FreeBSD Porter 手册

目□

表格清□

□例清□

Chapter 1. 介□

几乎□个人都是通□ FreeBSD Ports Collection 在 FreeBSD 上面装□用程序 (“ports”)的。 就像FreeBSD的其它部分一□， 它主要来自于志□者的努力。 所以在□□□□文□的□候□□必□住□些。

在 FreeBSD 的世界里， 任何人都能提交新的 port， 或志□地□□一个已有的 port， 如果那个 port 没人□□的□ - 不需要任何特殊的□限来做□件事情。

Chapter 2. 自行制作新 port

那□， □有□趣□建自己的 port 或升□□有的 port？ 太好了。

下面的内容将会提供一些□建FreeBSD port的指□。 如果想升□一个□有的 port， 那□□□□在看完□些内容并□□升□一个 *port*。

因□□□文□不是十分□□， □□□□再参考一下 /usr/ports/Mk/bsd.port.mk， 所有 port 的 Makefile 文件都会包含它。 即使不是□天都去□弄 Makefile， □也会从那个文件里面□得很多知□， 里面的注□非常□□。 □有要□充一下，如果□有其它的□□， 可以□FreeBSD ports □件列表 □个 mailing list □信。

在□□文□里提到的大部分的□量 (VAR) 是不能修改的。 大多 (但不是全部) 都在 /usr/ports/Mk/bsd.port.mk 的□始部分□行了介□； 其它一些也□□可以在那里□到。 注意□些文件使用了非□准的制表符： Emacs 和 Vim □□能在打□文件的□候自□□□它， 而 vi(1) 和 ex(1) □需要在打□文件的□候通□ `:set tabstop=4` 来修正默□的□置。

想□□手□？ □参□我□的 希望移植的□件列表 来看看□是否有□趣完成其中的任□。

Chapter 3. □□的 port

□一章将介□如何快速□建一个全新的 port。 很多□候， □点内容是不□的， □需要□□□□文□中更深入的内容。

首先， 需要取得包含源代□的 tar包， 并把它放到 DISTDIR□量所指的地方。 默□的情况下， □□□是 /usr/ports/distfiles。

下面的内容假定□不需要修改□件的源代□就能在 FreeBSD 上□□通□。 如果需要修改代□，就需要参考下一章的内容了。

3.1. □写 Makefile

最□□的 Makefile □□是□个□子的：

```
# New ports collection makefile for:   oneko
# Date created:        5 December 1994
# Whom:                asami
#
# $FreeBSD$
#

PORTNAME=      oneko
PORTVERSION=   1.1b
CATEGORIES=    games
MASTER_SITES=  ftp://ftp.cs.columbia.edu/archives/X11R5/contrib/

MAINTAINER=    asami@FreeBSD.org
COMMENT=       A cat chasing a mouse all over the screen

MAN1=          oneko.1
MANCOMPRESSED= yes
USE_IMAKE=     yes

.include <bsd.port.mk>
```

看看□是否能□看□。 不必担心 $FreeBSD$ 那一行， 当□个 port 被□入到 ports □里的□候， CVS 会自□填写它。 □可以在 示□的 Makefile那章□到更多的□□。

3.2. □建描述文件

有 2 个描述文件□于任何一个 port 来□是必□的， 不□它是不是打算成□ package。 它□是 pkg-descr 和 pkg-plist。 □□个文件使用 pkg- 前□以区□于其它文件。

3.2.1. pkg-descr (□于 port 的冗□描述文件)

□是 port 里一个□□的描述文件。 使用一段或几段文件文字来□明的描述□个 ports 是用来做什□的。

□ 不是 手册或者□如何 深入使用/□□□个port的□明！ 要是□从 *README* 或者□机手册里面中□制文字的□， □□必小心； 通常， 它□不是□□个 port □明□要的描述，或者用了□以使用的格式 (比如， □机手册里有迫使□端□□的空格)。 如果要移植的□件有官方的WWW网□， □□□在□里列出来。 使用 `WWW:` 作□前□来表示 一个网站，□□其它的自□工具就能正常工作了。

下面是一个□□的 pkg-descr 例子：

```
This is a port of oneko, in which a cat chases a poor mouse all over
the screen.
 :
(etc.)

WWW: http://www.oneko.org/
```

3.2.2. pkg-plist (port 的装箱□)

□□文件列出了 port 所要安装的所有文件。 由于 package 也是据此□行打包， 因此它也被称作 "装箱□(packing list)". □个文件中， 路径是相□于安装的路径的 (通常是 /usr/local 或 /usr/X11R6)。 如果□使用 `MAN`□ □量的□， □不要在□里列出任何□机手册。 假如 port 在安装□程中会□建一些目□， □□必□加□□的 `@dirrm` 行， 以便在 package 被卸□□予以自□□除。

下面是一个□□的例子：

```
bin/oneko
lib/X11/app-defaults/Oneko
lib/X11/oneko/cat1.xpm
lib/X11/oneko/cat2.xpm
lib/X11/oneko/mouse.xpm
@dirrm lib/X11/oneko
```

参考 pkg_create(1) 的□机手册以□得更多有□装箱□的□□

建□□将□个文件里的所有的文件名按字母排序。 □□， 在升□□个port的□候就能□更方便地核□所做的修改。

手工□建□□一□列表可能是一件非常枯燥的事情。 如果□的 port 需要安装大量的文件，自□□建装箱□ 会□□省下不少□□。

只有一□情况可以不用 pkg-plist文件。 如果□个 port 只安装很少量的一些文件或目□的□， □些文件和目□就可以分□列在 Makefile 的 `PLIST_FILES`和`PLIST_DIRS` □量里。 □个例子来□， 我□可以在上面那个 oneko port 里面不用 pkg-plist， 而把下面的□几行加到 Makefile 里面：

```
PLIST_FILES=    bin/oneko \
                lib/X11/app-defaults/Oneko \
                lib/X11/oneko/cat1.xpm \
                lib/X11/oneko/cat2.xpm \
                lib/X11/oneko/mouse.xpm
PLIST_DIRS=     lib/X11/oneko
```

当然， 如果一个 port 不需要它自己建目的， 就不用置 PLIST_DIRS 量了。

不， 如果用方式来列出 port 要安装的文件和目的， 也就无法利用在 pkg_create(1) 里介的命令来制作 package 了。 因此， 方法只用于那些的 port， 使它更化。 同， 做法也有助于少 ports collection 中的文件数量。 在采用 pkg-plist 之前， 考一下使用方法。

后我将看到 pkg-plist 以及 PLIST_FILES 如何理 更的任。

3.3. 建校和文件

只要入 make makesum， port 便会自建 distinfo文件。

如果下的文件的校和常化， 而又能保它的来源可 (比如， 来自于CD制造商， 或天生成的文文件)， 就在 IGNOREFILES 里面明些文件。 ， 再行 make makesum 的候便不会把些 IGNORE 的文件算在内了。

3.4. port

当定的 port 做了希望它做的事情， 包括打包。下面是需要重点的一些重要的工作。

- pkg-plist 中没有包括任何不想安装的文件
- pkg-plist 包含了所有安装的文件
- 的 port 能使用 reinstall 多次安装。
- 的 port 能在卸 (deinstall) ， 自完成 清理

Procedure: 推的序

1. make install
2. make package
3. make deinstall
4. pkg_add package-name
5. make deinstall
6. make reinstall
7. make package

信在 package 和 deinstall 段没有任何警告。 第三以后， 是否所有新建的目都被正除了。 在第四

□以后， □着□行一下所装的□件， □保当它以 package 方式安装的□候也能正常工作。

自□化□些□□最□□的方法是通□ ports tinderbox 来□行□□。 它可以□□ jails 并在其中完成全部□□工作，而不会破坏正在□行的系□的状□。 □参□ ports/ports-mgmt/tinderbox 以了解更多的信息。

3.5. 用 portlint 来□□ port

□使用 portlint 命令来□□□的 port 是否符合我□的□□。 ports-mgmt/portlint 程序是 ports 套件的一部分。□个程序的主要功能是□助□□□ Makefile 的□式是否符合□□， 以及 package 的命名是否得体。

3.6. 提交新 port

在提交新 port 之前， □先□□ □做什□和不□做什□ 一□。

既然已□□所制作的 port 相当□意了， 剩下的工作， 便是将它放□ FreeBSD 的主 ports □， 以便□更多的人从中受益。 我□并不需要□的 work 目□以及 pkgname.tgz 包， 因此□在可以□除它□了。 假定□的 port 的名字是 oneko， 接下来要做的是 cd 到 oneko 所在的目□， 然后□入命令： shar find oneko > oneko.shar

将□个 oneko.shar 文件作□附件， 使用 send-pr(1) 程序提交 (□参□ Bug Reports and General Commentary 以了解□于 send-pr(1) 的□一□□情) 将其送出。 □□必将□的 bug □告分□ (category) □ ports 并把子分□ (class) □置□ change-request (不要把□告表及□机密的， 即 confidential ！)。 此外， 在 PR 的描述 ("Description") 一□中的内容□□是 port 的□要介□ (例如 COMMENT 内容的□化版本)， 而 shar 文件□□填入修正 ("Fix") □中。

在□□□告里面使用了一段好的描述， 能使我□的工作□得更容易。 □□上， 我□会使用□似： "New port: <category>/<portname> <short description of the port>" □□的□□来□明□是新的 port。 如果□也使用□□的□□， 那□我□将更容易更方便地□□□的 PR， 从而加快□理速度。

再次声明， 不要包含原始的*distfile*， *work*目□， 或者□用 make package 制作的包； 此外， □于新的 port □□必使用 shar(1) 而不是 diff(1)。

在□提交的□的 port 以后□耐心等待。 有□在一个 port 正式加入 FreeBSD 之前需要花□好几个月，尽管也有可能是几天。 □可以□看 正等待被 commit 到 FreeBSD 的 port PR。

一旦我□看□了□的□告， 有必要的□我□会□系□， 并把它放到 ports □里。 □的名字也会出□在 Additional FreeBSD Contributors 和其它的文件。 不是很棒□!? :-)

Chapter 4. □□的 Porting

好了， 也□工作没那□□□， port 需要做些修改才能□在 FreeBSD 上□起来。 在□一章里， 我□将会一□□□例来介□□□如何修改来使□的 port 能在 FreeBSD 上面□行。

4.1. 整个系□是如何□□的?

首先， □一系列的□作是由用□在□的 port 目□里敲入 make 后□生的。 □也□会□□在□外的一个□口里□□一下 bsd.port.mk 将会有助于□的理解。

要是□不是非常明白 bsd.port.mk 是做什□的□， 也不用太担心， 很多人都不知道的... :→

1. fetch 会首先被□行。 fetch 将□□在本地的 DISTDIR 目□里是否存在 tar 包。 如果 fetch 没有□到就会□□ Makefile 中定□的 MASTER_SITES URL， □有我□的主 FTP 站点 ftp://ftp.FreeBSD.org/pub/FreeBSD/ports/distfiles/， 在那里我□□□了所有被□可的 distfile。 假□那个 MASTER_SITES 站点是直接□在 Internet 上的， 就会□着用 FETCH 指定的程序取回 distfile。 如果成功的□， 文件会被保存在DISTDIR 所指定的目□以□□后使用。
2. 接下来会□行 extract。 它会在 DISTDIR 中□□□的 tar 包 (通常是用 gzip □□的 tar 包)，然后解□□到由 WRKDIR 所指定的□□目□里 (默□□work目□)。
3. 下一□是□行 patch。 首先任何在 PATCHFILES 中定□的□丁都会被打上。 然后， 在由 PATCHDIR 指定的目□ (默□□ files目□) 中□□的patch-*， 它□将会以文件名的字母□序被先后打上。
4. configure会被□行。 □一□□可能会有以下几□情形。
   a. 如果存在 scripts/configure， 就会□行它
   b. 如果定□了 HAS_CONFIGURE 或者 GNU_CONFIGURE， 就会□行 WRKSRC/configure。
   c. 如果定□了USE_IMAGE， 就会□行 XMKMF (默□□: xmkmf -a)。
5. build会被□行。 □一□将会□入ports的工作目□ (WRKSRC) 然后□行□□。如果定□了USE_GMAKE，就会使用 GNU make， 反之， □会使用系□默□的 make。

以上都是系□默□的□□。 □也可以定□ pre-something 或者 post-something， 或者把以此命名的脚本放到 scripts 目□， 它□会在默□的□作之前或之后被□行。

□个例子， 如果□在□的 Makefile 里定□了post-extract， 并在 script 目□里放了一个 pre-build 脚本， 那□在 tar 包解□之后 post-extract 将被□用， pre-build 脚本会在默□的□□之前被□行。 我□推□□在 Makefile 定□所有的□作， 如果不是十分□□的□， □□， □人能更容易明白□的 port 需要□行□些非默□的□作。

默□的行□都是由 bsd.port.mk 定□的 do-something 来表示的。 例如， port 中用来解□□的命令是由 do-extract 来定□的。 如果□□默□的□置不□意， 可以通□在 Makefile 重新定□ do-someting 来做些改□。

"主" □作 (例如 extract、 configure， 等等) □□是用来□定所有相□的□段都完成了， 以及□用真□的□作或脚本， 它□不□被修改。 如果□想要修改解□□□个□作， 可以修改 do-extract， 但永□都不要改□ extract 的操作！

我□已□介□了在用□敲入 make 之后会□生□些事情了。 接下来我□将□行□一□的学□， 来看一看如何□建一个理想的 port。

4.2. □取源代□

□取源代□的 tar 包 (通常是 foo.tar.gz 或者 foo.tar.Z) 并把它□放□ DISTDIR。 最好使用 主流 的版本。

□需要□置□量 MASTER_SITES 来指向原始 tar 包的□取位置。 □可以在 bsd.sites.mk 里□到一些速度□快的主流站点。 □使用□些站点 - 和相□的定□ - 如果可能的□， □尽量避免在同一个源代□□里出□大量重□的信息。 □些站点会随着□□而□化， 如果□个人都随意加入的□会使□□□得非常困□。

如果□□不到一个有很好网□□接的 FTP/HTTP 站点， 或者它□使用了非□准的格式， □也□就会想在□自己的 FTP 或 HTTP 服□器上放上一□副本。

如果□□不到可□的地方放置 distfiles， 我□也可以提供□□一些空□来保存它。 我□自己的 ftp.FreeBSD.org； 然而□只是一个折衷的□法。 distfile 必□放□某人在 freefall 上的 ~/public_distfiles/ 目□中。 可以要求□助□ commit port 的人来放□个 distfile， 而□个人也需要把 MASTER_SITES、 MASTER_SITE_LOCAL 以及 MASTER_SITE_SUBDIR 的□置， 改□在 freefall 上的用□名。

如果□的 port 的 distfile 一直在□化， 而作者拒□改□其版本号， □可以考□把 distfiles 放在自己的主□， 并在 MASTER_SITES 里把原作者的列□首□位置。 如果可能， □着与 port 的作者□通一下□他不要□□做， □将有助于建立□源代□的控制。 在□的主□上放置□自己的 distfile 会避免用□得到 checksum mismatch 的□□， 而且能□□我□ FTP 站点□□人□的工作量。 如果□的port只有一个主站点的□， 我□建□□在自己的网站上做一□□□， 并他列□ MASTER_SITES的第2□。

如果□的 port 需要来自网□上的一些□丁， □把它□放到 DISTDIR里。 不用担心它□跟源代□不是来自同一站点。 我□有□法□理 (参□下面的 □丁文件)。

4.3. 修改 port

解□ tar 包， □源代□做出合理的修改使得□个 port 能在最新版本的 FreeBSD 上面□行。 一定要 仔□□□所做的□□改□， 包括□除、添加、修改的文件等等， □些修改以后会在□的 port 中以脚本或□丁的方式出□， 并且能通□□行它□来自□完成□□ port 的改□要求。

如果□的 port 要求用与用□交互/配置来完成□□或安装的□， □可以看一下 Larry Wall 的□典的 Configure 脚本， □当地模□一下。 Port collection 的目的， 就是使□个 port 占用最少的空□， 并做到□件的 "即□即用"。

除非明□地声明， 否□□提交□ FreeBSD ports collection 的□丁， 脚本和其它的文件都将以□准的 BSD 版□□布。

4.4. 打□丁

在□准□制作 port 的□程中， □加或修改的文件， 都可以通□ diff(1) 来做成□丁。 希望□用到源代□上的□个□丁， 都□保存□□独的文件， 并命名□ patch-*， 其中 * 表示将要修改的文件的完整路径名， 例如 patch-Imakefile 或 patch-src-config.h。 □些文件， 都□保存在 PATCHDIR (通常是 files/)， □里的□丁都会自□□用到源代□上。 所有的□丁必□是相□于 WRKSRC 的 (一般而言， □的 port 会将其 tarball 解□□在那里， 并完成余下的工作)。 □了□修正和升□更容易， □□避免使用多个 patch 去修改同一个文件 (例如， patch-file 以及 patch-file2 都修改 WRKSRC/foobar.c) □□情况。 需要注意的是， 如果修改的文件的路径中包含下□□ (_) 字符， □在□丁文件名中□使用□个下□□来代替。 例如， 如果需要修改名□ src/freeglut_joystick.c 的文件， □丁文件的名字□□ patch-src-freeglut__joystick.c。

只有 [-+._a-zA-Z0-9] □些字符， 可以出□在□丁的文件名中， □□必不要使用除□些字符以外的其它字符。 不要把□的□丁命名成 patch-aa 或 patch-ab 等□□的名字， 最好能在□丁名中提到路径和文件名。

不要把 RCS 字符串放□□丁。 我□把文件放□ ports □的□候， CVS 会□坏它□， 当我□再 check out 出来的□候， 它□就会和原来的不一□， 从而□致打□丁失□。 RCS 字符串 是由美元符号 ($) □□的， 通常由 $Id 或 $RCS □□。

使用 diff(1) 的□□□□(-r) 很好， 但是□□□一下最后□出的 patch， □保没有任何的□□信息。 特□地， 有 2 □文件不需要 diff， 并且□□□除： 一□是 Makefile， 当□的port使用了Imake， 或者 GNU configure 等等的 □。 如果□不得不□□configure.in 以使 autoconf 去生成 configure， 不要使用 configure 来做 diff (□常常会有好几千行□!)； □定□ USE_AUTOTOOLS=autoconf:261 并□□ configure.in 来制作 diff。

□外， □□□尽量□少□丁中非功能性的空格及空白□□。 在□源世界中， 遵循不同的□□□□的□目共享大量代□ 是很常□的事情。 如果□从某个□目中提取一部分功能用来修正□一个程序中的□□□， □□必小心： □丁中很可能到□都是非功能性的□□行。 □不□会□致 CVS □的膨□， 而且也会□□致□□的故障点， 以及 □到底修改了什□□得不甚清晰。

假如需要□除文件， □□在 post-extract target， 而不是作□□丁的一部分来完成。

除此之外， port 的 Makefile □可以通□ in-place 模式的 sed(1) 来直接□行□□的替□操作。 如果□丁需要使用 □量□， □就非常有用了。 例如：

```
post-patch:
    @${REINPLACE_CMD} -e 's|for Linux|for FreeBSD|g' ${WRKSRC}/README
    @${REINPLACE_CMD} -e 's|-pthread|${PTHREAD_LIBS}|' ${WRKSRC}/configure
```

往往在移植某些□件的□候会遇到□□一□情况， 特□是□个□件是在 Windows® 上□□的□候， 大多数的源代 □都需要□行CR/LF的□□。 □很可能会□以后打□丁□来□□， □可能触□□□警告， 并□脚本的□行□来麻□ (/bin/sh^M not found)， 等等。 要迅速将所有文件中的 CR/LF 改□只用 LF， 可以在 port 的 Makefile 中加入 USE_DOS2UNIX=yes。 除此之外， □可以指定一个需要□行□□□□操作的文件列表：

```
USE_DOS2UNIX=    util.c util.h
```

如果希望□□一系列目□中的一□文件， 也可以使用 DOS2UNIX_REGEX。 它的参数是与 find 兼容的正□表□式。 □于□□格式的□明， □参□ re_format(7)。 □个□□□□□所有指定□展名的文件， 例如只□□源代□文件□□的 □用非常有用：

```
USE_DOS2UNIX=    yes
DOS2UNIX_REGEX=  .*\.(c|cpp|h)
```

如果□希望基于□存的文件□建□丁， 可以把文件□制□□ .orig □展名的名字， 然后修改原文件。 然后使用 makepatch 目□根据修改在 port 的 files 目□中生成□丁文件。

4.5. 配置

把任何附加的配置命令加□□的 configure 脚本并把它保存到 scripts 子目□。 如前面提到的那□， □也能在 Makefile 和/或 使用 pre-configure 或 post-configure 的脚本来做同□的事情。

4.6. □理用□□入

如果□的 port 要求用□的□入以便配置□□、 或安装配置□程， 就必□在 Makefile 里□置 IS_INTERACTIVE □量。 如果用□□置了 BATCH 的□， □将□用□能跳□□的 port 来完成 "通宵□□" (如果用□□置了 INTERACTIVE的□， 那□只有 那些要求互□的 port 才会被□□) □将□那些不停□□ ports 的机器省下很多□□。

通常我□□建□， 如果□于那些□□能有合理的缺省答案的□， □□□一下 PACKAGE_BUILDING □量， 并根据其□置决定是否□行□□交互脚本。 □将允□我□□ CDROM 和 FTP 来□□ package。

Chapter 5. 配置 Makefile

配置 Makefile 是相当□□的， 我□在此建□□在□始之前看一下□有的例子。 在□□手册里也有一个 Makefile 例子， 照着里面□量的□序来写能使得□的 port 更容易地被其它人看□。

□在， 当□□始□写□新的Makefile 的□候， 可以依次思考一下以下的□□：

5.1. 作者□布的代□

放在 DISTDIR 中的是不是□准的用 gzip □□的 tar 包， 例如 foozolix-1.2.tar.gz？ 如果是， 可以先略□□一□。 如果不是， □□当看看是不是要覆□□些□量： DISTVERSION、 DISTNAME、 EXTRACT_CMD、 EXTRACT_BEFORE_ARGS、 EXTRACT_AFTER_ARGS、 EXTRACT_SUFX， DISTFILES，取决于□ port 的 distfile 格式有多□怪□。 (最常□的一个例子便是 EXTRACT_SUFX=.tar.Z， 一般□是因□ tar 包是用 compress 而不是 gzip □□的□候。)

最糟的情况是， □需要自己□写 do-extract 来覆□默□的定□， 尽管□不常□， 但如果遇到了， □是需要□□做。

5.2. 命名

Makefile 的第一部分便是 port 的名字、 版本号， 以及它所属的分□。

5.2.1. PORTNAME 和 PORTVERSION

□□□把 PORTNAME □置□□ port 的名字， PORTVERSION □是 port 的版本号。

5.2.2. PORTREVISION 和 PORTEPOCH

5.2.2.1. PORTREVISION (port 的修□版本号)

PORTEREVISION □量是一个□□□□的□， 如果不□ 0， 就会被加到包名的后面， 当 PORTVERSION □加 的□候□被置 0 (也就是当官方有新版本□布的□候)。 PORTREVISION 会被自□化工具 (比如 pkg_version(1)) 用来□□是否存在可用的新版本。

□当 port □生□化并□生成的 package 的内容或□□有□著影□□， 都□□加 PORTREVISION □。

下面是一些□当修改 PORTREVISION 的情况：

- 有新的□丁用来修正安全漏洞、 □□， 或□ port 添加了新的功能。
- 修改了 Makefile 里□□□□□或禁用的□□。
- 修改了要安装文件的列表或安装□的行□ (例如， 修改了一个用来□ package 初始化数据的脚本， 如 ssh host keys)。
- 一个port依□的共享□版本改□ (在□□情况下， 当安装了新版本的共享□， 后再去安装□早的□件就会出□，因□它□要依□老的 libfoo.x 而不是libfoo.(x+1))。
- 原作者修改了 port distfile， 并且 distfile 的新老版本之□用 diff -ru 只能□□一些□微的□化， □□我□只需要□ distinfo 做相□的修正， 而不需要修改 PORTVERSION。

不需要修改 PORTREVISION 的例子：

- port □□□格的改□， 但□于打成的包没有功能的上的□化。
- MASTER_SITES □生□化， 或□行了□ port 功能的修改， 但不致影□最后打成的包。
- □ distfiles □如修正□写□□之□的□丁， □用□而言没有升□上的麻□。
- □一个原本□□失□的包的修改， 使其可□□， 而没有加入新功能。 因□ PORTREVISION 表示包的内容□生了□化， 如果先前没有可□□的包， 也就不需要修改 PORTREVISION 来表示□化。

一个修改并提交 port 的原□是： 使得□人能从中受益 (改□、 修改已有□□， 或使新的 package 能□□行)， □□要□衡一下□是否□□那些□常更新 ports □的人升□， 如果回答是 "是" 的□， PORTREVISION 就□□修改了。

5.2.2.2. PORTEPOCH (port 的加□版本号)

有□□件商或 FreeBSD 的 porter 会使用比旧版的版本号小的数字做□新版本号的情况。 □例来□， 从 foo-20000801 到 foo-1.0 (从形式上来□□是不□的， 因□ 20000801 在数□上比1大很多)。

在□□情况下， PORTEPOCH □当□加。 如果 PORTEPOCH 非 0， 就□当加到包名字的后面。 PORTEPOCH 永□不能被□少或清零， 因□那□会□致与前一□期的 package 比□版本□□生不正□的□果。 (就是□， 那个 package 就不会被□□到已□□□了。) 新的版本号 (比如前面在前面那个例子中的 1.0,1) 在数□上比前一个版本 (20000801) 小， 但多数自□化的工具会□□ ,1 后□意味着比前一个包的后□ ,0 大。

□□的去除或重置 PORTEPOCH 会□致很多不幸□生； 如果□□不明白前面的□□， □多□□几次直至明白□止， 或到□件列表上来提□。

大多数 port 都不会用到 PORTEPOCH， 并且如果某个□件的下一个版本改□了版本号□□的□， 用巧妙的方法来□定 PORTVERSION 也能避免使用 PORTEPOCH。 然而， FreeBSD porter 也需要注意， 当有新版本的□件□布， 但并非正式版本□ - 比如 "snapshot" 版本， 原作者可能会使用当□的日期来命名， □在新的 "官方" 版本□布的□候， 就很容易引起前面提到的□□。

□个例子， 如果 snapshot 版本的□布日期是 20000917， □个□件的上一个版本是1.2， 那□□个版本的 PORTVERSIN □□□□ 1.2.20000917 或□似的□子， 而不是20000917， □□在 1.3 □布以后， 新版本就可以在数□上大于旧的版本了。

5.2.2.3. □于 PORTREVISION 和 PORTEPOCH 的用例

gtkmumble port，版本号 0.10， 被提交到 ports collection:

```
PORTNAME=       gtkmumble
PORTVERSION=    0.10
```

PKGNAME □成 gtkmumble-0.10。

然后有人□□了一个安全漏洞， 需要用一个FreeBSD的□丁。 PORTREVISION 就要相□的□加。

```
PORTNAME=       gtkmumble
PORTVERSION=    0.10
PORTREVISION=   1
```

PKGNAME□成了 gtkmumble-0.10_1

□件的作者□布了新的版本， 版本□ 0.2 (作者本来的意思是， 用 0.10 表示 0.1.0，"而不是指 0.9 之后的那个版本" - 但是□在太□了)。 因□□在的次版本号 2 在数□上比上一个版本 10 小， PORTEPOCH 必□□加，以使新的 package 被□□是 "更新的"。 由于那是作者□布的一个新版本， 因此 PORTREVISION □被置0 (或者从 Makefile 里面□除它)。

```
PORTNAME=       gtkmumble
PORTVERSION=    0.2
PORTEPOCH=      1
```

PKGNAME □成了 gtkmumble-0.2,1

下一个版本将会是 0.3。 由于 PORTEPOCH 从不□少， 那□就无□改□：

```
PORTNAME=       gtkmumble
PORTVERSION=    0.3
PORTEPOCH=      1
```

PKGNAME □成 gtkmumble-0.3,1

如果在□次升□中 PORTEPOCH 被置□了0， 那□在装了 gtkmumble-0.10_1 包的机器上就无法□□到 gtkmumble-0.3 包的更新， 因□ 3 在数□上比 10 小。 □住， □是 PORTEPOCH 最重要的地方。

5.2.3. PKGNAMEPREFIX 和 PKGNAMESUFFIX

2 个可□的□量， PKGNAMEPREFIX 和 PKGNAMESUFFIX 可以和 PORTNAME □有 PORTVERSION 配合使用， 形成像□□的 PKGNAME： ${PKGNAMEPREFIX}${PORTNAME}${PKGNAMESUFFIX}-${PORTVERSION}。 □□定符合我□的 包命名□□。 当然， 不 允□在 PORTVERSION 中使用□字符 (-)。 如果包名有 *language-* 或 *-compiled.specifics* 部分 (□下文)， □分□用 PKGNAMEPREFIX 和 PKGNAMESUFFIX， 不要直接加到 PORTNAME 中。

5.2.4. LATEST_LINK

LATEST_LINK 在□□包的□程中用于□定可以□ pkg_add -r 使用的□短的名字。 □例来□， 在安装最新版本的 perl 的□候， 只需指定 pkg_add -r perl 而无需知道具体的版本号。 □个名字□□是独一无二的， 并且□用□而言□□是□而易□的名字。

有□， 在 ports 套件中可能会存在同一程序的多个版本。 索引和□□□包的□□系□都需要能□将它□□□不同的□件包， 尽管其 PORTNAME、 PKGNAMEPREFIX， 甚至 PKGNAMESUFFIX 可能是一模一□的。 遇到□□情况□，就需要将除了 "主" port 之外的其他 port 中的 LATEST_LINK □量□□不同的□ - □参□ lang/gcc46 和 lang/gcc port， 以及 www/apache* 系列， 以了解它的用法。 如果□置了 NO_LATEST_LINK， □系□便不会生成□□的□接， □于非 "主" port 来□是一个可行的□□。 需要注意的是， 如何□定 "主" 版本 - "最流行"、 "受支持最好"，"□□最少"， 等等 - 已□超□了本□能□□出的建□□□； □里只是向□介□在□定了一个 "主" port 之后如何指定其他 port 的版本。

5.2.5. 包命名□□

以下是□在命名□的包□□当遵守的□□。 □将使得我□放包的目□更利于□□， 因□我□已□有数以万□的包了，

如果用□□得□看包名很困□的□， 他□会很快走□的。

一个包的名字□□看起来像□□： language_region-name-compiled.specifics-version.numbers。

要像□□来定□包的名字： ${PKGNAMEPREFIX}${PORTNAME}${PKGNAMESUFFIX}-${PORTVERSION}。 □保所有的□量符合上面的格式。

1. FreeBSD 会尽力去支持用□当地的□言。 如果□个 port 是某□□言□用的， 那□ *language-* 部分□□是 由 ISO-639 定□的自然□言的 2 个字母□写。 比如， ja是表示日本， ru 是表示俄□斯， vi 表示越南， zh 表示中国， ko 表示□国， de 表示德国。

   如果是□□某□□言的某一地区的□， 再要加上2个字母的国家代□。 例如， en_US 表示美国英□， fr_CH 表示瑞士法□。

   language- 部分□□在 PKGNAMEPREFIX □量里□置。

2. name 部分的首字母□□ 小写。 (余下的部分可以包含大写字母， 所以当□ 要□□一个包含大写字母□件的名字□， □需要 自己做出判断。) □于 Perl 5 模□的命名， 有个□□的□□是， 在前面 加上 p5- 并把□个冒号的部分改□□字号， 如： Data::Dumper 模□□□的名字， 就□□是 p5-Data-Dumper。

3. □□ port 的名字和版本之□有清晰的分隔， 并放入 PORTNAME 和 PORTVERSION □量。 在 PORTNAME 中包含版本部分的唯一理由是上游□件包真的采用□□的命名方式， □似 textproc/libxml2 或 japanese/kinput2-freewnn port □□。 否□， 在 PORTNAME 中就不□包含任何版本信息。 □多 port 采用同□的 PORTNAME 名字是很正常的， www/apache* port 便是如此； 在□□情况下， 不同的版本 (以及不同的索引□) 是由 PKGNAMEPREFIX、 PKGNAMESUFFIX， 以及 LATEST_LINK 的□的不同而有所区□的。

4. 如果 port 可以使用不同的 硬□□默□配置 □行□□ (通常是一系列 port 的一部分目□名)， □ *-compiled.specifics* 部分就□□明示□□□去的默□□ (此□□字号是可□的)。 通常的用例包括□型和不同的字体尺寸。

   -compiled.specifics 部分□□通□ PKGNAMESUFFIX □量来□置。

5. 版本号□□□随在□字号 (-) 后面并由数字和字母□成。 特□指出， □外的□字号是不允□出□在版本号里的。 唯一例外的是字符串 pl (表示 "patchlevel")， 只能 用在□件没有主版本号和次版本号的情况下。 如果□件的版本号里出□了像 "alpha"， "beta"， "rc"， "pre"， 取第一个字母把它放在小数点的后面。 如果在版本号里一直出□那些名字， 那□在数字和字母之□不□有多余的小数点。

   □个方法是□了更容易得凭版本号来排序 port。 特□注意的是， □保版本号之□的□部分都由小数点来分隔， 如果日期也是版本号的一部分， 就用□□的格式， 0.0.yyyy.mm.dd □□的格式， 而非 dd.mm.yyyy 甚至 yy.mm.dd □□不□合表示千年的格式。 在版本号上使用 0.0. 前□十分重要， 因□当□件□行正式的版本□， 其版本号数字很可能会小于表示年□的 yyyy 数字。

□里是一些真□的例子， 我□藉此□明如何把□件作者□□件的命名， □□□□合我□包的命名方式：

□行版的名字	**PKGNAMEPREFIX**	**PORTNAME**	**PKGNAMESUFFIX**	**PORTVERSION**	□明
mule-2.2.2	(空)	mule	(空)	2.2.2	没什□需要修改的
EmiClock-1.0.2	(空)	emiclock	(空)	1.0.2	程序的名字不能使用大写字母

□行版的名字	PKGNAMEPREFIX	PORTNAME	PKGNAMESUFFIX	PORTVERSION	□明
rdist-1.3alpha	(空)	rdist	(空)	1.3.a	像 alpha □□的字符串是不允□出□的
es-0.9-beta1	(空)	es	(空)	0.9.b1	像 beta □□的字符串是不允□出□的
mailman-2.0rc3	(空)	mailman	(空)	2.0.r3	像 rc □□的字符串是不允□出□的
v3.3beta021.src	(空)	tiff	(空)	3.3	那个是□鬼□西？
tvtwm	(空)	tvtwm	(空)	pl11	□需要有个版本号□
piewm	(空)	piewm	(空)	1.0	□需要有个版本号□
xvgr-2.10pl1	(空)	xvgr	(空)	2.10.1	pl 只允□在没有主/次 版本号的情况下才能出□
gawk-2.15.6	ja-	gawk	(空)	2.15.6	日文版
psutils-1.13	(空)	psutils	-letter	1.13	□□大小已□在□□的□候被硬□□到程序里了
pkfonts	(空)	pkfonts	300	1.0	300dpi 字体的包

如果在原始的代□里没有版本号， 或者原作者并不打算□□□外的版本， 就□把版本号□成 1.0 (就像前面 piewm 的例子那□)。 否□， 要求原始的作者加上版本号或使用日期 (0.0.yyyy.mm.dd) 来作□版本号。

5.3. 分□

5.3.1. CATEGORIES (所属分□)

在包制作完成之后， 它会被放在 /usr/ports/packages/All， 并建立一系列来自 /usr/ports/packages 下子目□的符号□接。 □些子目□的名称是由 CATEGORIES 指定的。 □将方便于那些用□在 FTP 站点或 CDROM 的一大堆包里面□□自己想要的包。 □□看一下 目前的分□表， 并□出一个□合□ port 的分□。

此列表也会决定□的 port 在 port 目□中的位置。 如果□在□里□定了 1 个以上的分□， □□□□ port 文件□放到以第一个分□命名的子目□中。 □参□ 后面 □于如何□□正□分□的更多□□。

5.3.2. 目前的分□表

□是目前 port 中的分□。 那些用星号 (*) □□的是 虚□分□ - 它□在ports□里没有相□的子目□， 因而只用来做

□次要的分□， 用以方便搜索。

□于非虚□的分□来□， □会看到在相□□子目□中的 Makefile 里有写在 COMMENT 里的□行描述。

分□	描述	注意事□
accessibility	□助残障人士的 port。	
afterstep*	□于 AfterStep □口管理器的支持。	
arabic	阿拉伯□言支持。	
archivers	□□与□□工具。	
astro	有□天文学的 port。	
audio	声音支持。	
benchmarks	□□程序。	
biology	生物学相□的□件。	
cad	□算机□助□□工具。	
chinese	中文□言支持。	
comms	通□□件。	大部分是用于串口通□的。
converters	字符□□□□。	
databases	数据□。	
deskutils	在□明□算机以前就已□在□面上使用的□西。	
devel	程序□□工具。	不要把□□□放在□里 - 除非□再也□不到更合□的分□， 否□就不□放在□个分□里。
dns	DNS 相□的□件。	
docs*	有□ FreeBSD 文□的 Meta-ports。	
editors	通用□□器。	有特殊用途的□□器□□被置于相□的分□中 (比如， 数学-方程式 □□器□□放在 math 分□里。
elisp*	Emacs-lisp相□的port。	
emulators	其它操作系□的模□器。	□端模□器 不□□ 属于□个分□ - 基于 X 的□□放在 x11 而基于文本模式的□□放到 comms 或 misc 中去， 取决于具体的功能。
finance	□□、 金融以及相□的□用程序。	
french	法□□言支持。	
ftp	FTP 客□端和服□器端的程序。	如果□的 port 同□支持 FTP 和 HTTP 的□， 把它放□ ftp 并把 www 做□第二分□。

分□	描述	注意事□
games	游□。	
geography*	与地理学有□的□件。	
german	德□□言支持。	
gnome*	□于 GNOME □目的支持。	
gnustep*	与 GNUstep □面□境有□的□件。	
graphics	□形□象程序。	
hamradio*	□余无□□□好者使用的□件。	
haskell*	有□ Haskell □程□言的□件。	
hebrew	希伯来□□言支持。	
hungarian	匈牙利□□言支持。	
ipv6*	IPv6 相□□件。	
irc	IRC 相□程序	
japanese	日□□言支持。	
java	与 Java™ □程□言有□的□件。	java 分□□ port 而言不□是其唯一的分□。 除了直接与 Java □言相□的 port 之外， □□人□□尽量避免使用 java 作□ port 的主分□。
kde*	K □面□境 (KDE) 相□的□件。	
kld*	可加□内核模□。	
korean	□□□言支持。	
lang	□程□言。	
linux*	Linux 相□的□用程序。	
lisp*	和 Lisp □程□言有□的□件。	
mail	□子□件□件。	
math	数□□算和其它数学相□的□件。	
mbone*	MBone □用程序。	
misc	各式各□的□用程序。	通常不属于其它的任何分□，如果可能的□， 尽量□□的 port □□ `misc` 以外的分□， 因□在□里的 port 比□容易被人忽略。
multimedia	多媒体□件。	
net	各□网□相□的□件。	
net-im	即□消息□件。	
net-mgmt	网□管理□件。	

分□	描述	注意事□
net-p2p	□等网 (Peer to peer network) □用程序。	
news	USENET新□□相□□件。	
palm	Palm™ 系列相□□件。	
parallel*	并行□算相□□件。	
pear*	Pear PHP 架□相□□件。	
perl5*	Perl5 相□的□件。	
plan9*	Plan9 相□程序。	
polish	波□□□言□言支持。	
ports-mgmt	用于管理、 安装和□□ FreeBSD ports 和□□□包的 port。	
portuguese	葡萄牙□□言支持。	
print	打印相□的□件。	□面出版工具 (打印□□工具等等) 也可以放在此分□里。
python*	Python □程□言相□的□件。	
ruby*	Ruby □程□言相□的□件。	
rubygems*	移植版本的 RubyGems □件包。	
russian	俄□□言支持。	
scheme*	与 Scheme □言有□的 port。	
science	科学相□但不□合放在 astro、biology， 以及 math 分□的 port。	
security	安全相□的□用程序。	
shells	命令行 shell。	
spanish*	西班牙□支持	
sysutils	系□相□的□用程序。	
tcl*	依□于 Tcl □行的 port。	
textproc	文本□理的□用程序。	□个分□并不□合于那些□□放到 print 的□面出版工具。
tk*	依□于 Tk □行的 port。	
ukrainian	□克□□□言支持。	
vietnamese	越南□□言支持。	
windowmaker*	WindowMaker □口管理器的相□支持。	

