FreeBSD Porter's Handbook

FreeBSD ####

FreeBSD Porter's Handbook

#
##: 44974
April 2000 # .
© 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008 FreeBSD

Legal Notice

FreeBSD is a registered trademark of The FreeBSD Foundation.

UNIX is a registered trademark of The Open Group in the US and other countries.

Sun, Sun Microsystems, SunOS, Solaris, Java, JDK, and OpenJDK are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

Apple and QuickTime are trademarks of Apple Computer, Inc., registered in the U.S. and other countries.

Macromedia and Flash are trademarks or registered trademarks of Macromedia, Inc. in the United States and/or other countries.

Microsoft, Windows, and Windows Media are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

PartitionMagic is a registered trademark of PowerQuest Corporation in the United States and/or other countries.

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and the FreeBSD Project was aware of the trademark claim, the designations have been followed by the ™ symbol.

Copyright

Redistribution and use in source (XML DocBook) and 'compiled' forms (XML, HTML, PDF, PostScript, RTF and so forth) with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code (XML DocBook) must retain the above copyright notice, this list of conditions and the following disclaimer as the first lines of this file unmodified.

2. Redistributions in compiled form (transformed to other DTDs, converted to PDF, PostScript, RTF and other formats) must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

##

THIS DOCUMENTATION IS PROVIDED BY THE FREEBSD DOCUMENTATION PROJECT "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE FREEBSD DOCUMENTATION PROJECT BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS

OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS DOCUMENTATION, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

####

1. ## 1
2. #### port 3
3. ## Port ##### 5
    3.1. ## Makefile 5
    3.2. ######### 5
    3.3. ## checksum ### distinfo # 7
    3.4. ## port ####### 7
    3.5. # portlint #### 8
    3.6. ##(Submit) port 8
4. Slow Porting 11
    4.1. How things work 11
    4.2. ##### source # 12
    4.3. #### port 12
    4.4. Patching 13
    4.5. ## 14
    4.6. ####### 14
5. ## Makefile 15
    5.1. The original source 15
    5.2. Naming 15
    5.3. Categorization 21
    5.4. The distribution files 29
    5.5. MAINTAINER 40
    5.6. COMMENT 41
    5.7. Dependencies 41
    5.8. MASTERDIR 47
    5.9. Manpages 48
    5.10. Info files 49
    5.11. Makefile Options 49
    5.12. Specifying the working directory 53
    5.13. CONFLICTS 54
    5.14. Installing files 54
6. Special considerations 59
    6.1. Shared Libraries 59
    6.2. Ports with distribution restrictions 59
    6.3. Building mechanisms 61
    6.4. Using GNU autotools 63
    6.5. Using GNU gettext 65
    6.6. Using perl 66
    6.7. Using X11 68
    6.8. Using GNOME 71
    6.9. Using KDE 71
    6.10. Using Java 75
    6.11. Web applications, Apache and PHP 79
    6.12. Using Python 82

6.13. Using Emacs 84
6.14. Using Ruby 84
6.15. Using SDL 85
6.16. Using wxWidgets 86
6.17. Using Lua 92
6.18. Using Xfce 98
6.19. Using databases 99
6.20. Starting and stopping services (rc scripts) 99
7. Advanced `pkg-plist` practices 103
7.1. Changing `pkg-plist` based on make variables 103
7.2. Empty directories 104
7.3. Configuration files 104
7.4. Dynamic vs. static package list 105
7.5. #### package list 105
8. The `pkg-*` files 107
8.1. `pkg-message` 107
8.2. `pkg-install` 107
8.3. `pkg-deinstall` 108
8.4. `pkg-req` 108
8.5. Changing the names of `pkg-*` files 108
8.6. Making use of `SUB_FILES` and `SUB_LIST` 109
9. Testing your port 111
9.1. Running `make describe` 111
9.2. Portlint 111
9.3. Port Tools 111
9.4. `PREFIX ## DESTDIR` 112
9.5. Tinderbox 113
10. Upgrading 115
11. Ports security 117
11.1. Why security is so important 117
11.2. Fixing security vulnerabilities 117
11.3. Keeping the community informed 118
12. Dos and Don'ts 125
12.1. Introduction 125
12.2. `WRKDIR` 125
12.3. `WRKDIRPREFIX` 125
12.4. Differentiating operating systems and OS versions 125
12.5. FreeBSD #####(__FreeBSD_version) 127
12.6. Writing something after `bsd.port.mk` 152
12.7. # wrapper scripts ### exec ## 154
12.8. UIDs # GIDs 154
12.9. Do things rationally 154
12.10. Respect both `CC` and `CXX` 155
12.11. Respect `CFLAGS` 155
12.12. Threading libraries 156
12.13. Feedback 156

12.14. README.html 156
12.15. Marking a port not installable with BROKEN, FORBIDDEN, or IGNORE 156
12.16. Marking a port for removal with DEPRECATED or EXPIRATION_DATE 159
12.17. Avoid use of the .error construct 159
12.18. sysctl #### 160
12.19. Rerolling distfiles 160
12.20. Necessary workarounds 160
12.21. Miscellanea 161
13. A Sample Makefile 163
14. Keeping Up 165
14.1. FreshPorts 165
14.2. The Web Interface to the Source Repository 165
14.3. The FreeBSD Ports Mailing List 165
14.4. The FreeBSD Port Building Cluster on pointyhat.FreeBSD.org 166
14.5. The FreeBSD Port Distfile Survey 166
14.6. The FreeBSD Ports Monitoring System 166

####

5.1. The USE_* variables 45
5.2. Common WITH_* and WITHOUT_* variables 50
6.1. Variables for ports related to gmake 61
6.2. Variables for ports that use configure 62
6.3. Variables for ports that use scons 62
6.4. Variables for ports that use perl 67
6.5. Variables for ports that use X 69
6.6. Variables for depending on individual parts of X11 69
6.7. Variables for ports that use KDE 71
6.8. Variables for ports that use Qt 71
6.9. Additional variables for ports that use Qt 4.x 72
6.10. Available Qt4 library components 73
6.11. Available Qt4 tool components 73
6.12. Available Qt4 plugin components 74
6.13. Variables that may be set by ports that use Java 75
6.14. Variables provided to ports that use Java 76
6.15. Constants defined for ports that use Java 78
6.16. Variables for ports that use Apache 79
6.17. port Apache ######## 80
6.18. Variables for ports that use PHP 81
6.19. Most useful variables for ports that use Python 83
6.20. Useful variables for ports that use Ruby 84
6.21. Selected read-only variables for ports that use Ruby 84
6.22. Variables to select wxWidgets versions 86
6.23. Available wxWidgets versions 87
6.24. wxWidgets version specifications 87
6.25. Variables to select preferred wxWidgets versions 87
6.26. Available wxWidgets components 88
6.27. Available wxWidgets dependency types 88
6.28. Default wxWidgets dependency types 88
6.29. Variables to select Unicode in wxWidgets versions 89
6.30. Variables defined for ports that use wxWidgets 90
6.31. Legal values for WX_CONF_ARGS 92
6.32. Variables to select Lua versions 92
6.33. Available Lua versions 93
6.34. Lua version specifications 93
6.35. Variables to select preferred Lua versions 93
6.36. Available Lua components 94
6.37. Available Lua dependency types 94
6.38. Default Lua dependency types 94
6.39. Variables defined for ports that use Lua 96
6.40. Variables for ports using databases 99
12.1. __FreeBSD_version values 127

####

5.1. Simplified use of MASTER_SITES:n with 1 file per site 33
5.2. Simplified use of MASTER_SITES:n with more than 1 file per site 33
5.3. Detailed use of MASTER_SITES:n in MASTER_SITE_SUBDIR 35
5.4. Detailed use of MASTER_SITES:n with comma operator, multiple files, multiple sites and multiple subdirectories 35
5.5. Detailed use of MASTER_SITES:n with MASTER_SITE_SOURCEFORGE 38
5.6. Simplified use of MASTER_SITES:n with PATCH_SITES 38
5.7. Use of ALWAYS_KEEP_DISTFILES 40
5.8. Simple use of OPTIONS 52
5.9. Wrong handling of an option 52
5.10. Correct handling of an option 53
6.1. USE_XORG example 68
6.2. Using some X11 related variables in port 70
6.3. Selecting Qt4 components 74
6.4. Example Makefile for PEAR class 82
6.5. Selecting wxWidgets components 89
6.6. Detecting installed wxWidgets versions and components 90
6.7. Using wxWidgets variables in commands 91
6.8. Selecting the Lua version 93
6.9. Selecting Lua components 95
6.10. Detecting installed Lua versions and components 95
6.11. Telling the port where to find Lua 97
6.12. Using Lua variables in commands 97
12.1. How to avoid using .error 159

1.

FreeBSD ####### FreeBSD Ports Collection ########("ports")### FreeBSD
ports ##########################

FreeBSD ############# port# #### port ############### —— ######## commit
#########

2. #### port

########## port ###############

######### port ################# port ####### # 10, *Upgrading* ###

####################### `/usr/ports/Mk/bsd.port.mk` ##### port # Makefile ####### ######### hacking Makefiles ########### port ###Makefile #### ########## ######## # port ######## FreeBSD ports #### ######

##

########## (*VAR*)### ########(overridden)#########(###)###### `/usr/ports/Mk/bsd.port.mk` ########## ############ tab ## ###### 1 # tab ## 4 # space# Emacs # Vim ################### vi(1) # ex(1) ########## `:set tabstop=4` #######

3. ## Port

############ port#### ############################

##################(tarball)###### DISTDIR ######## /usr/ports/distfiles #

##

############################## FreeBSD ##################
############

3.1. ## Makefile

Makefile

```
# 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>
```

#################### ## $FreeBSD$ ########## ### RCS ID ##### port #### port tree ## CVS ############ ############## sample Makefile ###

3.2.

########### package## 2 ######## port (Slave port####)######## # 2 ##### pkg-descr # # pkg-plist ## ########### pkg- ##########

3.2.1. pkg-descr

port ##################### port ####### WWW ##(####)#

##

############################## port ##### ####### README # manpage ######################### #####(## manpage ###### #)# ############################### ####### ###### WWW: ### ## ##################

port # pkg-descr ##########

```
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 ############### package ################packing list (####)## # ${PREFIX} ##### #########(${PREFIX} ### /usr/local # /usr/X11R6) ########## man page #### #### MAN*n*= ##### Makefile ###### pkg-plist ## ############ port ############ ## ######### pkg-plist ##### ###@dirrm # ############ Makefile ##### PLIST_FILES=

port # pkg-plist ### #######:

```
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
```

packing list ####### pkg_create(1) ####

##

############################ ################

##

################## port ########## #### #### packing list ######
###

############### pkg-plist ## #### port ############ ################ Makefile
PLIST_FILES # PLIST_DIRS #### ######### oneko port ##### pkg-plist
Makefile ########

