之后直接安装就是了，没必要重启切换到单用户再安装。 
- [5] 新特性一旦启用就不可禁用，没有回头路。 
- [6] 某些新特性只对新写入的数据有效，无法作用于旧数据。当然你也可以把旧数据重写一遍。 
- [7] `gpart` 第一个字母“g”并不是指代 GPT。 
- [8] `portmaster` 是纯 shell 脚本，它的开发者同时也是 FreeBSD 开发者。 
- [9] `make deinstall` 也不例外，它最终还是要调用 `pkg delete`。 
- [10] `pkg-rmleaf` 是纯 shell 脚本，需要用 `ports` 或 `pkg` 安装。 
-]
- [11] 如果你会修 ABI，那更新公告里的 `portmaster -r` 就不必照做了，可以减少很多不必要的编译。 
-]
- [12] 若已有该文件夹，应当先退出 Xorg 并将其删去。 
-]