分□	描述	注意事□
www	Word Wide Web的相□□件。	HTML□言相□的支持也可以放在□个分□里。
x11	X Window System以及相□□件。	□个分□是□那些直接支持X Window System 的□件的。不要把常□的 X □用程序也放□□里；它□中的大多数都□被□□到 x11-* (参□下文)。 如果□的 port 是 X □用程序， □定□ USE_XLIB (使用 USER_IMAKE □含包括它)，然后把它放到合□的分□里。
x11-clocks	X11 下的□□程序。	
x11-drivers	X11 □□程序。	
x11-fm	X11 下的文件管理器。	
x11-fonts	X11 下的字体以及相□工具。	
x11-servers	X11 服□器。	
x11-themes	X11 主□。	
x11-toolkits	X11 工具包。	
x11-wm	X11 □口管理器。	
xfce*	与 Xfce □面□境有□的 port。	
zope*	Zope 相□的支持。	

5.3.3. □□正□的分□

由于不少分□是重□的， □通常在用□个分□作□□ port 的主分□上做出□□。 下面有几条□□能□□解决□个□□。□是一个□□先□的表， 按□先□降序□列：

- 第一个分□必□是个物理的分□ (参□ 前面)。 □□于制作包是必要的。 虚□分□和物理分□可能在包制作完成后混合在一起。
- □于特定□言的分□通常放在第一位。 例如， 如果□的 port 会安装一些 X11 的日文字体， 那□ CATEGORIES那行 就□□是 japanese x11-fonts。
- 有特定意□的分□□当被列在无特定意□的前面。 例如， HTML □□器□□是□□的 www editors，而不是其它的什□。 同□地， □不□□列出 net， 如果 port 属于 irc、 mail、 news、 security， 或是 www， 因□ net 可以表示它□的超集。
- 只有当主要的分□是一□自然□言的□候， x11 能被做□第二分□。 需要特□指出的是， □不□把 X 的□用程序也□□□ x11。
- Emacs 模式□当于相□的□用程序放在同一个分□里， 而不是 editors 分□。 □例来□， 一个用于□□某□□程□言源代□的 Emacs 模式□□被□□ lang 一□。
- 需要安装可加□内核模□的 port □在其 CATEGORIES 中□入虚□分□ kld。
- misc 分□的 port 不能有其它非虚□的分□。 如果□在□的 CATEGORIES 里□了 misc 和□外的分□，那意味着可以安全地□除 misc 并把 port 放到其它的子目□中了！

- 如果的 port 不属于有的分，才把它放到 misc。

如果不能定使用个分，在提交的 send-pr(1) 里加上一行注，我就能在入 port 之前一下。如果是 committer，一忘到 FreeBSD ports 件列表 先一下。很多情况是新的 port 被加到的分里，然后又立即被移走。会造成源代不必要和不良的膨。

5.3.4. 如何提建立新的分

由于 Ports Collection 在持，已引入了多新的分。新的分既可以是 虚的 分 - 些分在整个 ports 目中没有属于自己的子目 - 或 物理的 分 - 它有自己的子目。接下来我将与建立新的物理分有的事，以便助理解如何提建立新的分。

我目前的做法是避免建立新的物理分，除非有非常多的 port 被入一分，或者 port 属于某一特定的小体 (例如，与某人言相)，或者皆是。

做的原因是修改会 committer 和用都不得不行 多工作 来在 Ports Collection 行或追踪修改。此外，提新的分通常都会引起争。(可能是因于某个分是否 "太大" 一直没有非常一致的意的故，一方面，分是否能能有助于 (以及多少个分是合的)，等等，也都是。)

下面是具体的：

1. 在 FreeBSD ports 件列表 提新的分。提供建立新分的依据，包括什有的分不，以及希望移位置的一系列 port 的名字。(如果有尚在 GNATS 而未 commit 的 port，也一一列出。) 如果是相 port 的人或提交者，明一情况可能有助于的提得到通。
2. 参与。
3. 如果有人支持的建，及提交一个 PR，其中包括提 PR 的理由，以及需要移的 port 的列表。理想情况下，个 PR 也包含下列文件的丁：
   - 行 repocopy 之后 Makefile 行的修改
   - 新分的 Makefile
   - 旧分的 Makefile
   - 依于旧 port 的 port 的 Makefile
   - (此外，作一加分因素，可以按照 Committer 指南所介的流程，提供一些其它需要修改的文件。)
4. 由于是一影 ports 基施的，它不及 repo-copy 的使用，而且也可能会影集群的回操作，因此 PR 分派 Ports 管理 <portmgr@FreeBSD.org>。
5. 如果一 PR 得到批准，某个 committer 将按照在 Committer 指南 中所介的来完成余下的工作。

提新的虚分和上述程似，但会容易多，因不需要地移任何 port。情况下，PR 附的丁，就只需要修改影到的 port 的 Makefile，以便在其中的 CATEGORIES 中加入新的分了。

5.3.5. 如何提分行重新

有些候会有一些人提重新将分 2- 或某基于字的。目前止，没有行任何相的改，因

尽管□些修改比□容易完成， 但修改整个 Ports Collection 所需要□行的工作， 至少也是令人生畏的。 在□表□的□点之前， □□□在□件列表存□中□史上所□行□的提□； 此外， □也会被要求提供一个可用的原形。

5.4. 源□包文件

在 Makefile 中的第二部分是描述用于□□ port 所必需下□的文件， 以及到什□地方去下□它□。

5.4.1. DISTVERSION/DISTNAME (源□包版本号/名称)

DISTNAME 是作者称呼□所 port □件的名字。 DISTNAME 的默□□是 ${PORTNAME}-${PORTVERSION}， 因此只有在需要□才□手工指定。 DISTNAME 只在□个地方用到。 第一□是源□包文件列表 (DISTFILES)， 其默□□是 ${DISTNAME}${EXTRACT_SUFX}。 第二□是源□包□被展□到的目□名， 即 WRKSRC 所指定的目□， 其默□□是 work/${DISTNAME}。

某些□件作者□布源□包的□候并不采取 ${PORTNAME}-${PORTVERSION} □□的模式， □可以通□□置 DISTVERSION 来自□□理。 PORTVERSION 和 DISTNAME 会自□地展□， 当然， 也可以改掉它。 下表□出了一些例子：

DISTVERSION	PORTVERSION
0.7.1d	0.7.1.d
10Alpha3	10.a3
3Beta7-pre2	3.b7.p2
8:f_17	8f.17

PKGNAMEPREFIX 和 PKGNAMESUFFIX 并不影□ DISTNAME。 此外□□注意 WRKSRC 等于 work/${PORTNAME}-${PORTVERSION}， 而源代□的□□包□可能是 ${PORTNAME}-${PORTVERSION}${EXTRACT_SUFX} 以外的其它名字。 一般情况下□□保持 DISTNAME 不□ - 更好的方法是定□ DISTFILES 而不是同□□置 DISTNAME 和 WRKSRC (可能□有 EXTRACT_SUFX)。

5.4.2. MASTER_SITES (主流下□站点)

□□ FTP/HTTP-URL 指向 MASTER_SITES 中原始□□□的目□部分。 不要忘了□尾的斜□ (/)！

make 宏将□□使用 FETCH 来□取所指定的源□包文件， 如果无法在本地系□中□到□些文件的□。

建□□指定多个□像站点， 最好是在不同的大洲上的。 □□将有效地防止由于大□□网□□□所□致无法下□的□□。 我□甚至打算□加自□□□距□最近的站点并从那里下□的功能； 使用多个站点是□□做的重要一□。

如果原始的源□包可以从比□流行的□件下□站点， 例如 SourceForge、 GNU 或是 Perl CPAN 等等来□得， □可能会希望使用□似 MASTER_SITE_* □□的□写来表示它□ (例如 MASTER_SITE_SOURCEFORGE、 MASTER_SITE_GNU 以及 MASTER_SITE_PERL_CPAN)。 只需将 MASTER_SITES □□□些□量， 并使用 MASTER_SITE_SUBDIR 来指定路径就可以了。 下面是一个例子：

```
MASTER_SITES=         ${MASTER_SITE_GNU}
MASTER_SITE_SUBDIR=   make
```

此外， □□可以用更□□略的格式：

```
MASTER_SITES=    GNU/make
```

□些□量是在 /usr/ports/Mk/bsd.sites.mk 中定□的。 新□目会随□□加， 因此在□提交 port 之前， □先看一看□个文件的最新版本。

□□常用□件下□站的□多 暗□魔法 宏， □能□自□判断目□的□□。 □于□些站点， 只要使用与之□□的□写， 系□便会自□□□生成相□的子目□配置。

```
MASTER_SITES=    SF
```

如果系□猜□的路径不□， □可以使用下面□□的配置来替□。

```
MASTER_SITES=    SF/stardict/WyabdcRealPeopleTTS/${PORTVERSION}
```

表 *1.* 常用的魔□ MASTER_SITES 宏

宏	自□猜□的子目□
APACHE_JAKARTA	/dist/jakarta/${PORTNAME:S,-,,/,}/source
BERLIOS	/${PORTNAME:L}
CHEESESHOP	/packages/source/source/${DISTNAME:C/(.).*/\1/}/${DISTNAME:C/(.*)-[0-9].*/\1/}
DEBIAN	/debian/pool/main/${PORTNAME:C/^((lib)?.).*$/\1/}/${PORTNAME}
GCC	/pub/gcc/releases/${DISTNAME}
GNOME	/pub/GNOME/sources/${PORTNAME}/${PORTVERSION:C/^([0-9]+\.[0-9]+).*/\1/}
GNU	/gnu/${PORTNAME}
MOZDEV	/pub/mozdev/${PORTNAME:L}
PERL_CPAN	/pub/CPAN/modules/by-module/${PORTNAME:C/-.*//}
PYTHON	/ftp/python/${PYTHON_PORTVERSION:C/rc[0-9]//}
RUBYFORGE	/${PORTNAME:L}
SAVANNAH	/${PORTNAME:L}
SF	/project/${PORTNAME:L}/${PORTNAME:L}/${PORTVERSION}

5.4.3. EXTRACT_SUFX (□□包所用的□展名)

如果□有一个源□包文件， 而它使用了某□怪□的□展名来表□□□方法， □□置 EXTRACT_SUFX。

例如， 如果源□包文件的名字是 foo.tgz 而非更□一般的 foo.tar.gz， □□写上：

```
DISTNAME=       foo
EXTRACT_SUFX=   .tgz
```

USE_BZIP2 和 USE_ZIP □量会自□根据需要将 EXTRACT_SUFX □置□ .tar.bz2 或 .zip。 如果□□个都没□置， □ EXTRACT_SUFX 的 默□□将是 .tar.gz。

任何□候都不需要同□□置 EXTRACT_SUFX 和 DISTFILES.

5.4.4. DISTFILES (全部源代□包)

有些□候所下□的文件名字和 port 的名字没有任何□系。 例如， 可能是 source.tar.gz， 或者与此□似的其它名字。 也有一些其它的□用□件， 它□的源代□可能被存放到了不同的□□包中， 而且全都需要下□。

如果遇到□□情况， 可以将 DISTFILES □置□以空格分隔的一□需要下□的文件列表。

```
DISTFILES=      source1.tar.gz source2.tar.gz
```

如果没有予以明□的□置， DISTFILES 的默□□将是 ${DISTNAME}${EXTRACT_SUFX}。

5.4.5. EXTRACT_ONLY (只解□□部分源文件)

如果只有一部分 DISTFILES 需要解□□ - 例如， 其中的一个是源代□， 而其它□是未□□的文□ - 此□□把那些需要解□□的文件加到 EXTRACT_ONLY 中。

```
DISTFILES=      source.tar.gz manual.html
EXTRACT_ONLY=   source.tar.gz
```

如果 DISTFILES 中 没有 需要解□□的文件， □□将 EXTRACT_ONLY □□空串。

```
EXTRACT_ONLY=
```

5.4.6. PATCHFILES (通□下□得到的□丁文件)

如果□的 port 需要来自 FTP 或 HTTP 的一些□外的□丁， □将 PATCHFILES □置□□些文件的名字， 并将 PATCH_SITES 指向包含□些文件的目□的 URL (格式与 MASTER_SITES 相同)。

如果□些□丁， 由于包含了其它的目□名， 而□致它□不是相□于源代□目□的□□目□ (也就是 WRKSRC) 的□， 就需要相□地□置 PATCH_DIST_STRIP 了。 例如， 如果□丁中所有的目□名前面都有一个多余的 foozolix-1.0/， 就□□置 PATCH_DIST_STRIP=-p1。

不需要担心□丁文件本身是否是□□的； 如果文件名以 .gz or .Z □尾， 系□会自□解□□。

如果□丁是同某些其它文件， 例如文□， 一同以 gzip □□的 tar 格式□布的， 就不能□□地使用 PATCHFILES 了。 □□情况下， □□将□些□丁包的文件和位置加入到 DISTFILES 和 MASTER_SITES 中。 然后， 用 EXTRA_PATCHES □量来指出□些文件， □□ bsd.port.mk 就会自□地□□□用□些□丁了。 需要特□注意的是， 不要 将□丁文件□制到

PATCHDIR 目□中 - □个目□可能是不可写的。

□□包会以同源代□一□的方式解□□， 因此不需要自行完成解□□操作， 并□制□丁文件。 如果□一定要□□做， 就要注意， 不要□解□□出来的文件覆□先前已□存在的文件。 此外， □□做□需要手工□加命令， 以便在 pre-clean target 中□除□些□制出来的文件。

5.4.7. 来自不同站点的多个源代□包或□丁文件 (MASTER_SITES:n)

(□一□在某□程度上□被□作 "□□□□"； □□始□□□□文□的□者可能会希望先跳□□一部分)。

□一□提供了被称作 MASTER_SITES:n 和 MASTER_SITES_NN 的下□控制机制。 □里我□把它□称□ MASTER_SITES:n。

首先□出一些背景。 OpenBSD 在其 DISTFILES 和 PATCHFILES □量中提供了一个很棒的功能， 即， 允□□些文件和□丁□有 :n 后□， 其中 n 可以使用 [0-9]， 来表□□。 例如：

```
DISTFILES=      alpha:0 beta:1
```

在 OpenBSD 中， 源□包文件 alpha □被□□到□量 MASTER_SITES0 而不是公共的 MASTER_SITES □量上； 而 beta □□□□到 MASTER_SITES1 上。

□是一个很有意思的功能， 它可以避免无休止地搜索正□的下□站点的□程。

想象 DISTFILES 中指定了 2 个文件， 而 MASTER_SITES 包含了 20 个站点的情形， □其中□多站点慢如□牛， 而 beta 可以在 MASTER_SITES 的所有站点□到， 而 alpha 只能在第 20 个上面□到。 如果□□人了解□一点， 那□□□所有的站点无疑是在浪□□□， 不是□？ □□然不是□始一个愉快周末的好□法！

□在□有了一个感性的□□了， 想象一下 DISTFILES 和更多的 MASTER_SITES。 □然， 我□的 "distfiles □□□先生" 会感□□□少他浪□在等待下□上所耗□的□□。

下一□中， 将按照 FreeBSD □上述想法的□□来加以□□。 我□□ OpenBSD 所提出的概念□行了一些改□。

5.4.7.1. □化信息

□一□将介□如何迅速地□从不同的站点以及子目□下□多个源□包和□丁□行精□的控制。 □里， 我□将描述 MASTER_SITES:n 的一□□化用法。 □于多数情况而言□□做是足□的。 然而， 如果□需要更多信息， □需要参考下面的几□。

一些□用程序需要从多个站点下□不同的源□包。 例如， Ghostscript 包括了程序核心本身， 以及大量的□□文件， 以及□取决于用□的打印机品牌和型号的□□程序。 某些□□文件已□随程序核心附□， 但也有很多需要从其它站点下□。

□了□□□□需要， □一个 DISTFILES □□跟随一个冒号， 以及一个 "□□名"。 在 MASTER_SITES 的□个站点也□跟随冒号和□□名， 以便指定从□个网站下□源□包文件。

例如， 考□一个将源代□包分□□部分， 即 source1.tar.gz 和 source2.tar.gz 的□件， 它必□从□个不同的站点下□。 port 的 Makefile □包括□似 □化的 MASTER_SITES:n 用法， □个文件来自一个站点的配置。

例 1. □化的 MASTER_SITES:n 用法， □个文件来自一个站点

```
MASTER_SITES=   ftp://ftp.example1.com/:source1 \
        ftp://ftp.example2.com/:source2
DISTFILES=      source1.tar.gz:source1 \
        source2.tar.gz:source2
```

多个源□包可以使用同一个□□。 □□前面的例子， 假定□加了第三个源□包， source3.tar.gz， □从 ftp.example2.com 下□。 Makefile 的□部分□写成 □化的 MASTER_SITES:n 用法，其中同一个站点上提供了不止一个文件 的□子。

例 2. □化的 MASTER_SITES:n 用法， 其中同一个站点上提供了不止一个文件

```
MASTER_SITES=   ftp://ftp.example1.com/:source1 \
        ftp://ftp.example2.com/:source2
DISTFILES=      source1.tar.gz:source1 \
        source2.tar.gz:source2 \
        source3.tar.gz:source2
```

5.4.7.2. 深入介□

前面的例子无法□足□的需求？ □一□， 我□将□□介□ MASTER_SITES:n 的精□控制是如何工作的， 以及如何修改□的 port 来使用它□。

1. 元素可以包含 :n □□的后□， 其中 *n* 是 [^:,]+， 概念上即 *n* 可以取任意数字或字母， 但我□目前将其限定□ [a-zA-Z_][0-9a-zA-Z_]+。

   此外， 字符串匹配□□大小写是敏感的； □言之， n 与 N 不同。

   但是， 由于表□特殊的意□， 下列□□不能用于后□： default、 all 和 ALL (它□会在 ii 中介□的部分用到)。 此外， DEFAULT 是一个有特殊用途的□ (□参□ 3)。

2. 后□□ :n 的□目属于 n □， 而 :m 属于 m □， 依此□推。

3. 没有后□的元素是无□的， 也就是它□都属于那个特殊的 DEFAULT □。 □元素加入 DEFAULT 后□通常是多余的， 除非□有同□属于 DEFAULT 和其它□的元素 (参□ 5)。

   下面的例子是等价的， 但通常□□用第一个：

   ```
   MASTER_SITES=   alpha

   MASTER_SITES=   alpha:DEFAULT
   ```

4. □之□不是互斥的， 同一元素可以同□隶属于多个□， 而□□可以□空或者有任意多个元素。 同一□中的重□元素， 并不会被自□消去。

5. 如果希望同一元素同□属于多个□， 可以用逗号 (,) 分□。

   □□□法可以避免□□指定不同的□而多次重□同一元素。 例如 :m,n,o 表示□个元素同□属于 m、 n 和 o □三□。

   下面□些写法都是等价的， 但只推□使用最后一□：

   ```
   MASTER_SITES=   alpha alpha:SOME_SITE

   MASTER_SITES=   alpha:DEFAULT alpha:SOME_SITE

   MASTER_SITES=   alpha:SOME_SITE,DEFAULT

   MASTER_SITES=   alpha:DEFAULT,SOME_SITE
   ```

6. 同一□中的所有站点， 会根据 MASTER_SORT_AWK 排序。 在 MASTER_SITES 和 PATCH_SITES 中的□也会□行排序。

7. 在 MASTER_SITES、 PATCH_SITES、 MASTER_SITE_SUBDIR、 PATCH_SITE_SUBDIR、 DISTFILES， 以及 PATCHFILES 中， 都可以使用□， 其□法□：

   a. 所有 MASTER_SITES、 PATCH_SITES、 MASTER_SITE_SUBDIR 以及 PATCH_SITE_SUBDIR 的元素， 都必□以 / 字符□尾。 如果有元素属于某些□， □□后□ :n 必□出□在□□符 / 之后。 MASTER_SITES:n 机制依□于 / 的存在， 以避免在 :n 是元素一部分， 而 :n 同□又表示□ n □□生混□。 □了兼容性的考□， 因□之前 / □□符在 MASTER_SITE_SUBDIR 和 PATCH_SITE_SUBDIR 元素中都不是必需的， 如果后□所□跟的字符不是 /， □ :n 将被□□是元素的一部分， 而不被当作□后□， 即使元素□有 :n 后□。 □参□ 在 MASTER_SITE_SUBDIR 中 MASTER_SITES:n 的□□用法 和 用到逗号分隔符、 多个文件， 多个站点和不同子目□的 MASTER_SITES:n □□用法 以了解□一□的□□。

      例 3. 在 MASTER_SITE_SUBDIR 中 MASTER_SITES:n 的□□用法

      ```
      MASTER_SITE_SUBDIR=     old:n new/:NEW
      ```

      - □ DEFAULT 中的目□ → old:n
      - □ NEW 中的目□ → new

例 4. 用到逗号分隔符、 多个文件， 多个站点和 不同子目□的 MASTER_SITES:n □□用法

```
MASTER_SITES=   http://site1/%SUBDIR%/ http://site2/:DEFAULT \
        http://site3/:group3 http://site4/:group4 \
        http://site5/:group5 http://site6/:group6 \
        http://site7/:DEFAULT,group6 \
        http://site8/%SUBDIR%/:group6,group7 \
        http://site9/:group8
DISTFILES=      file1 file2:DEFAULT file3:group3 \
        file4:group4,group5,group6 file5:grouping \
        file6:group7
MASTER_SITE_SUBDIR=     directory-trial:1 directory-n/:groupn \
            directory-one/:group6,DEFAULT \
            directory
```

前述的例子的□果是下述的□于下□行□的精□控制。 站点的列表按照使用的□序□出。

- file1 将从
  - MASTER_SITE_OVERRIDE
  - http://site1/directory-trial:1/
  - http://site1/directory-one/
  - http://site1/directory/
  - http://site2/
  - http://site7/
  - MASTER_SITE_BACKUP

  下□。
- file2 将和 file1 以同□的方式下□， 因□它□属于同一□
  - MASTER_SITE_OVERRIDE
  - http://site1/directory-trial:1/
  - http://site1/directory-one/
  - http://site1/directory/
  - http://site2/
  - http://site7/
  - MASTER_SITE_BACKUP
- file3 将从
  - MASTER_SITE_OVERRIDE
  - http://site3/
  - MASTER_SITE_BACKUP

下□。

- file4 将从
  - MASTER_SITE_OVERRIDE
  - http://site4/
  - http://site5/
  - http://site6/
  - http://site7/
  - http://site8/directory-one/
  - MASTER_SITE_BACKUP

  下□。
- file5 将从
  - MASTER_SITE_OVERRIDE
  - MASTER_SITE_BACKUP

  下□。
- file6 将从
  - MASTER_SITE_OVERRIDE
  - http://site8/
  - MASTER_SITE_BACKUP

  下□。

8. 如何□来自 bsd.sites.mk 的特殊□量， 例如 MASTER_SITE_SOURCEFORGE □行分□？

   参□ MASTER_SITE_SOURCEFORGE 中 MASTER_SITES:n 的□□用法。

   例 *5.* MASTER_SITE_SOURCEFORGE 中 MASTER_SITES:n 的□□用法

   ```
   MASTER_SITES=   http://site1/ ${MASTER_SITE_SOURCEFORGE:S/$/:sourceforge,TEST/}
   DISTFILES=      something.tar.gz:sourceforge
   ```

   something.tar.gz 将从所有 MASTER_SITE_SOURCEFORGE 中的站点下□。

9. 如何与 PATCH* □量□用？

   前面的例子介□的都是 MASTER* □量， 但□于 PATCH* 也是完全一□的， 它□在 □化的 PATCH_SITES 中的 MASTER_SITES:n 用法。 有所介□。

例 *6.* □化的 PATCH_SITES 中的 MASTER_SITES:n 用法。

```
PATCH_SITES=    http://site1/ http://site2/:test
PATCHFILES=     patch1:test
```

5.4.7.3. 会改□ **ports** 的□些行□？ □些不会？

i. 所有普通的 ports 的行□都会保持不□。 MASTER_SITES:n 功能的代□， 只有在某些元素包含了前述， 特□是 7 中所提及□法的 :n 后□□， 才会□用。

ii. 不受影□的 port target： checksum、 makesum、 patch、 configure、 build， 等等。 □然， do-fetch、 fetch-list、 master-sites 和 patch-sites 的行□会□生□化。

- do-fetch： 会按照新的、 □有□后□的 DISTFILES 和 PATCHFILES 在 MASTER_SITES 和 PATCH_SITES 所匹配的□元素， 以及 MASTER_SITE_SUBDIR 和 PATCH_SITE_SUBDIR 来□行。 □参□ 用到逗号分隔符、 多个文件， 多个站点和 不同子目□的 MASTER_SITES:n □□用法。
- fetch-list： 和旧式的 fetch-list □似， 但以同 do-fetch 相似的方式□理□。
- master-sites 和 patch-sites： (与旧版本不兼容) □返回□ DEFAULT 的元素； 事□上， 它□会□行 master-sites-default 和 patch-sites-default □□个 target。

  更□一□， 使用 master-sites-all 或 patch-sites-all □□个 target 之一， 要比直接□□ MASTER_SITES 或 PATCH_SITES 更好。 此外， 未来版本可能不再保□直接□□能□正□工作。 □参□ [porting-master-sites-n-new-port-targets-master-sites-all] 以了解□于□些新 target 的更多技□□□。

iii. port 中的新 target

a. 一系列 master-sites-n 和 patch-sites-n target 可以分□用来列出 MASTER_SITES 和 PATCH_SITES 中的 *n* □的内容。 例如， master-sites-DEFAULT 和 patch-sites-DEFAULT 都会返回 DEFAULT □的内容， 而 master-sites-test 和 patch-sites-test □返回 test □的内容， 等等。

b. 新□的 master-sites-all 和 patch-sites-all □□个 target， 会完成先前 master-sites 和 patch-sites 所做的工作。 它□会返回所有□的元素， 就像□些元素都属于同一□一□， 并且会列出与 MASTER_SITE_BACKUP 或 MASTER_SITE_OVERRIDE 中在 DISTFILES 或 PATCHFILES 中指定的同□多个； 分□□于 master-sites-all 和 patch-sites-all。

5.4.8. DIST_SUBDIR (独立的源□包子目□)

避免□□的 port 使 /usr/ports/distfiles 陷入混乱。 如果□的 port 需要下□很多文件， 或者需要下□可能与其它 port 的源文件名冲突的文件 (例如， Makefile)， □□将 DIST_SUBDIR □置□ port 的名字 (通常可以用 ${PORTNAME} 或 ${PKGNAMEPREFIX}${PORTNAME})。 □将把 DISTDIR 从默□的 /usr/ports/distfiles 改□ /usr/ports/distfiles/DIST_SUBDIR， 并将与□的 port 有□的文件放到那个目□中。

此外， 它也会在□□文件主服□器 ftp.FreeBSD.org 上□□同一子目□下的文件 (直接在□的 Makefile 中□置 DISTDIR □不会有□□的效果， 因此□□使用 DIST_SUBDIR。)

□一□置并不影□□在 Makefile 中定□的 MASTER_SITES。

5.4.9. ALWAYS_KEEP_DISTFILES (一直保存源□包)

如果□的 port 采用的是□□□的包， 但却采用了某□要求源代□必□与□□□版本一同提供的授□， 例如 GPL，□□使用 ALWAYS_KEEP_DISTFILES 来告□ FreeBSD □□集群保留一□在 DISTFILES 中文件的副本。 一般来□□些 port 的用□并不需要□些文件， 因此， 只在定□了 PACKAGE_BUILDING 符的□候， 才将源代□包文件加入 DISTFILES 是个好主意。

例 7. 如何使用 ALWAYS_KEEP_DISTFILES。

```
.if defined(PACKAGE_BUILDING)
DISTFILES+=             foo.tar.gz
ALWAYS_KEEP_DISTFILES=  yes
.endif
```

当□在 DISTFILES 加入其它文件□， □□必□保□些文件也出□在了 distinfo 中。 此外， □些□外的文件通常也会展□到 WRKDIR 中， □于某些 ports， □可能□致一些不希望的副作用， 因而需要□行特□的□理。

5.5. MAINTAINER (□□人)

□在此□写上□的□子□件地址。 *:-)*

需要注意一点， MAINTAINER □量的□只能是一个不包括注□部分的□子□件地址， 其格式□□ user@hostname.domain。 □不要在此□写任何□明性的文字， 例如□的真□姓名 - □会□ bsd.port.mk □来麻□。

□□人有□任保持 port 随□更新， 并□保其能□正□地□行。 □□的 port □□人□□□明， □参□ port □□人面□的挑□ 一□。

□于 port 的修改， □被□□ port 的□□人□行□□， 且在 commit 之前需要□得其□□人的同意。 假如某一 port 的□□人没有在□周之内 (不包括主要的公共假日) □□来自用□的更新□求， □可□□□□人超□， 在□□情况下可以在没有□□人明□同意的情形下□行更新。 如果□□人在多□三个月的□□内没有□行任何□□， □可以□□□□□人不辞而□， 允□□出□此□□□的 port □行□□人□更。 尽管如此， □□人□ Ports 管理□□ <portmgr@FreeBSD.org> 或者 Security Officer □□ <security-officer@FreeBSD.org> 的 port 不受此限。□□□人□□些小□的 port □行未□□可的 commit 是不允□的。

我□保留□□□人所提交修正案□行改□的□力， 以便使其更符合□行的 Ports Collection □□， 而无需提交□丁的人明□批准。 此外， 大□模的基□性修改， 也可能使 port 在没有得到□□人同意的情形下□行修改。 但□□修改都不□影□ port 本身的功能。

Ports 管理□□ <portmgr@FreeBSD.org> 保留以任何原因收回或□□任何人□□□的□力， 而 Security Officer □□ <security-officer@FreeBSD.org> □保留以安全原因收回或□□□□□的□力。

5.6. COMMENT (一句□□明)

□一□量用于指定 port 的一句□□明。 □ 勿将 package 的名字 (或□件的版本) 放在□明中。 □一□明的第一个字母□大写， □尾不用句点。 下面是一个例子：

```
COMMENT=	A cat chasing a mouse all over the screen
```

Makefile 中的 COMMENT □量□□□接着 MAINTAINER □量出□。

□□必将 COMMENT □行限制在不超□ 70 个字符之内， 因□□行内容会成□ pkg_info(1) 呈□□用□的 port 的一句□□介。

5.7. 依□□系

□多 ports 会依□其它 port。 □是包括 FreeBSD 在内的多数 □-Unix 系□的很方便的功能。 □□功能，可以避免在□个 port 或□□□包中都□上重□的依□的代□， 而可以以依□□系的方式去共享它□。 有七个□量用于□助□□保所需的文件都存在于用□的机器上。 此外， 也提供了用于支持常□情形的依□□系□量， 以及□依□□系行□的更多控制。

5.7.1. LIB_DEPENDS (依□的函数□/共享□)

□个□量用于指定 port 所依□的共享□。 其内容是由一系列 lib:dir:target 元□□成的表， 其中 *lib* 是共享□的名字， 而 *dir* □是在□不到□□□从□里□□和安装， 最后， *target* 用于指定在那个目□中□用的 target。 例如，

```
LIB_DEPENDS=	jpeg.9:${PORTSDIR}/graphics/jpeg
```

会□□主版本号□ 9 的 jpeg 共享□， 如果它不存在， □会□入到□的 ports 目□中的 graphics/jpeg 子目□， 并□□和安装它。 如果□指定的 *target* 就是 DEPENDS_TARGET (默□是 install)， □可以略去不写。

lib 部分是一个正□表□式， 用于在 ldconfig -r 的□出中□行□□。 可以使用□似 intl.[5-7] 和 intl □□的□。 前一□模式， 即 intl.[5-7]， 能□匹配 intl.5、 intl.6 和 intl.7 中的任意一个。 第二□模式， 即 intl □可以匹配任意版本的 intl □。

依□□系会被□□□次， 一次是在 extract target 中， 而□一次□是在 install target。 □外， 依□□系的名字会放到 package 中， 以便□ pkg_add(1) 能□自□地在用□系□上安装所需的未安装的其它 package。

5.7.2. RUN_DEPENDS (依□的□行□境)

□个□量可以用来指定 port 在□行□所需要的可□行文件， 以及□源文件。 它是一系列 path:dir:target 元□的列表， □里， *path* □所需的可□行， 或者□源文件的名字， *dir* 是在无法□到□些文件或目□□， 去什□地方完成□□和安装以便□得□些文件； 而 *target* □用来指定在□个目□中所□用的 target 的名字。 假如 *path* 以斜□ (/) □始， □会当作普通文件， 使用 test -e 来□□； 反之， □系□会假定□是一个可□行文件， 并且用 which -s 来□□程序是否存在于搜索路径中。

例如，

```
RUN_DEPENDS=	${LOCALBASE}/etc/innd:${PORTSDIR}/news/inn \
		xmlcatmgr:${PORTSDIR}/textproc/xmlcatmgr
```

将□□文件， 或者目□ /usr/local/etc/innd 是否存在， 如果□不到， □将从 port 目□的 news/inn 子目□加以安装。 系□也会□□是否能□在搜索路径中□到名□ xmlcatmgr 的文件， 如果□不到的□， □会□入 ports 目□中的 textproc/xmlcatmgr 子目□， 并□行□□和安装的操作。

□□情况下， innd □□上是一个可□行文件； 如果可□行文件不会出□在搜索路径中，□就需要指定完整路径了。

ports □□集群上官方的搜索 PATH 是

```
/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/sbin:/usr/local/bin:/usr/X11R6
/bin
```

□个依□□系会在 install target 的□程中□行□□。 此外， 依□□系的名字会被放到 package 中， 以便 pkg_add(1) 能□在用□的系□中尚未安装相□□件□自□地安装那些 package。 如果□希望指定一个的 *target* 和默□的 DEPENDS_TARGET 相同， □可以略去不写。

一□比□常□的情形是 RUN_DEPENDS 和 BUILD_DEPENDS 完全一□， □□情况在移植的□件是采用脚本□言□写，或□□□境与□行□境需求相同□尤其普遍。 □□情况可以用下面□□明了的方式直接将其中一个□量□□□一个□量：

```
RUN_DEPENDS= ${BUILD_DEPENDS}
```

不□， □□□□有可能会令□行□境被某些没有在 port 原本的 BUILD_DEPENDS 明□定□的依□□系□染。 □致□□情况的原因是 make(1) □算□量□□□默□采用的是延后□算 (lazy evaluation)。 例如， 如果在 Makefile 中使用了 USE_* □量， □些□量就会由 ports/Mk/bsd.*.mk □理， 并填写与之□□的□□依□□系。 例如， USE_GMAKE=yes 会把 devel/gmake 加入到 BUILD_DEPENDS。 如果希望避免□些附加的依□□系□染 RUN_DEPENDS， 在使用□□的□候就需要小心考□□□□展的情况， 例如， 可以在□□展□之前□制□量的□：

```
RUN_DEPENDS:=  ${BUILD_DEPENDS}
```

5.7.3. BUILD_DEPENDS (依□的□□□境)

此□量用于指定用来□□ port 的可□行文件或□源文件。 与 RUN_DEPENDS □似， 它是一个 path:dir:target 元□的列表。 例如，

```
BUILD_DEPENDS=
        unzip:${PORTSDIR}/archivers/unzip
```

将□□名□ unzip 的可□行文件是否存在， 如果不存在， □会□入到□的 ports 目□中的 archivers/unzip 并完成□□和安装工作。

□里的 "build" 表示从解□□到□□的全部□程。 依□□系是在 extract target 的□程中□□的。 假如□要指定的 *target* 和 DEPENDS_TARGET 相同， □可以略去不写。

5.7.4. FETCH_DEPENDS (依□的下□□境)

□一□量用于指定 port 在下□□所需的可□行文件或□源文件。 和前□个□似， 它是一□ path:dir:target 元□。 例如，

```
FETCH_DEPENDS=
        ncftp2:${PORTSDIR}/net/ncftp2
```

将□□名□ ncftp2 的可□行文件是否存在， 如果□不到， □将□入到□ ports 目□中的 net/ncftp2 子目□并加以□□和安装。

□个依□□系是在 fetch target □程中□□的。 如果与 DEPENDS_TARGET 相同， □可以省略 *target* 部分。

5.7.5. EXTRACT_DEPENDS (依□的解□□□境)

此□量用于指定 port 在解□□□所需的可□行文件或其它□源文件。 和前一个□量□似， 它是一系列 path:dir:target 元□的列表。 例如，

```
EXTRACT_DEPENDS=
        unzip:${PORTSDIR}/archivers/unzip
```

将□□名□ unzip 的可□行文件是否存在， 如果不存在， □会□入到□的 ports 目□中的 archivers/unzip 子目□，予以□□和安装。

□个依□□系是在 extract target 的□程中□□的。 如果与 DEPENDS_TARGET 相同， □可以略去 *target* 部分。

只有在其它方式都不可用 (默□是 gzip) 而且无法通□ USE_* 所介□的 USE_ZIP 或 USE_BZIP2 都不能□到需要□， 才□使用□个□量。

5.7.6. PATCH_DEPENDS (依□的打□丁□境)

□个□量用于指定 port 在□行 patch 操作□所需的可□行文件或其它□源文件。 和前一个□量□似， 它是一□ path:dir:target 元□的表。 例如，

```
PATCH_DEPENDS=
        ${NONEXISTENT}:${PORTSDIR}/java/jfc:extract
```

表示□入到□的 ports 目□中的 java/jfc 子目□， 并将其解□□。

□个依□□系是在 patch target 的□程中□□的。 *target* 部分如果和 DEPENDS_TARGET 相同， 就可略去不写。

5.7.7. USE_*

提供了一系列□量， 用以封装大量 port 都用到的依□□系。 □然使用□些□量是可□的， 但它□能□著□少 port 的 Makefile □□性。 □些□量的共同特征在于， 它□的名字都是 USE_* □□的形式。 □些□量的使用， □□格限制于 port 的 Makefile 以及 ports/Mk/bsd.*.mk， 而□不□用于表□用□能□□置的□□ - □□情况下□采用 WITH* 和 WITHOUT* □□的□量。

在 任何 情况下， 都不□在 /etc/make.conf 中配置任何 USE_*。 例如， □置

```
USE_GCC=3.4
```

将□致□个 port 都依□ gcc34， 甚至包括 gcc34 本身！

表 2. 常用的 USE_* □量

□量	含□
USE_BZIP2	此 port 的源□包是使用 bzip2 □□的。
USE_ZIP	此 port 的源□包是用 zip □□的。
USE_BISON	此 port 在□□□使用 bison。
USE_CDRTOOLS	此 port 需要使用 cdrecord， 根据用□的喜好， 可能是 sysutils/cdrtools 或 sysutils/cdrtools-cjk。
USE_GCC	此 port 需要使用某一特定版本的 gcc 才能完成□□。 可以使用□似 3.4 □□的□来精□指定版本。 如果希望使用不低于某一版本的□□器， □可以用 3.4+ □□的形式。 如果与所希望的版本吻合， □将使用基本系□中所提供的 gcc， 反之， 系□会从 ports 中安装所希望版本的 gcc， 并□整 CC 以及 CXX □量的□置。

与 gmake 和 configure 脚本有□的□量在 □□机制 中□行了介□， 而 autoconf、 automake 以及 libtool 的介□□可以在 利用 GNU autotools □到。 使用 perl 介□了与 Perl 有□的的□量。 使用 X11 中列出了□于 X11 的□量。 □于 GNOME 的□量在 使用 GNOME， 而□于 KDE 的□在 使用 KDE。 使用 Java □述了和 Java 有□的□量， 而 Web □用， Apache 和 PHP □包含了□于 Apache、 PHP 以及 PEAR 的介□性信息。 □于 Python， 在 使用 Python □行了□□， 而□于 Ruby 的介□， □可以在 使用 Ruby 中□到。 使用 SDL 提供了用于 SDL □用程序的□量介□， 最后， 使用 Xfce 包含了□于 Xfce 的信息。