```
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
```

port ############### PLIST_DIRS

PLIST_FILES # PLIST_DIRS ######## #### pkg_create(1) #### command sequences# ############# port ###### port #### ################ ports collection
pkg-plist ###

########## pkg-plist # PLIST_FILES ####### #######

3.3. ## checksum ### distinfo

####make makesum ##### ############ distinfo ### #

########## checksum ###### ################## (##################)##########
IGNOREFILES # ####### make makesum ############ checksum ###### IGNORE

3.4. ## port

############ port ########## port # package# ##############

- ## port ########## pkg-plist ##
- ## port ########## pkg-plist ##
- # port ### reinstall ##### #
- # port ########## cleans up#

3.1.

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

package # deinstall ################ ############################## ### ## ########### package #############

############ ports tinderbox# #### jails ################## ############### ## ### ports/ports-mgmt/tinderbox #

3.5. # portlint

portlint #### port ########### ### ports-mgmt/portlint### ports collection ##### ### ######### Makefile ######## package

3.6. ##(Submit) port

########### DOs and DON'Ts #####

########### port ################## FreeBSD ports tree ############### port# #### work ####### pkgname.tgz # package ##### ### ### shar `find port_dir` ### shar # ## ### send-pr(1) ########(send-pr(1) ####### #########)

PR ####(Category)## ports # #####(Class)## change-request (######## PR ##Confidential(##)### yes#)# ######("Description")# ############## shar ####### ("Fix")

##

Synopsis ###### PR ################## new ports ######### "New port: <category>/<portname> <# port ###>" #### port

> “Update port: <category>/<portname> <## update ###>”# ######## ####################

############ *source # distfile #### work ### ######## make package #### package#*

port ########## ##################### FreeBSD ports tree ###### ######### ## committed to FreeBSD # port

############ port ################# #### feedback ### ############ port tree ### # ####### Additional FreeBSD Contributors ######################### :-)

4. Slow Porting

Ok...#############port ################ ### ##########################

4.1. How things work

############# port #### make ############## ######### bsd.port.mk ############# ###

########### bsd.port.mk ##############... :->

1. ##### fetch ### fetch ### tarball ###### DISTDIR #### fetch # DISTDIR ########## Makefile ## MASTER_SITES URL ##### FTP ###### distfiles ### ftp://ftp.FreeBSD.org/pub/FreeBSD/ports/distfiles/ # ################ Internet ######## FETCH ######### ############ DISTDIR ############

2. ##### extract ###### DISTDIR #### port #####(### gzip ### tarball)####### WRKDIR ##########(### work ##)# #

3. ##### patch ### ####### PATCHFILES ###### patch ## ### PATCHDIR (### files ## #) ##### patch-* ################# patch#

4. ### configure ###### port ##################

    1. ### scripts/configure ### ###########

    2. ### HAS_CONFIGURE ## GNU_CONFIGURE ######## WRKSRC/configure

    3. ### USE_IMAKE ######## XMKMF (### xmkmf -a) #

5. ### build ####### port # working directory(# WRKSRC ###) ###### ### USE_GMAKE # ## ###### GNU make #### ######### make ####

####### make ####### ########################## pre-*something* # post-*something* # ######## script ## scripts ##### ######################

####### Makefile ### post-extract #### scripts ###### pre-build ############### ### post-extract ############## ### build ########## pre-build ## script ###### # ############### Makefile ##### ############################### #

########### bsd.port.mk ## do-*something* ####### ###do-extract ############## # ################# port # Makefile #####

##

The “main” targets (e.g., `extract`, `configure`, etc.) do nothing more than make sure all the stages up to that one are completed and call the real targets or scripts, and they are not intended to be changed. If you want to fix the extraction, fix `do-extract`, but never ever change the way `extract` operates!

######### make ########## ############# port#

4.2. ##### source

source #(##### `foo.tar.gz` # `foo.tar.Z` ######)############# `DISTDIR` ## ##### ##########################

`MASTER_SITES` ############# ##### `bsd.sites.mk` ############## ############ #### ################### port tree ## ######################## port

############ FTP ####(HTTP)## ############## ######################## FTP #### (HTTP) ##############

######(######)####### #### “house(##)” # `ftp.FreeBSD.org` ## committer ###### # ############ ###### committer # `freefall` ## `~/public_distfiles/` ####### #### commit ## port ### committer ### ############# ## committer ##### `MASTER_SITES` ## `MASTER_SITE_LOCAL` #### `MASTER_SITE_SUBDIR` ###### `freefall` #####

port ######################### ############################ `MASTER_SITES` ### ### ################(#########)###### ########## source code ##### ############ ################### checksum mismatch(########) ############ FTP ############ ## port ########### ################### `MASTER_SITES`

port ###### `patches'(### Internet ##)#### `DISTDIR` ######## patch ############## ### ##########(##### PATCHFILES ####)#

4.3. #### port

################ port ###### FreeBSD ######### ######### ##### ################ ### port ######################## ######### script ## patch #############

port ##################### ######## Larry Wall ##### Configure scripts ######## Ports collection ########## port ######## ###### “plug-and-play(####)”#

##

############### FreeBSD ports collection # patch #### script #### ############ BSD ########

4.4. Patching

port ################ #### diff(1) ######## ##### patch(1) ### ####### patch ###### patch-* ### * ### patch ########## ## patch-Imakefile # patch-src-config.h # ## ######## PATCHDIR (### files/ ############### ## patch ######## WRKSRC (#### port # tarball ###### port #########)# ################### patch ####### (#####patch-file # patch-file2 #### WRKSRC/foobar.c)#

[-+._a-zA-Z0-9] ####### patch ############### ####### patch #### patch-aa ## patch-ab ####

RCS ###### patch ##CVS ######## ports tree ################## check out ####### ######## patch ### RCS ####### ($) ##### ### $Id # $RCS

diff(1) ## recurse (-r) ## ### patch ############ patch ######## ############## ######## Imake # GNU configure #### Makefile ### patch# ########## patch ###### ########## configure.in ### autoconf ##### configure #### configure ## patch # (##########)# ### USE_AUTOTOOLS=autoconf:261 ## configure.in ## patch

######### whitespace ####### Open Source ### project ######### code base########## ###### #coding style# ################## ############ ####### CVS repository ### ## ######################### patch ###### #

################## post-extract ####### #### patch ###

port # Makefile ############### sed(1) # in-place mode ####### patch ###### ######

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

############ Windows® ######## ################ source file ## ## CR/LF ####### ##### patching#compiler warnings### scripts execution (### /bin/sh^M ###) ## ###### CR/LF # LF#### USE_DOS2UNIX=yes ## port # Makefile ### ##################

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

###################### DOS2UNIX_REGEX # ##### find ######### ####### re_format(7)# ###################### ################ binary ###

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

4.5.

############### configure script ######## scripts ####### ######## ##### Makefile #### ### pre-configure # post-configure # script ########

4.6.

port ################ #### Makefile ## IS_INTERACTIVE ### ########## BATCH # ########## port ### "overnight builds"(############ BATCH ##### ## ########## port #####)# ########## port ##########(#######)

####################### ###### PACKAGE_BUILDING ####### ################ ### ######### CDROM # FTP #####

5. ## Makefile

Makefile ######### ##############Also, there is a sample Makefile in this handbook, so take a look and please follow the ordering of variables and sections in that template to make your port easier for others to read.

Now, consider the following problems in sequence as you design your new Makefile:

5.1. The original source

Does it live in DISTDIR as a standard gzip'd tarball named something like foozolix-1.2.tar.gz ? If so, you can go on to the next step. If not, you should look at overriding any of the DISTVERSION , DISTNAME , EXTRACT_CMD , EXTRACT_BEFORE_ARGS , EXTRACT_AFTER_ARGS , EXTRACT_SUFX , or DISTFILES variables, depending on how alien a format your port's distribution file is. (The most common case is EXTRACT_SUFX=.tar.Z , when the tarball is condensed by regular compress , not gzip.)

In the worst case, you can simply create your own do-extract target to override the default, though this should be rarely, if ever, necessary.

5.2. Naming

The first part of the port's Makefile names the port, describes its version number, and lists it in the correct category.

5.2.1. PORTNAME and PORTVERSION

You should set PORTNAME to the base name of your port, and PORTVERSION to the version number of the port.

5.2.2. PORTREVISION and PORTEPOCH

5.2.2.1. PORTREVISION

The PORTREVISION variable is a monotonically increasing value which is reset to 0 with every increase of PORTVERSION (i.e. every time a new official vendor release is made), and appended to the package name if non-zero. Changes to PORTREVISION are used by automated tools (e.g. pkg_version(1)) to highlight the fact that a new package is available.

PORTREVISION should be increased each time a change is made to the port which significantly affects the content or structure of the derived package.

Examples of when PORTREVISION should be bumped:

- Addition of patches to correct security vulnerabilities, bugs, or to add new functionality to the port.
- Changes to the port `Makefile` to enable or disable compile-time options in the package.
- Changes in the packing list or the install-time behavior of the package (e.g. change to a script which generates initial data for the package, like ssh host keys).
- Version bump of a port's shared library dependency (in this case, someone trying to install the old package after installing a newer version of the dependency will fail since it will look for the old libfoo.x instead of libfoo.(x+1)).
- Silent changes to the port distfile which have significant functional differences, i.e. changes to the distfile requiring a correction to `distinfo` with no corresponding change to `PORTVERSION`, where a `diff -ru` of the old and new versions shows non-trivial changes to the code.

Examples of changes which do not require a `PORTREVISION` bump:

- Style changes to the port skeleton with no functional change to what appears in the resulting package.
- Changes to `MASTER_SITES` or other functional changes to the port which do not affect the resulting package.
- Trivial patches to the distfile such as correction of typos, which are not important enough that users of the package should go to the trouble of upgrading.
- Build fixes which cause a package to become compilable where it was previously failing (as long as the changes do not introduce any functional change on any other platforms on which the port did previously build). Since `PORTREVISION` reflects the content of the package, if the package was not previously buildable then there is no need to increase `PORTREVISION` to mark a change.

A rule of thumb is to ask yourself whether a change committed to a port is something which everyone would benefit from having (either because of an enhancement, fix, or by virtue that the new package will actually work at all), and weigh that against that fact that it will cause everyone who regularly updates their ports tree to be compelled to update. If yes, the `PORTREVISION` should be bumped.

5.2.2.2. PORTEPOCH

From time to time a software vendor or FreeBSD porter will do something silly and release a version of their software which is actually numerically less than the previous version. An example of this is a port which goes from foo-20000801 to foo-1.0 (the former will be incorrectly treated as a newer version since 20000801 is a numerically greater value than 1).

In situations such as this, the PORTEPOCH version should be increased. If PORTEPOCH is nonzero it is appended to the package name as described in section 0 above. PORTEPOCH must never be decreased or reset to zero, because that would cause comparison to a package from an earlier epoch to fail (i.e. the package would not be detected as out of date): the new version number (e.g. 1.0,1 in the above example) is still numerically less than the previous version (20000801), but the ,1 suffix is treated specially by automated tools and found to be greater than the implied suffix ,0 on the earlier package.

Dropping or resetting PORTEPOCH incorrectly leads to no end of grief; if you do not understand the above discussion, please keep after it until you do, or ask questions on the mailing lists.

It is expected that PORTEPOCH will not be used for the majority of ports, and that sensible use of PORTVERSION can often pre-empt it becoming necessary if a future release of the software should change the version structure. However, care is needed by FreeBSD porters when a vendor release is made without an official version number —— such as a code "snapshot" release. The temptation is to label the release with the release date, which will cause problems as in the example above when a new "official" release is made.

For example, if a snapshot release is made on the date 20000917, and the previous version of the software was version 1.2, the snapshot release should be given a PORTVERSION of 1.2.20000917 or similar, not 20000917, so that the succeeding release, say 1.3, is still a numerically greater value.

5.2.2.3. Example of PORTREVISION and PORTEPOCH usage

The gtkmumble port, version 0.10, is committed to the ports collection:

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

PKGNAME becomes gtkmumble-0.10 .

A security hole is discovered which requires a local FreeBSD patch. PORTREVISION is bumped accordingly.

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

PKGNAME becomes gtkmumble-0.10_1

A new version is released by the vendor, numbered 0.2 (it turns out the author actually intended 0.10 to actually mean 0.1.0, not "what comes after 0.9" - oops, too late now). Since the new minor version 2 is numerically less than the previous version 10, the PORTEPOCH must be bumped to manually force the new package to be detected as "newer".

Since it is a new vendor release of the code, PORTREVISION is reset to 0 (or removed from the Makefile).

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

PKGNAME becomes gtkmumble-0.2,1

The next release is 0.3. Since PORTEPOCH never decreases, the version variables are now:

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

PKGNAME becomes gtkmumble-0.3,1

##

If PORTEPOCH were reset to 0 with this upgrade, someone who had installed the gtkmumble-0.10_1 package would not detect the gtkmumble-0.3 package as newer, since 3 is still numerically less than 10. Remember, this is the whole point of PORTEPOCH in the first place.

5.2.3. PKGNAMEPREFIX and PKGNAMESUFFIX

Two optional variables, PKGNAMEPREFIX and PKGNAMESUFFIX, are combined with PORTNAME and PORTVERSION to form PKGNAME as ${PKGNAMEPREFIX}${PORTNAME}${PKGNAMESUFFIX}-${PORTVERSION}. Make sure this conforms to our guidelines for a good package name. In particular, you are *not* allowed to use a hyphen (-) in PORTVERSION. Also, if the package name has the *language*- or the *-compiled.specifics* part (see below), use PKGNAMEPREFIX and PKGNAMESUFFIX, respectively. Do not make them part of PORTNAME.

5.2.4. LATEST_LINK

########################## ########## PORTNAME # PKGNAMEPREFIX ### PKGNAMESUFFIX ################## ### port # index ## package ###### ####### the optional LATEST_LINK variable should be set to a different value for all ports except the “main” one —— see the editors/vim5 and editors/vim ports, and the www/apache* family for examples of its use. Note that how to choose a “main” version —— “most popular”, “best supported”, “least patched”, and so on —— is outside the scope of this handbook's recommendations; we only tell you how to specify the other ports' versions after you have picked a “main” one.

5.2.5. Package Naming Conventions

The following are the conventions you should follow in naming your packages. This is to have our package directory easy to scan, as there are already thousands of packages and users are going to turn away if they hurt their eyes!

The package name should look like `language_region-name-compiled.specifics-version.numbers`.

The package name is defined as `${PKGNAMEPREFIX}${PORTNAME}${PKGNAMESUFFIX}-${PORTVERSION}`. Make sure to set the variables to conform to that format.

1. FreeBSD strives to support the native language of its users. The *`language-`* part should be a two letter abbreviation of the natural language defined by ISO-639 if the port is specific to a certain language. Examples are `ja` for Japanese, `ru` for Russian, `vi` for Vietnamese, `zh` for Chinese, `ko` for Korean and `de` for German.

   If the port is specific to a certain region within the language area, add the two letter country code as well. Examples are `en_US` for US English and `fr_CH` for Swiss French.

   The *`language-`* part should be set in the `PKGNAMEPREFIX` variable.

2. The first letter of the `name` part should be lowercase. (The rest of the name may contain capital letters, so use your own discretion when you are converting a software name that has some capital letters in it.) There is a tradition of naming `Perl 5` modules by prepending `p5-` and converting the double-colon separator to a hyphen; for example, the `Data::Dumper` module becomes `p5-Data-Dumper`.

3. Make sure that the port's name and version are clearly separated and placed into the `PORTNAME` and `PORTVERSION` variables. The only reason for `PORTNAME` to contain a version part is if the upstream distribution is really named that way, as in the `textproc/libxml2` or `japanese/kinput2-freewnn` ports. Otherwise, the `PORTNAME` should not contain any version-specific information. It is quite normal for several ports to have the same `PORTNAME`, as the `www/apache*` ports do; in that case, different versions (and different index entries) are distinguished by the `PKGNAMEPREFIX`, `PKGNAMESUFFIX`, and `LATEST_LINK` values.

4. If the port can be built with different hardcoded defaults (usually part of the directory name in a family of ports), the *`-compiled.specifics`* part should state the compiled-in defaults (the hyphen is optional). Examples are papersize and font units.

   The *`-compiled.specifics`* part should be set in the `PKGNAMESUFFIX` variable.

5. The version string should follow a dash (-) and be a period-separated list of integers and single lowercase alphabetics. In particular, it is not permissible to have another dash inside the version string. The only exception is the string `pl` (meaning "patchlevel"), which can be used *only* when there are no major and minor version

numbers in the software. If the software version has strings like “alpha”, “beta”, “rc”, or “pre”, take the first letter and put it immediately after a period. If the version string continues after those names, the numbers should follow the single alphabet without an extra period between them.

The idea is to make it easier to sort ports by looking at the version string. In particular, make sure version number components are always delimited by a period, and if the date is part of the string, use the `yyyy.mm.dd` format, not `dd.mm.yyyy` or the non-Y2K compliant `yy.mm.dd` format.

Here are some (real) examples on how to convert the name as called by the software authors to a suitable package name:

Distribution Name	**PKGNAMEPREFI**	**PORTNAME**	**PKGNAMESUFFI**	**PORTVERSION**	Reason
mule-2.2.2	(empty)	mule	(empty)	2.2.2	No changes required
EmiClock-1.0.2	(empty)	emiclock	(empty)	1.0.2	No uppercase names for single programs
rdist-1.3alpha	(empty)	rdist	(empty)	1.3.a	No strings like `alpha` allowed
es-0.9-beta1	(empty)	es	(empty)	0.9.b1	No strings like `beta` allowed
mailman-2.0rc3	(empty)	mailman	(empty)	2.0.r3	No strings like `rc` allowed
v3.3beta021.src	(empty)	tiff	(empty)	3.3	What the heck was that anyway?
tvtwm	(empty)	tvtwm	(empty)	pl11	Version string always required
piewm	(empty)	piewm	(empty)	1.0	Version string always required
xvgr-2.10pl1	(empty)	xvgr	(empty)	2.10.1	`pl` allowed only when

Distribution Name	PKGNAMEPREFI	PORTNAME	PKGNAMESUFFI	PORTVERSION	Reason
					no major/ minor version numbers
gawk-2.15.6	ja-	gawk	(empty)	2.15.6	Japanese language version
psutils-1.13	(empty)	psutils	-letter	1.13	Papersize hardcoded at package build time
pkfonts	(empty)	pkfonts	300	1.0	Package for 300dpi fonts

If there is absolutely no trace of version information in the original source and it is unlikely that the original author will ever release another version, just set the version string to `1.0` (like the `piewm` example above). Otherwise, ask the original author or use the date string (`yyyy.mm.dd`) as the version.

5.3. Categorization

5.3.1. CATEGORIES

When a package is created, it is put under `/usr/ports/packages/All` and links are made from one or more subdirectories of `/usr/ports/packages` . The names of these subdirectories are specified by the variable `CATEGORIES` . It is intended to make life easier for the user when he is wading through the pile of packages on the FTP site or the CDROM. Please take a look at the current list of categories and pick the ones that are suitable for your port.

This list also determines where in the ports tree the port is imported. If you put more than one category here, it is assumed that the port files will be put in the subdirectory with the name in the first category. See below for more discussion about how to pick the right categories.

5.3.2. Current list of categories

Here is the current list of port categories. Those marked with an asterisk (*) are *virtual* categories——those that do not have a corresponding subdirectory in the ports tree. They are only used as secondary categories, and only for search purposes.

##

For non-virtual categories, you will find a one-line description in the `COMMENT` in that subdirectory's `Makefile`.

Category	Description	Notes
`accessibility`	Ports to help disabled users.	
`afterstep*`	Ports to support the AfterStep window manager.	
`arabic`	Arabic language support.	
`archivers`	Archiving tools.	
`astro`	Astronomical ports.	
`audio`	Sound support.	
`benchmarks`	Benchmarking utilities.	
`biology`	Biology-related software.	
`cad`	Computer aided design tools.	
`chinese`	Chinese language support.	
`comms`	Communication software.	Mostly software to talk to your serial port.
`converters`	Character code converters.	
`databases`	Databases.	
`deskutils`	Things that used to be on the desktop before computers were invented.	
`devel`	Development utilities.	Do not put libraries here just because they are libraries ——unless they truly do not belong anywhere else, they should not be in this category.
`dns`	DNS-related software.	
`editors`	General editors.	Specialized editors go in the section for those

Category	Description	Notes
		tools (e.g., a mathematical-formula editor will go in `math`).
`elisp*`	Emacs-lisp ports.	
`emulators`	Emulators for other operating systems.	Terminal emulators do *not* belong here——X-based ones should go to `x11` and text-based ones to either `comms` or `misc`, depending on the exact functionality.
`finance`	Monetary, financial and related applications.	
`french`	French language support.	
`ftp`	FTP client and server utilities.	If your port speaks both FTP and HTTP, put it in `ftp` with a secondary category of `www`.
`games`	`###`	
`geography*`	`##########`	
`german`	`#######`	
`gnome*`	Ports from the GNOME Project.	
`gnustep*`	GNUstep `##########`	
`graphics`	`##########`	
`hamradio*`	Software for amateur radio.	
`haskell*`	Software related to the Haskell language.	
`hebrew`	Hebrew language support.	
`hungarian`	Hungarian language support.	
`ipv6*`	IPv6 related software.	
`irc`	Internet Relay Chat utilities.	
`japanese`	Japanese language support.	
`java`	Software related to the Java language.	The `java` category shall not be the only one for a port. Save for ports directly related to the Java language,

Category	Description	Notes
		porters are also encouraged not to use `java` as the main category of a port.
`kde*`	Ports from the K Desktop Environment (KDE) Project.	
`kld*`	Kernel loadable modules#	
`korean`	Korean language support.	
`lang`	Programming languages.	
`linux*`	Linux applications and support utilities.	
`lisp*`	Software related to the Lisp language.	
`mail`	Mail software.	
`math`	Numerical computation software and other utilities for mathematics.	
`mbone`	MBone applications.	
`misc`	Miscellaneous utilities	Basically things that do not belong anywhere else. If at all possible, try to find a better category for your port than `misc`, as ports tend to get overlooked in here.
`multimedia`	Multimedia software.	
`net`	Miscellaneous networking software.	
`net-im`	Instant messaging software.	
`net-mgmt`	Networking management software.	
`net-p2p`	Peer to peer network applications.	
`news`	USENET news software.	
`palm`	Software support for the Palm™ series.	
`parallel*`	Applications dealing with parallelism in computing.	

Category	Description	Notes
`pear*`	Ports related to the Pear PHP framework.	
`perl5*`	Ports that require Perl version 5 to run.	
`plan9*`	Various programs from Plan9.	
`polish`	Polish language support.	
`ports-mgmt`	FreeBSD ports # packages ## ########	
`portuguese`	Portuguese language support.	
`print`	Printing software.	Desktop publishing tools (previewers, etc.) belong here too.
`python*`	Software related to the Python language.	
`ruby*`	Software related to the Ruby language.	
`rubygems*`	Ports of RubyGems packages.	
`russian`	Russian language support.	
`scheme*`	Software related to the Scheme language.	
`science`	Scientific ports that do not fit into other categories such as `astro`, `biology` and `math`.	
`security`	Security utilities.	
`shells`	Command line shells.	
`spanish*`	##########	
`sysutils`	System utilities.	
`tcl*`	## Tcl ######	
`textproc`	Text processing utilities.	It does not include desktop publishing tools, which go to `print`.
`tk*`	## Tk ######	

Category	Description	Notes
ukrainian	Ukrainian language support.	
vietnamese	Vietnamese language support.	
windowmaker*	Ports to support the WindowMaker window manager.	
www	Software related to the World Wide Web.	HTML language support belongs here too.
x11	The X Window System and friends.	This category is only for software that directly supports the window system. Do not put regular X applications here; most of them should go into other x11-* categories (see below). If your port *is* an X application, define USE_XLIB (implied by USE_IMAKE) and put it in the appropriate category.
x11-clocks	X11 clocks.	
x11-drivers	X11 #####	
x11-fm	X11 file managers.	
x11-fonts	X11 fonts and font utilities.	
x11-servers	X11 servers.	
x11-themes	X11 themes.	
x11-toolkits	X11 toolkits.	
x11-wm	X11 window managers.	
xfce*	Xfce ##########	
zope*	Zope support.	

5.3.3. Choosing the right category

As many of the categories overlap, you often have to choose which of the categories should be the primary category of your port. There are several rules that govern this issue. Here is the list of priorities, in decreasing order of precedence:

- The first category must be a physical category (see above). This is necessary to make the packaging work. Virtual categories and physical categories may be intermixed after that.
- Language specific categories always come first. For example, if your port installs Japanese X11 fonts, then your `CATEGORIES` line would read `japanese x11-fonts` .
- Specific categories are listed before less-specific ones. For instance, an HTML editor should be listed as `www editors` , not the other way around. Also, you should not list `net` when the port belongs to any of `irc`, `mail`, `mbone`, `news`, `security` , or `www`, as `net` is included implicitly.
- `x11` is used as a secondary category only when the primary category is a natural language. In particular, you should not put `x11` in the category line for X applications.
- Emacs modes should be placed in the same ports category as the application supported by the mode, not in `editors` . For example, an Emacs mode to edit source files of some programming language should go into `lang`.
- `###` kernel loadable modules `#` port `##` `CATEGORIES` `###` `kld` `#######`
- `misc` should not appear with any other non-virtual category. If you have `misc` with something else in your `CATEGORIES` line, that means you can safely delete `misc` and just put the port in that other subdirectory!
- If your port truly does not belong anywhere else, put it in `misc`.

If you are not sure about the category, please put a comment to that effect in your send-pr(1) submission so we can discuss it before we import it. If you are a committer, send a note to the FreeBSD ports `####` so we can discuss it first. Too often, new ports are imported to the wrong category only to be moved right away. This causes unnecessary and undesirable bloat in the master source repository.

5.3.4. Proposing a new category

As the Ports Collection has grown over time, various new categories have been introduced. New categories can either be *virtual* categories——those that do not have a corresponding subdirectory in the ports tree—— or *physical* categories——those that do. The following text discusses the issues involved in creating a new physical category so that you can understand them before you propose one.

Our existing practice has been to avoid creating a new physical category unless either a large number of ports would logically belong to it, or the ports that would belong to it are a logically distinct group that is of limited general interest (for instance, categories related to spoken human languages), or preferably both.

The rationale for this is that such a change creates a fair amount of work for both the committers and also for all users who track changes to the Ports Collection. In addition,

proposed category changes just naturally seem to attract controversy. (Perhaps this is because there is no clear consensus on when a category is “too big”, nor whether categories should lend themselves to browsing (and thus what number of categories would be an ideal number), and so forth.)

Here is the procedure:

1. Propose the new category on FreeBSD ports ####. You should include a detailed rationale for the new category, including why you feel the existing categories are not sufficient, and the list of existing ports proposed to move. (If there are new ports pending in GNATS that would fit this category, list them too.) If you are the maintainer and/or submitter, respectively, mention that as it may help you to make your case.

2. Participate in the discussion.

3. If it seems that there is support for your idea, file a PR which includes both the rationale and the list of existing ports that need to be moved. Ideally, this PR should also include patches for the following:

   - `Makefile`s for the new ports once they are repocopied
   - `Makefile` for the new category
   - `Makefile` for the old ports' categories
   - `Makefile`s for ports that depend on the old ports
   - (for extra credit, you can include the other files that have to change, as per the procedure in the Committer's Guide.)

4. Since it affects the ports infrastructure and involves not only performing repocopies but also possibly running regression tests on the build cluster, the PR should be assigned to the Ports Management Team <portmgr@FreeBSD.org >.

5. If that PR is approved, a committer will need to follow the rest of the procedure that is outlined in the Committer's Guide.

Proposing a new virtual category should be similar to the above but much less involved, since no ports will actually have to move. In this case, the only patches to include in the PR would be those to add the new category to the `CATEGORIES` of the affected ports.

5.3.5. Proposing reorganizing all the categories

Occasionally someone proposes reorganizing the categories with either a 2-level structure, or some other kind of keyword structure. To date, nothing has come of any of these proposals because, while they are very easy to make, the effort involved to retrofit

the entire existing ports collection with any kind of reorganization is daunting to say the very least. Please read the history of these proposals in the mailing list archives before you post this idea; furthermore, you should be prepared to be challenged to offer a working prototype.

5.4. The distribution files

The second part of the `Makefile` describes the files that must be downloaded in order to build the port, and where they can be downloaded from.

5.4.1. `DISTVERSION/DISTNAME`

`DISTNAME` is the name of the port as called by the authors of the software. `DISTNAME` defaults to `${PORTNAME}-${PORTVERSION}`, so override it only if necessary. `DISTNAME` is only used in two places. First, the distribution file list (`DISTFILES`) defaults to `${DISTNAME}${EXTRACT_SUFX}`. Second, the distribution file is expected to extract into a subdirectory named `WRKSRC`, which defaults to `work/${DISTNAME}`.

Some vendor's distribution names which do not fit into the `${PORTNAME}-${PORTVERSION}`-scheme can be handled automatically by setting `DISTVERSION`. `PORTVERSION` and `DISTNAME` will be derived automatically, but can of course be overridden. The following table lists some examples:

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

##

`PKGNAMEPREFIX` and `PKGNAMESUFFIX` do not affect `DISTNAME`. Also note that if `WRKSRC` is equal to `work/${PORTNAME}-${PORTVERSION}` while the original source archive is named something other than `${PORTNAME}-${PORTVERSION}${EXTRACT_SUFX}`, you should probably leave `DISTNAME` alone—— you are better off defining `DISTFILES` than having to set both `DISTNAME` and `WRKSRC` (and possibly `EXTRACT_SUFX`).

5.4.2. MASTER_SITES

Record the directory part of the FTP/HTTP-URL pointing at the original tarball in MASTER_SITES . Do not forget the trailing slash (/)!

The make macros will try to use this specification for grabbing the distribution file with FETCH if they cannot find it already on the system.

It is recommended that you put multiple sites on this list, preferably from different continents. This will safeguard against wide-area network problems. We are even planning to add support for automatically determining the closest master site and fetching from there; having multiple sites will go a long way towards helping this effort.

If the original tarball is part of one of the popular archives such as X-contrib, GNU, or Perl CPAN, you may be able refer to those sites in an easy compact form using MASTER_SITE_ * (###MASTER_SITE_XCONTRIB # MASTER_SITE_GNU # MASTER_SITE_PERL_CPAN)# Simply set MASTER_SITES to one of these variables and MASTER_SITE_SUBDIR to the path within the archive. Here is an example:

```
MASTER_SITES=         ${MASTER_SITE_XCONTRIB}
MASTER_SITE_SUBDIR=   applications
```

These variables are defined in /usr/ports/Mk/bsd.sites.mk . There are new entries added all the time, so make sure to check the latest version of this file before submitting a port.

The user can also set the MASTER_SITE_* variables in /etc/make.conf to override our choices, and use their favorite mirrors of these popular archives instead.

5.4.3. EXTRACT_SUFX

If you have one distribution file, and it uses an odd suffix to indicate the compression mechanism, set EXTRACT_SUFX .

For example, if the distribution file was named foo.tgz instead of the more normal foo.tar.gz , you would write:

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

The USE_BZIP2 and USE_ZIP variables automatically set EXTRACT_SUFX to .tar.bz2 or .zip as necessary. If neither of these are set then EXTRACT_SUFX defaults to .tar.gz .

##

You never need to set both EXTRACT_SUFX and DISTFILES .

5.4.4. DISTFILES

Sometimes the names of the files to be downloaded have no resemblance to the name of the port. For example, it might be called source.tar.gz or similar. In other cases the application's source code might be in several different archives, all of which must be downloaded.

If this is the case, set DISTFILES to be a space separated list of all the files that must be downloaded.

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

If not explicitly set, DISTFILES defaults to ${DISTNAME}${EXTRACT_SUFX} .

5.4.5. EXTRACT_ONLY

If only some of the DISTFILES must be extracted——for example, one of them is the source code, while another is an uncompressed document——list the filenames that must be extracted in EXTRACT_ONLY .

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

If *none* of the DISTFILES should be uncompressed then set EXTRACT_ONLY to the empty string.

```
EXTRACT_ONLY=
```

5.4.6. PATCHFILES

If your port requires some additional patches that are available by FTP or HTTP, set PATCHFILES to the names of the files and PATCH_SITES to the URL of the directory that contains them (the format is the same as MASTER_SITES).

If the patch is not relative to the top of the source tree (i.e., WRKSRC) because it contains some extra pathnames, set PATCH_DIST_STRIP accordingly. For instance, if all the pathnames in the patch have an extra foozolix-1.0/ in front of the filenames, then set PATCH_DIST_STRIP=-p1 .

Do not worry if the patches are compressed; they will be decompressed automatically if the filenames end with .gz or .Z.

If the patch is distributed with some other files, such as documentation, in a gzip'd tarball, you cannot just use PATCHFILES. If that is the case, add the name and the location of the patch tarball to DISTFILES and MASTER_SITES. Then, use the EXTRA_PATCHES variable to point to those files and bsd.port.mk will automatically apply them for you. In particular, do *not* copy patch files into the PATCHDIR directory——that directory may not be writable.

##

The tarball will have been extracted alongside the regular source by then, so there is no need to explicitly extract it if it is a regular gzip'd or compress'd tarball. If you do the latter, take extra care not to overwrite something that already exists in that directory. Also, do not forget to add a command to remove the copied patch in the pre-clean target.

5.4.7. Multiple distribution files or patches from different sites and subdirectories (MASTER_SITES:n)

(Consider this to be a somewhat "advanced topic"; those new to this document may wish to skip this section at first).

This section has information on the fetching mechanism known as both MASTER_SITES:n and MASTER_SITES_NN. We will refer to this mechanism as MASTER_SITES:n hereon.

A little background first. OpenBSD has a neat feature inside both DISTFILES and PATCHFILES variables, both files and patches can be postfixed with :n identifiers where n both can be [0-9] and denote a group designation. For example:

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

In OpenBSD, distribution file alpha will be associated with variable MASTER_SITES0 instead of our common MASTER_SITES and beta with MASTER_SITES1.

This is a very interesting feature which can decrease that endless search for the correct download site.

Just picture 2 files in DISTFILES and 20 sites in MASTER_SITES, the sites slow as hell where beta is carried by all sites in MASTER_SITES, and alpha can only be found in the 20th site. It would be such a waste to check all of them if maintainer knew this beforehand, would it not? Not a good start for that lovely weekend!

Now that you have the idea, just imagine more DISTFILES and more MASTER_SITES. Surely our "distfiles survey meister" would appreciate the relief to network strain that this would bring.

In the next sections, information will follow on the FreeBSD implementation of this idea. We improved a bit on OpenBSD's concept.

5.4.7.1. Simplified information

This section tells you how to quickly prepare fine grained fetching of multiple distribution files and patches from different sites and subdirectories. We describe here a case of simplified `MASTER_SITES:n` usage. This will be sufficient for most scenarios. However, if you need further information, you will have to refer to the next section.

Some applications consist of multiple distribution files that must be downloaded from a number of different sites. For example, Ghostscript consists of the core of the program, and then a large number of driver files that are used depending on the user's printer. Some of these driver files are supplied with the core, but many others must be downloaded from a variety of different sites.

To support this, each entry in `DISTFILES` may be followed by a colon and a "tag name". Each site listed in `MASTER_SITES` is then followed by a colon, and the tag that indicates which distribution files should be downloaded from this site.

For example, consider an application with the source split in two parts, `source1.tar.gz` and `source2.tar.gz` , which must be downloaded from two different sites. The port's `Makefile` would include lines like ## 5.1, "Simplified use of `MASTER_SITES:n` with 1 file per site".

5.1. Simplified use of `MASTER_SITES:n` with 1 file per site

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

Multiple distribution files can have the same tag. Continuing the previous example, suppose that there was a third distfile, `source3.tar.gz` , that should be downloaded from `ftp.example2.com` . The `Makefile` would then be written like ## 5.2, "Simplified use of `MASTER_SITES:n` with more than 1 file per site".

5.2. Simplified use of `MASTER_SITES:n` with more than 1 file per site

```
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. Detailed information

Okay, so the previous section example did not reflect your needs? In this section we will explain in detail how the fine grained fetching mechanism `MASTER_SITES:n` works and how you can modify your ports to use it.

1. Elements can be postfixed with `:n` where *n* is `[^:,]+`, i.e., *n* could conceptually be any alphanumeric string but we will limit it to `[a-zA-Z_][0-9a-zA-Z_]+` for now.

   Moreover, string matching is case sensitive; i.e., `n` is different from `N`.

   However, the following words cannot be used for postfixing purposes since they yield special meaning: `default`, `all` and `ALL` (they are used internally in item ii). Furthermore, `DEFAULT` is a special purpose word (check item 3).

2. Elements postfixed with `:n` belong to the group `n`, `:m` belong to group `m` and so forth.

3. Elements without a postfix are groupless, i.e., they all belong to the special group `DEFAULT`. If you postfix any elements with `DEFAULT`, you are just being redundant unless you want to have an element belonging to both `DEFAULT` and other groups at the same time (check item 5).

   The following examples are equivalent but the first one is preferred:

   ```
   MASTER_SITES=   alpha

   MASTER_SITES=   alpha:DEFAULT
   ```

4. Groups are not exclusive, an element may belong to several different groups at the same time and a group can either have either several different elements or none at all. Repeated elements within the same group will be simply that, repeated elements.

5. When you want an element to belong to several groups at the same time, you can use the comma operator (`,`).

   Instead of repeating it several times, each time with a different postfix, we can list several groups at once in a single postfix. For instance, `:m,n,o` marks an element that belongs to group `m`, `n` and `o`.

   All the following examples are equivalent but the last one is preferred:

   ```
   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. All sites within a given group are sorted according to `MASTER_SORT_AWK` . All groups within `MASTER_SITES` and `PATCH_SITES` are sorted as well.

7. Group semantics can be used in any of the following variables `MASTER_SITES` , `PATCH_SITES` , `MASTER_SITE_SUBDIR` , `PATCH_SITE_SUBDIR` , `DISTFILES` , and `PATCHFILES` according to the following syntax:

   a. All `MASTER_SITES` , `PATCH_SITES` , `MASTER_SITE_SUBDIR` and `PATCH_SITE_SUBDIR` elements must be terminated with the forward slash `/` character. If any elements belong to any groups, the group postfix `:n` must come right after the terminator `/`. The `MASTER_SITES:n` mechanism relies on the existence of the terminator `/` to avoid confusing elements where a `:n` is a valid part of the element with occurrences where `:n` denotes group `n`. For compatibility purposes, since the `/` terminator was not required before in both `MASTER_SITE_SUBDIR` and `PATCH_SITE_SUBDIR` elements, if the postfix immediate preceding character is not a `/` then `:n` will be considered a valid part of the element instead of a group postfix even if an element is postfixed with `:n`. See both ## 5.3, “Detailed use of `MASTER_SITES:n` in `MASTER_SITE_SUBDIR` ” and ## 5.4, “Detailed use of `MASTER_SITES:n` with comma operator, multiple files, multiple sites and multiple subdirectories”.

5.3. Detailed use of `MASTER_SITES:n` in `MASTER_SITE_SUBDIR`

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

- Directories within group `DEFAULT` -> old:n
- Directories within group `NEW` -> new

5.4. Detailed use of `MASTER_SITES:n` with comma operator, multiple files, multiple sites and multiple subdirectories

```
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
```

The previous example results in the following fine grained fetching. Sites are listed in the exact order they will be used.

- `file1` will be fetched from
  - `MASTER_SITE_OVERRIDE`
  - http://site1/directory-trial:1/
  - http://site1/directory-one/
  - http://site1/directory/
  - http://site2/
  - http://site7/
  - `MASTER_SITE_BACKUP`
- `file2` will be fetched exactly as `file1` since they both belong to the same group
  - `MASTER_SITE_OVERRIDE`
  - http://site1/directory-trial:1/
  - http://site1/directory-one/
  - http://site1/directory/
  - http://site2/
  - http://site7/
  - `MASTER_SITE_BACKUP`
- `file3` will be fetched from

- MASTER_SITE_OVERRIDE
- http://site3/
- MASTER_SITE_BACKUP

- file4 will be fetched from
  - MASTER_SITE_OVERRIDE
  - http://site4/
  - http://site5/
  - http://site6/
  - http://site7/
  - http://site8/directory-one/
  - MASTER_SITE_BACKUP
- file5 will be fetched from
  - MASTER_SITE_OVERRIDE
  - MASTER_SITE_BACKUP
- file6 will be fetched from
  - MASTER_SITE_OVERRIDE
  - http://site8/
  - MASTER_SITE_BACKUP

8. How do I group one of the special variables from bsd.sites.mk, e.g., MASTER_SITE_SOURCEFORGE ?

   See ## 5.5, "Detailed use of MASTER_SITES:n with MASTER_SITE_SOURCEFORGE ".

5.5. Detailed use of MASTER_SITES:n with MASTER_SITE_SOURCEFORGE

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

something.tar.gz will be fetched from all sites within MASTER_SITE_SOURCEFORGE .

9. How do I use this with PATCH* variables?

   All examples were done with MASTER* variables but they work exactly the same for PATCH* ones as can be seen in ## 5.6, "Simplified use of MASTER_SITES:n with PATCH_SITES .".

5.6. Simplified use of MASTER_SITES:n with PATCH_SITES .

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

5.4.7.3. What does change for ports? What does not?

i. All current ports remain the same. The MASTER_SITES:n feature code is only activated if there are elements postfixed with :n like elements according to the aforementioned syntax rules, especially as shown in item 7.

ii. The port targets remain the same: checksum , makesum , patch , configure , build , etc. With the obvious exceptions of do-fetch , fetch-list , master-sites and patch-sites .

- do-fetch : deploys the new grouping postfixed DISTFILES and PATCHFILES with their matching group elements within both MASTER_SITES and PATCH_SITES which use matching group elements within both MASTER_SITE_SUBDIR and PATCH_SITE_SUBDIR . Check ## 5.4, "Detailed use of MASTER_SITES:n with comma operator, multiple files, multiple sites and multiple subdirectories".

- `fetch-list` : works like old `fetch-list` with the exception that it groups just like `do-fetch` .

- `master-sites` and `patch-sites` : (incompatible with older versions) only return the elements of group `DEFAULT` ; in fact, they execute targets `master-sites-default` and `patch-sites-default` respectively.

  Furthermore, using target either `master-sites-all` or `patch-sites-all` is preferred to directly checking either `MASTER_SITES` or `PATCH_SITES` . Also, directly checking is not guaranteed to work in any future versions. Check item B for more information on these new port targets.

iii. New port targets

A. There are `master-sites-` *n* and `patch-sites-` *n* targets which will list the elements of the respective group *n* within `MASTER_SITES` and `PATCH_SITES` respectively. For instance, both `master-sites-DEFAULT` and `patch-sites-DEFAULT` will return the elements of group `DEFAULT` , `master-sites-test` and `patch-sites-test` of group `test` , and thereon.

B. There are new targets `master-sites-all` and `patch-sites-all` which do the work of the old `master-sites` and `patch-sites` ones. They return the elements of all groups as if they all belonged to the same group with the caveat that it lists as many `MASTER_SITE_BACKUP` and `MASTER_SITE_OVERRIDE` as there are groups defined within either `DISTFILES` or `PATCHFILES` ; respectively for `master-sites-all` and `patch-sites-all` .

5.4.8. `DIST_SUBDIR`

Do not let your port clutter `/usr/ports/distfiles` . If your port requires a lot of files to be fetched, or contains a file that has a name that might conflict with other ports (e.g., `Makefile`), set `DIST_SUBDIR` to the name of the port (`${PORTNAME}` or `${PKGNAMEPREFIX}${PORTNAME}` should work fine). This will change `DISTDIR` from the default `/usr/ports/distfiles` to `/usr/ports/distfiles/DIST_SUBDIR` , and in effect puts everything that is required for your port into that subdirectory.

It will also look at the subdirectory with the same name on the backup master site at `ftp.FreeBSD.org` . (Setting `DISTDIR` explicitly in your `Makefile` will not accomplish this, so please use `DIST_SUBDIR` .)

##

This does not affect the MASTER_SITES you define in your Makefile.

5.4.9. ALWAYS_KEEP_DISTFILES

If your port uses binary distfiles and has a license that requires that the source code is provided with packages distributed in binary form, e.g. GPL, ALWAYS_KEEP_DISTFILES will instruct the FreeBSD build cluster to keep a copy of the files specified in DISTFILES. Users of these ports will generally not need these files, so it is a good idea to only add the source distfiles to DISTFILES when PACKAGE_BUILDING is defined.

5.7. Use of ALWAYS_KEEP_DISTFILES.

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

When adding extra files to DISTFILES, make sure you also add them to distinfo. Also, the additional files will normally be extracted into WRKDIR as well, which for some ports may lead to undesirable sideeffects and require special handling.

5.5. MAINTAINER

Set your mail-address here. Please. *:-)*

Note that only a single address without the comment part is allowed as a MAINTAINER value. The format used should be user@hostname.domain. Please do not include any descriptive text such as your real name in this entry——that merely confuses bsd.port.mk.

The maintainer is responsible for keeping the port up to date, and ensuring the port works correctly. For a detailed description of the responsibilities of a port maintainer, refer to the The challenge for port maintainers section.

Changes to the port will be sent to the maintainer of a port for a review and an approval before being committed. If the maintainer does not respond to an

update request after two weeks (excluding major public holidays), then that is considered a maintainer timeout, and the update may be made without explicit maintainer approval. If the maintainer does not respond within three months, then that maintainer is considered absent without leave, and can be replaced as the maintainer of the particular port in question. Exceptions to this are anything maintained by the Ports Management Team <portmgr@FreeBSD.org >, or the Security Officer Team <security-officer@FreeBSD.org >. No unauthorized commits may ever be made to ports maintained by those groups.

We reserve the right to modify the maintainer's submission to better match existing policies and style of the Ports Collection without explicit blessing from the submitter. Also, large infrastructural changes can result in a port being modified without maintainer's consent. This kind of changes will never affect the port's functionality.

The Ports Management Team <portmgr@FreeBSD.org > reserves the right to revoke or override anyone's maintainership for any reason, and the Security Officer Team <security-officer@FreeBSD.org > reserves the right to revoke or override maintainership for security reasons.

5.6. COMMENT

This is a one-line description of the port. *Please* do not include the package name (or version number of the software) in the comment. The comment should begin with a capital and end without a period. Here is an example:

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

The COMMENT variable should immediately follow the MAINTAINER variable in the `Makefile`.

Please try to keep the COMMENT line less than 70 characters, as it is displayed to users as a one-line summary of the port.

5.7. Dependencies

Many ports depend on other ports. There are seven variables that you can use to ensure that all the required bits will be on the user's machine. There are also some pre-supported dependency variables for common cases, plus a few more to control the behavior of dependencies.

5.7.1. LIB_DEPENDS

This variable specifies the shared libraries this port depends on. It is a list of *`lib`*:*`dir`*[:target] tuples where *`lib`* is the name of the shared library, *`dir`* is the directory in

which to find it in case it is not available, and *target* is the target to call in that directory. For example,

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

will check for a shared jpeg library with major version 9, and descend into the `graphics/jpeg` subdirectory of your ports tree to build and install it if it is not found. The *target* part can be omitted if it is equal to `DEPENDS_TARGET` (which defaults to `install`).

##

The *lib* part is a regular expression which is being looked up in the `ldconfig -r` output. Values such as `intl.[5-7]` and `intl` are allowed. The first pattern, `intl.[5-7]`, will match any of: `intl.5`, `intl.6` or `intl.7`. The second pattern, `intl`, will match any version of the `intl` library.

The dependency is checked twice, once from within the `extract` target and then from within the `install` target. Also, the name of the dependency is put into the package so that pkg_add(1) will automatically install it if it is not on the user's system.

5.7.2. RUN_DEPENDS

This variable specifies executables or files this port depends on during run-time. It is a list of *path*:*dir*[:target] tuples where *path* is the name of the executable or file, *dir* is the directory in which to find it in case it is not available, and *target* is the target to call in that directory. If *path* starts with a slash (/), it is treated as a file and its existence is tested with `test -e`; otherwise, it is assumed to be an executable, and `which -s` is used to determine if the program exists in the search path.

For example,

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

will check if the file or directory `/usr/local/etc/innd` exists, and build and install it from the `news/inn` subdirectory of the ports tree if it is not found. It will also see if an executable called `xmlcatmgr` is in the search path, and descend into the `textproc/xmlcatmgr` subdirectory of your ports tree to build and install it if it is not found.

##

In this case, `innd` is actually an executable; if an executable is in a place that is not expected to be in the search path, you should use the full pathname.

##

The official search `PATH` used on the ports build cluster is

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

The dependency is checked from within the `install` target. Also, the name of the dependency is put into the package so that pkg_add(1) will automatically install it if it is not on the user's system. The *`target`* part can be omitted if it is the same as `DEPENDS_TARGET` .

5.7.3. `BUILD_DEPENDS`

This variable specifies executables or files this port requires to build. Like `RUN_DEPENDS` , it is a list of *`path`*:*`dir`*[:target] tuples. For example,

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

will check for an executable called `unzip`, and descend into the `archivers/unzip` subdirectory of your ports tree to build and install it if it is not found.

##

“build” here means everything from extraction to compilation. The dependency is checked from within the `extract` target. The *`target`* part can be omitted if it is the same as `DEPENDS_TARGET`

5.7.4. FETCH_DEPENDS

This variable specifies executables or files this port requires to fetch. Like the previous two, it is a list of *path*:*dir*[:target] tuples. For example,

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

will check for an executable called `ncftp2`, and descend into the `net/ncftp2` subdirectory of your ports tree to build and install it if it is not found.

The dependency is checked from within the `fetch` target. The *`target`* part can be omitted if it is the same as `DEPENDS_TARGET` .

5.7.5. EXTRACT_DEPENDS

This variable specifies executables or files this port requires for extraction. Like the previous, it is a list of *path*:*dir*[:target] tuples. For example,

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

will check for an executable called `unzip`, and descend into the `archivers/unzip` subdirectory of your ports tree to build and install it if it is not found.

The dependency is checked from within the `extract` target. The *`target`* part can be omitted if it is the same as `DEPENDS_TARGET` .

##

Use this variable only if the extraction does not already work (the default assumes `gzip`) and cannot be made to work using `USE_ZIP` or `USE_BZIP2` described in # 5.7.7, “USE_*”.

5.7.6. PATCH_DEPENDS

This variable specifies executables or files this port requires to patch. Like the previous, it is a list of *path*:*dir*[:target] tuples. For example,

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

will descend into the `java/jfc` subdirectory of your ports tree to extract it.

The dependency is checked from within the `patch` target. The *`target`* part can be omitted if it is the same as `DEPENDS_TARGET` .

5.7.7. USE_*

A number of variables exist in order to encapsulate common dependencies that many ports have. Although their use is optional, they can help to reduce the verbosity of the port Makefile s. Each of them is styled as USE_*. The usage of these variables is restricted to the port Makefile s and ports/Mk/bsd.*.mk and is not designed to encapsulate user-settable options —— use WITH_* and WITHOUT_* for that purpose.

##

It is *always* incorrect to set any USE_* in /etc/make.conf . For instance, setting