5.7.8. 在依□□系中指定最低版本

在依□某个其他 port □， 可以采用下面的句法， 通□除 LIB_DEPENDS 之外的 *_DEPENDS □量来指定最低版本：

```
p5-Spiffy>=0.26:${PORTSDIR}/devel/p5-Spiffy
```

第一个字段指明了所依□ package 的名字， 用以与 package 数据□中的某□匹配， 然后是比□算符， 以及 package 的版本号。 前面的例子中， 如果系□中安装了 p5-Spiffy-0.26 □□□□足了依□条件。

5.7.9. □于依□□系的□充□明

如前面所提到的那□， 在需要某一依□的 port □， 将□用 DEPENDS_TARGET 所指定的 target。 □一□量的默□□是 install。 □不是一个用□□量， 它不□在 port 的 Makefile 中予以定□。 如果□的 port 需要使用特殊的 target 来□理依□□系， □使用 *_DEPENDS 的 :target 部分， 而不是重定□ DEPENDS_TARGET 来完成。

当□□入 make clean □， 其依□的 port 也会自□□行清理。 如果□不希望如此， □定□□境□量 NOCLEANDEPENDS。

如果 port 依□一些重新□□需要花□很□□□的 port □， 例如 KDE， GNOME 或 Mozilla □，□一方法会非常有用。

要无条件地依□某个 port， 可以使用 ${NONEXISTENT} 作□ BUILD_DEPENDS 或 RUN_DEPENDS 的第一部分。只有在□需要使用其它 port 提供的源代□□才□□□做。 通常也可以通□□□指定来□短□□所需的□□。 例如

```
BUILD_DEPENDS=   ${NONEXISTENT}:${PORTSDIR}/graphics/jpeg:extract
```

表示依□ jpeg port 并将其解□□。

5.7.10. 循□的依□□系是致命的

不要在 ports tree 中引入任何循□依□□系!

ports □□技□不能□容忍循□依□□系。 如果□引入了□□的□系， 就一定会有人安装的 FreeBSD 会因此而□坏，而且□□□象会越来越多。 □些情形很□□□； 如果有疑□， 在□行□□的修改之前， □必□行： cd /usr/ports; make index。 □个□程在旧的机器上会很慢， 但能□□大量的用□ - 也包括□自己 - □救于由□□□□所造成的困惑之中。

5.8. MASTERDIR (主 port 所在的目□)

如果 port 需要依某些□量的□置 (□例来□， 分辨率或□型) 来□□略有不同的□□□包， □可以□□一个□□的包建立不同的目□， □□可以□用□更容易地看到他□想要安装的版本， 但又能在□些 port 之□共用尽可能多的文件。 一般情况下， 如果□用得当， 除主目□之外都只需要很短的 Makefile。 □些 Makefile 中， 可以用 MASTERDIR 来指定其它文件所在的目□。 □外， □□使用一个□量作□ PKGNAMESUFFIX 的一部分， 以便□不同的包□出不同的命名。

用例子来□述□些会更□明晰。 以下是 japanese/xdvi300/Makefile 的部分代□：

```
PORTNAME=       xdvi
PORTVERSION=    17
PKGNAMEPREFIX=  ja-
PKGNAMESUFFIX=  ${RESOLUTION}
 :
# default
RESOLUTION?=   300
.if ${RESOLUTION} != 118 && ${RESOLUTION} != 240 && \
       ${RESOLUTION} != 300 && ${RESOLUTION} != 400
       @${ECHO_MSG} "Error: invalid value for RESOLUTION: \"${RESOLUTION}\""
       @${ECHO_MSG} "Possible values are: 118, 240, 300 (default) and 400."
       @${FALSE}
.endif
```

japanese/xdvi300 也提供了全部常□的□丁， 以及打包用到的文件等等内容。 如果□在那里□入 make，它将使用默□的分辨率□ (300) 并正常地□□ port。

□于其它分辨率而言， 以下是 完整的 xdvi118/Makefile：

```
RESOLUTION=     118
MASTERDIR=      ${.CURDIR}/../xdvi300

.include "${MASTERDIR}/Makefile"
```

(xdvi240/Makefile 和 xdvi400/Makefile 是相似的)。 MASTERDIR 定□会告□ bsd.port.mk 常□的目□， 例如 FILESDIR 以及 SCRIPTDIR □在 xdvi300 中□□。 RESOLUTION=118 □行将覆□在 xdvi300/Makefile 中所作的 RESOLUTION=300 □置， 从而 port 将以分辨率□ 118 的□置来□□。

5.9. □机手册

MAN[1-9LN] □些□量， 会自□地将□机手册加到 pkg-plist (□也意味着 不能 在 pkg-plist 中列出□机手册 - 参□ PLIST 的生成 来了解更多□□)。 此外， □也会□安装□段自□地根据在 /etc/make.conf 中所作的 NO_MANCOMPRESS □置来自□□□机手册文件□行□□或解□□操作。

如果 port □□通□使用符号□接或硬□接将□机手册安装□多个名字， 就必□使用 MLINKS □量来予以明示。 由 port □建的□接， 将由 bsd.port.mk □除和重建， 以□□它□指向了正□的文件。 任何在 MLINKS 中列出的文件都不□在 pkg-plist 中再出□。

要指定是否在安装□□□机手册□行□□， 可以使用 MANCOMPRESSED □量。 □一□量可以取三□□， yes、 no 和 maybe 之一。 yes 表示□机手册已□以□□的形式安装， no 表示□没有， 而 maybe □表示所安装的□件会尊重 NO_MANCOMPRESS 的□置□， 因此 bsd.port.mk 不需要特□做什□事情。

如果□置了 USE_IMAKE 而未定□ NO_INSTALL_MANPAGES， MANCOMPRESSED 会自□□□ yes， 反之□是 no。 除非默□□不合□， 否□就不需要在 port 中明□地加以改□。

如果 port 将□机手册放到了 PREFIX 之外的其它目□， □□使用 MANPREFIX 来加以□置。 此外， 如果只有某些部分的□机手册会安装到不□准的位置， 例如某些 perl 模□的 port， □可以使用 MAN_sect_PREFIX (此□ *sect* 是 1-9、 L 或 N 之一) 来指定。

如果□的□机手册需要装入□用于某一□言□用的子目□， 需要将 MANLANG □□那□□言的名字。 此□量的默□□是 "" (也就是只有英□)。

下面是一个□合的例子。

```
MAN1=           foo.1
MAN3=           bar.3
MAN4=           baz.4
MLINKS=         foo.1 alt-name.8
MANLANG=        "" ja
MAN3PREFIX=     ${PREFIX}/shared/foobar
MANCOMPRESSED= yes
```

□表示 port 会安装六个文件；

```
${MANPREFIX}/man/man1/foo.1.gz
${MANPREFIX}/man/ja/man1/foo.1.gz
${PREFIX}/shared/foobar/man/man3/bar.3.gz
${PREFIX}/shared/foobar/man/ja/man3/bar.3.gz
${MANPREFIX}/man/man4/baz.4.gz
${MANPREFIX}/man/ja/man4/baz.4.gz
```

此外， ${MANPREFIX}/man/man8/alt-name.8.gz 可能会通□□的 port 安装， 也可能不会。 无□如何，都会□建一个符号□接， 把 foo(1) 和 alt-name(8) □机手册□起来。

假如只有部分□机手册是翻□□的， □可以使用一些根据 MANLANG 内容□□生成的□量：

```
MANLANG=       "" de ja
MAN1=          foo.1
MAN1_EN=       bar.1
MAN3_DE=       baz.3
```

□相当于下列文件：

```
${MANPREFIX}/man/man1/foo.1.gz
${MANPREFIX}/man/de/man1/foo.1.gz
${MANPREFIX}/man/ja/man1/foo.1.gz
${MANPREFIX}/man/man1/bar.1.gz
${MANPREFIX}/man/de/man3/baz.3.gz
```

5.10. Info 文件

如果□件包需要安装 GNU info 文件， □需要在 INFO □量中一一列出 (不需要指定 .info 后□)。 系□假定□些文件均会安装到 PREFIX/INFO_PATH 目□中。 如果□件包有需要， 也可以通□修改 INFO_PATH 来指定不同的位置。 不□， 并不推□□□做。 所有列出的□目均是相□于 PREFIX/INFO_PATH 的文件路径。例如， lang/gcc34 表示将 info 文件安装到 PREFIX/INFO_PATH/gcc34， 因此 INFO □写成□似□□：

```
INFO= gcc34/cpp gcc34/cppinternals gcc34/g77 ...
```

□□安装/卸□代□就会自□地在注册包之前将它□加入到□□的 pkg-plist 中了。

5.11. Makefile □□

某些大型□用程序可以在□□□使用一系列配置□□， 用以在系□中已□安装了某些□或□用程序□□加一些功能。例如， □□某□自然 (人□的) □言， GUI 或命令行界面， 由于并不是所有的用□都希望使用□些□或者□用程序，port 系□提供了一□方便的机制， 来□ port 的作者控制□□□的配置。 支持□些特性可以□用□体□更好， 并□到事半功倍的效果。

5.11.1. □□ (Knobs)

5.11.1.1. WITH_* 和 WITHOUT_*

□些□量是□系□管理□准□的。 □多□□的□量被□准化并置于 ports/KNOBS 文件。

在□建一个 port 的□候，不要使用某个□用程序□有的 knob 名称，比如□于 Avahi □个 port，□□用 WITHOUT_MDNS 而不是 WITHOUT_AVAHI_MDNS。

□不□假定□一个 WITH_* 都会有□□的 WITHOUT_* □量， 反之亦然。 一般而言， 会使用默□□。

除非□有□明， □些□量都是□□是否定□， 而不是它□□置了 YES 或 NO。

表 3. 常□的 WITH_* 和 WITHOUT_* □量

□量	意□
WITHOUT_NLS	表示不需要国□化支持， □可以□省□□所消耗的□□。 默□情况下， 会□用国□化支持。
WITH_OPENSSL_BASE	使用基本系□中的 OpenSSL 版本。
WITH_OPENSSL_PORT	从 security/openssl 安装 OpenSSL，即使基本系□中的版本是最新的。
WITHOUT_X11	如果 port 能□在是否包含 X 支持的情况下分□□□， □一般情况□□默□以包含 X 支持的配置来□□。 如果定□了□一□量， □□□□不包含 X 支持的版本。

5.11.1.2. □□ (knob) 的命名

我□建□ port 的□□人□使用相似的□□， 以便最□用□使用， 并□少□□名称的□数。 最□常用的□□名字可以在 KNOBS 文件中□到。

□□的名字□反映其功能。 如果 port 的 PORTNAME 包括 lib- 前□， □□□名中□□去 lib- 前□。

5.11.2. OPTIONS (菜□式可□□)

5.11.2.1. 背景

OPTIONS 将□正在安装 port 的用□提供一个包含可用□□的□□框， 并将用□的□□保存到 /var/db/ports/portname/options 中。 下次重新□□ port □， □些□□将被再次使用。 □□一来， 就不需要□神去□□□之前□□ port □的那几十个 WITH_* 和 WITHOUT_*□□了！

当用□□行 make config (或首次□行 make build) □， 框架会首先□□ /var/db/ports/portname/options。 如果□个文件不存在， □它会使用 OPTIONS 的□来生成一个可以□用或禁用各个□□的□□框。 随后， 用□的□□将保存到 options 文件中， 并被用于□□ port。

如果新版本的 port 新□了 OPTIONS， □系□会再次□出□□框， 并根据先前的 OPTIONS 配置□□先前存在的配置。

使用 make showconfig 可以□看保存的配置。 此外， make rmconfig 可以□除已□保存的配置。

5.11.2.2. □法

OPTIONS □量的□法是：

```
OPTIONS=    OPTION  "□明性文字" 默□□ ...
```

默□□必□是 ON 和 OFF 之一。 □□三元□可以使用多次。

定□ OPTIONS □量的□， 必□在引入 bsd.port.options.mk 之前□行。 而 WITH_* 和 WITHOUT_* □量□只能在引入了 bsd.port.options.mk 之后才可以□行□□。 使用 bsd.port.pre.mk 也可以□到同□的目的，在系□□始提供 bsd.port.options.mk 之前的□多 port 都在使用□□用法。 不□， □注意 bsd.port.pre.mk 会要求某些□量已□□行□定□， 如 USE_* 等。

例 8. □□的 OPTIONS 用法

```
OPTIONS=      FOO "□用 foo □□" On \
              BAR "支持 bar 功能" Off

.include <bsd.port.options.mk>

.if defined(WITHOUT_FOO)
CONFIGURE_ARGS+=    --without-foo
.else
CONFIGURE_ARGS+=    --with-foo
.endif

.if defined(WITH_BAR)
RUN_DEPENDS+=   bar:${PORTSDIR}/bar/bar
.endif

.include <bsd.port.mk>
```

例 9. *Old style use of* OPTIONS

```
OPTIONS=      FOO "Enable option foo" On

.include <bsd.port.pre.mk>

.if defined(WITHOUT_FOO)
CONFIGURE_ARGS+=    --without-foo
.else
CONFIGURE_ARGS+=    --with-foo
.endif

.include <bsd.port.post.mk>
```

5.11.3. 自□激活的特性

在使用 GNU configure 脚本□， 一定要小心有些特性会由其自□□□而激活。 □□通□明□地指定相□的 --without-xxx 或 --disable-xxx 参数到 CONFIGURE_ARGS 来禁用不希望的特性。

例 *10.* □理□□□的□□做法

```
.if defined(WITH_FOO)
LIB_DEPENDS+=       foo.0:${PORTSDIR}/devel/foo
CONFIGURE_ARGS+=    --enable-foo
.endif
```

在前面的例子中， 假□系□中已□安装了 libfoo □。 用□可能并不希望□用程序使用 libfoo， 因此他在 make config □□框中□掉了□个□□。 但是， □用程序的 configure 脚本□□到了系□中存在□个□， 并将其加入到了最□可□行文件支持的功能中。 □在， 如果用□决定从系□中卸□ libfoo □， ports 系□就无法保□□个□用程序免遭破坏了 (因□没有□□ libfoo 的依□□系)。

例 *11.* □理□□□的正□做法

```
.if defined(WITH_FOO)
LIB_DEPENDS+=       foo.0:${PORTSDIR}/devel/foo
CONFIGURE_ARGS+=    --enable-foo
.else
CONFIGURE_ARGS+=    --disable-foo
.endif
```

在第二个例子中， libfoo □被明□禁用。 即使系□中已□安装了□个□， configure 脚本也不会□用相□的功能了。

5.12. 指定工作□□目□

□个 port 都会被解□□到一个工作□□目□中， □个目□必□是可写的。 ports 系□默□情况下会将 DISTFILES 解□□到一个叫做 ${DISTNAME} 的目□中。 □言之， 如果□了：

```
PORTNAME=       foo
PORTVERSION=    1.0
```

□ port 的源□包文件的□□目□将是 foo-1.0。

如果□不是所希望的情形， □可以修改一系列□量的□置。

5.12.1. WRKSRC (□始□□操作的目□名)

□个□量□出了在□用程序的源代□包解□□之后所生成的目□的名字。 如果我□之前的例子解□□生成一个叫做 foo

(而不是 foo-1.0) 的目□， □□：

```
WRKSRC=      ${WRKDIR}/foo
```

或者， 也可能是

```
WRKSRC=      ${WRKDIR}/${PORTNAME}
```

5.12.2. NO_WRKSUBDIR (不需要□□的□□目□)

如果 port 完全不需要写入到某个子目□中， □□□置 NO_WRKSUBDIR 以明示□一点。

```
NO_WRKSUBDIR= yes
```

5.13. □理冲突

□□不同的 package 或 port 之□的冲突情形， 系□提供了不同的□量来□助□□人□□行表□： CONFLICTS、CONFLICTS_INSTALL 和 CONFLICTS_BUILD。

□些用于描述冲突的□量会自□地□置 IGNORE， 后者的完整介□， 可以在 使用 BROKEN、FORBIDDEN 或 IGNORE 阻止用□安装 port □到。

在□去相互冲突的 port □， 建□将 CONFLICTS 保留几个月， 以便□那些不□常更新系□的用□能□看到。

5.13.1. CONFLICTS_INSTALL

如果□的□件包不能与某些□件包同□安装 (例如由于安装同□的文件到相同的位置、 □行□不兼容等等)， □□把其它□件包的名字列在 CONFLICTS_INSTALL □量中。 此□可以使用 shell 通配符， 如 * 和 ?。 列出其它□件包的名字□需要遵循它□在 /var/db/pkg 中出□的□子。 □□保 CONFLICTS_INSTALL 不会匹配到□正制作的□个□□□包的名字， 否□， 使用 FORCE_PKG_REGISTER 来□制安装就没有□法□行了。 □于 CONFLICTS_INSTALL 的□□是在□□□程之后、 安装□始之前□行的。

5.13.2. CONFLICTS_BUILD

如果□的□件包在系□中存在某些其它□件包□不能完成□□， □□把其它□件包的名字列在 CONFLICTS_BUILD □量中。 此□可以使用 shell 通配符， 如 * 和 ?。 列出其它□件包的名字□需要遵循它□在 /var/db/pkg 中出□的□子。 □于 CONFLICTS_BUILD 的□□是在□□□程□始之前□行的。 □□□的冲突不会在□□好的包中予以□□。

5.13.3. CONFLICTS

如果□的 port 在某些其它 port 已□存在的情况下既不能□□， 也不能安装， □□把其它□件包的名字列在 CONFLICTS □量中。 此□可以使用 shell 通配符， 如 * 和 ?。 列出其它□件包的名字□需要遵循它□在 /var/db/pkg 中出□的□子。 □□保 CONFLICTS 不会匹配到□正制作的□个□□□包的名字， 否□， 使用 FORCE_PKG_REGISTER 来□制安装就没有□法□行了。 □于 CONFLICTS 的□□是在□□□程之后、 安装□始之前□行的。

5.14. 安装文件

5.14.1. INSTALL_* 宏

一定要使用由 bsd.port.mk 提供的宏， 以□保在□自己的 *-install target 中能□以正□的属主和□限模式安装文件。

- INSTALL_PROGRAM 是安装可□行二□制文件的命令。
- INSTALL_SCRIPT 是安装可□行脚本文件的命令。
- INSTALL_LIB 是安装□□□接□的命令。
- INSTALL_KLD 是用于安装可加□式内核模□的命令。 在某些平台上， 当□内核模□□行 strip 之后会□致一些□□， 因此□□使用□个宏而不是 INSTALL_PROGRAM 来安装内核模□。
- INSTALL_DATA 是安装可共享数据的命令。
- INSTALL_MAN 是安装□机手册和其他文□的命令 (注意它并不会□行□□操作)。

□些宏展□后基本上都是包含□当参数的 install 命令。

5.14.2. □可□行文件和□□□接□做脱模 (strip) 操作

除非不得不□行， 否□不要手工□可□行文件作脱模操作。 所有文件在安装□都□脱模， 但 INSTALL_PROGRAM 宏会在安装的同□□其□行脱模 (参□下一□的内容)。 INSTALL_LIB 宏

如果□需要□某一文件□行脱模， 但不希望使用 INSTALL_PROGRAM 及 INSTALL_LIB 宏， □□使用 ${STRIP_CMD} 来□理程序。 一般而言□□□在 post-install target 中□行。 例如：

```
post-install:
    ${STRIP_CMD} ${PREFIX}/bin/xdl
```

可以使用 file(1) 命令来□□所安装的可□行文件是否□行□脱模。 如果它没有□出 not stripped 的提示， □表示已□做□脱模了。 □外， strip(1) 不会□已□脱模□的文件重新脱模， 它会直接退出的。

5.14.3. 安装一个目□下的全部文件

有□， 会有需要安装大量的文件， 并保持其□次□□， 例如， 将整个目□□□从 WRKSRC □制到 PREFIX 的目□目□。

□□□□情况， 系□提供了□个宏。 使用□些宏， 而不是直接使用 cp 的□□是它□能□□保目□文件的属主和□限正□。 第一个宏， COPYTREE_BIN 将所有安装的文件□□可□行文件， 因而□合安装文件到 PREFIX/bin。 第二个宏， COPYTREE_SHARE， □不会□置可□行□限， 因此□合于将文件安装到 PREFIX/share 下。

```
post-install:
    ${MKDIR} ${EXAMPLESDIR}
    (cd ${WRKSRC}/examples/ && ${COPYTREE_SHARE} \* ${EXAMPLESDIR})
```

□个例子将原作者提供的整个 examples 目□□制到□ port 指定的安装示□文件的位置。

```
post-install:
    ${MKDIR} ${DATADIR}/summer
    (cd ${WRKSRC}/temperatures/ && ${COPYTREE_SHARE} "June July August"
${DATADIR}/summer/)
```

□个例子将把夏季的三个月的数据， □制到 DATADIR 中的 summer 子目□。

□由□置 COPYTREE_* 宏的第三个参数， □□可以□ find 指定□外的参数。 例如， 如果希望安装除了 Makefile 之外的其他所有文件， 可以使用下述命令。

```
post-install:
    ${MKDIR} ${EXAMPLESDIR}
    (cd ${WRKSRC}/examples/ && \
        ${COPYTREE_SHARE} \* ${EXAMPLESDIR} "! -name Makefile")
```

需要注意的是， □些宏并不能自□将所安装的文件加到 pkg-plist 中， □□是需要自行列出它□。

5.14.4. 安装附加的文□

如果□的□件包含了□准的□机手册和 info 手册以外的文□， 而且□□□它□□用□会有用， □把□些文□安装到 PREFIX/shared/doc 下。 和前面□似， □也可以在 post-install target 中完成。

□□的 port 建立一个新的目□。 □个目□的名字□□反映它是属于□个 port 的。 通常建□使用 PORTNAME。 不□， 如果□□□不同版本的 port 可能会同□安装， 也可以用完整的 PKGNAME。

□外， □□□是否安装取决于□量 NOPORTDOCS 的□置， □□用□就能□在 /etc/make.conf 中禁止安装它。 例如：

```
post-install:
.if !defined(NOPORTDOCS)
    ${MKDIR} ${DOCSDIR}
    ${INSTALL_MAN} ${WRKSRC}/docs/xvdocs.ps ${DOCSDIR}
.endif
```

□里是一些便于使用的□量， 以及它□在 Makefile 中默□的展□方式：

- DATADIR 会展□成 PREFIX/shared/PORTNAME。
- DATADIR_REL 会展□成 share/PORTNAME。
- DOCSDIR 会展□成 PREFIX/shared/doc/PORTNAME。
- DOCSDIR_REL 会展□成 share/doc/PORTNAME。
- EXAMPLESDIR 会展□成 PREFIX/shared/examples/PORTNAME。
- EXAMPLESDIR_REL 会展□成 share/examples/PORTNAME。

NOPORTDOCS 只控制将要安装到 DOCSDIR 的那些文□， 而不□影□□准的□机手册以及 info 手册的安装。 安装到 DATADIR 和 EXAMPLESDIR 的文件□相□地受 NOPORTDATA 和 NOPORTEXAMPLES 控制。

□些□量也会被□出到 PLIST_SUB 中。 只要可能， 它□的□就将在那里以相□于 PREFIX 的路径形式出□。 也就是□， share/doc/PORTNAME 在装箱□中默□情况下会替□掉 %%DOCSDIR%%， 等等。 (更多的 pkg-plist 代□可以在 □里 □到。)

所有非无条件安装的文□文件和目□， 都□在 pkg-plist 出□， 并且使用 %%PORTDOCS%% 前□， 例如：

```
%%PORTDOCS%%%%DOCSDIR%%/AUTHORS
%%PORTDOCS%%%%DOCSDIR%%/CONTACT
%%PORTDOCS%%@dirrm %%DOCSDIR%%
```

如果不希望在 pkg-plist 中逐个列□文□文件， port 也可以将 PORTDOCS □置□一□文件及其 shell glob 模式， 通□□□方式来加入到最□的装箱□中。 □些名字□是相□于 DOCSDIR 的。 因此， 使用了 PORTDOCS， 并将文□安装到非□准位置的 port， □相□地□置 DOCSDIR。 如果有在 PORTDOCS 中列出目□， 或者□一□量中的 glob 模式匹配到了目□， □整个子□中的文件和目□， 都将被注册到最□的装箱□中。 如果定□了 NOPORTDOCS， □ PORTDOCS 中定□的文件和目□将不被安装或加入装箱□。 是否安装文□到前面所□的 PORTDOCS 仍取决于 port 本身。 下面是一个典型的使用 PORTDOCS 的例子：

```
PORTDOCS=       README.* ChangeLog docs/*
```

与 PORTDOCS □似， □□于 DATADIR 和 EXAMPLESDIR 的□量分□是 PORTDATA 和 PORTEXAMPLES。

□也可以使用 pkg-message □个文件， 来在安装□□示一些信息。 参□ □于使用 pkg-message 的□一□ 以了解□一□的□情。 需要□明的是， 并不需要把 pkg-message 加到 pkg-plist 中。

5.14.5. 子目□

尽可能□ port 将它□建的文件， 放置到 PREFIX 中正□的位置。 一些 port 会把各式各□的□西混在一起， 并放到一个同名的目□中， □是不□的。 □外， □多 port 会把除了可□行文件、 □文件和□机手册之外的所有文件， 全都一股□地放到 lib 中， □在和 BSD 配合使用□会有□□。 多数文件， □被放到下列位置之一： etc (安装/配置文件)、 libexec (由系□内部□用的可□行文件)、 sbin (□超□用□/管理□提供的可□行文件)、 info (用于 info □□器的文□) 或 share (平台无□的其它文件)。 □参□ hier(7) 以了解□一□的□情； □□ /usr 的那些□□， 同□也□用于 /usr/local。 例外情况是那些需要和 USENET "news" 打交道的 port， 它□可以□□采用 PREFIX/news 作□文件的目的地。

Chapter 6. 特殊情况

有一些□在□建port□的特殊情况，我□在□里提一下。

6.1. 共享□

如果□的port安装了一个或多个共享□,那□□定□一个 USE_LDCONFIG make □量， 在post-install□□把它注册□共享□ □冲□会□用bsd.port.mk去□行 ${LDCONFIG} -m来指向新□的安装目□。 (通常是 PREFIX/lib) 同□，□也可以□当的在□的 pke-plist文件 中定□一□@exec /sbin/ldconfig -m 和@unexec /sbin/ldconfig -R, 那□用□可以在安装后□上 就能使用，并且在卸□□件包后系□也不会□□□些共享□仍然存在。

```
USE_LDCONFIG= yes
```

如果□需要把共享□安装在缺省的位置之外， 可以通□定□ make □量 USE_LDCONFIG 来改□默□的安装路径，它包含安装共享□的目□列表 例如： 如果□的共享□安装到 PREFIX/lib/foo 和 PREFIX/lib/bar directories目□，□可以在□的 Makefile中□□□置:

```
USE_LDCONFIG= ${PREFIX}/lib/foo ${PREFIX}/lib/bar
```

□□必仔□□□， 通常□是完全不必要的， 或者可以通□ -rpath 或在□接□□置 LD_RUN_PATH 来避免 (参□lang/moscow_ml □出的例子)， 或者用一个 shell 封装程序来在□行可□行文件之前□置 LD_LIBRARY_PATH，□似 www/seamonkey 那□。

当在 64-位系□上安装 32-位 的函数□□， □使用 USE_LDCONFIG32。

尽量将共享□版本号保持□ libfoo.so.0 □□的格式。 我□的□行□境□接器只会□□主 (第一个) 版本数字。

如果在更新 port □升□了其□的主版本号， □其它所有□接了受影□的□的 port 的 PORTREVISION 都□□□， 以□制它□采用新版本的□重新□□。

6.2. Ports 的□行限制

□多□□，并且其中的一些致力于 限制□□的□用程序能被打包， 是否能用于□售□利等等。

做□一名porter□有□□去□□□件的□□ 并且□保FreeBSD □目不必□通□FTP/HTTP 或CD-ROM重新□布源□或□□的二□制而解□ 什□。如果有任何疑□， □□系 FreeBSD ports □件列表。

□于□□情况，就可以□置以下描述 的□量。

6.2.1. NO_PACKAGE (禁止□□□果打包)

□个□量表示我□可能不能生成□个□用 程序的二□制文件。例如，他的□□不允□ 二□制文件的再次□行，或者他可能禁止从 □丁□的源代□打包的□行。

不管□□□，port的 DISTFILES 可以 随意的□像到FTP/HTTP。除非NO_CDROM □量也被□置，□件包也可以□

行在一□CD-ROM （或□似的媒介上）。

NO_PACKAGE也能用在当二□制包 不是非常有用，并且□个□用□件□常要 从源代□□□。例如：当□个□用□件在 □□的□候要在配置信息中指定特定的硬件 代□□，可以□置NO_PACKAGE。

NO_PACKAGE□□□置成字符串 来描述□什□□个□件 不能打包。

6.2.2. NO_CDROM (禁止以 CDROM □行□□□包)

□个□量□□指出□然我□允□ 生成二□制包，但也□我□既不能把□个 □件包也不能把port的DISTFILES 放在光□（或□似的媒介）上□售。但不管□□□， 二□制包和port的DISTFILES 可以从FTP/HTTP上□得。

如果□个□量和 NO_PACKAGE一起被□置， 那□□个port的DISTFIELS 将只能从FTP/HTTP上□得。

NO_CDROM □□被□置成一个字符串 来描述□什□□个port不能重新□布在CD-ROM上。 例如：如果□个port的□□□□是用于"非商□活□" ,那□□个□量就能□置了。

6.2.3. NOFETCHFILES (不自□□取指定的文件)

在 NOFETCHFILES □量中定□的文件， 不会自□从 MASTER_SITES □取。 一□典型的用例是， 使用来自某个□件供□商提供的 CD-ROM 上的文件。

用于□□在 MASTER_SITES 上是否包含了所需文件的工具， □忽略□些文件， 而不是□告它□不存在。

6.2.4. RESTRICTED (禁止任何形式的再分□)

如果□用程序既不允□□像其 DISTFILES， 也不允□□布其□□□版本的包， □置它就可以了。

NO_CDROM 或 NO_PACKAGE 不□与 RESTRICTED 同□□置， 因□它包含了□些情形。

RESTRICTED □□置□一个□明 port □何不能□布的串。 典型情况可能是由于 port 包含了□有的□件， 因而用□需要自行下□ DISTFILES， 可能是注册或者同意某一 EULA 的条款。

6.2.5. RESTRICTED_FILES (禁止某些文件的再分□)

当□置了 RESTRICTED 或 NO_CDROM □， □个□量会默□□置□ ${DISTFILES} ${PATCHFILES}， 否□它会□空。如果只有某些源□包文件是受限的， □可以用□个□量来指明它□。

注意， port committer □□在 /usr/ports/LEGAL 中□□一个源□包文件撰写□□的□目， 并介□□些限制的原因。

6.3. □□机制

6.3.1. Ports 的并行□□

FreeBSD ports 框架支持使用多个 make 子□程来□行并行□□， 在 SMP 上□可以全面地利用系□的 CPU □算能力， 令 port 的□□□程更快、 更有效率。

目前□是通□向原作者的代□□□ make(1) 参数 -jX 来□□的。 □憾的是， 并不是所有的 port 都能□很好地□理□个□□。 因此， 必□通□明□地在 Makefile 中指定 MAKE_JOBS_SAFE=yes 来□用□一功能。

从 port □□人的角度□有一个控制的方法是□置 MAKE_JOBS_UNSAFE=yes □量。 □个□量主要是用于已知不能与 -jX 配合使用的 port， 即使用□在 /etc/make.conf 中定□了 FORCE_MAKE_JOBS=yes □量， 系□也不会使用并行□□。

6.3.2. make、 gmake， 以及 imake

如果 port 用到了 GNU make， □□置 USE_GMAKE=yes。

表 4. 与 *gmake* 有□的 *port* □量

□量	意□
USE_GMAKE	此 port 需要使用 gmake 来完成□□□程。
GMAKE	不在 PATH 中□， gmake 的完整路径。

□于 X □用程序的 port， 如果它使用 imake 根据 Imakefile 文件来生成 Makefile， □□□置 USE_IMAKE=yes。 □会使□□□程中的配置 (configure) □段自□□行 xmkmf -a。 如果 -a □志会□□的 port □来麻□， □需□置 XMKMF=xmkmf。 如果 port 用到了 imake 但并不使用 install.man target， □□□置 NO_INSTALL_MANPAGES=yes。

如果 port 源文件的 Makefile 的主□□ target 是 all 以外的名字， □□□地□置 ALL_TARGET。 □于 install 而言， □□的□量是 INSTALL_TARGET。

6.3.3. configure 脚本

假如 port 使用 configure 脚本来从 Makefile.in 生成 Makefile 文件， 需要□置 GNU_CONFIGURE=yes。 如果希望□□外的参数□ configure 脚本 (默□参数□ --prefix=${PREFIX} --infodir=${PREFIX}/${INFO_PATH} --mandir=${MANPREFIX}/man --build=${CONFIGURE_TARGET})， □通□ CONFIGURE_ARGS 来指定□些参数。 □似地， 可以通□ CONFIGURE_ENV □量来□□一些□境□量。

如果□的□件包使用 GNU configure， 而生成的可□行文件命名方式 "怪□" 如 i386-portbld-freebsd4.7- □用程序名， □需要更□一□地通□改□ CONFIGURE_TARGET □量来按照□新版本的 autoconf 生成的脚本所希望的方式指定 target。 其方法是， □随 Makefile 中 GNU_CONFIGURE=yes 一行之后加入：

```
CONFIGURE_TARGET=--build=${MACHINE_ARCH}-portbld-freebsd${OSREL}
```

表 5. 用于用到了 configure 脚本的 *port* 的□量

□量	意□
GNU_CONFIGURE	此 port 需要用 configure 脚本来准□□□。
HAS_CONFIGURE	与 GNU_CONFIGURE □似， 但默□的 configure target 并不加入 CONFIGURE_ARGS。
CONFIGURE_ARGS	希望□□ configure 脚本的□外参数。
CONFIGURE_ENV	希望在□行 configure 脚本□□置的□境□量。
CONFIGURE_TARGET	替□默□的 configure target。 其默□□是 ${MACHINE_ARCH}-portbld-freebsd${OSREL}。

6.3.4. 使用 scons

如果的 port 使用 SCons， 就需要定 USE_SCONS=yes 了。

表 6. 使用 scons 的 *port* 会用到的量

量	含
SCONS_ARGS	当前 port 希望 SCons 境的参数。
SCONS_BUILDENV	希望在系境中置的量。
SCONS_ENV	希望在 SCons 境中置的量。
SCONS_TARGET	SCons 的最后一个参数， 似于 MAKE_TARGET。

如果希望第三方的 SConstruct 尊重通 SCONS_ENV (其中最重要的是 CC/CXX/CFLAGS/CXXFLAGS 配置) Scons 的配置， 需要 SConstruct 行修改， 使的 Environment 按下列方式建立：

```
env = Environment(**ARGUMENTS)
```

其后， 可以通 env.Append 和 env.Replace 来它行修改。

6.4. 利用 GNU autotools

6.4.1. 入

多 GNU autotools 提供了一在多重操作系和机器架之上件的抽象机制。 在 Ports Collection 中， port 可以通的方法来使用些工具：

```
USE_AUTOTOOLS= 工具:版本[:操作] ...
```

撰写本， 工具 可以置 libtool、 libltdl、 autoconf、 autoheader、 automake 或 aclocal 之一。

版本 用来指定希望使用的工具的特定版本 (参 devel/{automake,autoconf,libtool}[0-9]+ 以了解有效的版本号)。

操作 是一个可的展， 用于修改如何使用工具。

可以同指定多个不同的工具， 可以在一行中指定， 也可以用 Makefile 的 += 。

最后， 可以使用一个特殊的名 autotools 的工具， 它会安装全部可用的 autotools 版本， 以跨平台 的需要。 可以通安装 devel/autotools port 来到一目的。

6.4.2. libtool

使用 GNU 框架的共享通常会使用 libtool 来整共享的和安装， 以便与所行的操作系相匹配。 通常的做法是使用用程序所附的 libtool 副本。 如果需要使用外部的 libtool， 可以使用 Ports 套件提供的版本：

```
USE_AUTOTOOLS= libtool:版本[:env]
```

如果不使用□外的操作符， libtool:版本 表示希望□□框架使用 configure 脚本来□系□所安装的 libtool □行修□。 □会暗含地定□ GNU_CONFIGURE。 更□一□， □□框架□会□置一系列 make 和 shell □量用于 port 后□的操作。 □参□ bsd.autotools.mk 了解□一□的□情。

如果指定了 :env 操作符， □表示只□置□境， 而跳□其他的操作。

最后， LIBTOOLFLAGS 和 LIBTOOLFILES 可以用来替□最常修改的参数， 以及将被 libtool 修□的文件。 多数 port 不需要□□做。 □参□ bsd.autotools.mk 以了解□一□的□□。

6.4.3. libltdl

一些 ports 会使用 libltdl □， 后者是 libtool □件包的一部分。 使用□个□并不意味着必□使用 libtool 本身， 因此提供了□一□□□。

```
USE_AUTOTOOLS= libltdl:版本
```

目前， □一□置所做的全部工作是将 LIB_DEPENDS □置□□当的 libltdl port， 并作□一□方便的功能， □助□□人□消除在 USE_AUTOTOOLS 框架以外的， □于 autotools port 的依□。 □个工具并不提供其它的操作符。

6.4.4. autoconf 和 autoheader

某些 port 并没有直接提供 configure 脚本， 但包含了作□ autoconf 模板的 configure.ac 文件。 可以用下列□置来要求 autoconf □建 configure 脚本， 并使用 autoheader 来□ configure 脚本□建模板□文件。

```
USE_AUTOTOOLS=  autoconf:版本[:env]
```

以及

```
USE_AUTOTOOLS=  autoheader:版本
```

上述□置会暗含使用 autoconf:版本。

□于 libtool， □置与前面□似。 如果指定可□的 :env 操作符， □表示只□置用于后□工作的□境。 如果不指定， □会□ port □行相□的修□和重新配置。

其它的可□□量， 如 AUTOCONF_ARGS 和 AUTOHEADER_ARGS 可以通□ port 的 Makefile 来□式地指定替□。 □似 libtool， 多数 port 并不需要□□做。

6.4.5. automake 和 aclocal

某些□件包只提供了 Makefile.am 文件。 □些文件必□首先用 automake □□□ Makefile.in 并使用 configure 来生成□□的 Makefile。

□似地， 偶□会有一些□件包不提供□□所需的 aclocal.m4 文件。 □些文件可以通□使用 aclocal 来□描 configure.ac 或 configure.in 自□生成。

aclocal 与 automake 有和 autoheader 与 autoconf 在前面一□中所介□的相□似的□系。 aclocal 会暗含使用 automake， 因此：

```
USE_AUTOTOOLS=  automake:版本[:env]
```

和

```
USE_AUTOTOOLS=  aclocal:版本
```

也自□暗含使用 automake:版本。

与 libtool □似， autoconf 如果使用了可□的 :env 操作符表示□□□置用于后□使用的□境， 如果不□置， □会□ port □行重新配置。

□于 autoconf 和 autoheader 而言， automake 和 aclocal 提供了□□的可□参数□量 AUTOMAKE_ARGS 和 ACLOCAL_ARGS， 如果需要的□， 可以在 port 的 Makefile 中指定。

6.5. 使用 GNU gettext

6.5.1. 基本用法

如果□的 port 需要使用 gettext， 只要将 USE_GETTEXT □置□ yes， □的 port 就会□加□ devel/gettext 的依□。 USE_GETTEXT 也可以指定□所需的 libintl □的版本， 它是 gettext 的基本□成部分， 尽管如此， □烈建□□不要 使用□个功能： □的 port □能与目前版本的 devel/gettext 配合工作。

在 port 中相当常□的情况下， 会需要同□使用 gettext 和 configure。 一般而言， GNU configure 能□自□定位到 gettext。 如果它没有成功地完成□□工作， □可以通□□似下面□□的 CPPFLAGS 和 LDFLAGS 将 gettext 的位置告□它：

```
USE_GETTEXT=    yes
CPPFLAGS+=      -I${LOCALBASE}/include
LDFLAGS+=       -L${LOCALBASE}/lib

GNU_CONFIGURE=  yes
CONFIGURE_ENV=  CPPFLAGS="${CPPFLAGS}" \
                LDFLAGS="${LDFLAGS}"
```

当然， 不需要□参数□ configure □， 代□可以更□□□：

```
USE_GETTEXT=    yes
GNU_CONFIGURE=  yes
CONFIGURE_ENV=  CPPFLAGS="-I${LOCALBASE}/include" \
                LDFLAGS="-L${LOCALBASE}/lib"
```

6.5.2. 可□用法

一些□件□品提供了禁用 NLS 的能力， 例如， 在 configure □， 指定 --disable-nls 参数。 如果□ port 的□件支持□□配置， □□根据 WITHOUT_NLS 的□置来有条件地使用 gettext。 □于比□□□和不太□□的 port，□可以使用下列□□：

```
GNU_CONFIGURE=          yes

.if !defined(WITHOUT_NLS)
USE_GETTEXT=            yes
PLIST_SUB+=             NLS=""
.else
CONFIGURE_ARGS+=        --disable-nls
PLIST_SUB+=             NLS="@comment "
.endif
```

□要做的下一件事是合理地安排装箱□文件， 使其根据用□配置来决定是否将消息□□ (message catalog) 文件放入最□的装箱□。 前面已□介□了在 Makefile 中所需的写法， □□做法在 高□ pkg-plist 用法 □□中□行了介□。 □□地□， 在 pkg-plist 中出□的 %%NLS%% 均会在禁用 NLS □自□替□□ “`@comment `”， 反之□替□□空串。 □□， 在最□的装箱□中□ %%NLS%% 的行， 在 NLS □□的情况下就会□□注□， 反之， □些前□就会自□□掉。 □在需要做的事情就是把 %%NLS%% □到 pkg-plist 中的消息□□文件的那些行□□， 例如：

```
%%NLS%%share/locale/fr/LC_MESSAGES/foobar.mo
%%NLS%%share/locale/no/LC_MESSAGES/foobar.mo
```

在比□□□的情形中， □可能需要使用更高□的技□， 例如 □□生成装箱□ 等。

6.5.3. □理消息□□目□

在安装消息□□文件□□有一个需要注意的地方。 □些文件会放到 LOCALBASE/shared/locale 下与□言□□的目□中， □些目□一般□的 port 不需要□建和□除。 最常用的□言的目□已□在 /etc/mtree/BSD.local.dist 中列出；也就是□， 它□是基本系□的一部分。 其他一些□言的目□， □由 devel/gettext 控制。 □最好□看一下 pkg-plist， 以□定是否正在安装某□不常用□言的文件。

6.6. 使用 perl

如果 MASTER_SITES □□ MASTER_SITE_PERL_CPAN， □□尽量把 MASTER_SITE_SUBDIR □置□□□目□的名字。 例如，□ p5-Module-Name 而言推□的名字是 Module。 □可以在 cpan.org □到□□目□的名字。 □可以□保在模□的作者□生□化□， 保持 port □□可用。

以上□□有一个例外， 即□□目□不存在或源□包不在那个目□中□， 允□使用作者的 id 作□ MASTER_SITE_SUBDIR。

所有□些□□均同□接受 YES 和版本串， □似 5.8.0+ □□的写法。 使用 YES 表示 port 能□配合所有受支持的 Perl 版本来使用。 如果 port 只能配合特定版本的 Perl 来使用， □可以用版本串来表示， 例如最低版本 (如 5.7.3+)、 最高版本 (如 5.8.0-) 或某个具体的版本 (如 5.8.3)。

表 7. 用于用到 perl 的 *port* 的□量

□量	意□
USE_PERL5	表示 port 将 perl 5 用于□□和□行。
USE_PERL5_BUILD	表示 port 将 perl 5 用于□□。
USE_PERL5_RUN	表示 port 将 perl 5 用于□行。
PERL	perl 5 的完整路径， 可能是系□自□的， 或者从 port 安装， 但没有版本号。 如果□需要在脚本中替□ “#!” 行， □□使用□个□量。
PERL_CONFIGURE	采用 Perl 的 MakeMaker □行配置。 □一□量□含□置 USE_PERL5。
PERL_MODBUILD	使用 Module::Build □行配置、 □□并安装。 □一□量□含□置 PERL_CONFIGURE。

Perl 模□通常并没有官方网站， □些 port □将 cpan.org 作□其 pkg-descr WWW 行的内容。 推□的 URL 格式□ http://search.cpan.org/dist/Module-Name/ (保留最后的斜□)。

6.7. 使用 X11

6.7.1. X.Org □件

在 Ports 套件中提供的 X11 □□是 X.Org。 如果□的□用程序用到了 X □件， □□将 USE_XORG □□所需要的那些□件。 目前可用的□件包括：

bigreqsproto compositeproto damageproto dmx dmxproto evieproto fixesproto fontcacheproto fontenc fontsproto fontutil glproto ice inputproto kbproto libfs oldx printproto randrproto recordproto renderproto resourceproto scrnsaverproto sm trapproto videoproto x11 xau xaw xaw6 xaw7 xaw8 xbitmaps xcmiscproto xcomposite xcursor xdamage xdmcp xevie xext xextproto xf86bigfontproto xf86dgaproto xf86driproto xf86miscproto xf86rushproto xf86vidmodeproto xfixes xfont xfontcache xft xi xinerama xineramaproto xkbfile xkbui xmu xmuu xorg-server xp xpm xprintapputil xprintutil xpr oto xproxymngproto xrandr xrender xres xscrnsaver xt xtrans xtrap xtst xv xvmc xxf86dga xxf86misc xxf86vm.

最新的列表， 可以在 /usr/ports/Mk/bsd.xorg.mk 中□到。

The Mesa Project 是一个致力于自由的 OpenGL □□的□□。 □可以使用 USE_GL □量来□ port 依□其不同的□件。 可用的□□包括： glut, glu, glw, glew, gl 和 linux。 □了□□向前兼容， 当使用 yes □系□会自□将其映射□ glu。

例 12. 使用 *USE_XORG* 的例子

```
USE_XORG=   xrender xft xkbfile xt xaw
USE_GL=     glu
```

□多 ports 会定□ USE_XLIB， □会□致 port 依□ 50 多个□□□接□。 由于它出□于 X.org 模□化之前， 因此□个□量□□向前兼容的原因提供， 新的 port 不□再使用它。

表 8. 用到 *X* 的 *port* 可以使用的□量

USE_XLIB	此 port 用到了 X □。 已□□ - □□使用 USE_XORG □量列出用到的 X.Org □件， 而不是使用□个□量。
USE_IMAKE	此 port 用到了 imake。
USE_X_PREFIX	已□□。 目前其作用与 USE_XLIB 相同，并可以直接用后者替□。
XMKMF	□置□ xmkmf 的完整路径名， 如果它不在 PATH 中的□。 默□□是 xmkmf -a。

表 9. 用于表示□ *X11* 某些□件的依□□系的□量

X_IMAKE_PORT	用以提供 imake 以及□多其它用于□□ X11 的工具的 port。
X_LIBRARIES_PORT	用以提供 X11 □的 port。
X_CLIENTS_PORT	用以提供 X 客□的 port。
X_SERVER_PORT	用以提供 X 服□的 port。
X_FONTSERVER_PORT	用以提供字体服□的 port。
X_PRINTSERVER_PORT	用以提供打印服□的 port。
X_VFBSERVER_PORT	用以提供在虚□□□存服□(virtual framebuffer server) 的 port。
X_NESTSERVER_PORT	用以提供嵌套 X 服□的 port。
X_FONTS_ENCODINGS_PORT	用以□字体提供□□的 port。
X_FONTS_MISC_PORT	用以提供多□位□字体的 port。
X_FONTS_100DPI_PORT	用以提供 100dpi 位□字体的 port。
X_FONTS_75DPI_PORT	用以提供 75dpi 位□字体的 port。
X_FONTS_CYRILLIC_PORT	用以提供西里□位□字体的 port。
X_FONTS_TTF_PORT	用以提供 TrueType® 字体的 port。
X_FONTS_TYPE1_PORT	用以提供 Type1 字体的 port。
X_MANUALS_PORT	用以提供面向□□人□的□机手册的 port。

例 13. 在□量中使用与 X11 有□的□量

```
# 使用某些 X11 □并依□字体服□和西里□字体。
RUN_DEPENDS=    ${LOCALBASE}/bin/xfs:${X_FONTSERVER_PORT} \

${LOCALASE}/lib/X11/fonts/cyrillic/crox1c.pcf.gz:${X_FONTS_CYRILLIC_PORT}

USE_XORG=       x11 xpm
```

6.7.2. 需要使用 Motif 的 port

如果□的 port 需要 Motif □， □□在 Makefile 中定□ USE_MOTIF。 默□的 Motif □□是 x11-toolkits/open-motif。 用□可以通□□置 WANT_LESSTIF □量来□□ x11-toolkits/lesstif 代替它。

bsd.port.mk 会将 MOTIFLIB □量□置□到合□的 Motif □的引用。 □使用□丁将□ port 中 Makefile 或 Imakefile 提到 Motif □的地方改□ ${MOTIFLIB}。

有□□比□常□的情况：

- 如果 port 中将 Motif 在其 Makefile 或 Imakefile 表□□ -lXm， □□□地将其替□□ ${MOTIFLIB}。
- 如果 port 在其 Imakefile 中使用 XmClientLibs， □将其改□ ${MOTIFLIB} ${XTOOLLIB} ${XLIB}.

注意 MOTIFLIB (通常) 会展□□ -L/usr/X11R6/lib -lXm 或 /usr/X11R6/lib/libXm.a， 所以不需要在其前加入 -L 或 -l。

6.7.3. X11 字体

如果 port 将□ X Window 系□安装字体， 将□些字体放到 LOCALBASE/lib/X11/fonts/local。

6.7.4. 通□ Xvfb 来□得虚□的 DISPLAY

某些□用程序必□在有可用的 X11 □示的□候才能成功□□。 当□□的机器没有控制台□， □会□来□□。 □了解决□个□□， 如果定□了□当的□量， □□基□□施会□□采用虚□□□存的 X server。 此□， □□□程中将会□出可用的 DISPLAY。

```
USE_DISPLAY=  yes
```

6.7.5. □面□

通□利用 DESKTOP_ENTRIES □量， 可以很容易地在□的 port 中□建□面□ (Freedesktop □准)。 □些□会在□似 GNOME 或 KDE □□的符合□一□准的□面□境中□示在□用程序菜□中。 □□做会自□□建、 安装 .desktop 文件，并将其加入 pkg-plist。 其□法□：

```
DESKTOP_ENTRIES=  "NAME" "COMMENT" "ICON" "COMMAND" "CATEGORY" StartupNotify
```

□可以在 Freedesktop 网站上 □到可用的分□名称。 StartupNotify 表示□用程序在支持□□通知的□境中清除状□信息。

例子：

```
DESKTOP_ENTRIES=  "ToME" "Roguelike game based on JRR Tolkien's work" \
                  "${DATADIR}/xtra/graf/tome-128.png" \
                  "tome -v -g" "Application;Game;RolePlaying;" \
                  false
```

6.8. 使用 GNOME

FreeBSD/GNOME □目□使用一□自己的□量来定□ port 所使用的 GNOME □件。 □些□量的□□列表 可以在 FreeBSD/GNOME □目的主□□到。

6.9. 使用 Qt

6.9.1. 在 port 中使用 Qt

表 *10.* 用于使用 *Qt* 的 *port* 的□量

USE_QT_VER	表示 port 用到了 Qt 工具套件。 可用的□包括 3 和 4； 用于指定使用的 Qt 的主版本。 此外， 系□会自□□ configure 脚本和 make 命令提供必要的参数。
QT_PREFIX	□个□量会自□□□ Qt 的安装路径 (只□□量)。
MOC	□个□量会自□□□ moc 的路径 (只□□量)。 默□□与 USE_QT_VER □量的□有□。
QTCPPFLAGS	通□ CONFIGURE_ENV □□ Qt 工具套件的□□参数。 默□配置与 USE_QT_VER 有□。
QTCFGLIBS	通□ CONFIGURE_ENV □□ Qt 工具套件的□接□。 默□配置与 USE_QT_VER 有□。
QTNONSTANDARD	禁止系□自□修改 CONFIGURE_ENV、 CONFIGURE_ARGS 和 MAKE_ENV。

表 *11.* 其他用于使用 *Qt 4.x* 的□量

QT_COMPONENTS	用于指定 Qt4 工具和函数□的依□。 □情□后。
UIC	□个□量会自□□□ uic 的路径 (只□□量)。 默□□与 USE_QT_VER 有□。
QMAKE	□个□量会自□□□ qmake 的路径 (只□□量)。 其默□□与 USE_QT_VER 有□。
QMAKESPEC	□个□量会自□□□ qmake 配置文件的路径 (只□□量)。 其默□□与 USE_QT_VER 有□。

当□置了 USE_QT_VER □， 系□自□会□ configure 脚本□一系列有用的参数：

```
CONFIGURE_ARGS+= --with-qt-includes=${QT_PREFIX}/include \
                 --with-qt-libraries=${QT_PREFIX}/lib \
                 --with-extra-libs=${LOCALBASE}/lib \
                 --with-extra-includes=${LOCALBASE}/include
CONFIGURE_ENV+=  MOC="${MOC}" CPPFLAGS="${CPPFLAGS} ${QTCPPFLAGS}" LIBS="${QTCFGLIBS}"
\
                 QTDIR="${QT_PREFIX}" KDEDIR="${KDE_PREFIX}"
```

如果将 USE_QT_VER □□ 4， □□会□行下列配置：

```
CONFIGURE_ENV+= UIC="${UIC}" QMAKE="${QMAKE}" QMAKESPEC="${QMAKESPEC}"
MAKE_ENV+=      QMAKESPEC="${QMAKESPEC}"
```

6.9.2. □件的□□ (□限 Qt 4.x)

当把 USE_QT_VER □□ 4 □， 就可以通□ QT_COMPONENTS □量来指定□ Qt4 工具和函数□的依□了。 通□在□件的名称后面添加 _build 或 _run □□的后□， □可相□地将□依□□系限于□□或□行□刻。 在没有指定后□□， 系□默□在□□和□行□刻均依□□□件。 通常情况下在指明函数□一□的□件□□不使用后□， □□工具□□件□使用 _build 后□， 而□件□□件， □□使用 _run 后□。 下表中列出了一些最常用的□件 (全部可用的□件， □在 /usr/ports/Mk/bsd.qt.mk 中的 _QT_COMPONENTS_ALL 列出)：

表 *12.* 可用的 *Qt4* 函数□□件

名字	描述
corelib	核心□ (在 port 只使用 corelib 而没有用到其他□□可以省略)
gui	□形用□界面□
network	网□函数□
opengl	OpenGL 函数□
qt3support	Qt3 兼容支持函数□
qtestlib	□元□□函数□
script	脚本函数□
sql	SQL 函数□
xml	XML 函数□

□可以通□在成功□□之后， 通□在主可□行文件上□行 ldd 来□定所需的□。

表 *13.* 可用的 *Qt4* 工具□件

名字	描述
moc	元□象□□器 (几乎所有的 Qt □用程序在□□□程中都需要它)

名字	描述
qmake	Makefile 生成器 / □□工具
rcc	□源□□器 (如果□用程序中包含 **.rc** 或 .qrc 文件，就需要它)
uic	用□界面□□器 (如果□用程序中包含使用 Qt Designer □建的 *.ui 文件□就需要它 - 一般□来 Qt □用程序都会使用 GUI 的)

表 *14.* 可用的 *Qt4* □件□件

名字	描述
iconengines	SVG □□引□□件 (如果□用程序使用 SVG □□)
imageformats	用于 GIF、 JPEG、 MNG 和 SVG 的 imageformat □件 (如果□用程序使用□片文件)

例 *14.* □□ *Qt4* □件

在□个例子中， 我□将要移植的□用程序用到了 Qt4 □形用□界面函数□、 Qt4 核心 (core) 函数□、 所有 Qt4 代□生成工具以及 Qt4 的 Makefile 生成器。 由于 gui 函数□会自□附□□核心函数□的依□，因此并不需要明□指出需要 corelib 的依□□系。 Qt4 代□生成工具 moc、 uic 和 rcc 以及 Makefile 生成器 qmake 只在□□□程中才会用到， 因此可以指定 `_build` 后□：

```
USE_QT_VER=    4
QT_COMPONENTS= gui moc_build qmake_build rcc_build uic_build
```

6.9.3. 其他考□

如果□用程序没有提供 configure 文件， 而是□了一个 .pro 文件， □□□□：

```
HAS_CONFIGURE=  yes

do-configure:
        @cd ${WRKSRC} && ${SETENV} ${CONFIGURE_ENV} \
                ${QMAKE} -unix PREFIX=${PREFIX} texmaker.pro
```

□注意， □与系□提供的 BUILD.sh 中的 `qmake` □似。 □□ `CONFIGURE_ENV` 能□□保 `qmake` 可以看到 `QMAKESPEC` □量， 否□它可能无法正常工作。 `qmake` 会生成□准的 Makefile， 因此无需自行□写 `build` target。

Qt □用程序通常会□写□能□跨平台使用， 通常 X11/Unix 并不是□□它的平台， 有□□会□致一些□□角角的□□，例如：

- 缺少必要的 *includepaths*。 □多□用程序会使用托□□□支持， 但忽略了□些□或□文件需要在 X11 目□中□□。 □可以通□命令行告□ `qmake` 将□些□文件和函数□加入到搜索路径中， 例如：

```
${QMAKE} -unix PREFIX=${PREFIX} INCLUDEPATH+=${LOCALBASE}/include \
    LIBS+=-L${LOCALBASE}/lib sillyapp.pro
```

- 有□□的安装路径。 有□， □似□□或 .desktop 文件□□的一些数据， 默□情况下没有安装到 XDG-兼容的程序会□描的路径中。 editors/texmaker 就是一个□□的例子 - □参考□个 port 的 files 目□中的 patch-texmaker.pro， 以了解如何在 Qmake 工程文件中修正□个□□。

6.10. 使用 KDE

6.10.1. □量定□ (只用于 KDE 3.x)

表 *15.* 用于使用 *KDE 3.x* 的 *port* 的□量

USE_KDELIBS_VER	表示 port 用到了 KDE □。 □个□量可以指定希望使用的 KDE 主版本号， 如果□置了□个□量， □系□也会将 USE_QT_VER □□□当的版本。 □□量目前唯一有效的□是 3。
USE_KDEBASE_VER	表示 port 用到了 KDE 的基本系□。 □个□量可以指定希望使用的 KDE 主版本号， 如果□置了□个□量， □系□也会将 USE_QT_VER □□□当的版本。 □□量目前唯一有效的□是 3。

6.10.2. 用于 KDE 4 的□量定□

如果□的□用程序需要使用 KDE 4.x， □□将 USE_KDE4 □□所需□件的列表。 下面列出一些最常用到的□件 (最新的□件列表位于 /usr/ports/Mk/bsd.kde4.mk 中的 _USE_KDE4_ALL)：

表 *16.* 可用的 *KDE4* □件

名称	□明
akonadi	个人信息管理 (PIM)存□服□
automoc4	令 port 使用 automoc4 □□工具集
kdebase	基本的 KDE □用程序 (Konqueror、 Dolphin、 Konsole)
kdeexp	□□性的 KDE □ (包含尚未完全□定不□的 API)
kdehier	常用的 KDE 目□□次□□
kdelibs	基本 KDE □
kdeprefix	如果□置了□个□□， □ port 将安装到 ${KDE4_PREFIX} 而不是 ${LOCALBASE}
pimlibs	PIM 函数□
workspace	用于□成□面的□用程序和函数□ (Plasma、 KWin)

KDE 4.x port 会安装到 ${KDE4_PREFIX}， 目前是 /usr/local/kde4， 以避免与 KDE 3.x ports 冲突。 □是通□指定 kdeprefix □件来□□的， 它表示替□默□的 PREFIX。 不□， port 仍会遵循通□ MAKEFLAGS □境□量□置的

PREFIX 以及其它 make 参数。

KDE 4.x ports 有可能和 KDE 3.x ports 冲突， 因此如果□用了 kdeprefix □件， 它□会安装到 ${KDE4_PREFIX}。 目前 KDE4_PREFIX 的默□□是 /usr/local/kde4。 也可以将 KDE 4.x ports 安装到自定□的 PREFIX。 当 PREFIX 是通□ MAKEFLAGS □境□量， 或直接在 make 命令行指定□， 它会替□ kdeprefix 提供的配置。

例 *15.* USE_KDE4 示例

下面是一个□□的 KDE 4 port。 USE_CMAKE 指定 port 使用 CMake - □多 KDE 4 □目所使用的配置工具。 USE_KDE4 □引入 KDE 函数□， 并令 port 在□□□段使用 automoc4。 需要的 KDE □件， 以及其他依□的 □件可以从 configure 的日志中□知。 USE_KDE4 并不会自□□置 USE_QT_VER。 如果 port 需要使用某些 Qt4 □件， □需要□置 USE_QT_VER 并指定所需要的□件。

```
USE_CMAKE=      yes
USE_KDE4=       automoc4 kdelibs kdeprefix
USE_QT_VER=     4
QT_COMPONENTS= qmake_build moc_build rcc_build uic_build
```

6.11. 使用 Java

6.11.1. □量定□

如果□的 port 需要 Java™ □□包 (JDK™) 来完成□□、 支持□行， 甚至完成解□源代□包□□的工作， 就□□定□ USE_JAVA。

在 Ports Collection 中有□多不同的 JDK， 它□的版本各不相同， 或是来自不同的供□商。 如果□的 port 必 □使用其中的某个特定的版本， 也可以予以定□。 最新的□定版本是 java/jdk16。

表 *17.* 用到 *Java* 的 *port* 可以使用的□量

□量	意□
USE_JAVA	只有定□它才能使其它□量生效。
JAVA_VERSION	用空格分□的□合 port 使用的 Java 版本。 可□的 "+" 可以用于指定某个□□的版本 (可以用： 1.5[+] 1.6[+] 1.7[+])。
JAVA_OS	用空格分□的□□ port 的 JDK port 操作系□□型 (可以用： native linux)。
JAVA_VENDOR	用空格分□的□□ port 的 JDK port 供□商 (可以用： freebsd bsdjava sun openjdk)。
JAVA_BUILD	□置□个□量表示所□的 JDK port □被列入 port 的□□ 依□□系。
JAVA_RUN	□置□个□量表示所□的 JDK port □被列入 port 的□行 □境依□□系。

□量	意□
JAVA_EXTRACT	□置□个□量表示所□的 JDK port □被列入 port 的解□□支持依□□系。

下面是在□置了 USE_JAVA 之后， port 能□从系□中□得的配置：

表 *18.* 向使用了 *Java* 的 *port* 提供的□量

□量	□
JAVA_PORT	JDK port 的名字 (例如 'java/diablo-jdk16')。
JAVA_PORT_VERSION	JDK port 的完整版本 (例如 '1.6.0')。 如果□只需要版本号的前□位， □可用 ${JAVA_PORT_VERSION:C/^([0-9])\.([0-9])(.*)$/\1.\2/}。
JAVA_PORT_OS	所用 JDK port 的操作系□ (例如 'native')。
JAVA_PORT_VENDOR	所用 JDK port 的供□商 (例如 'freebsd')。
JAVA_PORT_OS_DESCRIPTION	所用 JDK port 操作系□的描述 (例如 'Native')。
JAVA_PORT_VENDOR_DESCRIPTION	所用 JDK port 供□商的描述 (例如 'FreeBSD Foundation')。
JAVA_HOME	JDK 的安装目□ (例如 '/usr/local/diablo-jdk1.6.0')。
JAVAC	所用 Java □□器的完整路径 (例如 '/usr/local/diablo-jdk1.6.0/bin/javac')。
JAR	所用 jar 工具的完整路径 (例如 '/usr/local/diablo-jdk1.6.0/bin/jar' 或 '/usr/local/bin/fastjar')。
APPLETVIEWER	所用 appletviewer 工具的完整路径 (例如 '/usr/local/diablo-jdk1.6.0/bin/appletviewer')。
JAVA	所用 java □行文件的完整路径。 □□使用它来□行 Java 程序 (例如 '/usr/local/diablo-jdk1.6.0/bin/java')。
JAVADOC	所用 javadoc 工具的完整路径。
JAVAH	所用 javah 程序的完整路径。
JAVAP	所用 javap 程序的完整路径。
JAVA_KEYTOOL	所用 keytool 工具的完整路径。
JAVA_N2A	所用 native2ascii 工具的完整路径。
JAVA_POLICYTOOL	所用 policytool 程序的完整路径。
JAVA_SERIALVER	所用 serialver 程序的完整路径。
RMIC	所用 RMI □/架 生成器， rmic 的完整路径。
RMIREGISTRY	所用 RMI 注册表程序， rmiregistry 的完整路径。
RMID	所用 RMI 服□程序 rmid 的完整路径。
JAVA_CLASSES	所用 JDK □文件目□的完整路径。 ${JAVA_HOME}/jre/lib/rt.jar。

□可以使用 java-debug make target 以□取用于□□ port 的信息。 大多数前述□量的□皆会予以呈□。

此外， □会定□下述常量， 以□保所有的 Java port 均以一致之方式安装：

表 19. □使用 *Java* 的 *port* 定□的常量

常量	□
JAVASHAREDIR	所有 Java 相□□料的安装根目□。 默□□： ${PREFIX}/shared/java.
JAVAJARDIR	用以安装 JAR 文件的目□。 默□□： ${JAVASHAREDIR}/classes。
JAVALIBDIR	其它 port 安装的 JAR 文件所在的目□。 默□□： ${LOCALBASE}/shared/java/classes。

相□的□也会定□在 PLIST_SUB (在 根据 make □量□ pkg-plist □行修改 中□行介□) 和 SUB_LIST 中。

6.11.2. 采用 Ant □行□□

如果 port 采用 Apache Ant □行□□， □需要定□ USE_ANT。 如是， □ Ant 将作□ 子-make 命令来使用。 如果 port 未定□ do-build target， □将默□依 MAKE_ENV、 MAKE_ARGS 和 ALL_TARGET。 的□置□行 Ant。 □□似于 □□机制 中介□的□于 USE_GMAKE 的机制。

6.11.3. 最佳□践

如果□正移植某个 Java □， □的 port □把 JAR 文件安装到 ${JAVAJARDIR}， 而其它文件□□放在 ${JAVASHAREDIR}/${PORTNAME} 下 (除了文□， 参□下文)。 要□少打包文件的尺寸， □可以直接在 Makefile 中引用□些 JAR 文件， 具体做法是使用下面的□句 (此□的 myport.jar 是作□ port 一部分安装的 JAR 文件的名字)：

```
PLIST_FILES+= %%JAVAJARDIR%%/myport.jar
```

移植 Java □用程序□， port 通常会希望将所有文件安装到同一目□ (包括其依□的 JAR)。 □□□烈建□使用 ${JAVASHAREDIR}/${PORTNAME}。 移植□件的□□人□， 可以自行决定是否将所依□的其它 JAR 安装到此目□， 或直接使用已□装好的那些 (来自 ${JAVAJARDIR})。

无□□正制作□一□的 port (□或者□用程序)， 附加的文□都□安装到和其它 port 同□的位置。 已□知道， JavaDoc 会根据 JDK 版本的不同而□生不同的文件。 □于那些不打算□制使用某一特定版本 JDK 的 port 而言， □无疑提高了制作装箱□ (pkg-plist) 的□度。 □是□什□□烈建□使用 PORTDOCS 宏的原因。 更□一□， 即使□能□□□ javadoc 将要生成的文件， 所需的 pkg-plist 的尺寸， 也是鼓吹使用 PORTDOCS 的一大理由。

DATADIR 的默□□是 ${PREFIX}/shared/${PORTNAME}。 □ Java port 而言将 DATADIR 改□ ${JAVASHAREDIR}/${PORTNAME} 是一个好主意。 当然， DATADIR 会自□加到 PLIST_SUB 中 (在 根据 make □量□ pkg-plist □行修改 有所介□) 因此□可以在 pkg-plist 中直接使用 %%DATADIR%%。

撰写本文□， □是□□从源代□□□， □是直接安装□□□的 Java ports 安装包并没有明□的□定。 尽管如此， FreeBSD Java Project 的□□人□仍鼓励移植□件的□□者在不麻□的情况下尽可能从源代□完成□□。

本□中所介□的全部特性， 均是在 bsd.java.mk 中□□的。 如果□感□自己的 port 需要更□□□的 Java 支持，

□首先参□ bsd.java.mk CVS 日志， 因□通常撰文介□最新特性需要一些□□。 此外， 如果□□□所缺少的支持□□多其它 Java port 亦属有益， □在 freebsd-java □其□行□□。

在 PR 中的 java □□， 主要是用于 FreeBSD Java project 移植 JDK 本身之用。 因而， 提交□的 Java port □， □□入 ports □□， 除非□正□□解决的□□是 JDK □□本身或 bsd.java.mk 的。

□似地， □□参考 分□ 中所□述的□于 CATEGORIES 在 Java port 中的使用□□。

6.12. Web □用， Apache 和 PHP

6.12.1. Apache

表 *20.* 用到 *Apache* 的 *port* 可以使用的□量

USE_APACHE	此 port 需要 Apache。 可用的□： yes (任意可用版本)、 1.3、 2.0、 2.2、 2.0+、 等等。 默□依□的版本是 1.3。
WITH_APACHE2	此 port 需要 Apache 2.0。 如果没有□个□量， □ port 将依□ Apache 1.3。 □一□量目前已□□□， 因而不□□□使用。
APXS	到 apxs 可□行文件的完整路径。 □可以在 port 中替代□□。
HTTPD	到 httpd 可□行文件的完整路径。 □可以在 port 中替代□□。
APACHE_VERSION	目前系□中安装的 Apache 版本 (只□□量)。 □一□量只有在引用了 bsd.port.pre.mk 之后才能使用，其可能的□□： 13、 20、 22。
APACHEMODDIR	Apache 模□所在的文件□。 在 pkg-plist 中， □一□量会自□展□。
APACHEINCLUDEDIR	Apache □文件所在的文件□。 在 pkg-plist 中， □一□量会自□展□。
APACHEETCDIR	Apache 配置文件所在的文件□。 在 pkg-plist 中， □一□量会自□展□。

表 *21.* 在移植 *Apache* 模□□比□有用的□量

MODULENAME	模□的名称。 默□□□ PORTNAME。 例如： mod_hello
SHORTMODNAME	模□的□略名字。 默□情况下会自□根据 MODULENAME □算， 但□也可以自行□置□来替代它。 例如： hello
AP_FAST_BUILD	使用 apxs 来□□和安装模□。
AP_GENPLIST	同□自□□建 pkg-plist。
AP_INC	在□□□程中， 将指定的目□加入到搜索□文件的目□中。
AP_LIB	在□□□程中， 将指定的目□加入到搜索函数□的目□中。
AP_EXTRAS	□□□ apxs 的□外参数。

6.12.2. Web □用

Web □用程序□安装到 PREFIX/www/□用程序的名字。 □方便起□， □个路径在 Makefile 和 pkg-plist 均以 WWWDIR □量的形式提供。 在 Makefile 中可以使用 WWWDIR_REL 来表示包含了 PREFIX 的□□量□。

web 服□器□程所用的用□和用□□， 分□以 WWWOWN 和 WWWGRP □量的形式提供， 如果□需要修改某些文件的属主的□。 □□个□量的默□□均□ www。 如果□的 port 希望使用其他□， □使用 WWWOWN?=myuser □□写法， 以便□用□能□更容易地修改它。

除非□的 port 必需使用 Apache， 否□不要将其写入依□□系。 □尊重□行□的□用程序的用□□□ Apache 以外的其他 web 服□器的需求。

6.12.3. PHP

表 *22.* 用到 *PHP* 的 *port* 中可以使用的□量

USE_PHP	此 port 需要 PHP。 取□□ yes 将把 PHP 加入依□□系。 此外， □可以在此指定将所需要的 PHP □展模□。 例如： pcre xml gettext
DEFAULT_PHP_VER	□□在没有安装 PHP □自□安装的 PHP 主版本。 默□是 4。 可□ 4、 5 之一。
IGNORE_WITH_PHP	此 port 无法与□定版本的 PHP 一同工作。 可□□□ 4、5 之一。
USE_PHPIZE	此 port 将作□ PHP □展模□□行□□。
USE_PHPEXT	此 port 将作□ PHP □展， 且需要作□□展模□注册。
USE_PHP_BUILD	□□依□于 PHP。
WANT_PHP_CLI	希望使用 CLI (命令行) 版本的 PHP。
WANT_PHP_CGI	希望使用 CGI 版本的 PHP。
WANT_PHP_MOD	希望使用 Apache 模□版本的 PHP。
WANT_PHP_SCR	希望使用 CLI 或 CGI 版本的 PHP。
WANT_PHP_WEB	希望使用 Apache 模□或 CGI 版本的 PHP。

6.12.4. PEAR 模□

移植 PEAR 模□的□程非常□□。

使用 FILES、 TESTS、 DATA、 SQLS、 SCRIPTFILES、 DOCS 以及 EXAMPLES □些□量来指明□希望安装的文件。 所有□里列出的文件都会自□安装到合□的位置， 并加入 pkg-plist。

在 Makefile 文件的最后一行引入 ${PORTSDIR}/devel/pear/bsd.pear.mk。

例 16. 用于 PEAR □的 Makefile 例子

```
PORTNAME=       Date
PORTVERSION=    1.4.3
CATEGORIES=     devel www pear

MAINTAINER=     example@domain.com
COMMENT=        PEAR Date and Time Zone Classes

BUILD_DEPENDS=  ${PEARDIR}/PEAR.php:${PORTSDIR}/devel/pear-PEAR
RUN_DEPENDS=    ${BUILD_DEPENDS}

FILES=          Date.php Date/Calc.php Date/Human.php Date/Span.php     \
                Date/TimeZone.php
TESTS=          test_calc.php test_date_methods_span.php testunit.php   \
                testunit_date.php testunit_date_span.php wknotest.txt   \
                bug674.php bug727_1.php bug727_2.php bug727_3.php       \
                bug727_4.php bug967.php weeksinmonth_4_monday.txt       \
                weeksinmonth_4_sunday.txt weeksinmonth_rdm_monday.txt   \
                weeksinmonth_rdm_sunday.txt
DOCS=           TODO
_DOCSDIR=       .

.include <bsd.port.pre.mk>
.include "${PORTSDIR}/devel/pear/bsd.pear.mk"
.include <bsd.port.post.mk>
```

6.13. 使用 Python

Ports 套件支持同□并行安装多个不同的 Python 版本。 Ports □□保能□根据用□配置的 PYTHON_VERSION □量使用正□的 python 解□器。 一般□来， □是通□将脚本中的 python 路径名替□□ PYTHON_CMD □量的□来□□的。

在 PYTHON_SITELIBDIR 下安装文件的 ports □在包名上使用 pyXY- 前□， 以便明示它□将会配合□个 Python 版本使用。

```
PKGNAMEPREFIX= ${PYTHON_PKGNAMEPREFIX}
```

表 23. □用到 Python 的 port 最有用的一些□量

USE_PYTHON	此 port 需要 Python。 可以用 2.3+ □□的形式来指定所希望的版本。 除此之外， 也可以用横□来分隔□个版本号， 以表示某个 □□的版本， 例如： 2.1-2.3

USE_PYDISTUTILS	使用 Python distutils 来完成配置、 □□和安装。 □包含 setup.py 的 port 而言□是必需的。 它会自□覆□默□的 do-build 以及 do-install □□个 target。 如未定□ GNU_CONFIGURE， 它□会改□ do-configure。
PYTHON_PKGNAMEPREFIX	作□ PKGNAMEPREFIX 来区分不同 Python 版本的 package。 例如： py24-
PYTHON_SITELIBDIR	全站 package 所在的目□， 它包括了 Python 的安装目□ (通常是 LOCALBASE)。 在安装 Python 模□□， PYTHON_SITELIBDIR □量会非常有用。
PYTHONPREFIX_SITELIBDIR	去掉了 PREFIX 部分的 PYTHON_SITELIBDIR。 □尽可能在 pkg-plist 中使用 %%PYTHON_SITELIBDIR%%。 %%PYTHON_SITELIBDIR%% 的默□□是 lib/python%%PYTHON_VERSION%%/site-packages
PYTHON_CMD	Python 解□器的命令行， 包括版本号。
PYNUMERIC	将数□□理□展模□加入依□□系。
PYNUMPY	□新□的数□□算□展, numpy的依□。 (PYNUMERIC 目前已被作者淘汰)。
PYXML	将 XML □展模□加入依□□系。 (□于 Python 2.0 和更高版本不再需要， 因□它已□成□了□准□件)。
USE_TWISTED	将 twistedCore 加入依□□系。 也可以用□个 □量指定所需的□件， 例如： web lore pair flow
USE_ZOPE	加入□ Zope， 一□ web □用平台的依□。 □会把 Python 依□改□ Python 2.3。 此外 ZOPEBASEDIR 也会自□□□ Zope 安装目□的位置。

完整的可用□量列表， 可以在 /usr/ports/Mk/bsd.python.mk 中□到。

6.14. 使用 Tcl/Tk

Ports 套件支持同□安装多个 Tcl/Tk 版本。 Ports □至少支持默□的 Tcl/Tk 版本， 以及通□ USE_TCL 和 USE_TK □量指定的更高版本。 希望使用的 tcl 版本， □可以通□ WITH_TCL_VER □量来使用。

表 *24.* 用到 *Tcl/Tk* 的 *port* 可以使用的□量

USE_TCL	表示 port 依□于 Tcl 函数□ (不是 shell)。 可以指定需要的最低版本， 例如 84+。 不支持的版本， 可以在 INVALID_TCL_VER □量中逐个指定。
USE_TCL_BUILD	表示 port 在□□□程中需要使用 Tcl。
USE_TCL_WRAPPER	需要使用 Tcl shell 而不需要特定版本的 tclsh 的 port 可以使用□个新□量。 系□中会安装 tclsh wrapper， 用□□可以指定所希望的 tcl shell。

WITH_TCL_VER	由用□定□的、 希望使用的 Tcl 版本。
UNIQUENAME_WITH_TCL_VER	和 WITH_TCL_VER □似， 但是□□ port 指定的。
USE_TCL_THREADS	需要包含□程支持的 Tcl/Tk。
USE_TK	表示 port 依□于 Tk □ (不是 wish shell)。 它同□会□含将 USE_TCL □置□相同的□。 更多的描述， □参考 USE_TCL □量。
USE_TK_BUILD	与 USE_TCL_BUILD □量表□□似的含□。
USE_TK_WRAPPER	与 USE_TCL_WRAPPER □量表□□似的含□。
WITH_TK_VER	表□与 WITH_TCL_VER □量□似的含□， 它同□会□含将 WITH_TCL_VER □置□相同的□。

可用的□量的完整列表， 可以在 /usr/ports/Mk/bsd.tcl.mk 中□到。

6.15. 使用 Emacs

本□尚有待撰写。

6.16. 使用 Ruby

表 *25.* 使用 *Ruby* 的 *port* 可以使用的□量

□量	□明
USE_RUBY	此 port 需要 Ruby。
USE_RUBY_EXTCONF	此 port 使用 extconf.rb 来完成配置。
USE_RUBY_SETUP	此 port 使用 setup.rb 来完成配置。
RUBY_SETUP	将此□量名□置□所用的 setup.rb 的文件名。 通常会是 install.rb。

下表展示了 ports 系□提供□ port 作者的一些□量。 □□使用□些□量， 以便把文件装到合□的位置。 □尽可能多地在 pkg-plist 中使用它□。 □些□量不□在 port 中重新定□。

表 *26.* 使用 *Ruby* 的 *port* 中的一些可用的只□□量

□量	□明	示□□
RUBY_PKGNAMEPREFIX	作□ PKGNAMEPREFIX 以区分用于不同 Ruby 版本的 package。	ruby18-
RUBY_VERSION	x.y.z 形式的完整 ruby 版本。	1.8.2
RUBY_SITELIBDIR	平台无□□的安装路径。	/usr/local/lib/ruby/site_ruby/1.8
RUBY_SITEARCHLIBDIR	平台相□的□的安装路径。	/usr/local/lib/ruby/site_ruby/1.8/amd64-freebsd6