```
USE_GCC=3.2
```

would adds a dependency on gcc32 for every port, including gcc32 itself!

5.1. The USE_* variables

Variable	Means
USE_BZIP2	The port's tarballs are compressed with bzip2.
USE_ZIP	The port's tarballs are compressed with zip.
USE_BISON	The port uses bison for building.
USE_CDRTOOLS	# port ## cdrecord# ### sysutils/cdrtools # sysutils/cdrtools-cjk #### cdrecord ###### ##### #
USE_GCC	The port requires a specific version of gcc to build. The exact version can be specified with value such as 3.2. The minimal required version can be specified as 3.2+. The gcc from the base system is used when it satisfies the requested version, otherwise an appropriate gcc is compiled from ports and the CC and CXX variables are adjusted.

Variables related to gmake and the configure script are described in # 6.3, “Building mechanisms”, while autoconf, automake and libtool are described in # 6.4, “Using GNU autotools”. Perl related variables are described in # 6.6, “Using perl”. X11 variables are listed in # 6.7, “Using X11”. # 6.8, “Using GNOME” deals with GNOME and # 6.9, “Using

KDE" with KDE related variables. # 6.10, "Using Java" documents Java variables, while # 6.11, "Web applications, Apache and PHP" contains information on Apache, PHP and PEAR modules. Python is discussed in # 6.12, "Using Python", while Ruby in # 6.14, "Using Ruby". # 6.15, "Using SDL" provides variables used for SDL applications and finally, # 6.18, "Using Xfce" contains information on Xfce.

5.7.8. Minimal version of a dependency

A minimal version of a dependency can be specified in any `*_DEPENDS` variable except `LIB_DEPENDS` using the following syntax:

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

The first field contains a dependent package name, which must match the entry in the package database, a comparison sign, and a package version. The dependency is satisfied if p5-Spiffy-0.26 or newer is installed on the machine.

5.7.9. Notes on dependencies

As mentioned above, the default target to call when a dependency is required is `DEPENDS_TARGET`. It defaults to `install`. This is a user variable; it is never defined in a port's `Makefile`. If your port needs a special way to handle a dependency, use the `:target` part of the `*_DEPENDS` variables instead of redefining `DEPENDS_TARGET`.

When you type `make clean`, its dependencies are automatically cleaned too. If you do not wish this to happen, define the variable `NOCLEANDEPENDS` in your environment. This may be particularly desirable if the port has something that takes a long time to rebuild in its dependency list, such as KDE, GNOME or Mozilla.

To depend on another port unconditionally, use the variable `${NONEXISTENT}` as the first field of `BUILD_DEPENDS` or `RUN_DEPENDS`. Use this only when you need to get the source of the other port. You can often save compilation time by specifying the target too. For instance

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

will always descend to the `jpeg` port and extract it.

5.7.10. Circular dependencies are fatal

##

Do not introduce any circular dependencies into the ports tree!

The ports building technology does not tolerate circular dependencies. If you introduce one, you will have someone, somewhere in the world, whose FreeBSD installation will break almost immediately, with many others quickly to follow. These can really be hard to detect; if in doubt, before you make that change, make sure you have done the following: `cd /usr/ports; make index` . That process can be quite slow on older machines, but you may be able to save a large number of people——including yourself—— a lot of grief in the process.

5.8. MASTERDIR

If your port needs to build slightly different versions of packages by having a variable (for instance, resolution, or paper size) take different values, create one subdirectory per package to make it easier for users to see what to do, but try to share as many files as possible between ports. Typically you only need a very short `Makefile` in all but one of the directories if you use variables cleverly. In the sole `Makefile` , you can use `MASTERDIR` to specify the directory where the rest of the files are. Also, use a variable as part of `PKGNAMESUFFIX` so the packages will have different names.

This will be best demonstrated by an example. This is part of `japanese/xdvi300/Makefile` ;

```
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 also has all the regular patches, package files, etc. If you type `make` there, it will take the default value for the resolution (300) and build the port normally.

As for other resolutions, this is the *entire* `xdvi118/Makefile` :

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

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

(`xdvi240/Makefile` and `xdvi400/Makefile` are similar). The `MASTERDIR` definition tells `bsd.port.mk` that the regular set of subdirectories like `FILESDIR` and `SCRIPTDIR` are to be found under `xdvi300`. The `RESOLUTION=118` line will override the `RESOLUTION=300` line in `xdvi300/Makefile` and the port will be built with resolution set to 118.

5.9. Manpages

The `MAN[1-9LN]` variables will automatically add any manpages to `pkg-plist` (this means you must *not* list manpages in the `pkg-plist` ——see generating PLIST for more). It also makes the install stage automatically compress or uncompress manpages depending on the setting of `NOMANCOMPRESS` in `/etc/make.conf`.

If your port tries to install multiple names for manpages using symlinks or hardlinks, you must use the `MLINKS` variable to identify these. The link installed by your port will be destroyed and recreated by `bsd.port.mk` to make sure it points to the correct file. Any manpages listed in MLINKS must not be listed in the `pkg-plist`.

To specify whether the manpages are compressed upon installation, use the `MANCOMPRESSED` variable. This variable can take three values, `yes`, `no` and `maybe`. `yes` means manpages are already installed compressed, `no` means they are not, and `maybe` means the software already respects the value of `NOMANCOMPRESS` so `bsd.port.mk` does not have to do anything special.

`MANCOMPRESSED` is automatically set to `yes` if `USE_IMAKE` is set and `NO_INSTALL_MANPAGES` is not set, and to `no` otherwise. You do not have to explicitly define it unless the default is not suitable for your port.

If your port anchors its man tree somewhere other than `MANPREFIX`, you can use the `MANPREFIX` to set it. Also, if only manpages in certain sections go in a non-standard place, such as some `perl` modules ports, you can set individual man paths using `MAN`*`sect`*`PREFIX` (where *`sect`* is one of `1-9`, `L` or `N`).

If your manpages go to language-specific subdirectories, set the name of the languages to `MANLANG`. The value of this variable defaults to `""` (i.e., English only).

Here is an example that puts it all together.

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

This states that six files are installed by this port;

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

Additionally `${MANPREFIX}/man/man8/alt-name.8.gz` may or may not be installed by your port. Regardless, a symlink will be made to join the foo(1) manpage and alt-name(8) manpage.

If only some manpages are translated, you can use several variables dynamically created from `MANLANG` content:

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

This translates into this list of files:

```
${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 files

If your package needs to install GNU info files, they should be listed in the `INFO` variable (without the trailing `.info`), one entry per document. These files are assumed to be installed to `PREFIX/INFO_PATH` . You can change `INFO_PATH` if your package uses a different location. However, this is not recommended. These entries contain just the path relative to `PREFIX/INFO_PATH` . For example, lang/gcc33 installs info files to `PREFIX/INFO_PATH/gcc33` , and `INFO` will be something like this:

```
INFO= gcc33/cpp gcc33/cppinternals gcc33/g77 ...
```

Appropriate installation/de-installation code will be automatically added to the temporary `pkg-plist` before package registration.

5.11. Makefile Options

Some large applications can be built in a number of configurations, adding functionality if one of a number of libraries or applications is available. Examples include choice of natural (human) language, GUI versus command-line, or type of database to support.

Since not all users want those libraries or applications, the ports system provides hooks that the port author can use to control which configuration should be built. Supporting these properly will make users happy, and effectively provide 2 or more ports for the price of one.

5.11.1. Knobs

5.11.1.1. `WITH_*` and `WITHOUT_*`

These variables are designed to be set by the system administrator. There are many that are standardized in `ports/KNOBS` file.

When creating a port, do not make knob names specific to a given application. For example in Avahi port, use `WITHOUT_MDNS` instead of `WITHOUT_AVAHI_MDNS` .

##

You should not assume that a `WITH_*` necessarily has a corresponding `WITHOUT_*` variable and vice versa. In general, the default is simply assumed.

##

Unless otherwise specified, these variables are only tested for being set or not set, rather than being set to some kind of variable such as `YES` or `NO`.

5.2. Common `WITH_*` and `WITHOUT_*` variables

Variable	Means
`WITHOUT_NLS`	If set, says that internationalization is not needed, which can save compile time. By default, internationalization is used.
`WITH_OPENSSL_BASE`	Use the version of OpenSSL in the base system.
`WITH_OPENSSL_PORT`	Installs the version of OpenSSL from security/openssl, even if the base is up to date.
`WITHOUT_X11`	If the port can be built both with and without X support, then it should normally

Variable	Means
	be built with X support. If this variable is defined, then the version that does not have X support should be built instead.

5.11.1.2. Knob naming

It is recommended that porters use like-named knobs, for the benefit of end-users and to help keep the number of knob names down. A list of popular knob names can be found in the KNOBS file.

Knob names should reflect what the knob is and does. When a port has a lib-prefix in the `PORTNAME` the lib-prefix should be dropped in knob naming.

5.11.2. `OPTIONS`

5.11.2.1. Background

The `OPTIONS` variable gives the user who installs the port a dialog with the available options and saves them to `/var/db/ports/portname/options` . Next time when the port has to be rebuild, the options are reused. Never again you will have to remember all the twenty `WITH_*` and `WITHOUT_*` options you used to build this port!

When the user runs `make config` (or runs `make build` for the first time), the framework will check for `/var/db/ports/portname/options` . If that file does not exist, it will use the values of `OPTIONS` to create a dialogbox where the options can be enabled or disabled. Then the `options` file is saved and the selected variables will be used when building the port.

If a new version of the port adds new `OPTIONS` , the dialog will be presented to the user, with the saved values of old `OPTIONS` prefilled.

Use `make showconfig` to see the saved configuration. Use `make rmconfig` to remove the saved configuration.

5.11.2.2. Syntax

The syntax for the `OPTIONS` variable is:

```
OPTIONS= OPTION "descriptive text" default ...
```

The value for default is either `ON` or `OFF`. Multiple repetitions of these three fields are allowed.

`OPTIONS` definition must appear before the inclusion of `bsd.port.pre.mk` . The `WITH_*` and `WITHOUT_*` variables can only be tested after the inclusion of `bsd.port.pre.mk` .

5.11.2.3. Example

5.8. Simple use of `OPTIONS`

```
OPTIONS=      FOO "Enable option foo" On \
              BAR "Support feature bar" Off

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

5.11.3. Feature auto-activation

When using a GNU configure script, keep an eye on which optional features are activated by auto-detection. Explicitly disable optional features you do not wish to be used by passing respective `--without-xxx` or `--disable-xxx` in `CONFIGURE_ARGS`.

5.9. Wrong handling of an option

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

In the example above, imagine a library libfoo is installed on the system. User does not want this application to use libfoo, so he toggled the option off in the `make config` dialog. But the application's configure script detects the library present in the system and includes its support in the resulting executable. Now when user decides to remove libfoo from the system, the ports system does not protest (no dependency on libfoo was recorded) but the application breaks.

5.10. Correct handling of an option

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

In the second example, the library libfoo is explicitly disabled. The configure script does not enable related features in the application, despite library's presence in the system.

5.12. Specifying the working directory

Each port is extracted in to a working directory, which must be writable. The ports system defaults to having the `DISTFILES` unpack in to a directory called `${DISTNAME}`. In other words, if you have set:

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

then the port's distribution files contain a top-level directory, `foo-1.0`, and the rest of the files are located under that directory.

There are a number of variables you can override if that is not the case.

5.12.1. `WRKSRC`

The variable lists the name of the directory that is created when the application's distfiles are extracted. If our previous example extracted into a directory called `foo` (and not `foo-1.0`) you would write:

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

or possibly

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

5.12.2. `NO_WRKSUBDIR`

If the port does not extract in to a subdirectory at all then you should set `NO_WRKSUBDIR` to indicate that.

```
NO_WRKSUBDIR= yes
```

5.13. CONFLICTS

If your package cannot coexist with other packages (because of file conflicts, runtime incompatibility, etc.), list the other package names in the CONFLICTS variable. You can use shell globs like * and ? here. Packages names should be enumerated the same way they appear in /var/db/pkg . Please make sure that CONFLICTS does not match this port's package itself, or else forcing its installation with FORCE_PKG_REGISTER will no longer work.

##

CONFLICTS automatically sets IGNORE, which is more fully documented in # 12.15, “Marking a port not installable with BROKEN, FORBIDDEN , or IGNORE”.

When removing one of several conflicting ports, it is advisable to retain the CONFLICTS entries in those other ports for a few months to cater for users who only update once in a while.

5.14. Installing files

5.14.1. INSTALL_* macros

Do use the macros provided in bsd.port.mk to ensure correct modes and ownership of files in your own *-install targets.

- INSTALL_PROGRAM is a command to install binary executables.
- INSTALL_SCRIPT is a command to install executable scripts.
- INSTALL_KLD is a command to install kernel loadable modules. Some architectures don't like it when the modules are stripped, therefor use this command instead of INSTALL_PROGRAM .
- INSTALL_DATA is a command to install sharable data.
- INSTALL_MAN is a command to install manpages and other documentation (it does not compress anything).

These are basically the install command with all the appropriate flags.

5.14.2. Stripping Binaries

Do not strip binaries manually unless you have to. All binaries should be stripped, but the `INSTALL_PROGRAM` macro will install and strip a binary at the same time (see the next section).

If you need to strip a file, but do not wish to use the `INSTALL_PROGRAM` macro, `${STRIP_CMD}` will strip your program. This is typically done within the `post-install` target. For example:

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

Use the file(1) command on the installed executable to check whether the binary is stripped or not. If it does not say `not stripped`, it is stripped. Additionally, strip(1) will not strip a previously stripped program; it will instead exit cleanly.

5.14.3. Installing a whole tree of files

Sometimes, there is a need to install a big number of files, preserving their hierarchical organization, ie. copying over a whole directory tree from `WRKSRC` to a target directory under `PREFIX`.

Two macros exists for this situation. The advantage of using these macros instead of `cp` is that they guarantee proper file ownership and permissions on target files. First macro, `COPYTREE_BIN`, will set all the installed files to be executable, thus being suitable for installing into `PREFIX/bin`. The second macro, `COPYTREE_SHARE`, does not set executable permissions on files, and is therefore suitable for installing files under `PREFIX/share` target.

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

This example will install the contents of `examples` directory in the vendor distfile to the proper examples location of your port.

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

And this example will install the data of summer months to the `summer` subdirectory of a `DATADIR`.

Additional `find` arguments can be passed via the third argument to the `COPYTREE_*` macros. For example, to install all files from the first example except Makefiles, one can use the following command.

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

Note that these macros does not add the installed files to `pkg-plist`. You still need to list them.

5.14.4. Install additional documentation

If your software has some documentation other than the standard man and info pages that you think is useful for the user, install it under `PREFIX/share/doc`. This can be done, like the previous item, in the `post-install` target.

Create a new directory for your port. The directory name should reflect what the port is. This usually means `PORTNAME`. However, if you think the user might want different versions of the port to be installed at the same time, you can use the whole `PKGNAME`.

Make the installation dependent on the variable `NOPORTDOCS` so that users can disable it in `/etc/make.conf`, like this:

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

Here are some handy variables and how they are expanded by default when used in the `Makefile`:

- `DATADIR` gets expanded to `PREFIX/share/PORTNAME`.
- `DATADIR_REL` gets expanded to `share/PORTNAME`.
- `DOCSDIR` gets expanded to `PREFIX/share/doc/PORTNAME`.
- `DOCSDIR_REL` gets expanded to `share/doc/PORTNAME`.
- `EXAMPLESDIR` gets expanded to `PREFIX/share/examples/PORTNAME`.
- `EXAMPLESDIR_REL` gets expanded to `share/examples/PORTNAME`.

##

`NOPORTDOCS` only controls additional documentation installed in `DOCSDIR`. It does not apply to standard man pages and info pages.

> Things installed in DATADIR and EXAMPLESDIR are controlled by NOPORTDATA and NOPORTEXAMPLES, respectively.

These variables are exported to PLIST_SUB. Their values will appear there as pathnames relative to PREFIX if possible. That is, share/doc/PORTNAME will be substituted for %%DOCSDIR%% in the packing list by default, and so on. (See more on pkg-plist substitution here.)

All conditionally installed documentation files and directories should be included in pkg-plist with the %%PORTDOCS%% prefix, for example:

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

As an alternative to enumerating the documentation files in pkg-plist, a port can set the variable PORTDOCS to a list of file names and shell glob patterns to add to the final packing list. The names will be relative to DOCSDIR. Therefore, a port that utilizes PORTDOCS and uses a non-default location for its documentation should set DOCSDIR accordingly. If a directory is listed in PORTDOCS or matched by a glob pattern from this variable, the entire subtree of contained files and directories will be registered in the final packing list. If NOPORTDOCS is defined then files and directories listed in PORTDOCS would not be installed and neither would be added to port packing list. Installing the documentation at PORTDOCS as shown above remains up to the port itself. A typical example of utilizing PORTDOCS looks as follows:

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

##

The equivalents of PORTDOCS for files installed under DATADIR and EXAMPLESDIR are PORTDATA and PORTEXAMPLES, respectively.

You can also use the pkg-message file to display messages upon installation. See the section on using pkg-message for details. The pkg-message file does not need to be added to pkg-plist.

5.14.5. Subdirectories under PREFIX

Try to let the port put things in the right subdirectories of PREFIX. Some ports lump everything and put it in the subdirectory with the port's name, which is incorrect. Also, many ports put everything except binaries, header files and manual pages in a subdirectory of lib, which does not work well with the BSD paradigm. Many of

the files should be moved to one of the following: `etc` (setup/configuration files), `libexec` (executables started internally), `sbin` (executables for superusers/managers), `info` (documentation for info browser) or `share` (architecture independent files). See hier(7) for details; the rules governing `/usr` pretty much apply to `/usr/local` too. The exception are ports dealing with USENET "news". They may use `PREFIX/news` as a destination for their files.

6. Special considerations

There are some more things you have to take into account when you create a port. This section explains the most common of those.

6.1. Shared Libraries

If your port installs one or more shared libraries, define a `USE_LDCONFIG` make variable, which will instruct a `bsd.port.mk` to run `${LDCONFIG} -m` on the directory where the new library is installed (usually `PREFIX/lib`) during `post-install` target to register it into the shared library cache. This variable, when defined, will also facilitate addition of an appropriate `@exec /sbin/ldconfig -m` and `@unexec /sbin/ldconfig -R` pair into your `pkg-plist` file, so that a user who installed the package can start using the shared library immediately and de-installation will not cause the system to still believe the library is there.

```
USE_LDCONFIG= yes
```

If you need, you can override the default directory by setting the `USE_LDCONFIG` value to a list of directories into which shared libraries are to be installed. For example if your port installs shared libraries into `PREFIX/lib/foo` and `PREFIX/lib/bar` directories you could use the following in your `Makefile`:

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

Please double-check, often this is not necessary at all or can be avoided through `-rpath` or setting `LD_RUN_PATH` during linking (see lang/moscow_ml for an example), or through a shell-wrapper which sets `LD_LIBRARY_PATH` before invoking the binary, like www/mozilla does.

When installing 32-bit libraries on 64-bit system, use `USE_LDCONFIG32` instead.

Try to keep shared library version numbers in the `libfoo.so.0` format. Our runtime linker only cares for the major (first) number.

When the major library version number increments in the update to the new port version, all other ports that link to the affected library should have their `PORTREVISION` incremented, to force recompilation with the new library version.

6.2. Ports with distribution restrictions

Licenses vary, and some of them place restrictions on how the application can be packaged, whether it can be sold for profit, and so on.

##

It is your responsibility as a porter to read the licensing terms of the software and make sure that the FreeBSD project will not be held accountable for violating them by redistributing the source or compiled binaries either via FTP/HTTP or CD-ROM. If in doubt, please contact the FreeBSD ports ####.

In situations like this, the variables described in the following sections can be set.

6.2.1. NO_PACKAGE

This variable indicates that we may not generate a binary package of the application. For instance, the license may disallow binary redistribution, or it may prohibit distribution of packages created from patched sources.

However, the port's DISTFILES may be freely mirrored on FTP/HTTP. They may also be distributed on a CD-ROM (or similar media) unless NO_CDROM is set as well.

NO_PACKAGE should also be used if the binary package is not generally useful, and the application should always be compiled from the source code. For example, if the application has configuration information that is site specific hard coded in to it at compile time, set NO_PACKAGE .

NO_PACKAGE should be set to a string describing the reason why the package should not be generated.

6.2.2. NO_CDROM

This variable alone indicates that, although we are allowed to generate binary packages, we may put neither those packages nor the port's DISTFILES onto a CD-ROM (or similar media) for resale. However, the binary packages and the port's DISTFILES will still be available via FTP/HTTP.