□量	□明	示□□
RUBY_MODDOCDIR	模□文□的安装路径。	/usr/local/shared/doc/ruby18/patsy
RUBY_MODEXAMPLESDIR	模□用例的安装路径。	/usr/local/shared/examples/ruby18/patsy

可用□量的完整列表， 可以在 /usr/ports/Mk/bsd.ruby.mk 中□到。

6.17. 使用 SDL

□量 USE_SDL 可以用于自□配置 port 的依□□系， 以□□使用□似 devel/sdl12 和 x11-toolkits/sdl_gui □些依□ SDL 的□的情形。

目前系□能□□□下列 SDL □：

- sdl: devel/sdl12
- gfx: graphics/sdl_gfx
- gui: x11-toolkits/sdl_gui
- image: graphics/sdl_image
- ldbad: devel/sdl_ldbad
- mixer: audio/sdl_mixer
- mm: devel/sdlmm
- net: net/sdl_net
- sound: audio/sdl_sound
- ttf: graphics/sdl_ttf

因此， 如果 port 需要依□ net/sdl_net 和 audio/sdl_mixer， □□□的写法将是：

```
USE_SDL=        net mixer
```

同□， net/sdl_net 和 audio/sdl_mixer 所依□的 devel/sdl12 也会被自□地加入。

加入□使用 USE_SDL， 它将自□地：

- 将□于 sdl12-config 的依□□系加入到 BUILD_DEPENDS
- 将□量 SDL_CONFIG 加入到 CONFIGURE_ENV
- 将□所□的□的依□， 加入到 LIB_DEPENDS

要□□某个特定的 SDL □是否可用， 可以通□ WANT_SDL □量来□到目的：

```
WANT_SDL=yes

.include <bsd.port.pre.mk>

.if ${HAVE_SDL:Mmixer}!=""
USE_SDL+=   mixer
.endif

.include <bsd.port.post.mk>
```

6.18. 使用 wxWidgets

□一□介□了在 ports tree 中的 wxWidgets □的□状， 以及它与 ports 系□的集成。

6.18.1. 介□

□多不同版本的 wxWidgets □之□是存在相互冲突的（它□会安装同名的文件）在 ports 系□中， □一□□是通□将不同的版本以包含版本号后□的名字安装来解决的。

□□做的一个最明□的缺点是， □用程序必□□行修改， 才能□到所希望的版本。 幸□的是， 多数□用程序会□用 `wx-config` 脚本来□定需要的□□器和□接器□□。 □个脚本会随可用的版本不同而有不同的名字。 主要的□用程序都会尊重□境□量的配置， 或提供一个 configure 参数， 用以指定□用□个 `wx-config`。 如果不是□□的□， 就需要□□用程序打□丁了。

6.18.2. 版本的□□

□了□□的 port 使用指定版本的 wxWidgets， 可以定□□个□量 (如果只定□了一个， □□一个会取默□□)：

表 *27.* 用于□□ *wxWidgets* 版本的□量

□量	□明	默□□
`USE_WX`	列出□个 port 能使用的版本	全部版本
`USE_WX_NOT`	列出□个 port 不能使用的版本	无

下面是可用的 wxWidgets 版本， 以及□□的 ports：

表 *28.* 可用的 *wxWidgets versions*

版本	**Port**
`2.4`	x11-toolkits/wxgtk24
`2.6`	x11-toolkits/wxgtk26
`2.8`	x11-toolkits/wxgtk28

从 `2.5` 版□始， 也提供了□□的 Unicode 版本， □□版本可以通□ slave port 安装， 与普通版本相比， 它会多一个 `-unicode` 后□， 不□□可以通□使用□量来□理 (□参□ Unicode)。

在 用于□□ wxWidgets 版本的□量 中的□量， 可以□□下列□或由空格分隔的□合：

表 29. wxWidgets 版本

□明	例子
□个版本	2.4
某版本以上版本	2.4+
某版本以下版本	2.6-
某段版本 (版本号□小的必□在前)	2.4-2.6

除此之外， □有一些用以从可用的本那本中□□所希望的版本的□量。 □□□量也可以□□一□版本， 而□前的版本的□先□更高。

表 30. 用于□□希望的版本的 wxWidgets versions

□量名	用于
WANT_WX_VER	port
WITH_WX_VER	用□

6.18.3. □□□件

也有一些其他□用， 尽管它□本身并不是 wxWidgets □， 但却与之相□。 □些□用程序可以在 WX_COMPS □量中使用， 以下是可用的□件：

表 31. 可用的 wxWidgets □件

名称	□明	版本限制
wx	主□	无
contrib	第三方□	无
python	wxPython (Python □定)	2.4-2.6
mozilla	wxMozilla	2.4
svg	wxSVG	2.6

□可以□□个依□的□件， 通□冒号分隔的后□指定其□型。 如果没有指定， □会使用默□的依□□型 (参□ 默□的 wxWidgets 依□□系□型)。 下面是可用的□型：

表 32. 可用的 wxWidgets 依□□型

名称	□明
build	□□□需要□□件， 相当于 BUILD_DEPENDS
run	□行□需要□□件， 相当于 RUN_DEPENDS
lib	□□和□行□均需要□□件， 相当于 LIB_DEPENDS

□件的默□依□□系□型， 如下表所示：

表 33. 默□的 wxWidgets 依□□系□型

□件	依□□系□型
wx	lib
contrib	lib
python	run
mozilla	lib
svg	lib

例 17. □□ *wxWidgets* □件

下面的片段展示了使用 wxWidgets 版本 2.4 及第三方□的方法。

```
USE_WX=       2.4
WX_COMPS=     wx contrib
```

6.18.4. Unicode

wxWidgets □从其 2.5 版□始支持 Unicode 了。 在 ports 系□中， □□□版本均有提供， 并可以通□下列□量来□□：

表 34. 用以在 *Unicode* 版本的 *wxWidgets* 的□量

□量	□明	作用
WX_UNICODE	□ port 只能 配合 Unicode 版本使用	port
WANT_UNICODE	port 能□与□□版本配合使用， 但希望使用 Unicode 版本	port
WITH_UNICODE	令 port 使用 Unicode 版本	用□
WITHOUT_UNICODE	令 port 使用普通版本， 如果支持的□ (即未定□ WX_UNICODE)	用□

如果 port 同□支持 Unicode 和普通版本， □不要使用 WX_UNICODE。 如果希望默□□用 Unicode， □定□ WANT_UNICODE。

6.18.5. □□已安装的版本

要□□系□中安装的版本， 就需要定□ WANT_WX。 如果没有将其□置□特定的版本， □□件将包含版本后□。 HAVE_WX □量在□□完成后会自□填入内容。

例 18. □□已安装的 wxWidgets 版本和□件

下面的片段可以在安装 wxWidgets 的系□中令 port 使用它， 反之□作□一□□□提供。

```
WANT_WX=        yes

.include <bsd.port.pre.mk>

.if defined(WITH_WX) || ${HAVE_WX:Mwx-2.4} != ""
USE_WX=         2.4
CONFIGURE_ARGS+=--enable-wx
.endif
```

下面的片段在系□中有安装□□□用 wxPython 支持， 或在没有安装□作□□□提供； □ wxWidgets 也是如此□理， 版本皆□ 2.6。

```
USE_WX=         2.6
WX_COMPS=       wx
WANT_WX=        2.6

.include <bsd.port.pre.mk>

.if defined(WITH_WXPYTHON) || ${HAVE_WX:Mpython} != ""
WX_COMPS+=      python
CONFIGURE_ARGS+=--enable-wxpython
.endif
```

6.18.6. 定□的□量

以下是一些可以在 port 中使用的□量 (□之前需要定□ 用于□□ wxWidgets 版本的□量 中的至少一个□量)。

表 35. □使用 wxWidgets 的 port 定□的□量

□量名	□明
WX_CONFIG	到 wxWidgets wx-config 脚本的路径 (名字会随版本不同而不同)
WXRC_CMD	到 wxWidgets wxrc 程序的路径 (名字会随版本不同而不同)
WX_VERSION	将要用到的 wxWidgets 版本 (例如， 2.6)
WX_UNICODE	如果没有定□， 而将会使用 Unicode □， 系□将自□定□此□量。

6.18.7. 在 bsd.port.pre.mk 中□行□理

如果□需要在引用了 bsd.port.pre.mk 之后立即□一些□量□行□理， □需要定□ WX_PREMK。

如果定了 WX_PREMK， 在此 之后 定的依系、 件和量将不会生效， 在引用 bsd.port.pre.mk 之前的 wxWidgets port 量将直接起作用。

例 *19.* 在命令中使用 *wxWidgets* 量

下面的片段以行 wx-config 脚本来得到完整的版本号， 将其到量中， 并一个程序例明了 WX_PREMK 的用法。

```
USE_WX=         2.4
WX_PREMK=       yes

.include <bsd.port.pre.mk>

.if exists(${WX_CONFIG})
VER_STR!=       ${WX_CONFIG} --release

PLIST_SUB+=     VERSION="${VER_STR}"
.endif
```

在 target 中的 wxWidgets 量可以直接使用， 而无需 WX_PREMK 的参与。

6.18.8. 外的 configure 参数

某些 GNU configure 脚本在只置了 WX_CONFIG 境量， 无法自到 wxWidgets， 而需要使用外的参数来加以指定。 可以使用 WX_CONF_ARGS 量来出些参数。

表 *36.* 可用于 WX_CONF_ARGS 的

可用	果
absolute	--with-wx-config=${WX_CONFIG}
relative	--with-wx=${LOCALBASE} --with-wx-config=${WX_CONFIG:T}

6.19. 使用 Lua

一描述了在 ports 系中的 Lua 的状， 以及它与 ports 系的集成。

6.19.1. 介

多不同版本的 Lua 和相的解器之是相互冲突的 (它会安装同名的文件)。 在 ports 系中， 一是通将不同版本的文件以不同的版本号作后名解决的。

做最大的一个是， 个程序都需要行修改才能到它所需要的版本。 不， 通将当的参数器和接器很容易解决个。

6.19.2. □□版本

要□□的 port 使用指定版本的 Lua， 可以定□□个□量的□ (如果只定□了其中的一个， □□一个会使用默□□)：

表 37. 用于□□ *Lua* 版本的□量

□量	□明	默□□
USE_LUA	port 能□使用的 Lua 版本列表	全部可用版本
USE_LUA_NOT	与 port 不兼容的版本列表	无

下面是目前 ports 系□提供的可用 Lua 版本和□□的目□：

表 38. 可用的 *Lua* 版本

版本	**Port**
4.0	lang/lua4
5.0	lang/lua50
5.1	lang/lua

在 用于□□ Lua 版本的□量 中的□量， 可以□置□下面的版本之一， 或用空格分隔的若干版本：

表 39. 指定 *Lua* 版本

□明	例子
一个版本	4.0
某个版本或更高版本	5.0+
不高于某个版本	5.0-
版本□□ (低版本必□在前)	5.0-5.1

除此之外， 也有一些用来从可用版本中□□推□版本的其它□量。 □些□量也可以□置□一□版本， 而前面的版本□先□□高。

表 40. 用于□□推□ *Lua* 版本的□量

□量名	用于
WANT_LUA_VER	port
WITH_LUA_VER	用□

例 20. □□ *Lua* 版本

下面是一个用到 Lua 版本 5.0 或 5.1， 并默□使用 5.0 的 port 的片段。 □个默□□可以通□ WITH_LUA_VER 来□外指定。

```
USE_LUA=      5.0-5.1
WANT_LUA_VER= 5.0
```

6.19.3. □件的□□

也有一些其它的□用， 尽管本身并不是 Lua □， 但却与它□相□。 □些□用可以通□ LUA_COMPS □量来指定。 可用的□件如下：

表 41. 可用的 *Lua* □件

名字	□明	版本限制
lua	主□	无
tolua	用于□□ C/C++ 代□的□	4.0-5.0
ruby	Ruby □定	4.0-5.0

□有一些其它的□件， 但□些□件是由解□器， 而不是由□用程序使用的 (也就是不被其它模□使用)。

□个□件的依□□系□型可以通□手工添加分隔符□冒号的后□来指定。 如果不指定， □会采用默□□型 (□参□ 默□的 Lua 依□□系□型)。 以下是可用的依□□系□型：

表 42. 可用的 *Lua* 依□□系□型

名字	□明
build	□个□件是□□□程所必需的， 相当于 BUILD_DEPENDS
run	在□行□需要□个□件， 相当于 RUN_DEPENDS
lib	□个□件在□□和□行□都需要， 相当于 LIB_DEPENDS

□件的默□依□□系□型如下：

表 43. 默□的 *Lua* 依□□系□型

□件	依□□系□型
lua	□于 4.0-5.0 是 lib (□□□接) 而□于 5.1 □是 build (静□□接)
tolua	build (静□□接)
ruby	lib (□□□接)

例 21. □□ *Lua* □件

下面是一个使用了 Lua 版本 4.0 及其 Ruby □定的 port 片段。

```
USE_LUA=      4.0
LUA_COMPS=    lua ruby
```

6.19.4. □□系□中已安装的版本

要□□系□中已安装的版本， □必□定□ WANT_LUA。 如果没有将其□定□具体的版本， □□件会包含版本后□。 在

□□之后， HAVE_LUA □量将□□□□到的版本。

例 22. □□已安装的 *Lua* 版本和□件

下面是一个如果系□中有安装 Lua 或□□了□□□使用它的 port 片段。

```
WANT_LUA=       yes

.include <bsd.port.pre.mk>

.if defined(WITH_LUA5) || ${HAVE_LUA:Mlua-5.[01]} != ""
USE_LUA=        5.0-5.1
CONFIGURE_ARGS+=--enable-lua5
.endif
```

下面的□段 port 在系□中已□有安装， 或用□□□了 tolua 和 Lua 支持□加以安装， 版本均□□ 4.0。

```
USE_LUA=        4.0
LUA_COMPS=      lua
WANT_LUA=       4.0

.include <bsd.port.pre.mk>

.if defined(WITH_TOLUA) || ${HAVE_LUA:Mtolua} != ""
LUA_COMPS+=     tolua
CONFIGURE_ARGS+=--enable-tolua
.endif
```

6.19.5. 定□的□量

在 port 中可以使用下列□量 (在定□了 用于□□ Lua 版本的□量 中至少一个□量之后)。

表 44. □用到 *Lua* 的 *port* 定□的□量

□量名	□明
LUA_VER	将要使用的 Lua 版本。 (例如， 5.1)
LUA_VER_SH	Lua □□□接□的主版本 (例如， 1)
LUA_VER_STR	不□点的 Lua 版本 (例如， 51)
LUA_PREFIX	安装 Lua (及其□件) 使用的后□
LUA_SUBDIR	在 ${PREFIX}/bin、 ${PREFIX}/share 和 ${PREFIX}/lib 中用于安装 Lua 的子目□
LUA_INCDIR	用以安装 Lua 和 tolua □文件的目□
LUA_LIBDIR	用以安装 Lua 和 tolua □文件的目□
LUA_MODLIBDIR	用以安装 Lua 模□□□接□ (.so) 的目□

□量名	□明
LUA_MODSHAREDIR	用以安装 Lua 模□ (.lua) 的目□
LUA_PKGNAMEPREFIX	Lua 模□包的后□名
LUA_CMD	到 Lua 解□器的路径
LUAC_CMD	到 Lua □□器的路径
TOLUA_CMD	到 tolua 程序的路径

例 23. 告□ *port* 到什□地方去□ *Lua*

下面的 port 片段展示了如何告□使用的 configure 脚本去什□地方□□ Lua 的□文件和□文件。

```
USE_LUA=        4.0
GNU_CONFIGURE=  yes
CONFIGURE_ENV=  CPPFLAGS="-I${LUA_INCDIR}" LDFLAGS="-L${LUA_LIBDIR}"
```

6.19.6. 在 bsd.port.pre.mk □□行□理

如果□需要在使用引用 bsd.port.pre.mk 之后就得到□量， 以便将其用于□行一些命令， 需要定□ LUA_PREMK。

如果□定□了 LUA_PREMK， □在□引用 bsd.port.pre.mk之后， 即使修改了 Lua port □量，版本和依□□系也都不会随之□生□化了。

例 24. 在命令中使用 *Lua* □量

下面的片段展示了如何利用 LUA_PREMK， 并□行 Lua 解□器得到完整的版本串， 将其□□□一个□量，并□□□程序。

```
USE_LUA=        5.0
LUA_PREMK=      yes

.include <bsd.port.pre.mk>

.if exists(${LUA_CMD})
VER_STR!=       ${LUA_CMD} -v

CFLAGS+=        -DLUA_VERSION_STRING="${VER_STR}"
.endif
```

在 target 中的 Lua □量可以在命令中安全的使用， 而无需使用 LUA_PREMK。

6.20. 使用 Xfce

USE_XFCE □量可以用来自□配置使用基于 Xfce □或□用程序， 如 x11-toolkits/libxfce4gui 和 x11-wm/xfce4-panel 的 port 的依□□系。

目前， 系□能□□□下列 Xfce □和□用程序：

- libexo：x11/libexo
- libgui：x11-toolkits/libxfce4gui
- libutil：x11/libxfce4util
- libmcs：x11/libxfce4mcs
- mcsmanager：sysutils/xfce4-mcs-manager
- panel：x11-wm/xfce4-panel
- thunar：x11-fm/thunar
- wm：x11-wm/xfce4-wm
- xfdev：dev/xfce4-dev-tools

除此之外， □能□使用下列参数：

- configenv： 如果□的 port 需要使用特殊的 CONFIGURE_ENV 来□□所需的□。

```
-I${LOCALBASE}/include -L${LOCALBASE}/lib
```

会加到 CONFIGURE_ENV 的 CPPFLAGS。

因此， 如果 port 有到 sysutils/xfce4-mcs-manager 的依□□系， 并需要在 configure 的□境中指定特殊的□ CPPFLAGS， □所用的□法□：

```
USE_XFCE=       mcsmanager configenv
```

6.21. 使用 Mozilla

表 *45.* 用到 *Mozilla* 的 *port* 使用的□量

USE_GECKO	port 支持的 Gecko 后端。 可□□： libxul (libxul.so)、 seamonkey (libgtkembedmoz.so， □□， 新 port □避免使用)。
USE_FIREFOX	port 需要使用 Firefox 作□□行□境依□。 可□□： yes (使用默□版本)、 40、 36、 35。 默□的依□采用的是版本 40。

USE_FIREFOX_BUILD	port 需要使用 Firefox 作□□□□境依□。 可□□： 参□ USE_FIREFOX。 □个□量会自□□置 USE_FIREFOX 使用相同的□。
USE_SEAMONKEY	port 需要使用 SeaMonkey 作□□行□境依□。 可□□： yes (使用默□版本)、 20、 11 (□□， 新 port □避免使用)。 默□的依□采用的是版本 20。
USE_SEAMONKEY_BUILD	port 需要使用 SeaMonkey 作□□□□境依□。 可□□： 参□ USE_SEAMONKEY。 □个□量会自□□置 USE_SEAMONKEY 使用相同的□。
USE_THUNDERBIRD	port 需要使用 Thunderbird 作□□行□境依□。 可□□： yes (使用默□版本)、 31、 30 (□□， 新 port □避免使用)。 默□的依□采用的是版本 31。
USE_THUNDERBIRD_BUILD	port 需要使用 Thunderbird 作□□□□境依□。 可□□： 参□ USE_THUNDERBIRD。 □个□量会自□□置 USE_THUNDERBIRD 使用相同的□。

可用□量的完整列表， □参□ /usr/ports/Mk/bsd.gecko.mk。

6.22. 使用数据□

表 *46. ports* 中有□数据□的□量

Variable	Means
USE_BDB	如果□个□量□ yes， □把 databases/db41 列□依□□系。 □个□量□可以被□置成的□有： 40，41， 42、 43、 44、 46、 47、 48 或 51。 □可以声明可接受□的□□， USE_BDB=42+ 将□□已安装的最高版本， 如果没有□到□退回到 42。
USE_MYSQL	如果□个□量□ yes， □把 databases/mysql55-server 列□依□□系。□有一个相□的□量，WANT_MYSQL_VER， 可以□置的□有 323，40，41，50，51， 52， 55 或者 60。
USE_PGSQL	如果□置成 yes，□把 databases/postgresql82 列□ 依□□系。□有一个相□的□量，WANT_PGSQL_VER， 可以 □置的□有 73，74，80，81，82， 83 或 90。

更多□情□参□ bsd.database.mk。

6.23. □□和停止服□ (rc 脚本)

rc.d 脚本在系□□□□用于□□服□， 并□管理□提供停止、 □□和重新□□某个服□的□准方法。 Ports 安装的脚本会集成到系□的 rc.d 框架中。 □于如何使用它的□明， 可以在 使用手册的 rc.d 章□ □到。 □于可用命令的□□解□， □可以在 rc(8) 和 rc.subr(8) □到。 最后， □可以参□ □篇文章 了解撰写 rc.d 脚本的最佳□践。

可以安装一或多个 rc.d 脚本：

```
USE_RC_SUBR=    doormand
```

些脚本必放到 files 目， 并附加 .in。 个文件中可以使用准的 SUB_LIST 替展。 除此之外， 我烈推使用 %%PREFIX%% 和 %%LOCALBASE%% 替展。 于 SUB_LIST 的介可以在 本的相章 到。

在 FreeBSD 6.1-RELEASE 之前， 与 rcorder(8) 的集成是通 USE_RCORDER 而不是 USE_RC_SUBR 来完成的。不， 除非 port 需要提供安装基本系的， 或者服需要在 rc.d 脚本 FILESYSTEMS 之前行特殊情况， 一般来是不需要使用个功能的。

从 FreeBSD 6.1-RELEASE 始， 本地安装的 rc.d 脚本 (包括由 port 安装的脚本) 会入基本系的 rcorder(8)。

以下是一个的 rc.d 脚本：

```
#!/bin/sh

# $FreeBSD$
#
# PROVIDE: doormand
# REQUIRE: LOGIN
# KEYWORD: shutdown
#
# 在 /etc/rc.conf.local 或 /etc/rc.conf 中加下述置可以用一服：
#
# doormand_enable (bool):   默 NO。
#                 YES 可以用 doormand。
# doormand_config (path):   默 %%PREFIX%%/etc/doormand/doormand.cf。
#

. /etc/rc.subr

name="doormand"
rcvar=${name}_enable

command=%%PREFIX%%/sbin/${name}
pidfile=/var/run/${name}.pid

load_rc_config $name

: ${doormand_enable="NO"}
: ${doormand_config="%%PREFIX%%/etc/doormand/doormand.cf"}

command_args="-p $pidfile -f $doormand_config"

run_rc_command "$1"
```

除非有很站得住脚的理由提前服，所有的 ports 脚本使用

```
REQUIRE: LOGIN
```

。 如果服□需要以特定用□ (除 root 之外) 身□□□， □必□□□做。 在前面的例子中， 我□□使用了

```
KEYWORD: shutdown
```

以便□ mythical port 在系□停机的□程中以正常的方式□止， 因□它需要在系□引□□程中□□服□。 如果脚本没有□□任何服□， □并不需要□□做。

□里， □□量的默□□□方法□采用 "="， 而非 ":=" □□的形式。 □是因□， 前一□□□方法只有在□量未被□置□才□置默□□， 而后一□方法□会在□量没有□置， 或者 其□□空□都□置默□□。 用□非常可能在其 rc.conf.local 中使用□似

```
doormand_flags=""
```

□□的□置， 而采用 ":=" 来□行□□， □会在不□意□覆□用□所希望的□置。

新□的脚本均不□使用 .sh 后□。 未来， 仍然包含□一后□的脚本将被批量改名。

6.23.1. 卸□□停止服□

可以在卸□的□程中自□地停止服□。 我□建□只有在□□必要， 例如必□在□除文件之前停止服□□□的情况下才使用□一功能。 通常来□， 决定是否在卸□□停止服□是系□管理□需要考□的事情。 □外要注意， □个功能也会影□升□□程。

需要□可以在 pkg-plist 中加入：

```
@stopdaemon doormand
```

□里的参数必□与 USE_RC_SUBR □量的内容匹配。

6.24. 添加用□和用□□

一些 port 需要在安装的系□中□建特定的用□或用□□。 如果有□□情况， □从 50 到 999 之□□□一个尚未使用的 UID， 并在 ports/UIDs (□□用□) 或 ports/GIDs (□□□) 中予以□□。 □□必□保□没有使用系□中已□在其他 ports 中使用的 UID。

如果□的 port 需要□建新用□或用□□， □在提交□丁的□候一并提交□□个文件的□丁。

接下来， 可以在□的 Makefile 中使用 USERS 和 GROUPS □□个□量， 系□会在安装□自□□建用□或□。

```
USERS=  pulse
GROUPS= pulse pulse-access pulse-rt
```

□有的保留 UID 和 GID 列表， 可以在 ports/UIDs 和 ports/GIDs □到。

6.25. 依□内核源代□的 Ports

某些 ports (例如可加□式内核模□) 需要内核的源文件才能□□。 下面是□□用□是否安装了源代□的例子：

```
.if !exists(${SRC_BASE}/sys/Makefile)
IGNORE=         requires kernel sources to be installed
.endif
```

Chapter 7. 高□ pkg-plist 用法

7.1. 根据 make □量□ pkg-plist □行修改

某些 port， 特□是 p5- port， 会需要根据配置□□ (或□于 p5- port 而言， perl 的版本) 来修改它□的 pkg-plist。 □□化□一工作， 在 pkg-plist 中的 %%OSREL%%、 %%PERL_VER%%， 以及 %%PERL_VERSION%% 将自□□行相□的替□。 其中， %%OSREL%% 的□是操作系□以数□表示的版本 (例如 4.9)。 %%PERL_VERSION%% 和 %%PERL_VER%% 是 perl 的完整版本号 (例如 5.8.9)。□多其它与 port 文□文件有□的 %%□量%% 在 相□章□ 中□行了介□。

如果□□需要□行其它的替□， 可以通□将 PLIST_SUB □量□置□一□ □量=□ □来□□。 其中， %%VAR%% 表示在 pkg-plist 中将被 □ 替□的那些文字。

□例来□， 如果 port 需要把很多文件放到和版本有□的目□中， 可以在 Makefile 中按照□似下面的例子：

```
OCTAVE_VERSION= 2.0.13
PLIST_SUB=      OCTAVE_VERSION=${OCTAVE_VERSION}
```

并在 pkg-plist 中将具体的版本替□□ %%OCTAVE_VERSION%%。 □□， 在升□ port □， 就不需要再到 pkg-plist 中修改那几十 (或者， 有□甚至是上百) 行的内容了。

如果□的 port 需要根据一定的配置来有条件地安装一些文件， 通常的做法是在 pkg-plist 中列出□些文件□， 在□□行的□□加上 %%TAG%%， 并将 TAG 写到 Makefile 中的 PLIST_SUB □量中， 根据需要替□掉， 或替□□ @comment， 后者表示□打包工具忽略□行：

```
.if defined(WITH_X11)
PLIST_SUB+= X11=""
.else
PLIST_SUB+= X11="@comment "
.endif
```

与之□□， 在 pkg-plist 中：

```
%%X11%%bin/foo-gui
```

□一替□□程 (以及加入 □机手册 的□程)， 会在 pre-install 和 do-install □个 target 之□， 通□□取 PLIST 并写入 TMPPLIST (默□情况下， 是： WRKDIR/.PLIST.mktmp) 来完成。 因此， 如果□的 port □□生成 PLIST， 就需要在 pre-install 之前完成。 □外， 如果□的 port 需要□□所生成的文件， □需要在 post-install 中操作名□ TMPPLIST 的那个文件。

□一□可行的修改装箱□的方法， □是根据 PLIST_FILES 和 PLIST_DIRS □□个□量的□置来□行。 它□的□会作□目□名□同 PLIST 的内容一起写入 TMPPLIST。 在 PLIST_FILES 和 PLIST_DIRS 中列出的名字， 会□□前面所介□的 %%□量%% 替□□程。 除此之外， 在 PLIST_FILES 中列出的文件， 会不加任何修改第出□在最□的装箱□中， 而 @dirrm 将作□前□加到 PLIST_DIRS 所列的名字之前。 □了□到目的， PLIST_FILES 和 PLIST_DIRS 必□在写 TMPPLIST 之前， 也就是在 pre-install 或更早的□段□行□置。

7.2. 空目□

7.2.1. 清理空目□

一定要□ port 在卸□□□行清理空目□。 通常， 可以通□□所有由 port □建的目□□加□□的 @dirrm 行来□□。 在□除父目□之前， 需要首先□除它的子目□。

```
 :
lib/X11/oneko/pixmaps/cat.xpm
lib/X11/oneko/sounds/cat.au
 :
@dirrm lib/X11/oneko/pixmaps
@dirrm lib/X11/oneko/sounds
@dirrm lib/X11/oneko
```

然而， 有□ @dirrm 会由于其它 port 使用了同一个目□而□生□□。 利用 @dirrmtry 可以只□除那些空目□，而避免□出警告。

```
@dirrmtry share/doc/gimp
```

按照上面的写法， 将不会□示任何□□信息， 而且，即使在 ${PREFIX}/shared/doc/gimp 由于其它 port 在其中安装了一些□的文件的□候， 也不会□致 pkg_delete(1) □常退出。

7.2.2. 如何建立空目□

在 port 安装□程中□建的空目□需要特□留意。 安装 package □并不会自□□建□些目□， □是因□ package 只保存文件。 要□保安装 package □会自□□建□些空目□， 需要在 pkg-plist 中加入与 @dirrm □□的行：

```
@exec mkdir -p %D/shared/foo/templates
```

7.3. 配置文件

如果 port 需要把一些文件放到 PREFIX/etc， 不要 □□地安装它□， 并将其列入 pkg-plist， 因□□□会□致 pkg_delete(1) □除用□精心□□的文件， 而新安装□□又会把□些文件覆□。

因此， □□把配置文件的例子按其它的后□来安装 (例如 filename.sample 就是一个不□的□□) 并□示一条 消息告□用□如何□制并□□□个配置文件， 以便□□件能□正□工作。

因此， □按其它的后□来安装配置文件的例子 (filename.sample 就是一个不□的□□)。 如果□□的配置文件不存在， □将其□制□□□文件的名字。 卸□□， 如果□□用□没有修改配置文件， □将其□除。□需要在 port 的 Makefile， 以及 pkg-plist (□于从 package 安装的情形) □行□理。

示例的 Makefile 部分：

```
post-install:
    @if [ ! -f ${PREFIX}/etc/orbit.conf ]; then \
        ${CP} -p ${PREFIX}/etc/orbit.conf.sample ${PREFIX}/etc/orbit.conf ; \
    fi
```

示例的 pkg-plist 部分：

```
@unexec if cmp -s %D/etc/orbit.conf.sample %D/etc/orbit.conf; then rm -f
%D/etc/orbit.conf; fi
etc/orbit.conf.sample
@exec if [ ! -f %D/etc/orbit.conf ] ; then cp -p %D/%F %B/orbit.conf; fi
```

□外， □□□示一条 消息 指出用□□在何□□制并□□□个文件， 以便□□件能□始正常工作。

7.4. □□装箱□与静□装箱□的□比

静□装箱□ 是指在 Ports Collection 中以 pkg-plist 文件 (可能包含□量替□)， 或以 PLIST_FILES 和 PLIST_DIRS 的形式嵌入到 Makefile 出□的装箱□。 即使它是由工具或 Makefile 中的某个 target 在□由 committer 加入到 Ports Collection 之前 自□生成的也是如此， 因□可以在不下□或□□源代□包的前提下□其□行□□。

□□装箱□ 是指在 port □□并安装□生成的装箱□。 在下□并□□□所移植的□用程序的源代□之前， 或在□行了 make clean 之后， 就无法□看其内容了。

尽管使用□□装箱□并不被禁止， 但□□人□尽可能使用静□装箱□， 因□它能□□用□使用 grep(1)来□□所需的 ports， 例如， 它是否会安装某个特定文件。 □□列表主要□用于□□的， 其装箱□随所□功能会□生巨□ (因而使得□□静□装箱□不再可行)， 或那些随版本而改□装箱□内容的 port (例如， 使用 Javadoc 来生成文□的那些 ports)。

我□鼓励那些□□使用□□装箱□的□□人提供一个能□生成 pkg-plist 的 target， 以便于用□□□其内容。

7.5. 装箱□ (package list) 的自□化制作

首先， □□□已□基本上完成了 port 的工作， □缺 pkg-plist。

接下来， 建立一个用于安装□的 port 的□□目□， 并在其中安装它所依□的所有其他□件包：

```
# mkdir /var/tmp/`make -V PORTNAME`
# mtree -U -f `make -V MTREE_FILE` -d -e -p /var/tmp/`make -V PORTNAME`
# make depends PREFIX=/var/tmp/`make -V PORTNAME`
```

将目□□□保存到一新文件中。

```
# (cd /var/tmp/`make -V PORTNAME` && find -d * -type d) | sort > OLD-DIRS
```

建立一空白 pkg-plist 文件：

```
# :>pkg-plist
```

如果□的 port 遵循 PREFIX (□□如此) □接下来□安装□ port 并□建装箱□。

```
# make install PREFIX=/var/tmp/`make -V PORTNAME`
# (cd /var/tmp/`make -V PORTNAME` && find -d * \! -type d) | sort > pkg-plist
```

此外□□把新建立的目□加入装箱□。

```
# (cd /var/tmp/`make -V PORTNAME` && find -d * -type d) | sort | comm -13 OLD-DIRS - |
sort -r | sed -e 's#^#@dirrm #' >> pkg-plist
```

最后需要手工整理 packing list； □一□程不是 完全 自□的。 □机手册□列入 port 的 Makefile 中的 MANn，而不是装箱□。 用□配置文件□被□除， 或以 filename.sample □□的名字来安装。 info/dir 文件， 也不□列入， 同□□按照 info 文件 的□明来□加一些 install-info 行。 所有由 port 安装的□， □按照 □□□接□ 小□中介□的方法□理。

□外， 也可以使用 /usr/ports/Tools/scripts/ 中的 plist 脚本来自□□建 package list。 plist 脚本是一个 Ruby 脚本， 它能□将前面介□的手工操作自□化。

□始的□□和上面的前三行一□， 也就是 mkdir， mtree 并 make depends。 然后□□和安装 port：

```
# make install PREFIX=/var/tmp/`make -V PORTNAME`
```

然后□ plist 生成 pkg-plist 文件：

```
# /usr/ports/Tools/scripts/plist -Md -m `make -V MTREE_FILE` /var/tmp/`make -V
PORTNAME` > pkg-plist
```

与前面□似， 如此生成的装箱□也需要手工□行一些清理工作。

□一个可以用来□建最初的 pkg-plist 的工具是 ports-mgmt/genplist。 和其他自□化工具□似， □□□它生成的 pkg-plist □手工□□并根据需要□行修改。

Chapter 8. pkg-* 文件

前面有一些没有提及的□于 pkg-* 文件的技巧， 它□可以方便地完成□多任□。

8.1. pkg-message (安装□□□包□□示的消息文件)

如果□需要在安装□□示一条消息□用□， 可以把□消息放在 pkg-message 中。 □一特性通常可以用于在 pkg_add(1) 之后□示一些附加的安装□□， 或□示□于授□的信息。

当需要□示一些□□□□或警告□， □使用 ECHO_MSG。 pkg-message 文件只是□□示安装后的□行操作指□使用的。 □似地， □需要留意 ECHO_MSG 和 ECHO_CMD 之□的区□。 前一个是在屏幕上□示消息性的文字， 而后一个□用于在□行命令□使用管道。

下面是用到了□□个宏的例子 shells/bash2/Makefile：

```
update-etc-shells:
    @${ECHO_MSG} "updating /etc/shells"
    @${CP} /etc/shells /etc/shells.bak
    @( ${GREP} -v ${PREFIX}/bin/bash /etc/shells.bak; \
        ${ECHO_CMD} ${PREFIX}/bin/bash) >/etc/shells
    @${RM} /etc/shells.bak
```

pkg-message 文件， 并不需要明□地加到 pkg-plist 中。 此外， 在用□使用 port 而不是 package 来安装□件□， 它并不会被□示出来。 因此如果需要的□， □□□在 post-install target 中指定□示它。

8.2. pkg-install (安装□□□包□□行的脚本文件)

如果□的 port 需要在□□□的安装包通□ pkg_add(1) 安装□□行一些命令， □□通□ pkg-install 脚本来完成。 □个脚本会自□地加入 package， 并被 pkg_add(1) □行□次： 第一次是 ${SH} pkg-install ${PKGNAME} PRE-INSTALL 而第二次是 ${SH} pkg-install ${PKGNAME} POST-INSTALL。 $2 可被用来□□脚本□行的模式。 □境□量 PKG_PREFIX 将□置□ package 的安装目□。 □参□ pkg_add(1) 以了解更□一□的□□。

在使用 make install □□个脚本不会被自□□行。 如果需要□行它， □必□在□的 port 中的 Makefile 里明□地予以□用， 其方法是加入□似 PKG_PREFIX=${PREFIX} ${SH} ${PKGINSTALL} ${PKGNAME} PRE-INSTALL □□的命令。

8.3. pkg-deinstall (卸□□□行的脚本文件)

□一脚本将在 package 被卸□□□行。

此脚本会被 pkg_delete(1) □行□次。 第一次是 ${SH} pkg-deinstall ${PKGNAME} DEINSTALL 而第二次□是 ${SH} pkg-deinstall ${PKGNAME} POST-DEINSTALL。

8.4. pkg-req (安装□□□包□□□是否□□行操作的脚本文件)

如果□的 port 需要□定它是否□被安装， 可以□建 pkg-req"requirements" 脚本。 它会在安装/卸□□自□□行，以决定操作是否□被□施。

□个脚本会在使用 pkg_add(1) 安装□以 pkg-req ${PKGNAME} INSTALL 的命令行□行。 卸□□， 它将由 pkg_delete(1) 以 pkg-req ${PKGNAME} DEINSTALL 的命令行□行。

8.5. 改□ pkg-* 文件的名字

所有 pkg- 文件的名字， 皆系采用□量予以定□， 因此在需要□可以在□的 Makefile 中加以改□。 当□需要在多个 port 之□共享某些 pkg- 文件， 或需要写入某些文件□就非常有用了。 (参□ 在 WRKDIR 以外的地方写文件， 以了解□什□直接将□更写入 pkg-* 子目□是个糟□的主意)

下面是一□□量以及它□的默□□ (PKGDIR 默□情况下是 ${MASTERDIR}。)

□量	默□□
DESCR	${PKGDIR}/pkg-descr
PLIST	${PKGDIR}/pkg-plist
PKGINSTALL	${PKGDIR}/pkg-install
PKGDEINSTALL	${PKGDIR}/pkg-deinstall
PKGREQ	${PKGDIR}/pkg-req
PKGMESSAGE	${PKGDIR}/pkg-message

□修改□些□量， 而不是直接覆□ PKG_ARGS 的□。 如果□改□了 PKG_ARGS， □些文件将无法在安装 port □正□地□制到 /var/db/pkg 目□。

8.6. 使用 SUB_FILES 和 SUB_LIST

SUB_FILES 和 SUB_LIST □□个□量可以用来在 port 文件中使用某些□□的□， 例如 pkg-message 中的 installation PREFIX。

用 SUB_FILES □量， 可以指定需要自□□行修改的文件列表。 在 SUB_FILES 中的□一个 文件， 在 FILESDIR 目□中都必□有一个□□的 文件.in。 修改后的版本将保存在 WRKDIR。 在 USE_RC_SUBR (或已□□□的 USE_RCORDER) 中定□的文件会自□加入到 SUB_FILES 中。 □于 pkg-message、 pkg-install、 pkg-deinstall and pkg-req， □□的 Makefile □量会被自□□置， 以指向□理□的版本。

SUB_LIST □个□量的内容是一系列 VAR=VALUE □。 SUB_FILES 所列出的文件中所有的 %%VAR%% 都将被替□□ VALUE。 系□自□定□了一些常用的替□□， 包括： PREFIX、 LOCALBASE、 DATADIR、 DOCSDIR， 以及 EXAMPLESDIR。 替□□果中所有以 @comment □□的行， 都将在□量替□之后被□去。