If this variable is set along with NO_PACKAGE , then only the port's DISTFILES will be available, and only via FTP/HTTP.

NO_CDROM should be set to a string describing the reason why the port cannot be redistributed on CD-ROM. For instance, this should be used if the port's license is for “non-commercial” use only.

6.2.3. NOFETCHFILES

Files defined in the NOFETCHFILES variable are not fetchable from any of the MASTER_SITES . An example of such a file is when the file is supplied on CD-ROM by the vendor.

Tools which check for the availability of these files on the MASTER_SITES should ignore these files and not report about them.

6.2.4. RESTRICTED

Set this variable alone if the application's license permits neither mirroring the application's DISTFILES nor distributing the binary package in any way.

NO_CDROM or NO_PACKAGE should not be set along with RESTRICTED since the latter variable implies the former ones.

RESTRICTED should be set to a string describing the reason why the port cannot be redistributed. Typically, this indicates that the port contains proprietary software and that the user will need to manually download the DISTFILES , possibly after registering for the software or agreeing to accept the terms of an EULA.

6.2.5. RESTRICTED_FILES

When RESTRICTED or NO_CDROM is set, this variable defaults to ${DISTFILES} ${PATCHFILES} , otherwise it is empty. If only some of the distribution files are restricted, then set this variable to list them.

Note that the port committer should add an entry to /usr/ports/LEGAL for every listed distribution file, describing exactly what the restriction entails.

6.3. Building mechanisms

6.3.1. make, gmake, and imake

If your port uses GNU make, set USE_GMAKE=yes .

6.1. Variables for ports related to gmake

Variable	Means
USE_GMAKE	The port requires gmake to build.
GMAKE	The full path for gmake if it is not in the PATH.

If your port is an X application that creates Makefile files from Imakefile files using imake, then set USE_IMAKE=yes . This will cause the configure stage to automatically do an

xmkmf -a. If the -a flag is a problem for your port, set XMKMF=xmkmf . If the port uses imake but does not understand the install.man target, NO_INSTALL_MANPAGES=yes should be set.

If your port's source Makefile has something else than all as the main build target, set ALL_TARGET accordingly. Same goes for install and INSTALL_TARGET .

6.3.2. configure script

If your port uses the configure script to generate Makefile files from Makefile.in files, set GNU_CONFIGURE=yes . If you want to give extra arguments to the configure script (the default argument is --prefix=${PREFIX} --infodir=${PREFIX}/${INFO_PATH} --mandir=${MANPREFIX}/man --build=${CONFIGURE_TARGET}), set those extra arguments in CONFIGURE_ARGS . Extra environment variables can be passed using CONFIGURE_ENV variable.

6.2. Variables for ports that use configure

Variable	Means
GNU_CONFIGURE	The port uses configure script to prepare build.
HAS_CONFIGURE	Same as GNU_CONFIGURE , except default configure target is not added to CONFIGURE_ARGS .
CONFIGURE_ARGS	Additional arguments passed to configure script.
CONFIGURE_ENV	Additional environment variables to be set for configure script run.
CONFIGURE_TARGET	Override default configure target. Default value is ${MACHINE_ARCH}-portbld-freebsd${OSREL} .

6.3.3. Using scons

If your port uses SCons, define USE_SCONS=yes .

6.3. Variables for ports that use scons

Variable	Means
SCONS_ARGS	Port specific SCons flags passed to the SCons environment.
SCONS_BUILDENV	Variables to be set in system environment.
SCONS_ENV	Variables to be set in SCons environment.

Variable	Means
SCONS_TARGET	Last argument passed to SCons, similar to MAKE_TARGET .

6.4. Using GNU autotools

6.4.1. Introduction

The various GNU autotools provide an abstraction mechanism for building a piece of software over a wide variety of operating systems and machine architectures. Within the Ports Collection, an individual port can make use of these tools via a simple construct:

```
USE_AUTOTOOLS= tool:version[:operation] ...
```

At the time of writing, *tool* can be one of libtool , libltdl , autoconf , autoheader , automake or aclocal .

version specifies the particular tool revision to be used (see devel/{automake,autoconf,libtool}[0-9]+ for valid versions).

operation is an optional extension to modify how the tool is used.

Multiple tools can be specified at once, either by including them all on a single line, or using the += Makefile construct.

Finally, there is the special tool, called autotools , which is a convenience function to bring in all available versions of the autotools to allow for cross-development work. This can also be accomplished by installing the devel/autotools port.

6.4.2. libtool

Shared libraries using the GNU building framework usually use libtool to adjust the compilation and installation of shared libraries to match the specifics of the underlying operating system. The usual practice is to use copy of libtool bundled with the application. In case you need to use external libtool , you can use the version provided by The Ports Collection:

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

With no additional operations, libtool:version tells the building framework to patch the configure script with the system-installed copy of libtool . The GNU_CONFIGURE is implied. Further, a number of make and shell variables will be assigned for onward use by the port. See bsd.autotools.mk for details.

With the :env operation, only the environment will be set up.

Finally, LIBTOOLFLAGS and LIBTOOLFILES can be optionally set to override the most likely arguments to, and files patched by, libtool. Most ports are unlikely to need this. See bsd.autotools.mk for further details.

6.4.3. libltdl

Some ports make use of the libltdl library package, which is part of the libtool suite. Use of this library does not automatically necessitate the use of libtool itself, so a separate construct is provided.

```
USE_AUTOTOOLS= libltdl:version
```

Currently, all this does is to bring in a LIB_DEPENDS on the appropriate libltdl port, and is provided as a convenience function to help eliminate any dependencies on the autotools ports outside of the USE_AUTOTOOLS framework. There are no optional operations for this tool.

6.4.4. autoconf and autoheader

Some ports do not contain a configure script, but do contain an autoconf template in the configure.ac file. You can use the following assignments to let autoconf create the configure script, and also have autoheader create template headers for use by the configure script.

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

and

```
USE_AUTOTOOLS= autoheader:version
```

which also implies the use of autoconf:version.

Similarly to libtool, the inclusion of the optional :env operation simply sets up the environment for further use. Without it, patching and reconfiguration of the port is carried out.

The additional optional variables AUTOCONF_ARGS and AUTOHEADER_ARGS can be overridden by the port Makefile if specifically requested. As with the libtool equivalents, most ports are unlikely to need this.

6.4.5. automake and aclocal

Some packages only contain Makefile.am files. These have to be converted into Makefile.in files using automake, and the further processed by configure to generate an actual Makefile.

Similarly, packages occasionally do not ship with included aclocal.m4 files, again required to build the software. This can be achieved with aclocal, which scans configure.ac or configure.in.

`aclocal` has a similar relationship to `automake` as `autoheader` does to `autoconf`, described in the previous section. `aclocal` implies the use of `automake`, thus we have:

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

and

```
USE_AUTOTOOLS= aclocal:version
```

which also implies the use of `automake:version`.

Similarly to `libtool` and `autoconf`, the inclusion of the optional `:env` operation simply sets up the environment for further use. Without it, reconfiguration of the port is carried out.

As with `autoconf` and `autoheader`, both `automake` and `aclocal` have optional argument variables, `AUTOMAKE_ARGS` and `ACLOCAL_ARGS` respectively, which may be overriden by the port `Makefile` if required.

6.5. Using GNU `gettext`

6.5.1. Basic usage

If your port requires `gettext`, just set `USE_GETTEXT` to `yes`, and your port will grow the dependency on devel/gettext. The value of `USE_GETTEXT` can also specify the required version of the `libintl` library, the basic part of `gettext`, but using this feature is *strongly discouraged*: Your port should work with just the current version of devel/gettext.

A rather common case is a port using `gettext` and `configure`. Generally, GNU `configure` should be able to locate `gettext` automatically. If it ever fails to, hints at the location of `gettext` can be passed in `CPPFLAGS` and `LDFLAGS` as follows:

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

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

Of course, the code can be more compact if there are no more flags to pass to `configure`:

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

6.5.2. Optional usage

Some software products allow for disabling NLS, e.g., through passing `--disable-nls` to `configure`. In that case, your port should use `gettext` conditionally, depending on the status of `WITHOUT_NLS`. For ports of low to medium complexity, you can rely on the following idiom:

```
GNU_CONFIGURE=          yes

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

The next item on your to-do list is to arrange so that the message catalog files are included in the packing list conditionally. The `Makefile` part of this task is already provided by the idiom. It is explained in the section on advanced `pkg-plist` practices. In a nutshell, each occurrence of `%%NLS%%` in `pkg-plist` will be replaced by "`@comment `" if NLS is disabled, or by a null string if NLS is enabled. Consequently, the lines prefixed by `%%NLS%%` will become mere comments in the final packing list if NLS is off; otherwise the prefix will be just left out. All you need to do now is insert `%%NLS%%` before each path to a message catalog file in `pkg-plist`. For example:

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

In high complexity cases, you may need to use more advanced techniques than the recipe given here, such as dynamic packing list generation.

6.5.3. Handling message catalog directories

There is a point to note about installing message catalog files. The target directories for them, which reside under `LOCALBASE/share/locale`, should rarely be created and removed by your port. The most popular languages have their respective directories listed in `/etc/mtree/BSD.local.dist`; that is, they are a part of the base system. The directories for many other languages are governed by the devel/gettext port. You may want to consult its `pkg-plist` and see whether your port is going to install a message catalog file for a unique language.

6.6. Using `perl`

If `MASTER_SITES` is set to `MASTER_SITE_PERL_CPAN`, then preferred value of `MASTER_SITE_SUBDIR` is top-level hierarchy name. For example, the recommend value

for `p5-Module-Name` is `Module`. The top-level hierarchy can be examined at cpan.org. This keeps the port working when the author of the module changes.

The exception to this rule is when the relevant directory does not exist or the distfile does not exist in the directory. In such case, using author's id as `MASTER_SITE_SUBDIR` is allowed.

All of the tunable knobs below accept both `YES` and a version string, like `5.8.0+`. Using `YES` means that the port can be used with all of the supported Perl versions. If a port only works with specific versions of Perl, it can be indicated with a version string, specifying a minimal version (e.g. `5.7.3+`), a maximal version (e.g. `5.8.0-`) or an exact version (e.g. `5.8.3`).

6.4. Variables for ports that use `perl`

Variable	Means
`USE_PERL5`	Says that the port uses `perl 5` to build and run.
`USE_PERL5_BUILD`	Says that the port uses `perl 5` to build.
`USE_PERL5_RUN`	Says that the port uses `perl 5` to run.
`PERL`	The full path of `perl 5`, either in the system or installed from a port, but without the version number. Use this if you need to replace "`#!`"lines in scripts.
`PERL_CONFIGURE`	Configure using Perl's MakeMaker. It implies `USE_PERL5` .
`PERL_MODBUILD`	Configure, build and install using Module::Build. It implies `PERL_CONFIGURE` .
Read only variables	Means
`PERL_VERSION`	The full version of `perl` installed (e.g., `5.00503`).
`PERL_VER`	The short version of `perl` installed (e.g., `5.005`).
`PERL_LEVEL`	The installed `perl` version as an integer of the form `MNNNPP` (e.g., `500503`).
`PERL_ARCH`	Where `perl` stores architecture dependent libraries. Defaults to `${ARCH}-freebsd` .
`PERL_PORT`	Name of the `perl` port that is installed (e.g., `perl5`).

Read only variables	Means
SITE_PERL	Directory name where site specific perl packages go. This value is added to PLIST_SUB.

##

Perl ### port############# pkg-descr ## WWW ##### cpan.org # #### URL ### http://search.cpan.org/dist/Module-Name/ (## ### / ####)#

6.7. Using X11

6.7.1. X.Org components

The X11 implementation available in The Ports Collection is X.Org. If your application depends on X components, set USE_XORG to the list of required components. Available components, at the time of writing, are:

```
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 .
```

Always up-to-date list can be found in /usr/ports/Mk/bsd.xorg.mk .

The Mesa Project is an effort to provide free OpenGL implementation. You can specify a dependency on various components of this project with USE_GL variable. Valid options are: glut, glu, glw, gl and linux. For backwards compatibility, the value of yes maps to glu.

6.1. USE_XORG example

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

```
USE_GL=     glu
```

Many ports define `USE_XLIB` , which makes the port depend on all the 50 or so libraries. This variable exists for backwards compatibility, as it predates modular X.Org, and should not be used on new ports.

6.5. Variables for ports that use X

`USE_XLIB`	The port uses the X libraries. Deprecated - use a list of X.Org components in `USE_XORG` variable instead.
`USE_IMAKE`	`### imake # port#`
`USE_X_PREFIX`	Deprecated. Today it is equivalent to `USE_XLIB` and can be replaced by it freely.
`XMKMF`	Set to the path of `xmkmf` if not in the `PATH`. Defaults to `xmkmf -a`.

6.6. Variables for depending on individual parts of X11

`X_IMAKE_PORT`	Port providing `imake` and several other utilities used to build X11.
`X_LIBRARIES_PORT`	Port providing X11 libraries.
`X_CLIENTS_PORT`	Port providing X clients.
`X_SERVER_PORT`	Port providing X server.
`X_FONTSERVER_PORT`	Port providing font server.
`X_PRINTSERVER_PORT`	Port providing print server.
`X_VFBSERVER_PORT`	Port providing virtual framebuffer server.
`X_NESTSERVER_PORT`	Port providing a nested X server.
`X_FONTS_ENCODINGS_PORT`	Port providing encodings for fonts.
`X_FONTS_MISC_PORT`	Port providing miscellaneous bitmap fonts.
`X_FONTS_100DPI_PORT`	Port providing 100dpi bitmap fonts.
`X_FONTS_75DPI_PORT`	Port providing 75dpi bitmap fonts.
`X_FONTS_CYRILLIC_PORT`	Port providing cyrillic bitmap fonts.
`X_FONTS_TTF_PORT`	Port providing TrueType® fonts.
`X_FONTS_TYPE1_PORT`	Port providing Type1 fonts.
`X_MANUALS_PORT`	Port providing developer oriented manual pages

6.2. Using some X11 related variables in port

```
# Use X11 libraries and depend on
# font server as well as cyrillic fonts.
RUN_DEPENDS=   ${LOCALBASE}/bin/xfs:${X_FONTSERVER_PORT} \
               ${LOCALBASE}/lib/X11/fonts/cyrillic/crox1c.pcf.↺
gz:${X_FONTS_CYRILLIC_PORT}

USE_XORG=      yes
```

6.7.2. Ports that require Motif

If your port requires a Motif library, define `USE_MOTIF` in the `Makefile`. Default Motif implementation is x11-toolkits/open-motif. Users can choose x11-toolkits/lesstif instead by setting `WANT_LESSTIF` variable.

The `MOTIFLIB` variable will be set by `bsd.port.mk` to reference the appropriate Motif library. Please patch the source of your port to use `${MOTIFLIB}` wherever the Motif library is referenced in the original `Makefile` or `Imakefile`.

There are two common cases:

- If the port refers to the Motif library as `-lXm` in its `Makefile` or `Imakefile`, simply substitute `${MOTIFLIB}` for it.
- If the port uses `XmClientLibs` in its `Imakefile`, change it to `${MOTIFLIB} ${XTOOLLIB} ${XLIB}`.

Note that `MOTIFLIB` (usually) expands to `-L/usr/X11R6/lib -lXm` or `/usr/X11R6/lib/libXm.a`, so there is no need to add `-L` or `-l` in front.

6.7.3. X11 fonts

If your port installs fonts for the X Window System, put them in `LOCALBASE/lib/X11/fonts/local`.

6.7.4. Getting fake `DISPLAY` using Xvfb

Some applications require a working X11 display for compilation to succeed. This pose a problem for machines which operates headless. When the following variable is used, the build infrastructure will start the virtual framebuffer X server. The working `DISPLAY` is then passed to the build.

```
USE_DISPLAY=  yes
```

6.7.5. Desktop entries

DESKTOP_ENTRIES ######## port # X #### (Desktop Entries#### Freedesktop standard)# ############## GNOME # KDE ########### .desktop ## ############## pkg-plist

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

########## Freedesktop ### # StartupNotify ############ startup noficication ####

###

```
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. Using GNOME

The FreeBSD/GNOME project uses its own set of variables to define which GNOME components a particular port uses. A comprehensive list of these variables exists within the FreeBSD/GNOME project's homepage.

6.9. Using KDE

6.9.1. Variable definitions

6.7. Variables for ports that use KDE

USE_KDELIBS_VER	The port uses KDE libraries. It specifies the major version of KDE to use and implies USE_QT_VER of the appropriate version. The only possible value is 3.
USE_KDEBASE_VER	The port uses KDE base. It specifies the major version of KDE to use and implies USE_QT_VER of the appropriate version. The only possible value is 3.

6.9.2. Ports that require Qt

6.8. Variables for ports that use Qt

USE_QT_VER	The port uses the Qt toolkit. Possible values are 3 and 4; each specify the major version

	of Qt to use. Appropriate parameters are passed to `configure` script and `make`.
`QT_PREFIX`	Set to the path where Qt installed to (read-only variable).
`MOC`	Set to the path of `moc` (read-only variable). Default set according to `USE_QT_VER` value.
`QTCPPFLAGS`	Additional compiler flags passed via `CONFIGURE_ENV` for Qt toolkit. Default set according to `USE_QT_VER` .
`QTCFGLIBS`	Additional libraries for linking passed via `CONFIGURE_ENV` for Qt toolkit. Default set according to `USE_QT_VER` .
`QTNONSTANDARD`	Suppress modification of `CONFIGURE_ENV` , `CONFIGURE_ARGS` , and `MAKE_ENV` .

6.9. Additional variables for ports that use Qt 4.x

`QT_COMPONENTS`	Specify tool and library dependencies for Qt4. See below for details.
`UIC`	Set to the path of `uic` (read-only variable). Default set according to `USE_QT_VER` value.
`QMAKE`	Set to the path of `qmake` (read-only variable). Default set according to `USE_QT_VER` value.
`QMAKESPEC`	Set to the path of configuration file for `qmake` (read-only variable). Default set according to `USE_QT_VER` value.

When `USE_QT_VER` is set, some useful settings are passed to `configure` script:

```
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}"
```

If `USE_QT_VER` is set to `4`, the following settings are also deployed:

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

6.9.3. Component selection (Qt 4.x only)

When `USE_QT_VER` is set to 4, individual Qt4 tool and library dependencies can be specified in the `QT_COMPONENTS` variable. Every component can be suffixed by either `_build` or `_run`, the suffix indicating whether the component should be depended on at buildtime or runtime, respectively. If unsuffixed, the component will be depended on at both build- and runtime. Usually, library components should be specified unsuffixed, tool components should be specified with the `_build` suffix and plugin components should be specified with the `_run` suffix. The most commonly used components are listed below (all available components are listed in `_QT_COMPONENTS_ALL` in `/usr/ports/Mk/bsd.qt.mk`):

6.10. Available Qt4 library components

Name	Description
`corelib`	core library (can be omitted unless the port uses nothing but `corelib`)
`gui`	graphical user interface library
`network`	network library
`opengl`	OpenGL library
`qt3support`	Qt3 compatibility library
`qtestlib`	unit testing library
`script`	script library
`sql`	SQL library
`xml`	XML library

You can determine which libraries the application depends on, by running `ldd` on the main executable after a successful compilation.