下面的例子中， 将把 pkg-message 中的 %%ARCH%% 替□□系□所□行的架□名称：

```
SUB_FILES=      pkg-message
SUB_LIST=       ARCH=${ARCH}
```

注意， 在上述例子中， FILESDIR 里必□有 pkg-message.in □个文件。

下面是一个正□的 pkg-message.in 例子：

```
Now it is time to configure this package.
Copy %%PREFIX%%/shared/examples/putsy/%%ARCH%%.conf into your home directory
as .putsy.conf and edit it.
```

Chapter 9. □□□的 port

9.1. □行 make describe

□多 FreeBSD port □□工具， 例如 portupgrade(1)， 会依□于一个名□ /usr/ports/INDEX 的数据□的正□性，它提供了□于 port 的相□信息， 例如依□□系等等。 INDEX 是由□□的 ports/Makefile 通□ make index 来建立的， □个命令会□入□一个 port 的子目□， 并在那里□行 make describe。 因此， 如果某个 port 的 make describe 失□， 就没有人能生成 INDEX， 人□很快会□得不高□。

无□在 make.conf 中□置了什□□□， □个文件都□能□正□地生成。 因此， □避免在 (例如) 某个依□□系无法□足□使用 .error。 (参□ 避免使用 .error □□。)

如果 make describe 只是□生一个字符串， 而不是□□信息， 可能就没什□□□。 □参□ bsd.port.mk 以了解所生成的串的意□。

最后要□明的是， 新版本的 portlint (在下一□中将□行介□) 将会自□地□行 make describe。

9.1.1. Portlint

在提交或 commit 之前， □使用 portlint 来□行□□。 portlint 会□常□的、 包括功能上的和格式上的□□□出警告。 □于新的 (或在 repocopy 代□□中□制的) port， portlint -A 可以完成全面□□； □于□存的 port， portlint -C 一般就足□了。

由于 portlint 采用□□式方法来□□□□， 有□它会□生□警。 □外， 有□由于 port 框架的限制可能没有□法修正它指出的□□。 如果□有疑□， □写信□□ FreeBSD ports □件列表。

9.1.2. 使用 Port Tools 来完成□□

在 Ports 套件中， 提供了一个 ports-mgmt/porttools 程序。

port 是一个能□□助□□化□□工具的前端脚本。 如果希望□新□的 port 或更新 port □□行□□， 可以用 port test 来完成□些□□工作， □也包含了 portlint □□。 □个命令会□□并列出没有在 pkg-plist 中列出的文件。 具体用法□参□下面的例子：

```
# port test /usr/ports/net/csup
```

9.1.3. PREFIX (安装□的□□目□名) 和 DESTDIR

PREFIX 能□决定 port 安装□的目的位置。 一般情况下□个位置是 /usr/local 或 /opt， 但也可以□□其它的任意□。 □的 port □必□遵循□个□量。

除此之外， 如果用□配置了 DESTDIR， □表示希望将 port 安装到□一个□境， 通常是 jail 或在 / 以外的其他位置挂接的系□中。 □□上， port 会安装到 DESTDIR/PREFIX， 并注册到位于 DESTDIR/var/db/pkg 的□□□包数据□中。 由于 DESTDIR 是由 ports 框架藉由 chroot(8) 来□□的， □在撰写符合 DESTDIR □□的 ports □并不需要什□□外的工作。

一般而言 PREFIX 会□□ LOCALBASE_REL (默□是 /usr/local)。 如果□置了 USE_LINUX_PREFIX， □ PREFIX 会□□

LINUXBASE_REL (默□是 /compat/linux)。

避免将 /usr/local 或 /usr/X11R6 硬□□到源代□中， 能□大大提高 port 的□活性， 并□□不同□境的需要。 □于使用 imake 的 X port， □一工作是自□完成的； 其他情况下， 通常可以□□地将 port 所用到的 Makefile 脚本中出□的 /usr/local (或□于没有使用 imake 的 X port 而言， /usr/X11R6) 替□□□取 ${PREFIX} □量就能□到目的了， 因□□个□量在□□和安装的□程中， 会自□向下□□。

一定要避免□□的 port 在 /usr/local 而不是正□的 PREFIX 中安装文件。 □□的□□方法是：

```
# make clean; make package PREFIX=/var/tmp/`make -V PORTNAME`
```

如果有文件安装到了 PREFIX 以外的地方， 打包□程将抱怨□不到□些文件。

□一□□并不能□助□□内部引用， 或□正在引用其它 port 中的文件□使用的 LOCALBASE。 □需要在 /var/tmp/make -V PORTNAME 中□□安装好的□件， 才能□□到□□的目的。

□可以在自己的 Makefile 中改□ PREFIX □量的□， 也可以通□用□□境□量来影□它。 然而， 一般情况下决不□□在 Makefile 中明□□置它的□。

此外， 引用其它 port 中的文件□， □使用前面介□的□量， 而不要直接指定它□的路径名。 例如， 如果□的 port 需要使用 PAGER □个宏来指明 less 的完整路径， □使用下面的□□□□：

```
-DPAGER=\"${LOCALBASE}/bin/less\"
```

而非 -DPAGER=\"/usr/local/bin/less\"。 □□方法能□□加在系□管理□把整个 /usr/local 目□□到其它位置□安装成功的机会。

9.1.4. Tinderbox

如果□是非常□心的 ports 参与者， □可以看看 Tinderbox。 □是一个□大的用于□□和□□ ports 的系□， 它基于 Pointyhat 的脚本。 □可以使用 ports-mgmt/tinderbox port 来安装 Tinderbox。 □一定仔□□□随它安装的文□， 因□配置并不□□。

□□□ Tinderbox 网站 以了解□一□的□□。

Chapter 10. 升□一个 port

如果□□□某个 port 相□原作者所□布的版本已□□□， □首先需要□□的是□的 port 是最新的。 □可以在 FreeBSD FTP □像的 ports/ports-current 目□中□到它□。 但是， 如果□正在使用□多的 port， □可能使用 CVSup 来保持 Ports Collection 最新更□□□， □在 使用手册 中□行了介□。 此外， □□做也有助于保持 port 依□□系的正□性。

下一□是□□是否已□有在等待的更新。 要完成□□工作， 可以采用下列□□方法之一。 有一个用于搜索 FreeBSD □□□告 (PR) 数据□ (也被称作 GNATS)。 在下拉框中□□ ports， 然后□入 port 的名字。

但是， 有些□候人□会忘□将避免混□的 port 的名字放到 Synopsis 字段中。 □□□候， □可以□□看 FreeBSD Ports □□系□ (也被叫做 portsmon)。 □个系□会□□按照 port 的名字来□行分□。 要搜索和某个特定 port 有□的 PR， 可以使用 port概□。

如果没有候□的 PR， 下一□是□ port 的□□者写信， □可以通□□行 make maintainer 看到。 □个人可能正在□行升□工作， 或者由于某□理由□□没有升□ (例如， 新版本有□定性□□)； 一般□不希望重□他□的工作。 注意没有□□者的 port 的□□者会□示□ ports@FreeBSD.org， □是一般性 port □□的□件列表， 因此□□件□它一般没什□意□。

如果□□者要求□去完成升□， 或者没有□□者， □就有机会通□自行完成升□来□助 FreeBSD 了！ □使用基本系□提供的 diff(1) 命令来完成相□的工作。

如果只修改一个文件， 可以直接使用 diff 来生成□丁， 将需要修改的文件□制成 *something.orig*， 并将改□放□ *something*， 接着生成□丁：

```
% /usr/bin/diff something.orig something > something.diff
```

如果不是□□的□， □□□使用 cvs diff 的方法 (使用 CVS 制作□丁)， 或将目□整个□制到□一个目□， 并使用 diff(1) 比□□个目□□在目□中□□□生的□出□果 (例如， 如果□修改后的 port 目□的名字是 superedit 而原始文件的目□是 superedit.bak， □□保存 diff -ruN superedit.bak superedit 的□果)。 一致式 (unified) 或 上下文式 (context) diff 都是可以的， 但一般来□ port committer 会更喜□一致式 diff。 □注意□里使用的□□ -N， 它的目的是□制 diff 正□地□理出□新文件， 或老文件被□除的情形。 在把 diff □□我□之前， □再次□□□出， 以便□□□一个修改都是有意□的。 (特□注意， 在□比目□之前要用 make clean 清理一下)。

□了□化常用的□丁文件操作， □可以使用 /usr/ports/Tools/scripts/patchtool.py。 使用之前， □首先□□ /usr/ports/Tools/scripts/README.patchtool。

如果 port 目前□无人□□， 而且□自己□常使用它， □考□自□□它的□□者。 FreeBSD 有超□ 4000 个没有□□者的 port， 而□正是最需要志□人□的□域。 (要了解□于□□者的任□描述， □参□ □□手册中的相□部分。)

将 diff □送□我□的最佳方式是通□ send-pr(1) (category 一□写 ports)。 如果□正□□那个 port， □□必在 synopsis 的□□写上 [maintainer update]， 并将□的 PR 的 "Class" □置□ maintainer-update。 反之， □的 PR 的 "Class" 就□□是 change-request。 □在信中逐个提及□一个□除或□加的文件， 因□□些都必□明□地在使用 cvs(1) □行 commit □明□地指定。 如果 diff 超□了 20K， □考□□□并□其□行 uuencode； 否□， □□地将其原□加入 PR 即可。

在□ send-pr(1) 之前， □再次□□ Problem Reports 一文中的 如何撰写□□□告 小□； 它□出了□富的

□于如何撰写更好的□□□告的介□。

如果□的更新是由于安全考□， 或修□已□ commit 的 port 中的□重□□， □通知 Ports 管理□□ <portmgr@FreeBSD.org> 来申□立即重建和分□□的 port 的 package。 否□， 不□□疑的使用 pkg_add(1) 的用□， 可能会在未来数周之内□□通□使用 pkg_add -r 安装旧版本。

再次□□， □使用 diff(1) 而非 shar(1) 来□送□有 port 的更新！ □可以□助 ports committer 理解需要修改的内容。

□在□已□了解了所需的所有操作， □可能会像要□□在 保持同□ 中□于如何保持最新的描述。

10.1. 使用 CVS 制作□丁

如果可能的□， □提交cvs(1) diff； □□情形要比直接比□ "新、旧" 目□要容易□理。 此外， □□方法也□□更容易看出到底改了什□， 并在其他人更新了 Ports Collection □容易合并□些改□， 在提交之前， □可以□少□□□丁所需的工作。

```
% cd ~/my_wrkdir ①
% cvs -d R_CVSROOT co pdnsd ② ③
% cd ~/my_wrkdir/pdnsd
```

① 当然， □可以是□指定的任意目□； □□ port 并不局限于 /usr/ports/ 的子目□。

② R_CVSROOT 是任何一个公共的 cvs □像服□器， □可以在 FreeBSD 使用手册 中挑□一个。

③ pdnsd 是 port 的模□名字； 通常□来它和 port 的名字一□， 不□也有些例外， 特□是那些本地化□□ (german/selfhtml □□的模□名字是 de-selfhtml)； □可以通□ cvsweb 界面 □□， 或者也可以指定完整路径， 例如在我□□个例子中是 ports/dns/pdnsd。

在工作目□中， □可以像往常一□□行任何更改。 如果□添加或□除了文件， □需要告□ cvs 来追踪□些改□：

```
% cvs add new_file
% cvs remove deleted_file
```

□反□□□ □□ port 列出的事□并使用 用 portlint 来□□ port □行□□。

```
% cvs status
% cvs update ①
```

① □会合并 CVS 中其他人做的改□和□的□丁； 在□个□程中， □需要仔□□察□出。 文件名前面的那个字母会□示做了什□， □参□ cvs update 文件名前字母前□的含□ 中□出的□明。

表 *47. cvs update* 文件名前字母前□的含□

U	文件更新无□。
P	文件更新无□ (通常只有在使用□程代□□□才会看到)。

M	文件有本地修改， 并合并成功而未□生任何冲突。
C	文件有本地修改， □行了合并并□生了冲突。

如果□在□行 `cvs update` □某些文件出□了 `C`， □□明有其他人在 CVS 中做了修改， 而 cvs(1) 无法将□些改□与□本地的改□□行合并。 不□， 无□如何， 最好都□□一下合并的□果， 因□ `cvs` 并不知道 port □□是什□□子，因此它所做的合并无□是否□生了冲突， 都有可能 (并且并不□□) □生没有意□的□果。

最后一□是以 CVS 中的文件□基□生成 unified diff(1)：

```
% cvs diff -uN > ../`basename ${PWD}`.diff
```

指定 `-N` 十分重要， 因□它□保了添加或□除的文件也出□在□丁中。 □丁将包含□除的文件，在打上□丁□， □些文件会被清空， 所以最好在 PR 中提醒 committer □除它□。

根据 升□一个 *port* 的指□提交□的□丁。

10.2. UPDATING 和 MOVED 文件

如果在升□ port □需要□似修改配置文件或□行特定的程序□□的特□□□， □□在 /usr/ports/UPDATING 文件中予以□明。 □个文件中的□目格式如下：

```
YYYYMMDD:
  AFFECTS: users of port□□/port名字
  AUTHOR: □的名字 <□的□子□件地址>

  所需□行的特□□□
```

如果□需要在内文中加入具体的 portmaster 或 portupgrade 的□明， □□保所用的 shell 命令使用了正□的□□字符。

如果 port 被□除或改名，□□在 /usr/ports/MOVED 中添加相□的□明□目。 □个文件中的□目格式如下：

```
原来的名字|新名字（如果□除□□留空）|□除或改名的日期|原因
```

Chapter 11. Ports 的安全

11.1. 安全□何如此重要

□件中偶□会引入 bug。 毋庸置疑， 安全漏洞是最□危□的。 从技□角度看， □些漏洞可以通□消除□致它□的 bug 来修□。 然而， □理一般的 bug 和安全漏洞的策略是截然不同的。

典型的小 bug 通常只影□那些□用了某些能□触□它的□□□合的用□。 □□人□最□会在□布没有那个□□的新版之后□出一个□丁来修正它， 而用□中的主体并不会立即升□， 因□他□并没有因存在□□而感到苦□。 □重的 bug 可能会□致数据□失和其它□□， 无□如何， □慎的用□知道， 除了□件 bug 之外□有很多其它事故可能会□致数据□失， 因此他□会□□重要数据； 此外， □重的 bug 通常会被很快□□。

安全漏洞□完全不同。 第一， 它□可能存在数年而不被□□， 因□它□可能并不□致□件无法正常工作。 第二， 通□利用漏洞， □意的一方可能会得到未□授□的□□□限， 并利用□些□限□掉或修改敏感数据； 而更糟□的情况□是用□可能根本注意不到□害已□□生。 第三， 暴露出安全漏洞的系□， 往往能□□助攻□者□入其它之前不可能□入的系□。 因此， 只是修正安全漏洞是不□的： 必□以清晰和全面的方式通知公□， □□他□就能□□估□□，并采取□当的措施。

11.2. 修□安全漏洞

当□起 port 或 package □， 安全漏洞往往是出□在原作者的□行包， 或移植□程中加入的文件里。 □于前一□情况， □件的原作者通常会立刻□布一个□丁甚至新版， □只需要按照原作者的修正去更新 port 就可以了。 如果由于某□原因修正被延□， □要□ 将 port □□□ FORBIDDEN， 要□在 port 中加入一个自己的□丁。 如果有存在漏洞的 port， 尽可能尽快修□其漏洞就是。 无□是□□情况， □□是需要按照 □准的提交流程 提交，除非□有直接在 ports tree 上 commit 的□限。

作□ ports committer 并不能□随便 commit 所有 port。 □注意通常 port 都有□□者，而他□□得到□的尊重。

在漏洞被修正之后， 一定要同□□加 port 的修□版本号。 □□， □律性地升□安装的 package 的用□就能□看到他□需要□行升□。 □外， □□□□□□□的安装包， 并通□ FTP 和 WWW □像□布， 以取代有漏洞的版本。注意要□加 PORTREVISION 数字， 除非在修正□□□ PORTVERSION □生了□化。 一般来□， 如果在 port 中□加了□丁文件， 就□□□加 PORTREVISION， 但例外的例子是□已□将□件升□到了最新版， 因□□□已□改掉 PORTVERSION 了。 □参□ 相□小□ 以了解□一□的信息。

11.3. 通知整个用□群体

11.3.1. VuXML 数据□

当□□了安全漏洞□的一□重要而□迫的□□， 就是□使用 port 的用□群了解其危□。 □□通知有□重目的。 首先，如果危害真的很□重， 可能理性的□法就是立即□用一□□解措施， 例如， 停止受到影□的服□， 甚至完全□除port， 直到□□被修正□止。 其次， □多用□只是偶□升□所安装的□件包， 通□通知， 他□能□知道已□到了 必□更新□件的□候， 因□已□有了修正□些□□的版本了。

由于□有的 port 数量□其□大， □□一个□□都□布安全公告， 毫无疑□地会□表和狼来了一□多的安全公告， 并□大受□在真的□生□重的□□□忽略□□的可能。 因此， 在 port 中□□的安全漏洞， 会在 FreeBSD VuXML 数据□

中□□。 安全官□□成□会持□地追踪□个数据□的修改， 以了解需要他□注意的内容。

如果□是 committer， □可以自行更新 VuXML 数据□。 □□， □就能□同□□助安全官□□， 并尽早将至□重要的信息□□□用□群体。 然而， 如果□不是 committer， 或者□相信自己□□了一个□常□重的漏洞， □不要□豫， 按照 FreeBSD 安全信息 □面上的方法□系安全官□□。

正如其名称所暗示的那□， VuXML 数据□是一个 XML 文□。 其源文件 vuln.xml 被保存在 security/vuxml port 的目□中。 所以， 它的全名是 PORTSDIR/security/vuxml/vuln.xml。 □当□□□ port 中的安全漏洞□，□把新的□□加入到那个文件中。 在熟悉 VuXML 之前， □最好先看看是否有□似的□□□的□□的其它□□， 并□制它作□模板。

11.3.2. VuXML □介

XML 是一个□□的□言， 它□□超越了□本□的□□。 不□， 只需了解□□的命名□□， 就能 VuXML □□的□□有一个大体的了解了。 XML □□的名字□出□在一□尖括号之□。 □一个 <tag> 必□有一个□□的 </tag>。□□可以嵌套， 如果嵌套的□， 内□的□□必□在外□□□之前□束。 □就形成了一个□□的□次□□， 也就是□于它□之□如何嵌套的□□。 听起来很像 HTML， 是不是？ 最主要的区□在于， XML 是可□展的 (e*X*tensible)，例如通□定□新的□□等等。 由于其□□的内在性□， XML 能□□予无□□的数据新的形□。 VuXML 是□□□描述安全漏洞□□的□言。

□在□我□来□察一个□□的 VuXML □□：

```
<vuln vid="f4bc80f4-da62-11d8-90ea-0004ac98a7b9"> ①
  <topic>Several vulnerabilities found in Foo</topic> ②
  <affects>
    <package>
      <name>foo</name> ③
      <name>foo-devel</name>
      <name>ja-foo</name>
      <range><ge>1.6</ge><lt>1.9</lt></range> ④
      <range><ge>2.*</ge><lt>2.4_1</lt></range>
      <range><eq>3.0b1</eq></range>
    </package>
    <package>
      <name>openfoo</name> ⑤
      <range><lt>1.10_7</lt></range> ⑥
      <range><ge>1.2,1</ge><lt>1.3_1,1</lt></range>
    </package>
  </affects>
  <description>
    <body xmlns="http://www.w3.org/1999/xhtml">
      <p>J. Random Hacker reports:</p> ⑦
      <blockquote
        cite="http://j.r.hacker.com/advisories/1">
        <p>Several issues in the Foo software may be exploited
          via carefully crafted QUUX requests.  These requests will
          permit the injection of Bar code, mumble theft, and the
          readability of the Foo administrator account.</p>
      </blockquote>
    </body>
  </description>
  <references> ⑧
    <freebsdsa>SA-10:75.foo</freebsdsa> ⑨
    <freebsdpr>ports/987654</freebsdpr> ⑩
    <cvename>CAN-2010-0201</cvename> ⑪
    <cvename>CAN-2010-0466</cvename>
    <bid>96298</bid> ⑫
    <certsa>CA-2010-99</certsa> ⑬
    <certvu>740169</certvu> ⑭
    <uscertsa>SA10-99A</uscertsa> ⑮
    <uscertta>SA10-99A</uscertta> ⑯
    <mlist
msgid="201075606@hacker.com">http://marc.theaimsgroup.com/?l=bugtraq&amp;m=20388660782
5605</mlist> ⑰
    <url>http://j.r.hacker.com/advisories/1</url> ⑱
  </references>
  <dates>
    <discovery>2010-05-25</discovery> ⑲
    <entry>2010-07-13</entry> ⑳
    <modified>2010-09-17</modified>
  </dates>
</vuln>
```

□□的名字都是□□明了的， 下面我□来介□一下需要由□填写的字段：

① □是 VuXML □□的□□ tag。 它有一个□制性的字段， vid， 用于□此□□ (它包含的部分) 指定一个全局唯一□□符 (UUID)。 □□□□一个新的 vuXML 生成新的 UUID (而且□忘了要把模板中的 UUID □成新的， 如果□不是从□□始的□)。 □可以使用 uuidgen(1) 来生成 VuXML UUID。

② □于□□的一句□描述。

③ 此□□出受到影□的 package 的名字。 可以□出多个名字， 因□可能有多个□件包基于同一个 master port 或□件□品。 □可能包括□定和□□分支、 本地化版本， 以及提供了不同的□□□□□的 slave port。

④ 受影□的 package 版本， 可以使用 <lt>, <le>, <eq>, <ge>, 和 <gt> 表□成一个或多个版本及其□□。 注意□出的版本□□不□存在重□。在描述□□的□候， * (星号) 表□最小的版本。 更具体地□， 2.* 小于 2.a。 因此， 星号可以用来匹配所有可能的 alpha、 beta， 以及 RC 版本。 例如， <ge>2.</ge><lt>3.</lt> 可以□□性地匹配□一个 2.x 的版本， 而 <ge>2.0</ge><lt>3.0</lt> □然不能， 因□它会漏掉 2.r3 而匹配 3.b。上面的例子指定了受影□的版本， 是包括 1.6 到 1.9 上下界的所有版本， 以及 2.x 在 2.4_1 之前的版本， 和 3.0b1 版。

⑤ 受到影□的一□ package (本□上是 ports) 可以列在 <affected> 小□中。 如果多个□件□品都采用了同□的基□代□， (比如□ FooBar、 FreeBar 和 OpenBar) 而且包含同□的 bug 或漏洞。 □注意列出多个名字□，□□在一个 <package> 小□中完成。

⑥ 如果可能， 版本的□□□包括 PORTEPOCH 和 PORTREVISION。 □必注意， 根据加□□□， □有非零 PORTEPOCH 的版本， 系□会□□比没有 PORTEPOCH 的版本高， 例如 3.0,1 高于 3.1 甚至 8.9。

⑦ □于□□的摘要性信息。 此□使用 XHTML。 □必要成□使用 和 。 可以使用□□□□的□□， 但□限于有助于□信息更准□和明了的修□： □不要□分地美化。

⑧ □部分包含了相□的可供参考的文□。 □尽可能多提供参考文献。

⑨ 指定 FreeBSD 安全公告。

⑩ 指定 FreeBSD □□□告。

⑪ 指定 Mitre CVE ID。

⑫ 指定 SecurityFocus Bug ID。

⑬ 指定 US-CERT 安全公告。

⑭ 指定 US-CERT 漏洞□明。

⑮ 指定 US-CERT □算机安全警□。

⑯ 指定 US-CERT 技□性□算机安全警□。

⑰ 指向□件列表存□的 URL。 属性 msgid 是可□□， 用以指定某一封信的 message ID。

⑱ 一般的 URL。 只有在没有其它更□合的参考文献□， 才□使用它。

⑲ □□被全面披露的日期 (*YYYY-MM-DD*)。

⑳ □□加入到数据□中的日期 (*YYYY-MM-DD*)。

□□最后一次被修改的日期 (*YYYY-MM-DD*)。 新□□不□包括□个字段。 只有在修改□□□才□加入它。

11.3.3. □□□□ VuXML 数据□所作的修改

假定□打算撰写， 或已□写好了一个□于 package clamav 的□□描述， 并且， 已□知道 0.65_7 版本修正了□个□□。

□需要做的准□工作， 是 安装 一个新版本的 ports ports-mgmt/portaudit 程序、 ports-mgmt/portaudit-db， 以及 security/vuxml。

要□行 packaudit， □必□□有其 DATABASEDIR， 通常是 /var/db/portaudit 的写入□限。

□可以通□ DATABASEDIR □境□量来指定一个不同的位置。

如果□的工作目□是 ${PORTSDIR}/security/vuxml 以外的其它地方， □□使用□境□量 VUXMLDIR 来指明 vuln.xml 的位置。

首先， □□一下是否已□有了□于□个漏洞的描述。 如果已□有□□□的□□， 那□它将匹配□早版本的 package，0.65_6：

```
% packaudit
% portaudit clamav-0.65_6
```

如果什□都没有□□， □就可以考□写一个新的□□来描述□个漏洞了。 □在可以生成一个新的 UUID (假□它是 74a9541d-5d6c-11d8-80e3-0020ed76ef5a)， 然后将□的新□□加入到 VuXML 数据□中。 接下来，用下面的命令来□□它是否符合□法：

```
% cd ${PORTSDIR}/security/vuxml && make validate
```

□需要安装下列 package 中的至少一个： textproc/libxml2、 textproc/jade。

接下来从 VuXML 文件重□ portaudit 数据□：

```
% packaudit
```

要□□□新加入的□的 <affected> 小□能□正□地匹配希望的 package， 可以使用下面的命令：

```
% portaudit -f /usr/ports/INDEX -r 74a9541d-5d6c-11d8-80e3-0020ed76ef5a
```

□参□ portaudit(1) 以了解□于□个命令□法的更多□□。

□□信新添加的□□不会在□出中匹配不□匹配的□。

□在□□□添加的□□所匹配的版本是否正□：

```
% portaudit clamav-0.65_6 clamav-0.65_7
Affected package: clamav-0.65_6 (matched by clamav<0.65_7)
Type of problem: clamav remote denial-of-service.
Reference: <http://www.freebsd.org/ports/portaudit/74a9541d-5d6c-11d8-80e3-
0020ed76ef5a.html>

1 problem(s) found.
```

□然， 前一个版本会匹配， 而后一个不会。

最后， □□□从 VuXML 数据□中能□正□地得到□期的网□效果：

```
% mkdir -p ~/public_html/portaudit
% packaudit
% lynx ~/public_html/portaudit/74a9541d-5d6c-11d8-80e3-0020ed76ef5a.html
```

Chapter 12. □做什□和不□做什□

12.1. 介□

□里是一些在移植□件□可能会遇到的常□□。 □□按照□个列表□□自己的 port， 同□地， □也可以□助□□ PR 数据□中由其它人提交的 port。 □按照在 □□□告和一般性注□ 中介□的方法提交□的看法。 □助□□ PR 数据□中的 ports 即能□□助我□更快地 commit 它□， 也能□明□清楚地了解如何完成□些工作。

12.2. WRKDIR (□□□使用的□□目□)

任何□候都不要在 WRKDIR 以外的位置写文件。 WRKDIR 是在 port □□□程中唯一的一□一定可写的地方 (参□如何从 CDROM 安装 port 以了解从只□的目□中□□和安装 port 的例子)。 如果□需要改□ pkg-* 文件， □按照重新定□某个□量 介□的方法， 而不是覆□它□来□□。

12.3. WRKDIRPREFIX (用于□□的□□目□的父目□名)

一定要□保□的 port 尊重 WRKDIRPREFIX 的□置。 □大多数 port 并不需要担心□个。 具体□来， 当引用其它 port 的 WRKDIR □， 需要注意正□的位置□□是 WRKDIRPREFIXPORTSDIR/subdir/name/work 而不是 PORTSDIR/subdir/name/work 或 .CURDIR/../../subdir/name/work， 或□的什□。

□外， 如果□自行定□了 WRKDIR， 也要把 ${WRKDIRPREFIX}${.CURDIR} 放到前面。

12.4. 区分不同的操作系□， 以及 OS 的版本

在不同版本的 Unix 下可能需要□代□□行一些修改或□加少□□□□， 才能□正□地□□和□行。 如果□需要根据一些条件来□代□□行修改， □尽可能□□些修改通用， □□， 我□就能□将□些代□移植回更早的 FreeBSD 系□， 并交叉移植到其它 BSD 系□， 例如来自 CSRG 的 4.4BSD， BSD/386， 386BSD， NetBSD 和 OpenBSD。

推□的□得 4.3BSD/Reno (1990) 以及更新版本 BSD 代□版本号的方式， 是使用 sys/param.h 中所定□的 BSD 宏的□。 一般来□□个文件已□被引用了； 如果没有的□， □加下述代□：

```
#if (defined(__unix__) || defined(unix)) && !defined(USG)
#include <sys/param.h>
#endif
```

到 .c 文件中合□的地方。 我□相信所有定□了□□个符号的系□中， 都提供了 sys/param.h。 如果□□□有不□□做的系□， □通□致信 FreeBSD ports □件列表 □我□了解□一情况。

□一□方法是使用 GNU Autoconf □格的方式：

```
#ifdef HAVE_SYS_PARAM_H
#include <sys/param.h>
#endif
```

采用□□方法□， 不要忘了把 -DHAVE_SYS_PARAM_H 加到 Makefile 中的 CFLAGS 里。

一旦引用了 sys/param.h， □就可以使用：

```
#if (defined(BSD) && (BSD >= 199103))
```

来□□代□是否正在 4.3 Net2 代□基□， 或更新的系□上□□ (例如 FreeBSD 1.x， 4.3/Reno， NetBSD 0.9， 386BSD， BSD/386 1.1 以及更高版本)。

使用：

```
#if (defined(BSD) && (BSD >= 199306))
```

来□□代□是否正在 4.4 或更新的系□ (例如 FreeBSD 2.x， 4.4， NetBSD 1.0、 BSD/386 2.0 或更高版本)。

□于 4.4BSD-Lite2 代□系来□， BSD 宏的□□□是 199506。 □里只是作□信息提供， □不□使用它来区分基于 4.4-Lite 的 FreeBSD 和基于 4.4-Lite2 的版本。 □些情况下， □□使用 {freebsd} 宏。

保守地使用：

- {freebsd} 在所有版本的 FreeBSD 中皆有定□。 如果□正□行的修改 只 影□ FreeBSD， □□使用□个宏。 □似 sys_errlist[] 之于 strerror() □□的移植□□是伯克利代□系公用的， 而并非 FreeBSD 所□有。
- 在 FreeBSD 2.x 中， {freebsd} 定□□ 2。 更早版本中， 它曾□是 1。 新的版本都会在主要的版本号□化□□更它。
- 如果□需要区分 FreeBSD 1.x 系□和 FreeBSD 2.x 及更高版本的区□， 通常□使用前述的 BSD 宏来□行。 如果事□上需要一个 FreeBSD □有的修改 (例如， 在使用 ld □需要特殊的共享□□□)， □可以用 {freebsd} 和 #if {freebsd} > 1 来□□ FreeBSD 2.x 和新系□上的□化。 如果需要更□粒度地□□ FreeBSD 2.0-RELEASE 之后版本的□化， □可以使用：

  ```
  #if __FreeBSD__ >= 2
  #include <osreldate.h>
  #    if __FreeBSD_version >= 199504
       /* □用于 2.0.5+ 版本的代□ */
  #    endif
  #endif
  ```

在已有的数百个 port 中， 只有一□个□□使用 {freebsd}。 早期的 port 在不□当的地方使用了它并引□□□， 并不意味着□也必定如此。

12.5. __FreeBSD_version □

下面是在 sys/param.h__FreeBSD_version 中定□的□及其意□的列表， □里□出以方便□□□：

表 *48. __FreeBSD_version* □

□	日期	版本
119411		2.0-RELEASE
199501, 199503	March 19, 1995	2.1-CURRENT
199504	April 9, 1995	2.0.5-RELEASE
199508	August 26, 1995	2.1 之前的 2.2-CURRENT
199511	November 10, 1995	2.1.0-RELEASE
199512	November 10, 1995	2.1.5 之前的 2.2-CURRENT
199607	July 10, 1996	2.1.5-RELEASE
199608	July 12, 1996	2.1.6 之前的 2.2-CURRENT
199612	November 15, 1996	2.1.6-RELEASE
199612		2.1.7-RELEASE
220000	February 19, 1997	2.2-RELEASE
(not changed)		2.2.1-RELEASE
(无□化)		在 2.2.1-RELEASE 之后的 2.2-STABLE
221001	April 15, 1997	texinfo-3.9 之后的 2.2-STABLE
221002	April 30, 1997	top 之后的 2.2-STABLE
222000	May 16, 1997	2.2.2-RELEASE
222001	May 19, 1997	2.2.2-RELEASE 之后的 2.2-STABLE
225000	October 2, 1997	2.2.5-RELEASE
225001	November 20, 1997	2.2.5-RELEASE 之后的 2.2-STABLE
225002	December 27, 1997	合并 ldconfig -R 之后的 2.2-STABLE
226000	March 24, 1998	2.2.6-RELEASE
227000	July 21, 1998	2.2.7-RELEASE
227001	July 21, 1998	2.2.7-RELEASE 之后的 2.2-STABLE
227002	September 19, 1998	semctl(2) 修改之后的 2.2-STABLE
228000	November 29, 1998	2.2.8-RELEASE
228001	November 29, 1998	2.2.8-RELEASE 之后的 2.2-STABLE
300000	February 19, 1996	mount(2) 修改之前的 3.0-CURRENT

□	日期	版本
300001	September 24, 1997	mount(2) 修改之后的 3.0-CURRENT
300002	June 2, 1998	semctl(2) 修改之后的 3.0-CURRENT
300003	June 7, 1998	ioctl 参数□化之后的 3.0-CURRENT
300004	September 3, 1998	ELF □□之后的 3.0-CURRENT
300005	October 16, 1998	3.0-RELEASE
300006	October 16, 1998	3.0-RELEASE 之后的 3.0-CURRENT
300007	January 22, 1999	3/4切分之后的 3.0-STABLE
310000	February 9, 1999	3.1-RELEASE
310001	March 27, 1999	3.1-RELEASE 之后的 3.1-STABLE
310002	April 14, 1999	C++ □建/析□函数□序□化之后的 3.1-STABLE
320000		3.2-RELEASE
320001	May 8, 1999	3.2-STABLE
320002	August 29, 1999	二□制不兼容的 IPFW 和 socket □化之后的 3.2-STABLE
330000	September 2, 1999	3.3-RELEASE
330001	September 16, 1999	3.3-STABLE
330002	November 24, 1999	libc 中加入 mkstemp(3) 之后的 3.3-STABLE
340000	December 5, 1999	3.4-RELEASE
340001	December 17, 1999	3.4-STABLE
350000	June 20, 2000	3.5-RELEASE
350001	July 12, 2000	3.5-STABLE
400000	January 22, 1999	3/4切分之后的 4.0-CURRENT
400001	February 20, 1999	修改□□□接器□理方式之后的 4.0-CURRENT
400002	March 13, 1999	C++ □建/析□函数□序□化之后的
400003	March 27, 1999	提供 dladdr(3) 之后的 4.0-CURRENT
400004	April 5, 1999	修正了 __deregister_frame_info 的 4.0-CURRENT (也表示在 EGCS 1.1.2 集成之后的 4.0-CURRENT)

□	日期	版本
400005	April 27, 1999	suser(9) API □化之后的 4.0-CURRENT (也表示 newbus 之后的 4.0-CURRENT)
400006	May 31, 1999	cdevsw 注册机制改□之后的 4.0-CURRENT
400007	June 17, 1999	加入了 socket □凭据的 so_cred 之后的 4.0-CURRENT
400008	June 20, 1999	在 libc_r 中加入 poll 系□□用接口之后的 4.0-CURRENT
400009	July 20, 1999	将内核中 `dev_t` □型改□ `struct specinfo` 指□之后的 4.0-CURRENT
400010	September 25, 1999	修正了一□ jail(2) 漏洞之后的 4.0-CURRENT
400011	September 29, 1999	`sigset_t` 数据□型改□之后的 4.0-CURRENT
400012	November 15, 1999	切□到 GCC 2.95.2 □□器之后的 4.0-CURRENT
400013	December 4, 1999	加入了可□的 linux模式 ioctl □理程序后的 4.0-CURRENT
400014	January 18, 2000	引入 OpenSSL 之后的 4.0-CURRENT
400015	January 27, 2000	GCC 2.95.2 中 ABI 默□□从 -fvtable-thunks 改□ -fno-vtable-thunks 之后的 4.0-CURRENT
400016	February 27, 2000	引入 OpenSSH 之后的 4.0-CURRENT
400017	March 13, 2000	4.0-RELEASE
400018	March 17, 2000	4.0-RELEASE 之后的 4.0-STABLE
400019	May 5, 2000	引入延□校□和之后的 4.0-STABLE
400020	June 4, 2000	将 libxpg4 的代□并入 libc 之后的 4.0-STABLE
400021	July 8, 2000	Binutils 升□到 2.10.0 之后的 4.0-STABLE， ELF □志□化， 以及将 tcsh 引入基本系□
410000	July 14, 2000	4.1-RELEASE
410001	July 29, 2000	4.1-RELEASE 之后的 4.1-STABLE
410002	September 16, 2000	setproctitle(3) 从 libutil 移入 libc 之后的 4.1-STABLE

□	日期	版本
411000	September 25, 2000	4.1.1-RELEASE
411001		4.1.1-RELEASE 之后的 4.1.1-STABLE
420000	October 31, 2000	4.2-RELEASE
420001	January 10, 2001	合并 libgcc.a 和 libgcc_r.a，并修改了相□的 GCC □接方式之后的 4.2-STABLE
430000	March 6, 2001	4.3-RELEASE
430001	May 18, 2001	引入 wint_t 之后的 4.3-STABLE
430002	July 22, 2001	PCI □源状□ API 合并之后的 4.3-STABLE
440000	August 1, 2001	4.4-RELEASE
440001	October 23, 2001	引入 d_thread_t 之后的 4.4-STABLE
440002	November 4, 2001	mount □□改□之后的 4.4-STABLE (影□文件系□ kld)
440003	December 18, 2001	用□□部分的 smbfs 被引入之后的 4.4-STABLE
450000	December 20, 2001	4.5-RELEASE
450001	February 24, 2002	usb □□元素改名之后的 4.5-STABLE
450004	April 16, 2002	在 rc.conf(5) □量 `sendmail_enable` 默□□改□ `NONE` 之后的 4.5-STABLE
450005	April 27, 2002	默□将 XFree86 4 用于□□□包 □□之后的 4.5-STABLE
450006	May 1, 2002	accept □□器修正了安全 □□并且不再会□易被 DoS 之后的 4.5-STABLE
460000	June 21, 2002	4.6-RELEASE
460001	June 21, 2002	修正了 sendfile(2) 以吻合文□，而不再根据□出的□□算□出数据量之后的 4.6-STABLE
460002	July 19, 2002	4.6.2-RELEASE
460100	June 26, 2002	4.6-STABLE
460101	June 26, 2002	MFC `sed -i` 之后的 4.6-STABLE
460102	September 1, 2002	MFC □多 pkg_install 新特性之后的 4.6-STABLE

□	日期	版本
470000	October 8, 2002	4.7-RELEASE
470100	October 9, 2002	4.7-STABLE
470101	November 10, 2002	□始生成 *std{in,out,err}p* 引用，而不是 sF。 □将 std{in,out,err} 从□□□表□式□成了□行□□。
470102	January 23, 2003	MFC mbuf 相□的将 m_aux mbuf 改□ m_tag 的修改之后的 4.7-STABLE
470103	February 14, 2003	OpenSSL 升□到 0.9.7 之后的 4.7-STABLE
480000	March 30, 2003	4.8-RELEASE
480100	April 5, 2003	4.8-STABLE
480101	May 22, 2003	realpath(3) □□□程安全的之后的 4.8-STABLE
480102	August 10, 2003	□ twe 的 3ware API 修改之后的 4.8-STABLE
490000	October 27, 2003	4.9-RELEASE
490100	October 27, 2003	4.9-STABLE
490101	January 8, 2004	kinfo_eproc 中加入 e_sid 之后的 4.9-STABLE
490102	February 4, 2004	MFC rtld 的 libmap 功能之后的 4.9-STABLE
491000	May 25, 2004	4.10-RELEASE
491100	June 1, 2004	4.10-STABLE
491101	August 11, 2004	MFC 20040629 版本的包 □□工具之后的 4.10-STABLE
491102	November 16, 2004	修正了 VM 当解除 wire 不存在□面□的□□之后的 4.10-STABLE
492000	December 17, 2004	4.11-RELEASE
492100	December 17, 2004	4.11-STABLE
492101	April 18, 2006	将 libdata/ldconfig 目□加入 mtree 文件之后的 4.11-STABLE。
500000	March 13, 2000	5.0-CURRENT
500001	April 18, 2000	加入 ELF □字段， 并改□我□的 ELF □行文件□□方式之后的 5.0-CURRENT

□	日期	版本
500002	May 2, 2000	kld 元数据修改之后的 5.0-CURRENT
500003	May 18, 2000	buf/bio 修改之后的 5.0-CURRENT
500004	May 26, 2000	binutils 升□后的 5.0-CURRENT
500005	June 3, 2000	将 libxpg4 并入 libc， 以及引入 TASKQ 之后的 5.0-CURRENT
500006	June 10, 2000	加入 AGP 接口之后的 5.0-CURRENT
500007	June 29, 2000	Perl 升□到 5.6.0 之后的 5.0-CURRENT
500008	July 7, 2000	KAME 代□升□到 2000/07 之后的 5.0-CURRENT
500009	July 14, 2000	ether_ifattach() 和 ether_detach() 修改之后的 5.0-CURRENT
500010	July 16, 2000	将 mtree 改□原先的默□□， 并使用 -L 来跟随符号□接之后的 5.0-CURRENT
500011	July 18, 2000	kqueue API 修改之后的 5.0-CURRENT
500012	September 2, 2000	setproctitle(3) 从 libutil □到 libc 之后的 5.0-CURRENT
500013	September 10, 2000	首个 SMPng commit 之后的 5.0-CURRENT
500014	January 4, 2001	<sys/select.h> 改□ <sys/selinfo.h> 之后的 5.0-CURRENT
500015	January 10, 2001	libgcc.a 和 libgcc_r.a 以及 GCC □接方式□□之后的 5.0-CURRENT
500016	January 24, 2001	修改以允□ libc 和 libc_r □接到一起， 不再鼓励使用 -pthread 之后的 5.0-CURRENT
500017	February 18, 2001	从 struct ucred 切□到 struct xucred 以便使内核□ mountd 等程序□出的 API □定下来之后的 5.0-CURRENT
500018	February 24, 2001	加入 CPUTYPE 用于 CPU □用的 □化的 make □量之后的 5.0-CURRENT

□	日期	版本
500019	June 9, 2001	machine/ioctl_fd.h 改□ sys/fdcio.h 之后的 5.0-CURRENT
500020	June 15, 2001	locale 名称改□之后的 5.0-CURRENT
500021	June 22, 2001	引入 bzip2 之后的 5.0-CURRENT， 同□也代表□去了 S/Key
500022	July 12, 2001	加入 SSE 支持之后的 5.0-CURRENT
500023	September 14, 2001	KSE 第2个里程碑之后的 5.0-CURRENT
500024	October 1, 2001	d_thread_t 之后的 5.0-CURRENT， 同□ UUCP 被移入 ports
500025	October 4, 2001	64-位平台上的描述符和 creds API □化之后的 5.0-CURRENT
500026	October 9, 2001	采用 XFree86 4 作□默□的□□□包，以及加入 strnstr() libc 函数之后的 5.0-CURRENT
500027	October 10, 2001	加入 strcasestr() libc 函数之后的 5.0-CURRENT
500028	December 14, 2001	引入了用□□的 smbfs □件之后的 5.0-CURRENT
(未予□加)		加入了新的 C99 指定位□整形 □量之后的 5.0-CURRENT
500029	January 29, 2002	修改了 sendfile(2) 的返回□之后的 5.0-CURRENT
500030	February 15, 2002	引入□合表□文件□志的 `fflags_t` □型之后的 5.0-
500031	February 24, 2002	usb □□元素改名之后的 5.0-CURRENT
500032	March 16, 2002	引入 Perl 5.6.1 之后的 5.0-CURRENT
500033	April 3, 2002	rc.conf(5) □量 `sendmail_enable` 默□□改□ `NONE` 之后的 5.0-CURRENT
500034	April 30, 2002	mtx_init() □加了第三个参数之后的 5.0-CURRENT

□	日期	版本
500035	May 13, 2002	包含 Gcc 3.1 的 5.0-CURRENT
500036	May 17, 2002	在 /usr/src 中□去了 Perl 的 5.0-CURRENT
500037	May 29, 2002	加入 dlfunc(3) 之后的 5.0-CURRENT
500038	July 24, 2002	一些 struct sockbuf 的成□□□□□，并重新排列□序之后的 5.0-CURRENT
500039	September 1, 2002	引入 GCC 3.2.1 之后的 5.0-CURRENT。 □文件也不再使用 *BSD_FOO_T* 而□始使用 _FOO_T_DECLARED。 □个□□可以用于作□一个包含使用 bzip2(1) 的□□□包支持的□期点。
500040	September 20, 2002	以去掉□ disklabel □□内容的依□的名□， □磁□相□的函数□行了□多修改之后的 5.0-CURRENT
500041	October 1, 2002	libc 中加入 getopt_long(3) 之后的 5.0-CURRENT
500042	October 15, 2002	Binutils 2.13 升□， 包含了新的 FreeBSD 模□， vec 以及□出格式之后的 5.0-CURRENT
500043	November 1, 2002	libc 中加入了弱 pthread_XXX 符号之后的 5.0-CURRENT，从而淘汰了 libXThrStub.so。 5.0-RELEASE。
500100	January 17, 2003	□建 RELENG_5_0 分支之后的 5.0-CURRENT
500101	February 19, 2003	<sys/dkstat.h> □成了一个空文件， 不□再被引用
500102	February 25, 2003	修改 d_mmap_t 接口之后的 5.0-CURRENT
500103	February 26, 2003	taskqueue_swi 以无全局□的方式□行之后的 5.0-CURRENT， 同□□加入了使用全局□的 taskqueue_swi_giant
500104	February 27, 2003	去掉了 cdevsw_add() 和 cdevsw_remove() 出□ MAJOR_AUTO 分配机制

□	日期	版本
500105	March 4, 2003	采用新的 cdevsw 初始化方法之后的 5.0-CURRENT
500106	March 8, 2003	devstat_add_entry() 被 devstat_new_entry() 取代
500107	March 15, 2003	修改 devstat 接口；□参□ sys/sys/param.h 1.149
500108	March 15, 2003	改□了 Token-Ring 接口
500109	March 25, 2003	加入 vm_paddr_t
500110	March 28, 2003	将 realpath(3) 改□□程安全之后的 5.0-CURRENT
500111	April 9, 2003	usbhid(3) 与 NetBSD 同□之后的 5.0-CURRENT
500112	April 17, 2003	加入新的 NSS □□， 以及 POSIX.1 getpw*_r, getgr*_r 函数之后的 5.0-CURRENT
500113	May 2, 2003	□去旧式 rc 系□之后的 5.0-CURRENT
501000	June 4, 2003	5.1-RELEASE.
501100	June 2, 2003	□建 RELENG_5_1 分支之后的 5.1-CURRENT
501101	June 29, 2003	改正 sigtimedwait(2) 和 sigwaitinfo(2) □□之后的 5.1-CURRENT
501102	July 3, 2003	在 bus_dma_tag_create(9) 中加入了 lockfunc 和 lockfuncarg 字段之后的 5.1-CURRENT
501103	July 31, 2003	集成了 GCC 3.3.1-pre 20030711 之后的 5.1-CURRENT
501104	August 5, 2003	twe 中 3ware API □化之后的 5.1-CURRENT
501105	August 17, 2003	允□□□□接 /bin 和 /sbin，以及将某些□移□到 /lib 之后的 5.1-CURRENT
501106	September 8, 2003	□加内核□ Coda 6.x 支持之后的 5.1-CURRENT

□	日期	版本
501107	September 17, 2003	将 16550 UART 常量从 <dev/sio/sioreg.h> □到 <dev/ic/ns16550.h> 之后的 5.1-CURRENT。 此外， rtld 也从此无条件支持 libmap 功能
501108	September 23, 2003	更新 PFIL_HOOKS API 之后的 5.1-CURRENT
501109	September 27, 2003	□加 kiconv(3) 之后的 5.1-CURRENT
501110	September 28, 2003	默□的 cdevsw open 和 close 操作□化之后的 5.1-CURRENT
501111	October 16, 2003	cdevsw 的布局□化之后的 5.1-CURRENT
501112	October 16, 2003	□加 kobj 多□承之后的 5.1-CURRENT
501113	October 31, 2003	修改 struct ifnet 中的 if_xname 之后的 5.1-CURRENT
501114	November 16, 2003	将 /bin 和 /sbin 改□□□□接之后的 5.1-CURRENT
502000	December 7, 2003	5.2-RELEASE
502010	February 23, 2004	5.2.1-RELEASE
502100	December 7, 2003	□建 RELENG_5_2 分支之后的 5.2-CURRENT
502101	December 19, 2003	libc 中加入了 *cxa_atexit*/cxa_finalize □个函数之后的 5.2-CURRENT
502102	January 30, 2004	默□□程□从 libc_r 改□ libpthread 之后的 5.2-CURRENT
502103	February 21, 2004	□□□□ API 大□模翻修之后的 5.2-CURRENT
502104	February 25, 2004	□加 getopt_long_only() 之后的 5.2-CURRENT
502105	March 5, 2004	C 的 NULL 定□改□ ((void *)0) 之后的 5.2-CURRENT， □会 □生更多的□□警告
502106	March 8, 2004	pf □入□□和安装□程之后的 5.2-CURRENT
502107	March 10, 2004	在 sparc64 上将 time_t 改□ 64-位 □之后的 5.2-CURRENT

□	日期	版本
502108	March 12, 2004	在一些□文件修改以支持 Intel C/C++ □□器， 以及□ execve(2) 更□格地符合 POSIX 之后的 5.2-CURRENT
502109	March 22, 2004	引入 bus_alloc_resource_any API 之后的 5.2-CURRENT
502110	March 27, 2004	加入 UTF-8 locale 之后的 5.2-CURRENT
502111	April 11, 2004	□去 getvfsent(3) API 之后的 5.2-CURRENT
502112	April 13, 2004	□ make(1) □加 .warning □句之后的 5.2-CURRENT
502113	June 4, 2004	所有串口□□都□制使用 ttyioctl() 之后的 5.2-CURRENT
502114	June 13, 2004	引入 ALTQ 框架之后的 5.2-CURRENT
502115	June 14, 2004	修改 sema_timedwait(9) 使其成功□返回 0， 失□□返回非 0 的□□代□之后的 5.2-CURRENT
502116	June 16, 2004	将内核 dev_t 改□指向 struct cdev * 的指□之后的 5.2-CURRENT
502117	June 17, 2004	将内核 udev_t 改□ dev_t 之后的 5.2-CURRENT
502118	June 17, 2004	□ clock_gettime(2) 和 clock_getres(2) □加 CLOCK_VIRTUAL 和 CLOCK_PROF 支持之后的 5.2-CURRENT
502119	June 22, 2004	□网□接口□制□行全面修改之后的 5.2-CURRENT
502120	July 2, 2004	package 工具升□□ 20040629 之后的 5.2-CURRENT
502121	July 9, 2004	不再将□牙代□□□□ i386 □用之后的 5.2-CURRENT
502122	July 11, 2004	引入 KDB □□器框架之后的 5.2-CURRENT。 同□□引入了 DDB 作□后台， 以及 GDB 后台。

□	日期	版本
502123	July 12, 2004	修改 VFS_ROOT 和 vflush 使其使用一个 struct thread 参数之后的 5.2-CURRENT。 struct kinfo_proc □加了一个用 □数据指□。 同□， 默□的 X □□切□□ xorg
502124	July 24, 2004	将使用 rc.d 和□□脚本的 port 分□□□之后的 5.2-CURRENT
502125	July 28, 2004	取消前一修改之后的 5.2-CURRENT
502126	July 31, 2004	□除 kmem_alloc_pageable() 并引入 gcc 3.4.2 的 5.2-CURRENT
502127	August 2, 2004	修改 UMA 内核 API 允 □□建函数和初始化失□之后的 5.2-CURRENT
502128	August 8, 2004	vfs_mount □名和全局替□ suser(9) API 的 PRISON_ROOT □ SUSER_ALLOWJAIL 之后的 5.2-CURRENT
503000	August 23, 2004	pfil API 修改之前的 5.3-BETA/RC
503001	September 22, 2004	5.3-RELEASE
503100	October 16, 2004	□建 RELENG_5_3 分支之后的 5.3-STABLE
503101	December 3, 2004	加入了 glibc □格的 strftime(3) 填充□□的 5.3-STABLE
503102	February 13, 2005	MFC OpenBSD 的 nc(1) 之后的 5.3-STABLE
503103	February 27, 2005	在 MFC 了 <src/include/stdbool.h> 和 <src/sys/i386/include/_types.h> 用于兼容 GCC 和 Intel C/C++ □□器的修正之后的 5.4-PRERELEASE
503104	February 28, 2005	MFC 了将 ifi_epoch 由 wall □□□□改□ uptime 之后的 5.4-PRERELEASE
503105	March 2, 2005	MFC 了 vswprintf(3) 中的 EOVERFLOW □□的 5.4-PRERELEASE
504000	April 3, 2005	5.4-RELEASE.

□	日期	版本
504100	April 3, 2005	□建 RELENG_5_4 分支之后的 5.4-STABLE
504101	May 11, 2005	加大默□□程堆□尺寸之后的 5.4-STABLE
504102	June 24, 2005	加入 sha256 之后的 5.4-STABLE
504103	October 3, 2005	MFC if_bridge 之后的 5.4-STABLE
504104	November 13, 2005	bsdiff 和 portsnap MFC 之后的 5.4-STABLE
504105	January 17, 2006	在 MFC 了 ldconfig_local_dirs 修改之后的 5.4-STABLE。
505000	May 12, 2006	5.5-RELEASE.
505100	May 12, 2006	在□建 RELENG_5_5 分支之后的 5.5-STABLE
600000	August 18, 2004	6.0-CURRENT
600001	August 27, 2004	内核中永久性□用 PFIL_HOOKS 之后的 6.0-CURRENT
600002	August 30, 2004	最初将 ifi_epoch 加入 if_data □□之后的 6.0-CURRENT。 此后不久即被撤□。 □不要使用□个□。
600003	September 8, 2004	if_data 中再次加入 ifi_epoch 成□之后的 6.0-CURRENT
600004	September 29, 2004	将 struct inpcb 参数加入 pfil API 之后的 6.0-CURRENT
600005	October 5, 2004	newsyslog 加入了 "-d DESTDIR" 参数之后的 6.0-CURRENT
600006	November 4, 2004	加入了 glibc □格的 strftime(3) 填充□□之后的 6.0-CURRENT
600007	December 12, 2004	加入了 802.11 框架更新之后的 6.0-CURRENT
600008	January 25, 2005	修改 VOP_*VOBJECT() 并□ 无全局□的文件系□引入 MNTK_MPSAFE □志之后的 6.0-CURRENT
600009	February 4, 2005	加入 cpufreq 框架和□□之后的 6.0-CURRENT
600010	February 6, 2005	引入 OpenBSD 的 nc(1) 之后的 6.0-CURRENT

□	日期	版本
600011	February 12, 2005	□去并不存在的 SVID2 `matherr()` 支持之后的 6.0-CURRENT
600012	February 15, 2005	□大默□□程堆□尺寸之后的 6.0-CURRENT
600013	February 19, 2005	□加了□□ <src/include/stdbool.h> 和 <src/sys/i386/include/_types.h> 的用于 Intel C/C++ □□器的 GCC-兼容性修正。
600014	February 21, 2005	修正了 vswprintf(3) 的 EOVERFLOW □□之后的 6.0-CURRENT
600015	February 25, 2005	将 struct if_data 成□ ifi_epoch 从 wall □□□□改□ uptime 之后的 6.0-CURRENT
600016	February 26, 2005	修改 LC_CTYPE 磁□格式之后的 6.0-CURRENT
600017	February 27, 2005	修改 NLS □□磁□格式之后的 6.0-CURRENT
600018	February 27, 2005	修改 LC_COLLATE 磁□格式之后的 6.0-CURRENT
600019	February 28, 2005	将 acpica □文件安装到 /usr/include
600020	March 9, 2005	□ send(2) API 加入了 MSG_NOSIGNAL
600021	March 17, 2005	在 cdevsw 上□加了一些字段
600022	March 21, 2005	基本系□中□去了 gtar
600023	April 13, 2005	unix(4) 中加入了 LOCAL_CREDS, LOCAL_CONNWAIT □个 socket □□
600024	April 19, 2005	加入了 hwpmc(4) 及其相□工具之后的 6.0-CURRENT
600025	April 26, 2005	加入 struct icmphdr 之后的 6.0-CURRENT
600026	May 3, 2005	pf 更新到了 3.7
600027	May 6, 2005	引入了内核 libalias 和 ng_nat
600028	May 13, 2005	将 ttyname_r(3) 接口改□符合 POSIX □准， 并通□ unistd.h 和 libc

□	日期	版本
600029	May 29, 2005	将 libpcap 升□□ v0.9.1 alpha 096 之后的 6.0-CURRENT
600030	June 5, 2005	引入 NetBSD 的 if_bridge(4) 之后的 6.0-CURRENT
600031	June 10, 2005	将 struct ifnet 从□□的 softc 中拆出之后的 6.0-CURRENT。
600032	July 11, 2005	引入了 libpcap v0.9.1 之后的 6.0-CURRENT。
600033	July 25, 2005	所有自 RELENG_5 以来没有修改□的共享□的版本□□之后的 6.0-STABLE。
600034	August 13, 2005	□ dev_clone 事件□理函数□加身□信息参数之后的 6.0-STABLE。6.0-RELEASE。
600100	November 1, 2005	6.0-RELEASE 之后的 6.0-STABLE
600101	December 21, 2005	将 local_startup 目□中的脚本集成到基本系□的 rcorder(8) 之后的 6.0-STABLE。
600102	December 30, 2005	更新 ELF □型和常量之后的 6.0-STABLE。
600103	January 15, 2006	MFC 了 pidfile(3) API 之后的 6.0-STABLE。
600104	January 17, 2006	在 MFC 了 ldconfig_local_dirs 修改之后的 6.0-STABLE。
600105	February 26, 2006	在 csh(1) 中加入了 NLS 目□支持之后的 6.0-STABLE。
601000	May 6, 2006	6.1-RELEASE
601100	May 6, 2006	6.1-RELEASE 之后的 6.1-STABLE。
601101	June 22, 2006	引入 csup 之后的 6.1-STABLE。
601102	July 11, 2006	更新了 iwi(4) 之后的 6.1-STABLE。
601103	July 17, 2006	将域名解析函数更新至 BIND9，并□出了可重入版本的 netdb 函数之后的 6.1-STABLE。
601104	August 8, 2006	在 OpenSSL 中□用了 DSO (□□共享□) 支持之后的 6.1-STABLE。

□	日期	版本
601105	September 2, 2006	由于 802.11 修正□□了 IEEE80211_IOC_STA_INFO ioctl API 之后的 6.1-STABLE。
602000	November 15, 2006	6.2-RELEASE
602100	September 15, 2006	6.2-RELEASE 之后的 6.2-STABLE。
602101	December 12, 2006	加入 Wi-Spy quirk 之后的 6.2-STABLE。
602102	December 28, 2006	□加 pci_find_extcap() 之后的 6.2-STABLE。
602103	January 16, 2007	MFC 了□ dlsym □行修改，使其在指定 dso 及其暗指的依□中□□符号之后的 6.2-STABLE。
602104	January 28, 2007	MFC 了 netgraph □点 ng_deflate(4) 和 ng_pred1(4) 以及用于 ng_ppp(4) □点的新□□及加密模式之后的 6.2-STABLE。
602105	February 20, 2007	MFC 了从 NetBSD 移植的 BSD 授□的 gzip(1) 之后的 6.2-STABLE。
602106	March 31, 2007	MFC 了 PCI MSI 和 MSI-X 支持之后的 6.2-STABLE。
602107	April 6, 2007	MFC 了包含□字符支持的 ncurses 5.6 之后的 6.2-STABLE。
602108	April 11, 2007	MFC 了□□ Linux SCSI SG 直通□□ API 子集的 CAM 'SG' □□之后的 6.2-STABLE。
602109	April 17, 2007	MFC 了 readline 5.2 patchset 002 之后的 6.2-STABLE。
602110	May 2, 2007	MFC 了用于 amd64 和 i386 的 pmap_invalidate_cache()、pmap_change_attr()、pmap_mapbios()、pmap_mapdev_attr()、 and pmap_unmapbios() 之后的 6.2-STABLE。
602111	June 11, 2007	由于 MFC 了 BOP_BDFLUSH □致文件系□模□ KBI □化之后的 6.2-STABLE。

□	日期	版本
602112	September 21, 2007	一系列 libutil(3) MFC 之后的 6.2-STABLE。
602113	October 25, 2007	MFC 了□字符和□字□ ctype 函数分拆之后的 6.2-STABLE。新□□的引用了 ctype.h 的可□行文件， 可能会需要一个在旧系□上不存在的新□符号 __mb_sb_limit。
602114	October 30, 2007	恢□了 ctype ABI 向前兼容性之后的 6.2-STABLE。
602115	November 21, 2007	回退了□字符和□字□ ctype 分拆之后的 6.2-STABLE。
603000	November 25, 2007	6.3-RELEASE
603100	November 25, 2007	在 6.3-RELEASE 之后的 6.3-STABLE。
603101	December 7, 2007	修正了 bit macro 的多字□支持之后的 6.3-STABLE。
603102	April 24, 2008	□ flock □□加入 l_sysid 之后的 6.3-STABLE。
603103	May 27, 2008	MFC 了 `memrchr` 函数之后的 6.3-STABLE。
603104	June 15, 2008	□ make(1) MFC `:u` □量修□符之后的 6.3-STABLE。
604000	October 4, 2008	6.4-RELEASE
604100	October 4, 2008	6.4-RELEASE 之后的 6.4-STABLE。
700000	July 11, 2005	7.0-CURRENT。
700001	July 23, 2005	所有自 RELENG_5 以来没有修改□的共享□的版本□□之后的 7.0-CURRENT。
700002	August 13, 2005	□ dev_clone 事件□理函数中□加身□信息参数之后的 7.0-CURRENT。
700003	August 25, 2005	将 memmem(3) 加入 libc 之后的 7.0-CURRENT。
700004	October 30, 2005	将 solisten(9) 改□接受一 backlog 参数之后的 7.0-CURRENT。
700005	November 11, 2005	将 IFP2ENADDR() 改□返回一 IF_LLADDR() 指□之后的 7.0-CURRENT。

□	日期	版本
700006	November 11, 2005	在 `struct ifnet` 中□加 `if_addr` 成□， 并□除 IFP2ENADDR() 之后的 7.0-CURRENT。
700007	December 2, 2005	将 local_startup 目□中的脚本集成到基本系□的 rcorder(8) 之后的 7.0-CURRENT。
700008	December 5, 2005	去掉 MNT_NODEV 挂接□□之后的 7.0-CURRENT。
700009	December 19, 2005	□ ELF-64 □型和符号版本□行□更之后的 7.0-CURRENT。
700010	December 20, 2005	□加 hostb 和 vgapci □□、pci_find_extcap()， 并将 AGP □□改□不再影射 aperature 之后的 7.0-CURRENT。
700011	December 31, 2005	除 Alpha 之外的所有平台上 tv_sec 改□ time_t 之后的 7.0-CURRENT。
700012	January 8, 2006	修改 ldconfig_local_dirs 之后的 7.0-CURRENT。
700013	January 12, 2006	在修改了 /etc/rc.d/abi 以支持 /compat/linux/etc/ld.so.cache 以某只□文件系□上的符号□接形式存在之后的 7.0-CURRENT。
700014	January 26, 2006	引入 pts 之后的 7.0-CURRENT。
700015	March 26, 2006	在引入 hwpmc(4) 的第 2 版 ABI 之后的 7.0-CURRENT。
700016	April 22, 2006	在 libc 中加入了 fcloseall(3) 之后的 7.0-CURRENT。
700017	May 13, 2006	□去 ip6fw 之后的 7.0-CURRENT。
700018	July 15, 2006	引入了 snd_emu10kx 之后的 7.0-CURRENT。
700019	July 29, 2006	引入了 OpenSSL 0.9.8b 之后的 7.0-CURRENT。
700020	September 3, 2006	□加了 bus_dma_get_tag 函数之后的 7.0-CURRENT。
700021	September 4, 2006	在引入了 libpcap 0.9.4 和 tcpdump 3.9.4 之后的 7.0-CURRENT。

□	日期	版本
700022	September 9, 2006	在□ dlsym □行修改， 使其在指定 dso 及其暗指的依□中 □□符号之后的 7.0-CURRENT。
700023	September 23, 2006	□ OSSv4 混音器 API 加入新的声音 IOCTL 之后的 7.0-CURRENT。
700024	September 28, 2006	□入 OpenSSL 0.9.8d 之后的 7.0-CURRENT。
700025	November 11, 2006	加入了 libelf 之后的 7.0-CURRENT。
700026	November 26, 2006	□音效相□的 sysctl □行大幅 □整之后的 7.0-CURRENT。
700027	November 30, 2006	加入 Wi-Spy quirk 之后的 7.0-CURRENT。
700028	December 15, 2006	在 libc 中加入 sctp □用之后的 7.0-CURRENT。
700029	January 26, 2007	将 GNU gzip(1) □□替□□从 NetBSD 移植的采用 BSD 授□版本之后的 7.0-CURRENT。
700030	February 7, 2007	在 IPv4 多播□□代□中□去了 IPIP 隧道封装 (VIFF_TUNNEL) 之后的 7.0-CURRENT。
700031	February 23, 2007	修改了 bus_setup_intr() (newbus) 之后的 7.0-CURRENT。
700032	March 2, 2007	引入了 ipw(4) 和 iwi(4) 固件之后的 7.0-CURRENT。
700033	March 9, 2007	在 ncurses 中引入了 □字符支持之后的 7.0-CURRENT。
700034	March 19, 2007	修改了 insmntque()、getnewvnode() 以及 vfs_hash_insert() 工作方式之后的 7.0-CURRENT。
700035	March 26, 2007	□加 CPU □率□□通知机制之后的 7.0-CURRENT。
700036	April 6, 2007	引入了 ZFS 文件系□之后的 7.0-CURRENT。
700037	April 8, 2007	新□了□□ Linux SCSI SG 直通□□ API 子集的 CAM 'SG' □□之后的 7.0-CURRENT。

□	日期	版本
700038	April 30, 2007	将 getenv(3)、 putenv(3)、 setenv(3) 和 unsetenv(3) 改□符合 POSIX 之后的 7.0-CURRENT。
700039	May 1, 2007	回退了 700038 中的□□之后的 7.0-CURRENT。
700040	May 10, 2007	在 libutil 中□加了 flopen(3) 之后的 7.0-CURRENT。
700041	May 13, 2007	□用了符号版本， 并将 libthr 改□默□□程□之后的 7.0-CURRENT。
700042	May 19, 2007	引入了 gcc 4.2.0 之后的 7.0-CURRENT。
700043	May 21, 2007	将 RELENG_6 之后未修改□版本的共享□版本□加之后的 7.0-CURRENT。
700044	June 7, 2007	将 vn_open()/VOP_OPEN() 的参数由文件描述符数□下□改□ struct file * 之后的 7.0-CURRENT。
700045	June 10, 2007	修改 pam_nologin(8) 使其向 PAM 框架提供□ 号管理功能而非身□□□功能之后的 7.0-CURRENT。
700046	June 11, 2007	更新 802.11 无□支持之后的 7.0-CURRENT。
700047	June 11, 2007	□加 TCP LRO 网□接口能力之后的 7.0-CURRENT。
700048	June 12, 2007	在 IPv4 □□□中加入了 RFC 3678 API 支持之后的 7.0-CURRENT。 先前 IP_MULTICAST_IF ioctl 的 RFC 1724 行□被□去； 0.0.0.0/8 不再能□用于指定接口索引下□, 而□使用 struct ipmreqn 代替。
700049	July 3, 2007	引入 OpenBSD 4.1 的 pf 之后的 7.0-CURRENT。
(not changed)		□ FAST_IPSEC □加 IPv6 支持， □去 KAME IPSEC， 并将 FAST_IPSEC 更名□ IPSEC 之后的 7.0-CURRENT。 (未□□)
700050	July 4, 2007	将 setenv/putenv/等等□用， 从□□ BSD 改□ POSIX □准之后的 7.0-CURRENT。

□	日期	版本
700051	July 4, 2007	□加新的 mmap/lseek/等等□些系□□用之后的 7.0-CURRENT。
700052	July 6, 2007	将 I4B □文件移□到 include/i4b 之后的 7.0-CURRENT。
700053	September 30, 2007	□加了 PCI domain 支持之后的 7.0-CURRENT。
700054	October 25, 2007	MFC 了□字符和□字□字符 ctype 分拆之后的 7.0-CURRENT。
700055	October 28, 2007	7.0-RELEASE， 以及 MFC 了恢□□ FreeBSD 4/5/6 版本的 PCIOCGETCONF、 PCIOCREAD 和 PCIOCWRITE IOCTL ABI 向下兼容之后的 7.0-CURRENT，□一□□□致 PCIOCGETCONF IOCTL 的 ABI 再次□生□化。
700100	December 22, 2007	7.0-RELEASE 之后的 7.0-STABLE
700101	February 8, 2008	MFC m_collapse() 之后的 7.0-STABLE。
700102	March 30, 2008	MFC kdb_enter_why() 之后的 7.0-STABLE。
700103	April 10, 2008	□ flock □□加入 l_sysid 之后的 7.0-STABLE。
700104	April 11, 2008	在 procstat(1) MFC 之后的 7.0-STABLE。
700105	April 11, 2008	在 MFC umtx 特性之后的 7.0-STABLE。
700106	April 15, 2008	□ psm(4) MFC write(2) 支持之后的 7.0-STABLE。
700107	April 20, 2008	□ fcntl(2) MFC F_DUP2FD 之后的 7.0-STABLE。
700108	May 5, 2008	□ lockmgr(9) 做了一些修改之后的 7.0-STABLE， 在使用 lockmgr(9) □必需包含 sys/lock.h。
700109	May 27, 2008	MFC 了 `memrchr` 函数之后的 7.0-STABLE。
700110	August 5, 2008	MFC 了内核 NFS locked 客□端之后的 7.0-STABLE。
700111	August 20, 2008	加入了□物理□□巨□支持之后的 7.0-STABLE。

□	日期	版本
700112	August 27, 2008	在 MFC 内核 DTrace 支持之后的 7.0-STABLE。
701000	November 25, 2008	7.1-RELEASE
701100	November 25, 2008	7.1-RELEASE 之后的 7.1-STABLE。
701101	January 10, 2009	合并了 `strndup` 之后的 7.1-STABLE。
701102	January 17, 2009	加入了 cpuctl(4) 支持之后的 7.1-STABLE。
701103	February 7, 2009	合并了 多/无-IPv4/v6 jail 之后的 7.1-STABLE。
701104	February 14, 2009	在 struct mount 中保存了挂起属主， 以及在 struct vfsops 中引入了 vfs_susp_clean 方法之后的 7.1-STABLE。
701105	March 12, 2009	□ kern.ipc.shmsegs sysctl □量不兼容的修改， 以允□在 64bit □架上分配更多的 SysV 共享内存段之后的 7.1-STABLE。
701106	March 14, 2009	合并了一个□ POSIX semaphore 等待操作修正之后的 7.1-STABLE。
702000	April 15, 2009	7.2-RELEASE
702100	April 15, 2009	7.2-RELEASE 之后的 7.2-STABLE。
702101	May 15, 2009	ichsmb(4) 改□使用左□接 □□址来保持与其它 SMBus 控制器□□一致性之后的 7.2-STABLE。
702102	May 28, 2009	MFC 了 `fdopendir` 函数之后的 7.2-STABLE。
702103	June 06, 2009	MFC 了 PmcTools 之后的 7.2-STABLE。
702104	July 14, 2009	MFC 了 `closefrom` 系□□用之后的 7.2-STABLE。
702105	July 31, 2009	MFC 了 SYSVIPC ABI 改□之后的 7.2-STABLE。

□	日期	版本
702106	September 14, 2009	MFC 了 x86 PAT □□， 并新□了 d_mmap_single() 以及 scatter/gather 型 VM □象 □型之后的 7.2-STABLE。
703000	February 9, 2010	7.3-RELEASE
703100	February 9, 2010	7.3-RELEASE 之后的 7.3-STABLE。
704000	December 22, 2010	7.4-RELEASE
704100	December 22, 2010	7.4-RELEASE 之后的 7.4-STABLE。
800000	October 11, 2007	8.0-CURRENT。 分拆了□字符和 □字□字符 ctype。
800001	October 16, 2007	引入了 libpcap 0.9.8 和 tcpdump 3.9.8 之后的 8.0-CURRENT。
800002	October 21, 2007	将 kthread_create() 系列函数改名□ kproc_create() 之后的 8.0-CURRENT。
800003	October 24, 2007	恢□了□ FreeBSD 4/5/6 版本的 PCIOCGETCONF、 PCIOCREAD 和 PCIOCWRITE IOCTL ABI 向下兼容之后的 8.0-CURRENT， □一□□□致 PCIOCGETCONF IOCTL 的 ABI 再次□生□化。
800004	November 12, 2007	将 agp(4) □□从 src/sys/pci □到 src/sys/dev/agp 之后的 8.0-CURRENT。
800005	December 4, 2007	修改了 jumbo frame 分配器之后的 8.0-CURRENT。
800006	December 7, 2007	在□ hwpmc(4) 加入了 callgraph 捕捉功能后的 8.0-CURRENT
800007	December 25, 2007	kdb_enter() □加 "why" 参数之后的 8.0-CURRENT。
800008	December 28, 2007	在去除 LK_EXCLUPGRADE □□后的 8.0-CURRENT。
800009	January 9, 2008	引入 lockmgr_disown(9) 之后的 8.0-CURRENT。
800010	January 10, 2008	修改 vn_lock(9) 原型之后的 8.0-CURRENT。

□	日期	版本
800011	January 13, 2008	修改 VOP_LOCK(9) 和 VOP_UNLOCK(9) 原型之后的 8.0-CURRENT。
800012	January 19, 2008	引入 lockmgr_recursed(9)、BUF_RECURSED(9) 和 BUF_ISLOCKED(9) 并□除了 `BUF_REFCNT()` 之后的 8.0-CURRENT。
800013	January 23, 2008	引入 "ASCII" □□之后的 8.0-CURRENT。
800014	January 24, 2008	修改 lockmgr(9) 并□除了 `lockcount()` 和 `LOCKMGR_ASSERT()` 之后的 8.0-CURRENT。
800015	January 26, 2008	□展了 fts(3) 数据□□之后的 8.0-CURRENT。
800016	February 1, 2008	□ MEXTADD(9) □加了一个参数之后的 8.0-CURRENT。
800017	February 6, 2008	□ lockmgr(9) 引入 LK_NODUP 和 LK_NOWITNESS □□后的 8.0-CURRENT。
800018	February 8, 2008	引入 m_collapse 之后的 8.0-CURRENT。
800019	February 9, 2008	□ sysctl □量 kern.proc.filedesc 加入 当前工作目□，root 目□和 jail 目□支持之后的 8.0-CURRENT。
800020	February 13, 2008	引入 lockmgr_assert(9) 之后的 8.0-CURRENT。
800021	February 15, 2008	引入 lockmgr_args(9) 和移除 LK_INTERNAL □志之后的 8.0-CURRENT。
800022	(backed out)	把 BSD ar(1) 作□系□默□的 ar 之后的 8.0-CURRENT。
800023	February 25, 2008	修改了 lockstatus(9) 和 VOP_ISLOCKED(9); 原型，特□□去掉 `struct thread` 参数之后的 8.0-CURRENT。

□	日期	版本
800024	March 1, 2008	□掉了 lockwaiters 和 BUF_LOCKWAITERS 函数， brelvp 的返回□从 void 修改成 int，并引入 lockinit(9) 新□志之后的 8.0-CURRENT。
800025	March 8, 2008	□ fcntl(2) 引入 F_DUP2FD 之后的 8.0-CURRENT。
800026	March 12, 2008	修改了 cv_broadcastpri □先□参数之后的 8.0-CURRENT，比如 0 表示无□先□。
800027	March 24, 2008	修改了 bpf □□ ABI, 加入了 zerocopy bpf buffer 之后的 8.0-CURRENT。
800028	March 26, 2008	□ flock □□□加了 l_sysid 之后的 8.0-CURRENT。
800029	March 28, 2008	重新整合了 BUF_LOCKWAITERS 函数并加入 lockmgr_waiters(9) 之后的 8.0-CURRENT。
800030	April 1, 2008	引入 rw_try_rlock(9) 和 rw_try_wlock(9) 之后的 8.0-CURRENT。
800031	April 6, 2008	引入 lockmgr_rw 和 lockmgr_args_rw 函数之后的 8.0-CURRENT。
800032	April 8, 2008	□□了 openat 和相□的系□□用，□ open(2) 引入了 O_EXEC □志，和提供了相□的 linux 兼容的系□□用之后的 8.0-CURRENT。
800033	April 8, 2008	□ psm(4) □加了原生的 write(2) 支持之后的 8.0-CURRENT。 □在任意命令可写入 /dev/psm%d 并□出状□。
800034	April 10, 2008	引入 memrchr 函数之后的 8.0-CURRENT。
800035	April 16, 2008	引入 fdopendir 函数之后的 8.0-CURRENT
800036	April 20, 2008	无□部分□向 multi-bss (也叫做 vaps) 支持之后的 8.0-CURRENT。

□	日期	版本
800037	May 9, 2008	加入多路由表支持 (也就是 setfib(1)、 stfib(2)） 后的 8.0-CURRENT。
800038	May 26, 2008	□去了 netatm 和 ISDN4BSD 后的 8.0-CURRENT。 □个版本也表示 □加了 Compact C Type (CTF) 工具。
800039	June 14, 2008	移除 sgtty 之后的 8.0-CURRENT。
800040	June 26, 2008	□加了内核□ NFS lockd 客□端的 8.0-CURRENT。
800041	July 22, 2008	□加了 arc4random_buf(3) 和 arc4random_uniform(3) 之后的 8.0-CURRENT。
800042	August 8, 2008	□加了 cpuctl(4) 之后的 8.0-CURRENT。
800043	August 13, 2008	修改 bpf(4) 使用□一的 □□□点而不是克隆之后的 8.0-CURRENT。
800044	August 17, 2008	在提交了 vimage □目第一□之后的 8.0-CURRENT。把全局□量重命名 □虚□化□ V_ 前 □并用宏映射到原来的全局名称。
800045	August 20, 2008	引入 MPSAFE TTY □之后的 8.0-CURRENT， 包括□相 □□□和工具的修改。
800046	September 8, 2008	将 amd64 架□上 GDT 拆分到不同 CPU 之后的 8.0-CURRENT。
800047	September 10, 2008	□去了 VSVTX、 VSGID 和 VSUID 之后的 8.0-CURRENT。
800048	September 16, 2008	将内核中 NFS 挂接部分的代□改 □能□通□ nmount() iovec， 而不再是大的 nfs_args □□体作 □参数之后的 8.0-CURRENT。
800049	September 17, 2008	□去了 suser(9) 和 suser_cred(9) 之后的 8.0-CURRENT。
800050	October 20, 2008	修改了□冲存□器 API 之后的 8.0-CURRENT。
800051	October 23, 2008	□去了 MALLOC(9) 和 FREE(9) 宏之后的 8.0-CURRENT。