6.11. Available Qt4 tool components

Name	Description
`moc`	meta object compiler (needed for almost every Qt application at buildtime)
`qmake`	Makefile generator / build utility
`rcc`	resource compiler (need if the application comes with `*.rc` or `*.qrc` files)
`uic`	user interface compiler (needed if the application comes with `*.ui` files created by Qt Designer - in practice, every Qt application with a GUI)

6.12. Available Qt4 plugin components

Name	Description
`iconengines`	SVG icon engine plugin (if the application ships SVG icons)
`imageformats`	imageformat plugins for GIF, JPEG, MNG and SVG (if the application ships image files)

6.3. Selecting Qt4 components

In this example, the ported application uses the Qt4 graphical user interface library, the Qt4 core library, all of the Qt4 code generation tools and Qt4's Makefile generator. Since the gui library implies a dependency on the core library, corelib does not need to be specified. The Qt4 code generation tools moc, uic and rcc, as well as the Makefile generator qmake are only needed at buildtime, thus they are specified with the `_build` suffix:

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

6.9.4. Additional considerations

If the application does not provide a `configure` file but a `.pro` file, you can use the following:

```
HAS_CONFIGURE=  yes

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

Note the similarity to the `qmake` line from the provided `BUILD.sh` script. Passing `CONFIGURE_ENV` ensures `qmake` will see the `QMAKESPEC` variable, without which it cannot work. `qmake` generates standard Makefiles, so it is not necessary to write our own `build` target.

Qt applications often are written to be cross-platform and often X11/Unix isn't the platform they are developed on, which in turn often leads to certain loose ends, like:

- *Missing additional includepaths.* Many applications come with system tray icon support, but neglect to look for includes and/or libraries in the X11 directories. You can tell `qmake` to add directories to the include and library searchpaths via the commandline, for example:

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

- *Bogus installation paths.* Sometimes data such as icons or .desktop files are by default installed into directories which aren't scanned by XDG-compatible applications. editors/texmaker is an example for this - look at `patch-texmaker.pro` in the `files` directory of that port for a template on how to remedy this directly in the Qmake project file.

6.10. Using Java

6.10.1. Variable definitions

If your port needs a Java™ Development Kit (JDK) to either build, run or even extract the distfile, then it should define `USE_JAVA`.

There are several JDKs in the ports collection, from various vendors, and in several versions. If your port must use one of these versions, you can define which one. The most current version is java/jdk15.

6.13. Variables that may be set by ports that use Java

Variable	Means
`USE_JAVA`	Should be defined for the remaining variables to have any effect.
`JAVA_VERSION`	List of space-separated suitable Java versions for the port. An optional `"+"` allows you to specify a range of versions (allowed values: `1.1[+] 1.2[+] 1.3[+] 1.4[+]`).
`JAVA_OS`	List of space-separated suitable JDK port operating systems for the port (allowed values: `native linux`).
`JAVA_VENDOR`	List of space-separated suitable JDK port vendors for the port (allowed values: `freebsd bsdjava sun ibm blackdown`).
`JAVA_BUILD`	When set, it means that the selected JDK port should be added to the build dependencies of the port.
`JAVA_RUN`	When set, it means that the selected JDK port should be added to the run dependencies of the port.

Variable	Means
JAVA_EXTRACT	When set, it means that the selected JDK port should be added to the extract dependencies of the port.
USE_JIKES	Whether the port should or should not use the jikes bytecode compiler to build. When no value is set for this variable, the port will use jikes to build if available. You may also explicitly forbid or enforce the use of jikes (by setting 'no' or 'yes'). In the later case, devel/jikes will be added to build dependencies of the port. In any case that jikes is actually used in place of javac, then the HAVE_JIKES variable is defined by bsd.java.mk .

Below is the list of all settings a port will receive after setting USE_JAVA :

6.14. Variables provided to ports that use Java

Variable	Value
JAVA_PORT	The name of the JDK port (e.g. 'java/jdk14').
JAVA_PORT_VERSION	The full version of the JDK port (e.g. '1.4.2'). If you only need the first two digits of this version number, use ${JAVA_PORT_VERSION:C/^([0-9])\.([0-9])(.*)$/\1.\2/} .
JAVA_PORT_OS	The operating system used by the JDK port (e.g. 'linux').
JAVA_PORT_VENDOR	The vendor of the JDK port (e.g. 'sun').
JAVA_PORT_OS_DESCRIPTION	Description of the operating system used by the JDK port (e.g. 'Linux').
JAVA_PORT_VENDOR_DESCRIPTION	Description of the vendor of the JDK port (e.g. 'FreeBSD Foundation').
JAVA_HOME	Path to the installation directory of the JDK (e.g. '/usr/local/jdk1.3.1').
JAVAC	Path to the Java compiler to use (e.g. '/usr/local/jdk1.1.8/bin/javac' or '/usr/local/bin/jikes').

Variable	Value
JAR	Path to the jar tool to use (e.g. '/usr/local/jdk1.2.2/bin/jar' or '/usr/local/bin/fastjar').
APPLETVIEWER	Path to the appletviewer utility (e.g. '/usr/local/linux-jdk1.2.2/bin/appletviewer').
JAVA	Path to the java executable. Use this for executing Java programs (e.g. '/usr/local/jdk1.3.1/bin/java').
JAVADOC	Path to the javadoc utility program.
JAVAH	Path to the javah program.
JAVAP	Path to the javap program.
JAVA_KEYTOOL	Path to the keytool utility program. This variable is available only if the JDK is Java 1.2 or higher.
JAVA_N2A	Path to the native2ascii tool.
JAVA_POLICYTOOL	Path to the policytool program. This variable is available only if the JDK is Java 1.2 or higher.
JAVA_SERIALVER	Path to the serialver utility program.
RMIC	Path to the RMI stub/skeleton generator, rmic.
RMIREGISTRY	Path to the RMI registry program, rmiregistry .
RMID	Path to the RMI daemon program rmid. This variable is only available if the JDK is Java 1.2 or higher.
JAVA_CLASSES	Path to the archive that contains the JDK class files. On JDK 1.2 or later, this is ${JAVA_HOME}/jre/lib/rt.jar . Earlier JDKs used ${JAVA_HOME}/lib/classes.zip .
HAVE_JIKES	Defined whenever jikes is used by the port (see USE_JIKES above).

You may use the java-debug make target to get information for debugging your port. It will display the value of many of the forecited variables.

Additionally, the following constants are defined so all Java ports may be installed in a consistent way:

6.15. Constants defined for ports that use Java

Constant	Value
JAVASHAREDIR	The base directory for everything related to Java. Default: ${PREFIX}/share/java .
JAVAJARDIR	The directory where JAR files should be installed. Default: ${JAVASHAREDIR}/classes .
JAVALIBDIR	The directory where JAR files installed by other ports are located. Default: ${LOCALBASE}/share/java/classes .

The related entries are defined in both PLIST_SUB (documented in # 7.1, "Changing pkg-plist based on make variables") and SUB_LIST .

6.10.2. Building with Ant

When the port is to be built using Apache Ant, it has to define USE_ANT . Ant is thus considered to be the sub-make command. When no do-build target is defined by the port, a default one will be set that simply runs Ant according to MAKE_ENV , MAKE_ARGS and ALL_TARGETS . This is similar to the USE_GMAKE mechanism, which is documented in # 6.3, "Building mechanisms".

If jikes is used in place of javac (see USE_JIKES in # 6.10.1, "Variable definitions"), then Ant will automatically use it to build the port.

6.10.3. Best practices

When porting a Java library, your port should install the JAR file(s) in ${JAVAJARDIR} , and everything else under ${JAVASHAREDIR}/${PORTNAME} (except for the documentation, see below). In order to reduce the packing file size, you may reference the JAR file(s) directly in the Makefile . Just use the following statement (where myport.jar is the name of the JAR file installed as part of the port):

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

When porting a Java application, the port usually installs everything under a single directory (including its JAR dependencies). The use of ${JAVASHAREDIR}/${PORTNAME} is strongly encouraged in this regard. It is up the porter to decide whether the port should install the additional JAR dependencies under this directory or directly use the already installed ones (from ${JAVAJARDIR}).

Regardless of the type of your port (library or application), the additional documentation should be installed in the same location as for any other port. The JavaDoc tool is known to produce a different set of files depending on the version of the JDK that is used. For ports that do not enforce the use of a particular JDK, it is therefore a complex task to specify the packing list (`pkg-plist`). This is one reason why porters are strongly encouraged to use the `PORTDOCS` macro. Moreover, even if you can predict the set of files that will be generated by `javadoc`, the size of the resulting `pkg-plist` advocates for the use of `PORTDOCS`.

The default value for `DATADIR` is `${PREFIX}/share/${PORTNAME}`. It is a good idea to override `DATADIR` to `${JAVASHAREDIR}/${PORTNAME}` for Java ports. Indeed, `DATADIR` is automatically added to `PLIST_SUB` (documented in # 7.1, "Changing `pkg-plist` based on make variables") so you may use `%%DATADIR%%` directly in `pkg-plist`.

As for the choice of building Java ports from source or directly installing them from a binary distribution, there is no defined policy at the time of writing. However, people from the FreeBSD Java Project encourage porters to have their ports built from source whenever it is a trivial task.

All the features that have been presented in this section are implemented in `bsd.java.mk`. If you ever think that your port needs more sophisticated Java support, please first have a look at the bsd.java.mk CVS log as it usually takes some time to document the latest features. Then, if you think the support you are lacking would be beneficial to many other Java ports, feel free to discuss it on the FreeBSD Java Language ####.

Although there is a `java` category for PRs, it refers to the JDK porting effort from the FreeBSD Java project. Therefore, you should submit your Java port in the `ports` category as for any other port, unless the issue you are trying to resolve is related to either a JDK implementation or `bsd.java.mk`.

Similarly, there is a defined policy regarding the `CATEGORIES` of a Java port, which is detailed in # 5.3, "Categorization".

6.11. Web applications, Apache and PHP

6.11.1. Apache

6.16. Variables for ports that use Apache

`USE_APACHE`	The port requires Apache. Possible values: `yes` (gets any version), `1.3`, `2.0`, `2.2`, `2.0+`, etc. Default dependency is on version `1.3`.
`WITH_APACHE2`	The port requires Apache 2.0. Without this variable, the port will depend on Apache

	1.3. This variable is deprecated and should not be used anymore.
APXS	Full path to the apxs binary. Can be overriden in your port.
HTTPD	Full path to the httpd binary. Can be overriden in your port.
APACHE_VERSION	The version of present Apache installation (read-only variable). This variable is only available after inclusion of bsd.port.pre.mk . Possible values: 13, 20, 22.
APACHEMODDIR	Directory for Apache modules. This variable is automatically expanded in pkg-plist.
APACHEINCLUDEDIR	Directory for Apache headers. This variable is automatically expanded in pkg-plist.
APACHEETCDIR	Directory for Apache configuration files. This variable is automatically expanded in pkg-plist.

6.17. port Apache

MODULENAME	##### #### PORTNAME . ### mod_hello
SHORTMODNAME	######## ###### MODULENAME ########### ### hello
AP_FAST_BUILD	## apxs ###########
AP_GENPLIST	########## pkg-plist #
AP_INC	###################
AP_LIB	###################
AP_EXTRAS	## apxs ### flags#

6.11.2. Web

Web ######### PREFIX/www/appname # For your convenience, this path is available both in Makefile and in pkg-plist as WWWDIR, and the path relative to PREFIX is available in Makefile as WWWDIR_REL .

The user and group of web server process are available as WWWOWN and WWWGRP, in case you need to change the ownership of some files. The default values of both are www. If you want different values for your port, use WWWOWN?= myuser notation, to allow user to override it easily.

Apache################## Apache ############### Apache # Web
##

6.11.3. PHP

6.18. Variables for ports that use PHP

USE_PHP	The port requires PHP. The value yes adds a dependency on PHP. The list of required PHP extensions can be specified instead. Example: pcre xml gettext
DEFAULT_PHP_VER	Selects which major version of PHP will be installed as a dependency when no PHP is installed yet. Default is 4. Possible values: 4, 5
IGNORE_WITH_PHP	The port does not work with PHP of the given version. Possible values: 4, 5
USE_PHPIZE	The port will be built as a PHP extension.
USE_PHPEXT	The port will be treated as a PHP extension, including installation and registration in the extension registry.
USE_PHP_BUILD	Set PHP as a build dependency.
WANT_PHP_CLI	Want the CLI (command line) version of PHP.
WANT_PHP_CGI	Want the CGI version of PHP.
WANT_PHP_MOD	Want the Apache module version of PHP.
WANT_PHP_SCR	Want the CLI or the CGI version of PHP.
WANT_PHP_WEB	Want the Apache module or the CGI version of PHP.

6.11.4. PEAR modules

Porting PEAR modules is a very simple process.

Use the variables FILES, TESTS, DATA, SQLS, SCRIPTFILES , DOCS and EXAMPLES to list the files you want to install. All listed files will be automatically installed into the appropriate locations and added to pkg-plist .

Include ${PORTSDIR}/devel/pear/bsd.pear.mk on the last line of the Makefile .

6.4. Example Makefile for PEAR class

```
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>
```

6.12. Using Python

The Ports Collection supports parallel installation of multiple Python versions. Ports should make sure to use a correct `python` interpreter, according to the user-settable `PYTHON_VERSION` variable. Most prominently, this means replacing the path to `python` executable in scripts with the value of `PYTHON_CMD` variable.

Ports that install files under `PYTHON_SITELIBDIR` should use the `pyXY-` package name prefix, so their package name embeds the version of Python they are installed into.

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

6.19. Most useful variables for ports that use Python

USE_PYTHON	The port needs Python. Minimal required version can be specified with values such as 2.3+. Version ranges can also be specified, by separating two version numbers with a dash, e.g.: 2.1-2.3
USE_PYDISTUTILS	Use Python distutils for configuring, compiling and installing. This is required when the port comes with setup.py. This overrides the do-build and do-install targets and may also override do-configure if GNU_CONFIGURE is not defined.
PYTHON_PKGNAMEPREFIX	Used as a PKGNAMEPREFIX to distinguish packages for different Python versions. Example: py24-
PYTHON_SITELIBDIR	Location of the site-packages tree, that contains installation path of Python (usually LOCALBASE). The PYTHON_SITELIBDIR variable can be very useful when installing Python modules.
PYTHONPREFIX_SITELIBDIR	The PREFIX-clean variant of PYTHON_SITELIBDIR. Always use %%PYTHON_SITELIBDIR%% in pkg-plist when possible. The default value of %%PYTHON_SITELIBDIR%% is lib/python%%PYTHON_VERSION%%/site-packages
PYTHON_CMD	Python interpreter command line, including version number.
PYNUMERIC	Dependency line for numeric extension.
PYNUMPY	Dependency line for the new numeric extension, numpy. (PYNUMERIC is deprecated by upstream vendor).
PYXML	Dependency line for XML extension (not needed for Python 2.0 and higher as it is also in base distribution).
USE_TWISTED	Add dependency on twistedCore. The list of required components can be specified as a value of this variable. Example: web lore pair flow

USE_ZOPE	Add dependency on Zope, a web application platform. Change Python dependency to Python 2.3. Set ZOPEBASEDIR containing a directory with Zope installation.

A complete list of available variables can be found in /usr/ports/Mk/bsd.python.mk .

6.13. Using Emacs

This section is yet to be written.

6.14. Using Ruby

6.20. Useful variables for ports that use Ruby

Variable	Description
USE_RUBY	The port requires Ruby.
USE_RUBY_EXTCONF	The port uses extconf.rb to configure.
USE_RUBY_SETUP	The port uses setup.rb to configure.
RUBY_SETUP	Set to the alternative name of setup.rb. Common value is install.rb .

The following table shows the selected variables available to port authors via the ports infrastructure. These variables should be used to install files into their proper locations. Use them in pkg-plist as much as possible. These variables should not be redefined in the port.

6.21. Selected read-only variables for ports that use Ruby

Variable	Description	Example value
RUBY_PKGNAMEPREFIX	Used as a PKGNAMEPREFIX to distinguish packages for different Ruby versions.	ruby18-
RUBY_VERSION	Full version of Ruby in the form of x.y.z.	1.8.2
RUBY_SITELIBDIR	Architecture independent libraries installation path.	/usr/local/lib/ruby/site_ruby/1.8
RUBY_SITEARCHLIBDIR	Architecture dependent libraries installation path.	/usr/local/lib/ruby/site_ruby/1.8/amd64-freebsd6

Variable	Description	Example value
RUBY_MODDOCDIR	Module documentation installation path.	/usr/local/share/doc/ruby18/patsy
RUBY_MODEXAMPLESDIR	Module examples installation path.	/usr/local/share/examples/ruby18/patsy

A complete list of available variables can be found in /usr/ports/Mk/bsd.ruby.mk .

6.15. Using SDL

The USE_SDL variable is used to autoconfigure the dependencies for ports which use an SDL based library like devel/sdl12 and x11-toolkits/sdl_gui.

The following SDL libraries are recognized at the moment:

- 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

Therefore, if a port has a dependency on net/sdl_net and audio/sdl_mixer, the syntax will be:

```
USE_SDL=        net mixer
```

The dependency devel/sdl12, which is required by net/sdl_net and audio/sdl_mixer, is automatically added as well.

If you use USE_SDL , it will automatically:

- Add a dependency on sdl12-config to `BUILD_DEPENDS`
- Add the variable `SDL_CONFIG` to `CONFIGURE_ENV`
- Add the dependencies of the selected libraries to the `LIB_DEPENDS`

To check whether an SDL library is available, you can do it with the `WANT_SDL` variable:

```
WANT_SDL=yes

.include <bsd.port.pre.mk>

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

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

6.16. Using wxWidgets

This section describes the status of the wxWidgets libraries in the ports tree and its integration with the ports system.

6.16.1. Introduction

There are many versions of the wxWidgets libraries which conflict between them (install files under the same name). In the ports tree this problem has been solved by installing each version under a different name using version number suffixes.

The obvious disadvantage of this is that each application has to be modified to find the expected version. Fortunately, most of the applications call the `wx-config` script to determine the necessary compiler and linker flags. The script is named differently for every available version. Majority of applications respect an environment variable, or accept a configure argument, to specify which `wx-config` script to call. Otherwise they have to be patched.

6.16.2. Version selection

To make your port use a specific version of wxWidgets there are two variables available for defining (if only one is defined the other will be set to a default value):

6.22. Variables to select wxWidgets versions

Variable	Description	Default value
`USE_WX`	List of versions the port can use	All available versions

Variable	Description	Default value
USE_WX_NOT	List of versions the port can not use	None

The following is a list of available wxWidgets versions and the corresponding ports in the tree:

6.23. Available wxWidgets versions

Version	Port
2.4	x11-toolkits/wxgtk24
2.6	x11-toolkits/wxgtk26
2.8	x11-toolkits/wxgtk28

##

The versions starting from 2.5 also come in Unicode version and are installed by a slave port named like the normal one plus a -unicode suffix, but this can be handled with variables (see # 6.16.4, “Unicode”).

The variables in ## 6.22, “Variables to select wxWidgets versions” can be set to one or more of the following combinations separated by spaces:

6.24. wxWidgets version specifications

Description	Example
Single version	2.4
Ascending range	2.4+
Descending range	2.6-
Full range (must be ascending)	2.4-2.6

There are also some variables to select the preferred versions from the available ones. They can be set to a list of versions, the first ones will have higher priority.

6.25. Variables to select preferred wxWidgets versions

Name	Designed for
WANT_WX_VER	the port
WITH_WX_VER	the user

6.16.3. Component selection

There are other applications that, while not being wxWidgets libraries, are related to them. These applications can be specified in the `WX_COMPS` variable. The following components are available:

6.26. Available wxWidgets components

Name	Description	Version restriction
`wx`	main library	none
`contrib`	contributed libraries	none
`python`	wxPython (Python bindings)	`2.4-2.6`
`mozilla`	wxMozilla	`2.4`
`svg`	wxSVG	`2.6`

The dependency type can be selected for each component by adding a suffix separated by a semicolon. If not present then a default type will be used (see ## 6.28, “Default wxWidgets dependency types”). The following types are available:

6.27. Available wxWidgets dependency types

Name	Description
`build`	Component is required for building, equivalent to `BUILD_DEPENDS`
`run`	Component is required for running, equivalent to `RUN_DEPENDS`
`lib`	Component is required for building and running, equivalent to `LIB_DEPENDS`

The default values for the components are detailed in the following table:

6.28. Default wxWidgets dependency types

Component	Dependency type
`wx`	`lib`
`contrib`	`lib`
`python`	`run`
`mozilla`	`lib`
`svg`	`lib`

6.5. Selecting wxWidgets components

The following fragment corresponds to a port which uses wxWidgets version `2.4` and its contributed libraries.

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

6.16.4. Unicode

The wxWidgets library supports Unicode since version `2.5`. In the ports tree both versions are available and can be selected with the following variables:

6.29. Variables to select Unicode in wxWidgets versions

Variable	Description	Designed for
`WX_UNICODE`	The port works *only* with the Unicode version	the port
`WANT_UNICODE`	The port works with both versions but prefers the Unicode one	the port
`WITH_UNICODE`	The port will use the Unicode version	the user
`WITHOUT_UNICODE`	The port will use the normal version if supported (when `WX_UNICODE` is not defined)	the user

##

Do not use `WX_UNICODE` for ports that can use both Unicode and normal versions. If you want the port to use Unicode by default define `WANT_UNICODE` instead.

6.16.5. Detecting installed versions

To detect an installed version you have to define `WANT_WX`. If you do not set it to a specific version then the components will have a version suffix. The `HAVE_WX` variable will be filled after detection.

6.6. Detecting installed wxWidgets versions and components

The following fragment can be used in a port that uses wxWidgets if it is installed, or an option is selected.

```
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
```

The following fragment can be used in a port that enables wxPython support if it is installed or if an option is selected, in addition to wxWidgets, both version `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
```

6.16.6. Defined variables

The following variables are available in the port (after defining one from ## 6.22, “Variables to select wxWidgets versions”).

6.30. Variables defined for ports that use wxWidgets

Name	Description
`WX_CONFIG`	The path to the wxWidgets `wx-config` script (with different name)
`WXRC_CMD`	The path to the wxWidgets `wxrc` program (with different name)
`WX_VERSION`	The wxWidgets version that is going to be used (e.g., `2.6`)

Name	Description
WX_UNICODE	If not defined but Unicode is going to be used then it will be defined

6.16.7. Processing in bsd.port.pre.mk

If you need to use the variables for running commands right after including bsd.port.pre.mk you need to define WX_PREMK.

##

If you define WX_PREMK, then the version, dependencies, components and defined variables will not change if you modify the wxWidgets port variables *after* including bsd.port.pre.mk .

6.7. Using wxWidgets variables in commands

The following fragment illustrates the use of WX_PREMK by running the wx-config script to obtain the full version string, assign it to a variable and pass it to the program.