□	日期	版本
800052	October 28, 2008	引入了 accmode_t 和重新命名 VOP_ACCES 'a_mode' □ a_accmode 之后的 8.0-CURRENT。
800053	November 2, 2008	修改了 vfs_busy(9) 原型并引入了 MBF_NOWAIT 和 MBF_MNTLSTLOCK □志之后的 8.0-CURRENT。
800054	November 22, 2008	□加了 buf_ring、 内存□以及 ifnet 函数， 以方便撰写支持多硬件 □□□列的□□， 以及无□□形□冲□□的 □□程序， 并更高效地管理包 □列功能之后的 8.0-CURRENT。
800055	November 27, 2008	引入了 hwpmc(4) □于 Intel™ Core，Core2 和 Atom 的支持之后的 8.0-CURRENT。
800056	November 29, 2008	引入了 multi-/no-IPv4/v6 jail 之后的 8.0-CURRENT。
800057	December 1, 2008	将 ath hal 改□使用源代□之后的 8.0-CURRENT。
800058	December 12, 2008	引入了 VOP_VPTOCNP 操作之后的 8.0-CURRENT。
800059	December 15, 2008	引入了新的 arp-v2 重写之后的 8.0-CURRENT。
800060	December 19, 2008	引入了 makefs 之后的 8.0-CURRENT。
800061	January 15, 2009	引入了 TCP Appropriate Byte Counting 之后的 8.0-CURRENT。
800062	January 28, 2009	□去了 minor()、 minor2unit()、 unit2minor() 等之后的 8.0-CURRENT。
800063	February 18, 2009	在 GENERIC 配置中改□使用 USB2 □之后的 8.0-CURRENT； □个数□ 同□也□志新□了 fdevname(3)。
800064	February 23, 2009	将 USB2 □移□并替□ dev/usb 之后的 8.0-CURRENT。
800065	February 26, 2009	在□ libmp(3) 中所有函数更名之后的 8.0-CURRENT。

□	日期	版本
800066	February 27, 2009	更改了 USB devfs 管理和布局之后的 8.0-CURRENT。
800067	February 28, 2009	加入了 getdelim()， getline()， stpncpy()， strnlen()， wcsnlen()， wcscasecmp()， 和 wcsncasecmp() 之后的 8.0-CURRENT。
800068	March 2, 2009	在 usbhub devclass 更名□ uhub 之后的 8.0-CURRENT。
800069	March 9, 2009	重命名 libusb20.so.1 □ libusb.so.1 之后的 8.0-CURRENT。
800070	March 9, 2009	合并 IGMPv3 和 Source-Specific Multicast (SSM) 入 IPv4 □之后的 8.0-CURRENT。
800071	March 14, 2009	□ gcc 打上了在 c99 和 gnu99 模式中使用 C99 inline □□□丁之后的 8.0-CURRENT。
800072	March 15, 2009	移除了 IFF_NEEDSGIANT □志; 不再支持非□程安全的网□□□□□之后的 8.0-CURRENT。
800073	March 18, 2009	□□了 rpath □□字符替□之后的 8.0-CURRENT.
800074	March 24, 2009	引入了 tcpdump 4.0.0 和 libpcap 1.0.0 之后的 8.0-CURRENT。
800075	April 6, 2009	修改了 structs vnet_net、vnet_inet 和 vnet_ipfw □□布局之后的 8.0-CURRENT。
800076	April 9, 2009	□ dummynet 新□了延□□估工具之后的 8.0-CURRENT。
800077	April 14, 2009	□去了 VOP_LEASE() 和 vop_vector.vop_lease 之后的 8.0-CURRENT
800078	April 15, 2009	在 struct rt_metrics 和 struct rt_metrics_lite 中添加了 rt_weight 字段， □致其□□□生□化之后的 8.0-CURRENT。 此后 RTM_VERSION □加，但又回退了。

□	日期	版本
800079	April 15, 2009	在 struct route 和 struct_in6 中添加了 struct llentry 指□之后的 8.0-CURRENT。
800080	April 15, 2009	改□了 struct inpcb 布局之后的 8.0-CURRENT。
800081	April 19, 2009	改□了 malloc_type 布局之后的 8.0-CURRENT。
800082	April 21, 2009	改□了 struct ifnet 布局， 并□加了 if_ref() 和 if_rele() 引用□数 □□功能之后的 8.0-CURRENT。
800083	April 22, 2009	□□了底□□牙 HCI API 之后的 8.0-CURRENT。
800084	April 29, 2009	修改了 IPv6 SSM 和 MLDv2 之后的 8.0-CURRENT。
800085	April 30, 2009	□用了包括一个活□映像的 VIMAGE 内核支持之后的 8.0-CURRENT。
800086	May 8, 2009	□ patch(1) □加任意 □□入行支持之后的 8.0-CURRENT。
800087	May 11, 2009	修改了一些 VFS KPI 之后的 8.0-CURRENT。 VFS 的 FSD 部分中 □去了□程参数。 `VFS_*` 函数并不需要□些上下文信息， 因 □它□是与 `curthread` 相□。 在某些特殊情况中， □保留了原先的行□。
800088	May 20, 2009	□ net80211 □□模式□行□整之后的 8.0-CURRENT。
800089	May 23, 2009	□加了 UDP 控制□支持之后的 8.0-CURRENT。
800090	May 23, 2009	将网□接口克隆虚□化之后的 8.0-CURRENT。
800091	May 27, 2009	□加了□次式 jail 并取消全局 securelevel 之后的 8.0-CURRENT。
800092	May 29, 2009	修改了 `sx_init_flags()` KPI 之后的 8.0-CURRENT。 SX_ADAPTIVESPIN 退役， 而新 □的 SX_NOADAPTIVE □志□表 □相反□□。

□	日期	版本
800093	May 29, 2009	□ struct mount □加 mnt_xflag 之后的 8.0-CURRENT。
800094	May 30, 2009	新□了 VOP_ACCESSX(9) 之后的 8.0-CURRENT。
800095	May 30, 2009	□整□□ KPI (polling KPI) 之后的 8.0-CURRENT。 □□□ 理程序会返回□理□的包的数量。 新□的 IFCAP_POLLING_NOCOUNT □表示返回□不重要， 并跳□□数。
800096	June 1, 2009	□新的 netisr □行了改□， 并 □整了保存和存取 FIB 方式之后的 8.0-CURRENT。
800097	June 8, 2009	引入了 vnet 析□挂□和相□基 □□施之后的 8.0-CURRENT。
800097	June 11, 2009	引入了 netgraph □出到□入路径□ 用□□和排□机制， 并□整了 struct thread 布局之后的 8.0-CURRENT。
800098	June 14, 2009	引入了 OpenSSL 0.9.8k 之后的 8.0-CURRENT。
800099	June 22, 2009	更新了 NGROUPS 并将路由虚□化 □到它自己的 VImage 模□之后的 8.0-CURRENT。
800100	June 24, 2009	修改了 SYSVIPC ABI 之后的 8.0-CURRENT。
800101	June 29, 2009	□去了与网□接口一一□□的 /dev/net/* 字符□□之后的 8.0-CURRENT。
800102	July 12, 2009	在 struct sackhint、 struct tcpcb 以及 struct tcpstat 上 □加占位元素之后的 8.0-CURRENT。
800103	July 13, 2009	将 TOE □□接口中的 struct tcpopt 替□□ TCP syncache 中的 struct toeopt 之后的 8.0-CURRENT。
800104	July 14, 2009	新□了基于 linker-set 的 per-vnet 分配器之后的 8.0-CURRENT。
800105	July 19, 2009	□□了所有未使用符号版本的□□□接□ 版本之后的 8.0-CURRENT。

□	日期	版本
800106	July 24, 2009	引入 VM □象□型 OBJT_SG 之后的 8.0-CURRENT。
800107	August 2, 2009	通□加入 newbus sxlock 使 newbus 子系□不再使用 Giant，以及 8.0-RELEASE。
800108	November 21, 2009	□□了 EVFILT_USER kevent □□器之后的 8.0-STABLE。
800500	January 7, 2010	令 `pkg_add -r` 使用 packages-8-stable 的 `__FreeBSD_version` 版本□化的 8.0-STABLE。
800501	January 24, 2010	□整了 `scandir(3)` 和 `alphasort(3)` 函数原型， 使其符合 SUSv4 之后的 8.0-STABLE。
800502	January 31, 2010	新□了 `sigpause(3)` 之后的 8.0-STABLE。
800503	February 25, 2010	新□了用于管理网□接口□明的 SIOCGIFDESCR 和 SIOCSIFDESCR ioctl 之后的 8.0-STABLE。 □□接口受到了 OpenBSD 的□□。
800504	March 1, 2010	MFC 了 x86emu， 来自 OpenBSD 的 x86 CPU □模式模□器之后的 8.0-STABLE。
800505	May 18, 2010	MFC 了添加 liblzma, xz, xzdec 以及 lzmainfo 之后的 8.0-STABLE。
801000	June 14, 2010	8.1-RELEASE
801500	June 14, 2010	8.1-RELEASE 之后的 8.1-STABLE。
801501	November 3, 2010	用于 PL_FLAG_SCE/SCX/EXEC/SI 的 struct sysentvec 的 KBI 以及用于 ptrace(PT_LWPINFO) 的 pl_siginfo 的 KBI 改□之后的 8.1-STABLE。
802000	December 22, 2010	8.2-RELEASE
802500	December 22, 2010	8.2-RELEASE 之后的 8.2-STABLE。
802501	February 28, 2011	合并了 DTrace □□， 包含用 □□跟踪支持之后的 8.2-STABLE。

□	日期	版本
802502	March 6, 2011	在 libm 中合并了 log2 和 log2f 之后的 8.2-STABLE。
802503	May 1, 2011	将 gcc 升□至 FSF gcc-4_2-branch 最后一个 GPLv2 版本之后的 8.2-STABLE。
802504	May 28, 2011	引入模□化□塞控制支持基□□施和 KPI 之后的 8.2-STABLE。
802505	May 28, 2011	引入了 Hhook 和 Khelp KPI 之后的 8.2-STABLE。
802506	May 28, 2011	在 tcpcb □□中□加 OSD 之后的 8.2-STABLE。
802507	June 6, 2011	引入 ZFS v28 之后的 8.2-STABLE。
802508	June 8, 2011	□去了 sv_schedtail struct sysvec 方法之后的 8.2-STABLE。
802509	July 14, 2011	在 binutils 中合并了 SSE3 支持之后的 8.2-STABLE。
802510	July 19, 2011	□ `rfork(2)` 添加了 RFTSIGZMB □志之后的 8.2-STABLE。
900000	August 22, 2009	9.0-CURRENT。
900001	September 8, 2009	引入了 x86emu， 来自 OpenBSD 的 x86 CPU □模式模□器之后的 9.0-CURRENT。
900002	September 23, 2009	□□了 EVFILT_USER kevent □□器之后的 9.0-CURRENT。
900003	December 2, 2009	新□了 `sigpause(3)` 以及 csu 的 PIE 支持之后的 9.0-CURRENT。
900004	December 6, 2009	新□了 libulog 及其 libutempter 兼容接口之后的 9.0-CURRENT。
900005	December 12, 2009	新□了用于□□指定休眠□列上等待者数量的 `sleepq_sleepcnt()` 函数之后的 9.0-CURRENT。
900006	January 4, 2010	□整了 `scandir(3)` 和 `alphasort(3)` 函数原型， 使其符合 SUSv4 之后的 9.0-CURRENT。
900007	January 13, 2010	□去了 utmp(5) 并□加了 utmpx (参□ `getutxent(3)`) 以改善用□登□日志和系□事件支持之后的 9.0-CURRENT。

□	日期	版本
900008	January 20, 2010	9.0-CURRENT 引入了 BSD 授□的 bc/dc 并将 GNU bc/dc □注 □□□之后的 9.0-CURRENT。
900009	January 26, 2010	新□了用于管理网□接口□明的 SIOCGIFDESCR 和 SIOCSIFDESCR ioctl 之后的 9.0-CURRENT。 □□接口受到了 OpenBSD 的□□。
900010	March 22, 2010	引入了 zlib 1.2.4 之后的 9.0-CURRENT。
900011	April 24, 2010	添加了 soft-updates 日志功能之后的 9.0-CURRENT。
900012	May 10, 2010	添加了 liblzma, xz, xzdec 以及 lzmainfo 之后的 9.0-CURRENT。
900013	May 24, 2010	添加了□□ linux(4) 的 USB 修正之后的 9.0-CURRENT。
900014	Jun 10, 2010	添加了 Clang 之后的 9.0-CURRENT。
900015	July 22, 2010	引入了 BSD grep 之后的 9.0-CURRENT。
900016	July 28, 2010	在 struct malloc_type_internal 中加入了 mti_zone 之后的 9.0-CURRENT。
900017	August 23, 2010	默□ grep 改回使用 GNU grep 并 □加 WITH_BSD_GREP □□之后的 9.0-CURRENT。
900018	August 24, 2010	将 `pthread_kill(3)` □生的信号在 si_code 中改□使用 SI_LWP □□之后的 9.0-CURRENT。 之前，si_code □□的□志□ SI_USER。
900019	August 28, 2010	□ `mmap(2)` 新□了 MAP_PREFAULT_READ □志之后的 9.0-CURRENT。
900020	September 9, 2010	□ sbuf □加了 drain 功能并改□了 struct sbuf 布局之后的 9.0-CURRENT。
900021	September 13, 2010	DTrace □加用□□跟踪支持之后的 9.0-CURRENT。

□	日期	版本
900022	October 2, 2010	新□了 BSDL man 工具， 并淘汰 GNU/GPL man 工具之后的 9.0-CURRENT。
900023	October 11, 2010	引入 20101010 git 快照版本 xz 之后的 9.0-CURRENT。
900024	November 11, 2010	将 libgcc.a 替□□ libcompiler_rt.a 之后的 9.0-CURRENT。
900025	November 12, 2010	引入了模□化□塞控制之后的 9.0-CURRENT。
900026	November 30, 2010	引入串行管理□□ (SMP) 直通，以及与之□□的 CAM CCB XPT_SMP_IO 和 XPT_GDEV_ADVINFO 之后的 9.0-CURRENT。
900027	December 5, 2010	在 libm 中□加 log2 之后的 9.0-CURRENT。
900028	December 21, 2010	添加了 Hhook (Helper Hook)、Khelp (Kernel Helpers) 和 Object Specific Data (OSD) KPI 之后的 9.0-CURRENT。
900029	December 28, 2010	修改 TCP □□□使其允□ Khelp 模□通□ helper hook 指□， 与 TCP 控制□交互并保存□接数据之后的 9.0-CURRENT。
900030	January 12, 2011	将 libdialog 更新至版本 20100428 之后的 9.0-CURRENT。
900031	February 7, 2011	添加了 `pthread_getthreadid_np(3)` 之后的 9.0-CURRENT。
900032	February 8, 2011	□除了 uio_yield 函数原型和符号之后的 9.0-CURRENT。
900033	February 18, 2011	将 binutils 更新至 2.17.50 之后的 9.0-CURRENT。
900034	March 8, 2011	修改了 struct sysvec (sv_schedtail) 之后的 9.0-CURRENT。
900035	March 29, 2011	将基本系□中 gcc 和 libstdc++ 升□至最后的 GPLv2 授□版本之后的 9.0-CURRENT。

□	日期	版本
900036	April 18, 2011	在基本系□中□去了 libobjc 和 Objective-C 支持之后的 9.0-CURRENT。
900037	May 13, 2011	在基本系□中引入了 libprocstat(3) 函数□以及 fuser(1) 工具之后的 9.0-CURRENT。
900038	May 22, 2011	□ VFS_FHTOVP(9) 添加 □□志参数之后的 9.0-CURRENT。
900039	June 28, 2011	引入了来自 OpenBSD 4.5 的 pf 之后的 9.0-CURRENT。
900040	July 19, 2011	将 amd64 和 ia64 平台上的 MAXCPU 提高到 64， 并把 XLP (mips) 上的□提高到 128 之后的 9.0-CURRENT。
900041	August 13, 2011	□□了 Capsicum capabilities 之后的 9.0-CURRENT。 fget(9) 新□了□限参数。
900042	August 28, 2011	提高修改□ ABI 的□□□接 □版本号之后的 9.0-CURRENT。
900043	September 2, 2011	□加了□不支持 SCSI 快取□存同 □功能的 USB 大容量存□□□自 □□□功能之后的 9.0-CURRENT。
900044	September 10, 2011	重□了 auto-quirk 之后的 9.0-CURRENT。
900045	Oct 13, 2011	将非兼容性系□□用入口点全部□加 sys_ 前□之后的 9.0-CURRENT。

□注意， 2.2.5-RELEASE 之后有一段□□的 2.2-STABLE 会声称自己是 "2.2.5-STABLE"。 □□模式的版本号表示的是年月。 但随后， 我□决定， 从 2.2 □始， 将它改□更□□□的 主/次 版本号的形式来命名版本。 □是因□并行地在多个分支上□行□□， 使得通□□□的 □布日期来区分不同的版本□得不再□□。 如果□正在做新的 port， □□不需要担心□早的 -CURRENT； 在此列出□供参考。

12.6. 在 bsd.port.mk 之后写一些内容

不要在 `.include <bsd.port.mk>` □行之后□加任何内容。 □通常可以通□在□的 Makefile 中□的某□引用 bsd.port.pre.mk， 并在□尾的地方引用 bsd.port.post.mk 来避免。

只能□采用 bsd.port.pre.mk/bsd.port.post.mk 或 bsd.port.mk □□写法之一； 任何 □候都不要同□使用□□写法。

bsd.port.pre.mk 只定□了很少的□量， 它□可以在 Makefile 中用于□行一些□□， 而 bsd.port.post.mk □定

□了所有其它的□量。

下面是一些由 bsd.port.pre.mk 定□的比□重要的□量 (□并不是一□完整的列表， □可以□□ bsd.port.mk 以□得全部□量的名字)。

□量	描述
ARCH	由 uname -m □出得到的硬件架□的名字 (例如， i386)
OPSYS	由 uname -s 返回的操作系□□型 (例如， FreeBSD)
OSREL	操作系□的版本号 (例如 2.1.5 或 2.2.7)
OSVERSION	操作系□的版本号的数□形式； 它等于 __FreeBSD_version。
PORTOBJFORMAT	系□默□的□行文件格式 (elf 或 aout； □注意， "□代的" FreeBSD 版本中， aout 已在淘汰之列。)
LOCALBASE	"local" 目□的根 (例如， /usr/local/)
PREFIX	port □被安装到□里 (参□ □于 PREFIX 的更多□明)。

如果□需要定□ USE_IMAKE， USE_X_PREFIX， 或 MASTERDIR □些□量， □□在引用 bsd.port.pre.mk 之前完成。

下面是一些在引用 bsd.port.pre.mk 之后可以□行的判断：

```
# 如果 perl5 已□在系□中提供， □不必□□ lang/perl5
.if ${OSVERSION} > 300003
BROKEN= perl is in system
.endif

# ELF 只使用一个 shlib 版本
.if ${PORTOBJFORMAT} == "elf"
TCL_LIB_FILE=  ${TCL_LIB}.${SHLIB_MAJOR}
.else
TCL_LIB_FILE=  ${TCL_LIB}.${SHLIB_MAJOR}.${SHLIB_MINOR}
.endif

# □件会自□□ ELF □建符号□接， 但 a.out □需要□行□建
post-install:
.if ${PORTOBJFORMAT} == "aout"
        ${LN} -sf liblinpack.so.1.0 ${PREFIX}/lib/liblinpack.so
.endif
```

□□□得□□在 BROKEN= 和 TCL_LIB_FILE= 后面使用制表符， 而不是空格， □□？ :-)

12.7. 在 wrapper 脚本中使用 exec □句

如果 port 安装了用以□□其他程序的脚本， 并且□行其他程序是□些脚本的最后一□操作， □□必使用 exec □句来□行□些程序， 例如：

```
#!/bin/sh
exec %%LOCALBASE%%/bin/java -jar %%DATADIR%%/foo.jar "$@"
```

使用 exec □句表示□行指定的程序来取代 shell □程。 如果省略了 exec， □ shell □程会一直在内存中，从而不必要地消耗了□外的系□□源。

12.8. 理性行事

任何 Makefile 都□□□□并理性地行事。 如果□能□其中的条目更□□□和易□， 一定要□□做。 例如， 使用 make 提供的 .if □□， 而不要使用 shell 的 if， 只要能重定□ EXTRACT* 就不要重□ do-extract， 尽量使用 GNU_CONFIGURE 而不是 CONFIGURE_ARGS += --prefix=${PREFIX}。

如果□在□□做什□事情的□候□□不得不写大量的代□， □回□□来□□一下 bsd.port.mk， 看看是否有□正打算做的事情的□成□□。 尽管□起来可能很□□， 但有很多貌似很□的□□， 在 bsd.port.mk 中都□出了十分□便的解决方案。

12.9. 遵循 CC 和 CXX □置

port □遵循 CC 和 CXX □量的□置。 □也就是□， port 不□使用□□的方式来□置□个□量的□， 而罔□已□存在的□置；与此相反， 它□□在其□后加入需要的其它□。 □□， 就可以□置全局的□□□□， 令其影□所有的 port □□□程了。

如果□在无法□□做， □在 Makefile 中加入 NO_PACKAGE=ignores cflags。

下面的 Makefile □例□出了如何遵循 CC 和 CXX □量的□置。 注意□里用到的 ?=：

```
CC?= gcc
```

```
CXX?= g++
```

下面□是没有遵循 CC 和 CXX 的例子：

```
CC= gcc
```

```
CXX= g++
```

在 FreeBSD 系□中， CC 和 CXX □□个□量都可以在 /etc/make.conf 中自行定□。 第一个例子只有在 /etc/make.conf 中没有定□□才□□□个□量□行定□， 从而保持了系□□□的配置。 而第二个例子□会覆□任何□有的配置。

12.10. 遵循 CFLAGS

□的 port □遵循 CFLAGS □量的□置。 □也就是□， port 不□使用□□的方式来□置□个□量的□， 而罔□已□存在的

□置； 与此相反， 它□□在其□后加入需要的其它□， □□， 就可以□置全局的□□□□， 令其影□所有的 port □□□程了。

如果□在无法□□做， □在 Makefile 中加入 NO_PACKAGE=ignores cflags。

下面的 Makefile 例子， 可以□助我□理解如何遵循 CFLAGS 的□置。 注意所用的 +=：

```
CFLAGS+= -Wall -Werror
```

下面是一个未能遵循 CFLAGS □置的例子：

```
CFLAGS= -Wall -Werror
```

一般来□， CFLAGS 在 FreeBSD 系□中是在 /etc/make.conf 里配置的。 第一个例子在 CFLAGS □量中□加了一些参数， 并保持了所有系□□定□的□志。 而第二个例子， □会覆□掉任何先前定□的参数。

□□从第三方□件的 Makefile 中去掉特殊的□化□置。 系□的 CFLAGS □出了全系□□□内的□化□置参数。 下面是一个未□修改的 Makefile □例：

```
CFLAGS= -O3 -funroll-loops -DHAVE_SOUND
```

如果使用系□的□化参数， □ Makefile 中的□置□□□似下面□□：

```
CFLAGS+= -DHAVE_SOUND
```

12.11. □程□

在 FreeBSD 上， □程□必□通□特殊的□接器参数 -pthread □接到可□行文件。 如果 port 一定要直接□接 -lpthread 或 -lc_r， □□将其改□使用由 ports 框架提供的 PTHREAD_LIBS。 □个□量的□通常是 -pthread， 但在某些特定平台上的 FreeBSD 版本中， 它可能是其它□， 因此， 不要将 -pthread 硬□□到□的□丁中， 而□使用 PTHREAD_LIBS □量。

如果□置了 PTHREAD_LIBS， 而在□□□出□ unrecognized option '-pthread' □□的□□， 可能需要通□将 CONFIGURE_ENV □□ LD=${CC} 来使用 gcc 作□□接器。 -pthread □一□□并不□ ld 所直接支持。

12.12. 反□

如果□行了一些很好的修改和□丁， 一定要把它□□回□原作者， 或□□者， 以便在下一版本的代□中包含它□。 □会□□在□件□布新版本的□候□得□松一些。

12.13. README.html

不要包含 README.html 文件。 □个文件并非 CVS 代□□中的一部分， 它是由 `make readme` 命令生成的。

12.14. 使用 BROKEN、 FORBIDDEN 或 IGNORE 阻止用□安装 port

某些□候会需要阻止用□安装某个 port。 想要告□用□某个 port 不□被安装， 有□多可以在 port 的 Makefile 中使用的 `make` □量。 下列 `make` 的□， 将是在用□□□安装□得到的提示信息。 □□使用正□的 `make` □量， 因□□一个都表□了截然不同的意□， 而且□多自□化系□， 例如 port □□集群、 FreshPorts， 以及 portsmon， 都依□于 Makefile 的正□性。

12.14.1. □量

- `BROKEN` □□用于表□目前无法正□□□、 安装或卸□□□□□。 如果是□□性的□□， □可以使用它。

  如果□行了相□的配置， □□□集群仍将□□□□它， 以□□□致□□的深□□□是否已被解决。 (不□， 一般情况下， □□集群并不会□□做。)

  □例来□， 当 port □生下述情况□， □使用 `BROKEN`：

  - 无法□□ (does not compile)
  - 无法正□□行配置或安装操作
  - 在 ${LOCALBASE} 以外的地方安装文件
  - 卸□□无法□除所安装的全部文件 (不□， 留下用□改□的文件可接受的， 因□可能希望□□作)

- `FORBIDDEN` 用于表示 ports 中包含安全漏洞， 或者可能会□安装了□个 port 的 FreeBSD 系□□来□重的安全□患 (例如： 一个很不安全的程序， 或包含了能□被□易攻陷的服□的□件)。 如果□□了安全漏洞， 而其作者没有□布升□版本， □□立即把那个 port □□□ `FORBIDDEN`。 理想情况下， 包含安全漏洞的 port □被尽快升□， 以便□少包含漏洞的 FreeBSD 主机的数量 (我□希望保持良好的安全□□)， 然而， 有□在安全漏洞的披露和□件更新之□可能会有一个□隔， 此□□予以□明。 除了安全之外， □不要以任何其它理由将 port □□□ `FORBIDDEN`。

- `IGNORE` 用来表示 port 由于某些其它原因不□予以□□。 如果□□□生了□□性的□□， □□使用它。 任何情况下， □□集群都不会□□□□□ `IGNORE` 的 port。 以下是使用 `IGNORE` 的一些例子：

  - 能□□□但无法正常□行
  - 无法与□行的 FreeBSD 版本一同工作
  - □□□需要 FreeBSD 内核的源代□， 但用□没有安装它□
  - 由于授□原因， 必□手工下□ distfile
  - 无法与的某个已安装的 port 一同工作 (例如， port 依□于 www/apache21 而安装的□是 www/apache13)

  

  如果 port 与某个已□安装的 port 冲突 (例如， 它□在同一位置安装同名但功能不同的文件)， □□ 使用 `CONFLICTS` 来□□它。 `CONFLICTS` 将自□地□置 `IGNORE`。

- 如果 port 只□在某些平台上□□□ `IGNORE`， □有□外□个方便使用的 `IGNORE` □量可供□□： `ONLY_FOR_ARCHS` 和

NOT_FOR_ARCHS。 例如：

```
ONLY_FOR_ARCHS= i386 amd64
```

```
NOT_FOR_ARCHS= alpha ia64 sparc64
```

可以使用 ONLY_FOR_ARCHS_REASON 和 NOT_FOR_ARCHS_REASON 来配置定制的 IGNORE 消息。 此外，□可以使用 ONLY_FOR_ARCHS_REASONARCH_ 和 NOT_FOR_ARCHS_REASONARCH_ 来分□指定与具体平台有□的信息。

- 如果 port 会下□并安装用于 i386 的□□□二□制文件， □□□置 IA32_BINARY_PORT。 如果□置了□个□量， □系□会□□是否已□在 /usr/lib32 目□中安装了 IA32 版本的函数□， 以及内核是否提供了 IA32 兼容支持。 如果□些依□条件不□足， □会自□□置 IGNORE。

12.14.2. □□□明

□些字串不□使用引号括起来。 此外， 由于□示□用□的方式不同， □些字串的措辞也□有所不同。 例如：

```
BROKEN= this port is unsupported on FreeBSD 5.x
```

```
IGNORE= is unsupported on FreeBSD 5.x
```

它□分□会在 make describe □□生下面的□出：

```
===>  foobar-0.1 is marked as broken: this port is unsupported on FreeBSD 5.x.
```

```
===>  foobar-0.1 is unsupported on FreeBSD 5.x.
```

12.15. 使用 DEPRECATED 或 EXPIRATION_DATE 表示某个 port 将被□除

一定要□得 BROKEN 和 FORBIDDEN 只□作□当某个 port 无法正常工作□的□□解决方案。 永久性地坏掉了的 port □被从 ports tree 中完全□除。

需要□□可以使用 DEPRECATED 和 EXPIRATION_DATE 来通知用□某个 port 不□被使用， 并即将被□除。 前一个□量用来表□□什□□□除 port； 而后一个是□是一个 ISO 8601 格式的日期 (YYYY-MM-DD)。 □者都会向用□呈□。

也可以□置 DEPRECATED 而不□出 EXPIRATION_DATE (例如， 建□使用某个新版本的 port)， 但反之□没有意□。

目前□没有□切的□于需要□出多少通知的政策。 当前的□践是， □于与安全有□的□□□一个月， 而与□□有□的□□□□□个月。 □也□有□趣的 committer 能□有一点□□来修正□□。

12.16. 避免使用 .error □□

在 Makefile 中□出信号， 表示由于某□外界因素 (例如， 用□指定了无效的□□□□) 而无法安装的方法是将□量 IGNORE □□一非空□。 □个□将被格式化， 并在用□□行 make install 是□出提示。

用 .error □□□一目的是一□常□的□用。 □□做的□□是， □多在 ports □上□行的自□化工具会因此而失□。 最常□的情况□于□□ /usr/ports/INDEX 的□程 (参□ □行 make describe)。 然而， 即使十分普通的命令， 例如 make maintainer， 在□□情况下也会失□。 □是不可接受的。

例 25. □□避免使用 .error

考□有人在 make.conf 中□置了

```
USE_POINTYHAT=yes
```

的情形。 接下来的例子中， 第一个 Makefile 中的□□将□致 make index 失□， 而第二个□不会：

```
.if USE_POINTYHAT
.error "POINTYHAT is not supported"
.endif
```

```
.if USE_POINTYHAT
IGNORE=POINTYHAT is not supported
.endif
```

12.17. □于 sysctl 的使用

除了在 target 中之外， 是不鼓励使用 sysctl 的。 □是因□□算 makevar， 例如在 make index 中所□行的那□， 都不得不□行一条命令， □会使□一操作□得更慢。

在使用 sysctl(8) □， □必通□ SYSCTL □量来□行， 因□此□量将展□成命令的完整路径， 并且用□可以根据需要□行指定。

12.18. 重新□布的 distfiles

有□， 一些□件的作者会修改□已□布的 distfile 的内容， 而并不修改文件名。 □□情况下， □需要□□□些□□是来自□件作者的官方改□。 在□去， 曾□□生□下□服□器上的 distfile 被悄悄□成注入□□意代□的版本， 并□用□安全造成威□或□害的事情。

□□保留一□旧的 distfile， 并下□一□新的， 分□展□， 用 diff(1) 来□比其内容。 如果没有□□可疑的□□， 就可以更新 distinfo 了。 □□必在□的 PR 或 commit log 中□□些差□□行描述， 以便□□人了解□已□仔□□比□差□， 并□□没有□□了。

除此之外， 也可以□系□件的作者， 以□□□些修改是否是他□做的。

12.19. □□

需要仔□地反□□□ pkg-descr 和 pkg-plist □□个文件。 如果□正在□□一个 port， 并□□□□个文件□□改□， □一定要□□做。

□不要在系□中□制多□ GNU General Public License。

一定要非常小心地□理法律□□！ 不要□我□□布没有得到合法授□的□件！

Chapter 13. 示□的 Makefile

□里是一个□可以在建立新 port □参考的 Makefile。 □□必□除不需要的那些注□ (方括号中□的文字)！

建□□按照下面□□的格式 (□量□序， 小□之□的空行等) 来□写。 □个格式的作用是便于□□重要的信息。 我□建□□使用 portlint 来□□ Makefile。

```
[□部... 主要是□我□更容易地分辨不同的 port。]
# New ports collection makefile for:   xdvi
[版本□行， 只有在 PORTVERSION □量不足以描述 port □才需要]
# Date created:                26 May 1995
[□是最初将□件移植到 FreeBSD 上的日期， 一般来□是建立□□ Makefile 的日期。
□注意不要在之后再次修改□个日期。]
# Whom:                        Satoshi Asami <asami@FreeBSD.org>
#
# $FreeBSD$
[ ^^^^^^^^^ □是 CVS 在文件 commit 到我□的代□□， 自□□行替□的 RCS ID。
如果□正在升□ port， 不要把它改回 "$FreeBSD$"。
CVS 会自□□行□理。]
#

[□个小□描述 port 本身以及主要下□站点 - PORTNAME 和 PORTVERSION
 □放在最前面， 随后是 CATEGORIES， 然后是 MASTER_SITES， 接下来是
 MASTER_SITE_SUBDIR。 如果需要的□， 接下来□指定
 PKGNAMEPREFIX 和 PKGNAMESUFFIX。 随后是 DISTNAME， EXTRACT_SUFX，
 以及 DISTFILES， EXTRACT_ONLY， 如果需要的□。]
PORTNAME=      xdvi
PORTVERSION=   18.2
CATEGORIES=    print
[如果不想使用 MASTER_SITE_* 宏， 一定不要忘□□尾的斜□ ("/")！]
MASTER_SITES=  ${MASTER_SITE_XCONTRIB}
MASTER_SITE_SUBDIR= applications
PKGNAMEPREFIX= ja-
DISTNAME=      xdvi-pl18
[如果源代□包不是□准的 ".tar.gz" 形式， 就需要□置□个]
EXTRACT_SUFX=  .tar.Z

[分散的□丁 -- 可以□空]
PATCH_SITES=   ftp://ftp.sra.co.jp/pub/X11/japanese/
PATCHFILES=    xdvi-18.patch1.gz xdvi-18.patch2.gz

[□□人(maintainer); *必□有*！ □是某个□源□理 port 更新、 □□失□，
 以及回答用□直接提□或□□ bug 的人。 □了保□ Ports Collection
 有尽可能高的品□， 我□不再接受指定□ "ports@FreeBSD.org" 的新 port。]
MAINTAINER=    asami@FreeBSD.org
COMMENT=       A DVI Previewer for the X Window System

[依□的其它□件包 -- 可以□空]
RUN_DEPENDS=   gs:${PORTSDIR}/print/ghostscript
LIB_DEPENDS=   Xpm.5:${PORTSDIR}/graphics/xpm
```

```
[□□是其它不□合上几□的□准 bsd.port.mk □量]
[如果需要在 configure、 build 或 install □程中提□...]
IS_INTERACTIVE=        yes
[如果解□□到 ${DISTNAME} 以外的目□...]
WRKSRC=                ${WRKDIR}/xdvi-new
[如果作者□布的□丁不是相□于 ${WRKSRC} 的， 可能需要□整□个]
PATCH_DIST_STRIP=      -p1
[如果需要□行由 GNU autoconf 生成的 "configure" 脚本]
GNU_CONFIGURE= yes
[如果需要使用 GNU make， 而不是 /usr/bin/make 来完成□□...]
USE_GMAKE=     yes
[如果是一个 X □用程序， 并使用 "xmkmf -a" 来□行...]
USE_IMAKE=     yes
[et cetera.]

[将在接下来的部分使用的非□准的□量]
MY_FAVORITE_RESPONSE=  "yeah, right"

[接下来是特殊□□， 按□用□序排列]
pre-fetch:
    i go fetch something, yeah

post-patch:
    i need to do something after patch, great

pre-install:
    and then some more stuff before installing, wow

[□□]
.include <bsd.port.mk>
```

Chapter 14. 保持同□

FreeBSD 的 Ports Collection 在持□地□行修改。 □里提供了一些□于如何保持同□的信息。

14.1. FreshPorts

最□□的了解已□被 commit 到 ports 中的更新的方法， 是□□ FreshPorts。 □可以□□多个 ports 并□其□行□□。 □烈建□□□人□□□它， □□就不□能接收到他□自己所做的修改， 而且能看到其它 FreeBSD committer 所做的改□。 (保持与所依□的 ports 框架同□是必要的-□然一般来□□会在□□的 commit 之前收到一个礼貌性的通知，但有□可能会有人没有注意到需要□□做， 或者□□做很困□。 □外， 有些□候通知的修改也可能是微不足道的。我□希望□一个人能□正□地□行判断。)

如果想使用 FreshPorts， 之需要建立一个□号。 如果□注册的□件地址是 @FreeBSD.org， □会看到 web □面右□的 opt-in □接。 如果□已□注册了 FreshPorts □号， 但没有使用 @FreeBSD.org □件地址， □只需把□件地址改□ @FreeBSD.org， 重新□□， 并将其改回。

FreshPorts 也会□□一个 FreeBSD ports tree 上的 commit □行自□的合法性□□。 如果□□□了□□服□， □如果□□了□□， 就会收到来自 FreshPorts 的□□□告。

14.2. 代□□的 Web □□界面

可以通□ web 界面来□□源代□□中的文件。 影□整个 ports 系□的修改， □在都会在 CHANGES 文件中□明。影□某一个 port 的□□， □在 UPDATING 文件中□明。 尽管如此， 所有□□最□□威的答案， 毫无疑□□□是 bsd.port.mk 的源代□， 以及相□的文件。

14.3. FreeBSD Ports □件列表

如果□□□了某个或某一些 ports， □□□考□□□ FreeBSD ports □件列表。 □于 ports 工作方式的重要修改都会在此宣示， 并提交到 CHANGES。

14.4. 位于 pointyhat.FreeBSD.org 的 FreeBSD Port □□集群

FreeBSD 的一个最不□人所知的□□是， 它□有一个□用于持□□□ Ports Collection 的集群， □个集群会□□所有主要的 OS 版本在□一个 Tier-1 架□上的 package。 □可以在 package □□和□□日志 □到其□果。

□一个 port 都会被□□， 除非□□□ IGNORE。 □□了 BROKEN 的 port 仍然会被□□□□， 以了解是否某些依□□系的□□解决了其□□ (□是通□□ port 的 Makefile □ TRYBROKEN 参数来完成的)。

14.5. FreeBSD 的 Ports Distfile □描器

□□集群是一□□□用于□□所有 port 最新版本的机器， 其上已□下□了所有的 distfiles。 然而， 由于 Internet 在持□地□生□化， distfile 可能很快就消失了。 FreeBSD Ports distfile □描器 □□□□□一个 port 的所有下□站点， 以期□出□些文件是否依然存在。 □□者□□律性地□□□些□告， □不□会提高用□□□的速度， 同□也避免了浪□那些□像了全部 distfile 的志□者的□□。

14.6. FreeBSD 的 Ports 追踪系□

□一个非常方便的□源， 就是 FreeBSD Ports 追踪系□ (也被称作 portsmon)。 □个系□包含了一个□理若干信息来源的数据□， 并提供了一个可以通□ web 方式□□的界面。 目前， 它利用到了和 ports 有□的□□□告 (PR)、 来自□□集群的□□日志， 以及来自 Ports Collection 的文件所提供的信息。 未来， □会□它□行□一□的□展， 从而提供包括 distfile 普□， 以及其它来源在内的更多信息。

要使用□个工具， 可以从□看□于某一个 port 的全部□料的 Port 的□□ □始。

本文撰写□, □是唯一一个能□将 GNATS PR □， 同□□的 port 名字映射起来的□源。 (提交 PR 的用□， 有□并不在 Synopsis (概要) 中指明 port 的名字， 尽管我□希望他□□□做)。 因此， portsmon 在□想要□□是否有人提交某个□存的 port 的 PR， 以及它的□□是否出□了□□； 或在□□建新的 port 之前想要□□一下是否已□有人提交□□， 就非常有用了。