```
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
```

##

The wxWidgets variables can be safely used in commands when they are inside targets without the need of WX_PREMK.

6.16.8. Additional configure arguments

Some GNU configure scripts can not find wxWidgets with just the WX_CONFIG environment variable set, requiring additional arguments. The WX_CONF_ARGS variable can be used for provide them.

6.31. Legal values for WX_CONF_ARGS

Possible value	Resulting argument
absolute	--with-wx-config=${WX_CONFIG}
relative	--with-wx=${LOCALBASE} --with-wx-config=${WX_CONFIG:T}

6.17. Using Lua

This section describes the status of the Lua libraries in the ports tree and its integration with the ports system.

6.17.1. Introduction

There are many versions of the Lua libraries and corresponding interpreters, which conflict between them (install files under the same name). In the ports tree this problem has been solved by installing each version under a different name using version number suffixes.

The obvious disadvantage of this is that each application has to be modified to find the expected version. But it can be solved by adding some additional flags to the compiler and linker.

6.17.2. Version selection

To make your port use a specific version of Lua there are two variables available for defining (if only one is defined the other will be set to a default value):

6.32. Variables to select Lua versions

Variable	Description	Default value
USE_LUA	List of versions the port can use	All available versions
USE_LUA_NOT	List of versions the port can not use	None

The following is a list of available Lua versions and the corresponding ports in the tree:

6.33. Available Lua versions

Version	Port
`4.0`	lang/lua4
`5.0`	lang/lua50
`5.1`	lang/lua

The variables in ## 6.32, “Variables to select Lua versions” can be set to one or more of the following combinations separated by spaces:

6.34. Lua version specifications

Description	Example
Single version	`4.0`
Ascending range	`5.0+`
Descending range	`5.0-`
Full range (must be ascending)	`5.0-5.1`

There are also some variables to select the preferred versions from the available ones. They can be set to a list of versions, the first ones will have higher priority.

6.35. Variables to select preferred Lua versions

Name	Designed for
`WANT_LUA_VER`	the port
`WITH_LUA_VER`	the user

6.8. Selecting the Lua version

The following fragment is from a port which can use Lua version `5.0` or `5.1`, and uses `5.0` by default. It can be overriden by the user using `WITH_LUA_VER` .

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

6.17.3. Component selection

There are other applications that, while not being Lua libraries, are related to them. These applications can be specified in the `LUA_COMPS` variable. The following components are available:

6.36. Available Lua components

Name	Description	Version restriction
`lua`	main library	none
`tolua`	Library for accesing C/C++ code	`4.0-5.0`
`ruby`	Ruby bindings	`4.0-5.0`

##

There are more components but they are modules for the interpreter, not used by applications (only by other modules).

The dependency type can be selected for each component by adding a suffix separated by a semicolon. If not present then a default type will be used (see ## 6.38, “Default Lua dependency types”). The following types are available:

6.37. Available Lua dependency types

Name	Description
`build`	Component is required for building, equivalent to `BUILD_DEPENDS`
`run`	Component is required for running, equivalent to `RUN_DEPENDS`
`lib`	Component is required for building and running, equivalent to `LIB_DEPENDS`

The default values for the components are detailed in the following table:

6.38. Default Lua dependency types

Component	Dependency type
`lua`	`lib` for `4.0-5.0` (shared) and `build` for `5.1` (static)
`tolua`	`build` (static)
`ruby`	`lib` (shared)

6.9. Selecting Lua components

The following fragment corresponds to a port which uses Lua version `4.0` and its Ruby bindings.

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

6.17.4. Detecting installed versions

To detect an installed version you have to define `WANT_LUA`. If you do not set it to a specific version then the components will have a version suffix. The `HAVE_LUA` variable will be filled after detection.

6.10. Detecting installed Lua versions and components

The following fragment can be used in a port that uses Lua if it is installed, or an option is selected.

```
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
```

The following fragment can be used in a port that enables tolua support if it is installed or if an option is selected, in addition to Lua, both version `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
```

6.17.5. Defined variables

The following variables are available in the port (after defining one from ## 6.32, “Variables to select Lua versions”).

6.39. Variables defined for ports that use Lua

Name	Description
LUA_VER	The Lua version that is going to be used (e.g., 5.1)
LUA_VER_SH	The Lua shared library major version (e.g., 1)
LUA_VER_STR	The Lua version without the dots (e.g., 51)
LUA_PREFIX	The prefix where Lua (and components) is installed
LUA_SUBDIR	The directory under ${PREFIX}/bin , ${PREFIX}/share and ${PREFIX}/lib where Lua is installed
LUA_INCDIR	The directory where Lua and tolua header files are installed
LUA_LIBDIR	The directory where Lua and tolua libraries are installed
LUA_MODLIBDIR	The directory where Lua module libraries (.so) are installed
LUA_MODSHAREDIR	The directory where Lua modules (.lua) are installed
LUA_PKGNAMEPREFIX	The package name prefix used by Lua modules
LUA_CMD	The path to the Lua interpreter
LUAC_CMD	The path to the Lua compiler
TOLUA_CMD	The path to the tolua program

6.11. Telling the port where to find Lua

The following fragment shows how to tell a port that uses a configure script where the Lua header files and libraries are.

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

6.17.6. Processing in `bsd.port.pre.mk`

If you need to use the variables for running commands right after including `bsd.port.pre.mk` you need to define `LUA_PREMK`.

##

If you define `LUA_PREMK`, then the version, dependencies, components and defined variables will not change if you modify the Lua port variables *after* including `bsd.port.pre.mk`.

6.12. Using Lua variables in commands

The following fragment illustrates the use of `LUA_PREMK` by running the Lua interpreter to obtain the full version string, assign it to a variable and pass it to the program.

```
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
```

##

The Lua variables can be safely used in commands when they are inside targets without the need of `LUA_PREMK` .

6.18. Using Xfce

The `USE_XFCE` variable is used to autoconfigure the dependencies for ports which use an Xfce based library or application like x11-toolkits/libxfce4gui and x11-wm/xfce4-panel.

The following Xfce libraries and applications are recognized at the moment:

- 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

The following additional parameters are recognized:

- configenv: Use this if your port requires a special modified `CONFIGURE_ENV` to find it's required libraries.

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

  gets added to CPPFLAGS to `CONFIGURE_ENV` .

Therefore, if a port has a dependency on sysutils/xfce4-mcs-manager and requires the special CPPFLAGS in its configure environment, the syntax will be:

```
USE_XFCE=       mcsmanager configenv
```

6.19. Using databases

6.40. Variables for ports using databases

Variable	Means
USE_BDB	If variable is set to yes, add dependency on databases/db41 port. The variable may also be set to values: 2, 3, 40, 41, 42, 43, 44, 45 46, or 47. You can declare a range of acceptable values, USE_BDB=42+ will find the highest installed version, and fall back to 42 if nothing else is installed.
USE_MYSQL	If variable is set to yes, add dependency on databases/mysql50-server port. An associated variable, WANT_MYSQL_VER , may be set to values such as 323, 40, 41, 50, 51 or 60.
USE_PGSQL	If set to yes, add dependency on databases/postgresql82 port. An associated variable, WANT_PGSQL_VER , may be set to values such as 73, 74, 80, 81, 82, or 83.

6.20. Starting and stopping services (rc scripts)

rc.d scripts are used to start services on system startup, and to give administrators a standard way of stopping, starting and restarting the service. Ports integrate into the system rc.d framework. Details on its usage can be found in the rc.d Handbook chapter. Detailed explanation of available commands is provided in rc(8) and rc.subr(8). Finally, there is an article on practical aspects of rc.d scripting.

One or more rc scripts can be installed:

```
USE_RC_SUBR= doormand
```

Scripts must be placed in the files subdirectory and a .in suffix must be added to their filename. The only difference from a base system rc.d script is that the . /etc/rc.subr line must be replaced with the . %%RC_SUBR%% , because older versions of FreeBSD do not have an /etc/rc.subr file. Standard SUB_LIST expansions are used too. Use of the %%PREFIX%% and %%LOCALBASE%% expansions is strongly encouraged as well. More on SUB_LIST in the relevant section.

Prior to FreeBSD 6.1-RELEASE, integration with rcorder(8) is available by using USE_RCORDER instead of USE_RC_SUBR . However, use of this method is deprecated.

As of FreeBSD 6.1-RELEASE, local `rc.d` scripts (including those installed by ports) are included in the overall rcorder(8) of the base system.

Example simple `rc.d` script:

```
#!/bin/sh

# PROVIDE: doormand
# REQUIRE: LOGIN
#
# Add the following lines to /etc/rc.conf.local or /etc/rc.conf
# to enable this service:
#
# doormand_enable (bool): Set to NO by default.
#    Set it to YES to enable doormand.
# doormand_config (path): Set to %%PREFIX%%/etc/doormand/doormand.cf
#    by default.
#

. %%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"
```

The "=" style of default variable assignment is preferable to the ":=" style here, since the former sets a default value only if the variable is unset, and the latter sets one if the variable is unset *or* null. A user might very well include something like

```
doormand_flags=""
```

in their `rc.conf.local` file, and a variable substitution using ":=" would inappropriately override the user's intention.

The suffix of the rc script is provided in `RC_SUBR_SUFFIX` for further use in the port's `Makefile`. Current versions of FreeBSD do not add any suffix to the script name, but older versions used to add `.sh` suffix.

##

No new scripts should be added with the `.sh` suffix. At some point there will be a mass repocopy of all the scripts that still have that suffix.

6.20.1. Stopping services at deinstall

It is possible to have a service stopped automatically as part of the deinstall routine. We advise using this feature only when it's absolutely necessary to stop a service before it's files go away. Usually, it's up to the administrator's discretion to decide, whether to stop the service on deinstall or not. Also note this affects upgrades, too.

Line like this goes to the `pkg-plist` :

```
@stopdaemon doormand
```

The argument must match the content of `USE_RC_SUBR` variable.

7. Advanced pkg-plist practices

7.1. Changing pkg-plist based on make variables

Some ports, particularly the p5- ports, need to change their pkg-plist depending on what options they are configured with (or version of perl, in the case of p5- ports). To make this easy, any instances in the pkg-plist of %%OSREL%% , %%PERL_VER%% , and %%PERL_VERSION%% will be substituted for appropriately. The value of %%OSREL%% is the numeric revision of the operating system (e.g., 4.9). %%PERL_VERSION%% is the full version number of perl (e.g., 5.00502) and %%PERL_VER%% is the perl version number minus the patchlevel (e.g., 5.005). Several other %%VARS%% related to port's documentation files are described in the relevant section.

If you need to make other substitutions, you can set the PLIST_SUB variable with a list of VAR=VALUE pairs and instances of %%VAR%% will be substituted with *VALUE* in the pkg-plist.

For instance, if you have a port that installs many files in a version-specific subdirectory, you can put something like

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

in the Makefile and use %%OCTAVE_VERSION%% wherever the version shows up in pkg-plist. That way, when you upgrade the port, you will not have to change dozens (or in some cases, hundreds) of lines in the pkg-plist .

This substitution (as well as addition of any manual pages) will be done between the pre-install and do-install targets, by reading from PLIST and writing to TMPPLIST (default: WRKDIR/.PLIST.mktmp). So if your port builds PLIST on the fly, do so in or before pre-install . Also, if your port needs to edit the resulting file, do so in post-install to a file named TMPPLIST .

Another possibility to modify port's packing list is based on setting the variables PLIST_FILES and PLIST_DIRS . The value of each variable is regarded as a list of pathnames to write to TMPPLIST along with PLIST contents. Names listed in PLIST_FILES and PLIST_DIRS are subject to %%VAR%% substitution, as described above. Except for that, names from PLIST_FILES will appear in the final packing list unchanged, while @dirrm will be prepended to names from PLIST_DIRS . To take effect, PLIST_FILES and PLIST_DIRS must be set before TMPPLIST is written, i.e. in pre-install or earlier.

7.2. Empty directories

7.2.1. Cleaning up empty directories

Do make your ports remove empty directories when they are de-installed. This is usually accomplished by adding `@dirrm` lines for all directories that are specifically created by the port. You need to delete subdirectories before you can delete parent directories.

```
 :
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
```

However, sometimes `@dirrm` will give you errors because other ports share the same directory. You can use `@dirrmtry` to remove only empty directories without warning.

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

This will neither print any error messages nor cause pkg_delete(1) to exit abnormally even if `${PREFIX}/share/doc/gimp` is not empty due to other ports installing some files in there.

7.2.2. Creating empty directories

Empty directories created during port installation need special attention. They will not get created when installing the package, because packages only store the files, and pkg_add(1) creates directories for them as needed. To make sure the empty directory is created when installing the package, add this line to `pkg-plist` above the corresponding `@dirrm` line:

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

7.3. Configuration files

If your port requires some configuration files in `PREFIX/etc` , do *not* just install them and list them in `pkg-plist` . That will cause pkg_delete(1) to delete files carefully edited by the user and a new installation to wipe them out.

Instead, install sample files with a suffix (`filename.sample` will work well). Copy the sample file as the real configuration file, if it does not exist. On deinstall, delete the configuration file, but only if it was not modified by the user. You need to handle this both in the port `Makefile` , and in the `pkg-plist` (for installation from the package).

Example of the `Makefile` part:

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

Example of the pkg-plist part:

```
@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
```

Alternatively, print out a message pointing out that the user has to copy and edit the file before the software can be made to work.

7.4. Dynamic vs. static package list

A *static package list* is a package list which is available in the Ports Collection either as a pkg-plist file (with or without variable substitution), or embedded into the Makefile via PLIST_FILES and PLIST_DIRS . Even if the contents are auto-generated by a tool or a target in the Makefile *before* the inclusion into the Ports Collection by a committer, this is still considered a static list, since it is possible to examine it without having to download or compile the distfile.

A *dynamic package list* is a package list which is generated at the time the port is compiled based upon the files and directories which are installed. It is not possible to examine it before the source code of the ported application is downloaded and compiled, or after running a make clean .

While the use of dynamic package lists is not forbidden, maintainers should use static package lists wherever possible, as it enables users to grep(1) through available ports to discover, for example, which port installs a certain file. Dynamic lists should be primarily used for complex ports where the package list changes drastically based upon optional features of the port (and thus maintaining a static package list is infeasible), or ports which change the package list based upon the version of dependent software used (e.g. ports which generate docs with Javadoc).

Maintainers who prefer dynamic package lists are encouraged to add a new target to their port which generates the pkg-plist file so that users may examine the contents.

7.5. #### package list

######## port ## pkg-plist ##############

########### port ############### #

```
# 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)
```

############

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

pkg-plist

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

port ### PREFIX (######)# ######## port

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

####################

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

############################Man page ##### MAN*n* ### ## port # Makefile ####### #### ###############filename.sample # The info/dir file should not be listed and appropriate install-info lines should be added as noted in the info files section. Any libraries installed by the port should be listed as specified in the shared libraries section.

Alternatively, use the plist script in /usr/ports/Tools/scripts/ to build the package list automatically. The first step is the same as above: take the first three lines, that is, mkdir, mtree and make depends. Then build and install the port:

```
# make install PREFIX=/var/tmp/port-name
```

And let plist create the pkg-plist file:

```
# /usr/ports/Tools/scripts/plist -Md -m /etc/mtree/BSD.port-type.↺
dist /var/tmp/port-name > pkg-plist
```

The packing list still has to be tidied up by hand as stated above.

8. The pkg-* files

There are some tricks we have not mentioned yet about the `pkg-*` files that come in handy sometimes.

8.1. pkg-message

If you need to display a message to the installer, you may place the message in `pkg-message`. This capability is often useful to display additional installation steps to be taken after a pkg_add(1) or to display licensing information.

When some lines about the build-time knobs or warnings have to be displayed, use `ECHO_MSG`. The `pkg-message` file is only for post-installation steps. Likewise, the distinction between `ECHO_MSG` and `ECHO_CMD` should be kept in mind. The former is for printing informational text to the screen, while the latter is for command pipelining.

A good example for both can be found in `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
```

##

The `pkg-message` file does not need to be added to `pkg-plist`. Also, it will not get automatically printed if the user is using the port, not the package, so you should probably display it from the `post-install` target yourself.

8.2. pkg-install

If your port needs to execute commands when the binary package is installed with pkg_add(1) you can do this via the `pkg-install` script. This script will automatically be added to the package, and will be run twice by pkg_add(1): the first time as `${SH} pkg-install ${PKGNAME} PRE-INSTALL` and the second time as `${SH} pkg-install ${PKGNAME} POST-INSTALL`. `$2` can be tested to determine which mode the script is being run in. The `PKG_PREFIX` environmental variable will be set to the package installation directory. See pkg_add(1) for additional information.

##

This script is not run automatically if you install the port with `make install`. If you are depending on it being run, you will have to explicitly call it from your port's `Makefile`, with a line like `PKG_PREFIX=${PREFIX} ${SH} ${PKGINSTALL} ${PKGNAME} PRE-INSTALL`.

8.3. pkg-deinstall

This script executes when a package is removed.

This script will be run twice by pkg_delete(1). The first time as `${SH} pkg-deinstall ${PKGNAME} DEINSTALL` and the second time as `${SH} pkg-deinstall ${PKGNAME} POST-DEINSTALL`.

8.4. pkg-req

If your port needs to determine if it should install or not, you can create a `pkg-req` "requirements" script. It will be invoked automatically at installation/de-installation time to determine whether or not installation/de-installation should proceed.

The script will be run at installation time by pkg_add(1) as `pkg-req ${PKGNAME} INSTALL`. At de-installation time it will be run by pkg_delete(1) as `pkg-req ${PKGNAME} DEINSTALL`.

8.5. Changing the names of pkg-* files

All the names of `pkg-*` files are defined using variables so you can change them in your `Makefile` if need be. This is especially useful when you are sharing the same `pkg-*` files among several ports or have to write to one of the above files (see writing to places other than `WRKDIR` for why it is a bad idea to write directly into the `pkg-*` subdirectory).

Here is a list of variable names and their default values. (`PKGDIR` defaults to `${MASTERDIR}`.)

Variable	Default value
DESCR	${PKGDIR}/pkg-descr

Variable	Default value
PLIST	${PKGDIR}/pkg-plist
PKGINSTALL	${PKGDIR}/pkg-install
PKGDEINSTALL	${PKGDIR}/pkg-deinstall
PKGREQ	${PKGDIR}/pkg-req
PKGMESSAGE	${PKGDIR}/pkg-message

Please change these variables rather than overriding PKG_ARGS. If you change PKG_ARGS, those files will not correctly be installed in /var/db/pkg upon install from a port.

8.6. Making use of SUB_FILES and SUB_LIST

The SUB_FILES and SUB_LIST variables are useful for dynamic values in port files, such as the installation PREFIX in pkg-message.

The SUB_FILES variable specifies a list of files to be automatically modified. Each *file* in the SUB_FILES list must have a corresponding file.in present in FILESDIR. A modified version will be created in WRKDIR. Files defined as a value of USE_RC_SUBR (or the deprecated USE_RCORDER) are automatically added to the SUB_FILES. For the files pkg-message, pkg-install, pkg-deinstall and pkg-reg, the corresponding Makefile variable is automatically set to point to the processed version.

The SUB_LIST variable is a list of VAR=VALUE pairs. For each pair %%VAR%% will get replaced with VALUE in each file listed in SUB_FILES. Several common pairs are automatically defined: PREFIX, LOCALBASE, DATADIR, DOCSDIR, EXAMPLESDIR. Any line beginning with @comment will be deleted from resulting files after a variable substitution.

The following example will replace %%ARCH%% with the system architecture in a pkg-message:

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

Note that for this example, the pkg-message.in file must exist in FILESDIR.

Example of a good pkg-message.in:

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

9. Testing your port

9.1. Running `make describe`

Several of the FreeBSD port maintenance tools, such as portupgrade(1), rely on a database called `/usr/ports/INDEX` which keeps track of such items as port dependencies. `INDEX` is created by the top-level `ports/Makefile` via `make index`, which descends into each port subdirectory and executes `make describe` there. Thus, if `make describe` fails in any port, no one can generate `INDEX`, and many people will quickly become unhappy.

##

It is important to be able to generate this file no matter what options are present in `make.conf`, so please avoid doing things such as using `.error` statements when (for instance) a dependency is not satisfied. (See # 12.17, "Avoid use of the `.error` construct".)

If `make describe` produces a string rather than an error message, you are probably safe. See `bsd.port.mk` for the meaning of the string produced.

Also note that running a recent version of `portlint` (as specified in the next section) will cause `make describe` to be run automatically.

9.2. Portlint

Do check your work with `portlint` before you submit or commit it. `portlint` warns you about many common errors, both functional and stylistic. For a new (or repocopied) port, `portlint -A` is the most thorough; for an existing port, `portlint -C` is sufficient.

Since `portlint` uses heuristics to try to figure out errors, it can produce false positive warnings. In addition, occasionally something that is flagged as a problem really cannot be done in any other way due to limitations in the ports framework. When in doubt, the best thing to do is ask on FreeBSD ports ####.

9.3. Port Tools

The ports-mgmt/porttools program is part of the Ports Collection.

port is the front-end script, which can help you simplify the testing job. Whenever you want to test a new port or update an existing one, you can use port test to test your port, including the portlint checking. This command also detects and lists any files that are not listed in pkg-plist . See the following example:

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

9.4. PREFIX ## DESTDIR

PREFIX ###### port ######## ### /usr/local # /opt # ######### PREFIX ######## ### ### port ########

####### DESTDIR ### ################## jail ########## / ##### ## port ### DESTDIR / PREFIX ### ###### DESTDIR /var/db/pkg ######## ##### DESTDIR ## ports #### chroot(8) ###################### DESTDIR ### ports#

PREFIX ########### LOCALBASE ## (### /usr/local)# #### USE_LINUX_PREFIX # ## PREFIX ## LINUXBASE (### /compat/linux)#

Avoiding the hard-coding of /usr/local or /usr/X11R6 anywhere in the source will make the port much more flexible and able to cater to the needs of other sites. For X ports that use imake, this is automatic; otherwise, this can often be done by simply replacing the occurrences of /usr/local (or /usr/X11R6 for X ports that do not use imake) in the various Makefile s in the port to read ${PREFIX} , as this variable is automatically passed down to every stage of the build and install processes.

Make sure your application is not installing things in /usr/local instead of PREFIX. A quick test for this is to do this is:

```
# make clean; make package PREFIX=/var/tmp/port-name
```

If anything is installed outside of PREFIX, the package creation process will complain that it cannot find the files.

This does not test for the existence of internal references, or correct use of LOCALBASE for references to files from other ports. Testing the installation in /var/tmp/port-name to do that while you have it installed would do that.

The variable PREFIX can be reassigned in your Makefile or in the user's environment. However, it is strongly discouraged for individual ports to set this variable explicitly in the Makefile s.

Also, refer to programs/files from other ports with the variables mentioned above, not explicit pathnames. For instance, if your port requires a macro PAGER to be the full pathname of less, use the compiler flag:

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

instead of `-DPAGER=\"/usr/local/bin/less\"` . This way it will have a better chance of working if the system administrator has moved the whole `/usr/local` tree somewhere else.

9.5. Tinderbox

If you're an avid ports contributor, you might want to take a look at Tinderbox. It is a powerful system for building and testing ports based on the scripts used on Pointyhat. You can install Tinderbox using ports-mgmt/tinderbox port. Be sure to read supplied documentation since the configuration is not trivial.

Visit the Tinderbox website for more details.

10. Upgrading

When you notice that a port is out of date compared to the latest version from the original authors, you should first ensure that you have the latest port. You can find them in the `ports/ports-current` directory of the FreeBSD FTP mirror sites. However, if you are working with more than a few ports, you will probably find it easier to use CVSup to keep your whole ports collection up-to-date, as described in the Handbook. This will have the added benefit of tracking all the ports' dependencies.

The next step is to see if there is an update already pending. To do this, you have two options. There is a searchable interface to the FreeBSD Problem Report (PR) database (also known as `GNATS`). Select `ports` in the dropdown, and enter the name of the port.

However, sometimes people forget to put the name of the port into the Synopsis field in an unambiguous fashion. In that case, you can try the FreeBSD Ports Monitoring System (also known as `portsmon`). This system attempts to classify port PRs by portname. To search for PRs about a particular port, use the Overview of One Port.

If there is no pending PR, the next step is to send an email to the port's maintainer, as shown by `make maintainer`. That person may already be working on an upgrade, or have a reason to not upgrade the port right now (because of, for example, stability problems of the new version); you would not want to duplicate their work. Note that unmaintained ports are listed with a maintainer of `ports@FreeBSD.org`, which is just the general ports mailing list, so sending mail there probably will not help in this case.

If the maintainer asks you to do the upgrade or there is no maintainer, then you have a chance to help out FreeBSD by preparing the update yourself! Please make the changes and save the result of the recursive `diff` output of the new and old ports directories (e.g., if your modified port directory is called `superedit` and the original is in our tree as `superedit.bak`, then save the result of `diff -ruN superedit.bak superedit`). Either unified or context diff is fine, but port committers generally prefer unified diffs. Note the use of the `-N` option——this is the accepted way to force diff to properly deal with the case of new files being added or old files being deleted. Before sending us the diff, please examine the output to make sure all the changes make sense. To simplify common operations with patch files, you can use `/usr/ports/Tools/scripts/patchtool.py`. Before using it, please read `/usr/ports/Tools/scripts/README.patchtool`.

If the port is unmaintained, and you are actively using it yourself, please consider volunteering to become its maintainer. FreeBSD has over 2000 ports without maintainers, and this is an area where more volunteers are always needed. (For a detailed description of the responsibilities of maintainers, refer to the section in the Developer's Handbook.)

`########` send-pr(1) `###` diff `####(####` `ports)#` `####` port `#########` synopsis `#######` `[maintainer update]` `####` PR # “Class” `####` `maintainer-update` # `####` PR # “Class” # `###` `change-request` # Please mention any added or deleted files in the message, as they have to be explicitly specified to cvs(1) when doing a commit. If the diff is more than about 20KB, please compress and uuencode it; otherwise, just include it in the PR as is.

Before you send-pr(1), you should review the Writing the problem report section in the Problem Reports article; it contains far more information about how to write useful problem reports.

##

If your upgrade is motivated by security concerns or a serious fault in the currently committed port, please notify the Ports Management Team <portmgr@FreeBSD.org > to request immediate rebuilding and redistribution of your port's package. Unsuspecting users of pkg_add(1) will otherwise continue to install the old version via `pkg_add -r` for several weeks.

##

Once again, please use diff(1) and not shar(1) to send updates to existing ports!

Now that you have done all that, you will want to read about how to keep up-to-date in # 14, *Keeping Up*.

11. Ports security

11.1. Why security is so important

Bugs are occasionally introduced to the software. Arguably, the most dangerous of them are those opening security vulnerabilities. From the technical viewpoint, such vulnerabilities are to be closed by exterminating the bugs that caused them. However, the policies for handling mere bugs and security vulnerabilities are very different.

A typical small bug affects only those users who have enabled some combination of options triggering the bug. The developer will eventually release a patch followed by a new version of the software, free of the bug, but the majority of users will not take the trouble of upgrading immediately because the bug has never vexed them. A critical bug that may cause data loss represents a graver issue. Nevertheless, prudent users know that a lot of possible accidents, besides software bugs, are likely to lead to data loss, and so they make backups of important data; in addition, a critical bug will be discovered really soon.

A security vulnerability is all different. First, it may remain unnoticed for years because often it does not cause software malfunction. Second, a malicious party can use it to gain unauthorized access to a vulnerable system, to destroy or alter sensitive data; and in the worst case the user will not even notice the harm caused. Third, exposing a vulnerable system often assists attackers to break into other systems that could not be compromised otherwise. Therefore closing a vulnerability alone is not enough: the audience should be notified of it in most clear and comprehensive manner, which will allow to evaluate the danger and take appropriate actions.

11.2. Fixing security vulnerabilities

While on the subject of ports and packages, a security vulnerability may initially appear in the original distribution or in the port files. In the former case, the original software developer is likely to release a patch or a new version instantly, and you will only need to update the port promptly with respect to the author's fix. If the fix is delayed for some reason, you should either mark the port as `FORBIDDEN` or introduce a patch file of your own to the port. In the case of a vulnerable port, just fix the port as soon as possible. In either case, the standard procedure for submitting your change should be followed unless you have rights to commit it directly to the ports tree.

##

Being a ports committer is not enough to commit to an arbitrary port. Remember that ports usually have maintainers, whom you should respect.

Please make sure that the port's revision is bumped as soon as the vulnerability has been closed. That is how the users who upgrade installed packages on a regular basis will see they need to run an update. Besides, a new package will be built and distributed over FTP and WWW mirrors, replacing the vulnerable one. `PORTREVISION` should be bumped unless `PORTVERSION` has changed in the course of correcting the vulnerability. That is you should bump `PORTREVISION` if you have added a patch file to the port, but you should not if you have updated the port to the latest software version and thus already touched `PORTVERSION`. Please refer to the corresponding section for more information.

11.3. Keeping the community informed

11.3.1. The VuXML database

A very important and urgent step to take as early as a security vulnerability is discovered is to notify the community of port users about the jeopardy. Such notification serves two purposes. First, should the danger be really severe, it will be wise to apply an instant workaround, e.g., stop the affected network service or even deinstall the port completely, until the vulnerability is closed. Second, a lot of users tend to upgrade installed packages just occasionally. They will know from the notification that they *must* update the package without delay as soon as a corrected version is available.

Given the huge number of ports in the tree, a security advisory cannot be issued on each incident without creating a flood and losing the attention of the audience by the time it comes to really serious matters. Therefore security vulnerabilities found in ports are recorded in the FreeBSD VuXML database. The Security Officer Team members are monitoring it for issues requiring their intervention.

If you have committer rights, you can update the VuXML database by yourself. So you will both help the Security Officer Team and deliver the crucial information to the community earlier. However, if you are not a committer, or you believe you have found an exceptionally severe vulnerability, or whatever, please do not hesitate to contact the Security Officer Team directly as described on the FreeBSD Security Information page.

All right, you elected the hard way. As it may be obvious from its title, the VuXML database is essentially an XML document. Its source file `vuln.xml` is kept right inside the port

security/vuxml. Therefore the file's full pathname will be `PORTSDIR/security/vuxml/vuln.xml` . Each time you discover a security vulnerability in a port, please add an entry for it to that file. Until you are familiar with VuXML, the best thing you can do is to find an existing entry fitting your case, then copy it and use as a template.

11.3.2. A short introduction to VuXML

The full-blown XML is complex and far beyond the scope of this book. However, to gain basic insight on the structure of a VuXML entry, you need only the notion of tags. XML tag names are enclosed in angle brackets. Each opening <tag> must have a matching closing </tag>. Tags may be nested. If nesting, the inner tags must be closed before the outer ones. There is a hierarchy of tags, i.e. more complex rules of nesting them. Sounds very similar to HTML, doesn't it? The major difference is that XML is eXtensible, i.e. based on defining custom tags. Due to its intrinsic structure, XML puts otherwise amorphous data into shape. VuXML is particularly tailored to mark up descriptions of security vulnerabilities.

Now let's consider a realistic VuXML entry:

```
<vuln vid="f4bc80f4-da62-11d8-90ea-0004ac98a7b9"> ❶
  <topic>Several vulnerabilities found in Foo</topic> ❷
  <affects>
    <package>
      <name>foo</name> ❸
      <name>foo-devel</name>
      <name>ja-foo</name>
      <range><ge>1.6</ge><lt>1.9</lt></range> ❹
      <range><ge>2.*</ge><lt>2.4_1</lt></range>
      <range><eq>3.0b1</eq></range>
    </package>
    <package>
      <name>openfoo</name> ❺
      <range><lt>1.10_7</lt></range> ❻
      <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> ❼
      <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> ❽
    <freebsdsa>SA-10:75.foo</freebsdsa> ❾
    <freebsdpr>ports/987654</freebsdpr> ❿
```

```
    <cvename>CAN-2010-0201</cvename>
    <cvename>CAN-2010-0466</cvename>
    <bid>96298</bid>
    <certsa>CA-2010-99</certsa>
    <certvu>740169</certvu>
    <uscertsa>SA10-99A</uscertsa>
    <uscertta>SA10-99A</uscertta>
    <mlist msgid="201075606@hacker.com">http://marc.theaimsgroup.↺
com/?l=bugtraq&amp;m=203886607825605</mlist>
    <url>http://j.r.hacker.com/advisories/1</url>
  </references>
  <dates>
    <discovery>2010-05-25</discovery>
    <entry>2010-07-13</entry>
    <modified>2010-09-17</entry>
  </dates>
</vuln>
```

The tag names are supposed to be self-descriptive, so we shall take a closer look only at fields you will need to fill in by yourself:

❶ This is the top-level tag of a VuXML entry. It has a mandatory attribute, `vid`, specifying a universally unique identifier (UUID) for this entry (in quotes). You should generate a UUID for each new VuXML entry (and do not forget to substitute it for the template UUID unless you are writing the entry from scratch). You can use uuidgen(1) to generate a VuXML UUID; alternatively, if you are using FreeBSD 4.x, you may install the port devel/p5-Data-UUID and issue the following command:

```
perl -MData::UUID -le 'print lc new Data::UUID->create_str'
```

❷ This is a one-line description of the issue found.

❸ The names of packages affected are listed there. Multiple names can be given since several packages may be based on a single master port or software product. This may include stable and development branches, localized versions, and slave ports featuring different choices of important build-time configuration options.

##

It is your responsibility to find all such related packages when writing a VuXML entry. Keep in mind that `make search name=foo` is your friend. The primary points to look for are as follows:

- the `foo-devel` variant for a `foo` port;
- other variants with a suffix like `-a4` (for print-related packages), `-without-gui` (for packages with X support disabled), or similar;

- `jp-`, `ru-`, `zh-`, and other possible localized variants in the corresponding national categories of the ports collection.

❹ Affected versions of the package(s) are specified there as one or more ranges using a combination of `<lt>`, `<le>`, `<eq>`, `<ge>`, and `<gt>` elements. The version ranges given should not overlap.

In a range specification, `*` (asterisk) denotes the smallest version number. In particular, `2.*` is less than `2.a`. Therefore an asterisk may be used for a range to match all possible `alpha`, `beta`, and `RC` versions. For instance, `<ge>2.*</ge><lt>3.*</lt>` will selectively match every `2.x` version while `<ge>2.0</ge><lt>3.0</lt>` will obviously not since the latter misses `2.r3` and matches `3.b`.

The above example specifies that affected are versions from `1.6` to `1.9` inclusive, versions `2.x` before `2.4_1`, and version `3.0b1`.

❺ Several related package groups (essentially, ports) can be listed in the `<affected>` section. This can be used if several software products (say FooBar, FreeBar and OpenBar) grow from the same code base and still share its bugs and vulnerabilities. Note the difference from listing multiple names within a single <package> section.

❻ The version ranges should allow for `PORTEPOCH` and `PORTREVISION` if applicable. Please remember that according to the collation rules, a version with a non-zero `PORTEPOCH` is greater than any version without `PORTEPOCH`, e.g., `3.0,1` is greater than `3.1` or even than `8.9`.

❼ This is a summary of the issue. XHTML is used in this field. At least enclosing `<p>` and `</p>` should appear. More complex mark-up may be used, but only for the sake of accuracy and clarity: No eye candy please.

❽ This section contains references to relevant documents. As many references as apply are encouraged.

❾ This is a FreeBSD security advisory.

❿ This is a FreeBSD problem report.

This is a Mitre CVE identifier.

This is a SecurityFocus Bug ID.

This is a US-CERT security advisory.

This is a US-CERT vulnerability note.

This is a US-CERT Cyber Security Alert.

This is a US-CERT Technical Cyber Security Alert.

This is a URL to an archived posting in a mailing list. The attribute `msgid` is optional and may specify the message ID of the posting.

This is a generic URL. It should be used only if none of the other reference categories apply.

This is the date when the issue was disclosed (*YYYY-MM-DD*).

This is the date when the entry was added (*YYYY-MM-DD*).

This is the date when any information in the entry was last modified (*YYYY-MM-DD*). New entries must not include this field. It should be added upon editing an existing entry.

11.3.3. Testing your changes to the VuXML database

Assume you just wrote or filled in an entry for a vulnerability in the package `clamav` that has been fixed in version `0.65_7`.

As a prerequisite, you need to install fresh versions of the ports ports-mgmt/portaudit and ports-mgmt/portaudit-db.

First, check whether there already is an entry for this vulnerability. If there were such entry, it would match the previous version of the package, `0.65_6`:

```
% packaudit
% portaudit clamav-0.65_6
```

##

To run `packaudit`, you must have permission to write to its `DATABASEDIR`, typically `/var/db/portaudit` .

If there is none found, you get the green light to add a new entry for this vulnerability. Now you can generate a brand-new UUID (assume it's `74a9541d-5d6c-11d8-80e3-0020ed76ef5a`) and add your new entry to the VuXML database. Please verify its syntax after that as follows:

```
% cd ${PORTSDIR}/security/vuxml && make validate
```

##

You will need at least one of the following packages installed: textproc/libxml2, textproc/jade.

Now rebuild the `portaudit` database from the VuXML file:

```
% packaudit
```

To verify that the `<affected>` section of your entry will match correct package(s), issue the following command:

```
% portaudit -f /usr/ports/INDEX -r ↺
74a9541d-5d6c-11d8-80e3-0020ed76ef5a
```

##

Please refer to portaudit(1) for better understanding of the command syntax.

Make sure that your entry produces no spurious matches in the output.

Now check whether the right package versions are matched by your entry:

```
% 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.
```

Obviously, the former version should match while the latter one should not.

Finally, verify whether the web page generated from the VuXML database looks like expected:

```
% mkdir -p ~/public_html/portaudit
% packaudit
% lynx ~/public_html/portaudit/74a9541d-5d6c-11d8-80e3-0020ed76ef5a.↺
html
```

12. Dos and Don'ts

12.1. Introduction

Here is a list of common dos and don'ts that you encounter during the porting process. You should check your own port against this list, but you can also check ports in the PR database that others have submitted. Submit any comments on ports you check as described in Bug Reports and General Commentary. Checking ports in the PR database will both make it faster for us to commit them, and prove that you know what you are doing.

12.2. `WRKDIR`

Do not write anything to files outside `WRKDIR`. `WRKDIR` is the only place that is guaranteed to be writable during the port build (see installing ports from a CDROM for an example of building ports from a read-only tree). If you need to modify one of the `pkg-*` files, do so by redefining a variable, not by writing over it.

12.3. `WRKDIRPREFIX`

Make sure your port honors `WRKDIRPREFIX`. Most ports do not have to worry about this. In particular, if you are referring to a `WRKDIR` of another port, note that the correct location is `WRKDIRPREFIX``PORTSDIR/subdir/name/work` not `PORTSDIR/subdir/name/work` or `.CURDIR/../../subdir/name/work` or some such.

Also, if you are defining `WRKDIR` yourself, make sure you prepend `${WRKDIRPREFIX}${.CURDIR}` in the front.

12.4. Differentiating operating systems and OS versions

You may come across code that needs modifications or conditional compilation based upon what version of Unix it is running under. If you need to make such changes to the code for conditional compilation, make sure you make the changes as general as possible so that we can back-port code to older FreeBSD systems and cross-port to other BSD systems such as 4.4BSD from CSRG, BSD/386, 386BSD, NetBSD, and OpenBSD.

The preferred way to tell 4.3BSD/Reno (1990) and newer versions of the BSD code apart is by using the `BSD` macro defined in sys/param.h. Hopefully that file is already included; if not, add the code:

```
#if (defined(__unix__) || defined(unix)) && !defined(USG)
#include <sys/param.h>
#endif
```

to the proper place in the `.c` file. We believe that every system that defines these two symbols has `sys/param.h` . If you find a system that does not, we would like to know. Please send mail to the FreeBSD ports ####.

Another way is to use the GNU Autoconf style of doing this:

```
#ifdef HAVE_SYS_PARAM_H
#include <sys/param.h>
#endif
```

Do not forget to add `-DHAVE_SYS_PARAM_H` to the `CFLAGS` in the `Makefile` for this method.

Once you have `sys/param.h` included, you may use:

```
#if (defined(BSD) && (BSD >= 199103))
```

to detect if the code is being compiled on a 4.3 Net2 code base or newer (e.g. FreeBSD 1.x, 4.3/Reno, NetBSD 0.9, 386BSD, BSD/386 1.1 and below).

Use:

```
#if (defined(BSD) && (BSD >= 199306))
```

to detect if the code is being compiled on a 4.4 code base or newer (e.g. FreeBSD 2.x, 4.4, NetBSD 1.0, BSD/386 2.0 or above).

The value of the `BSD` macro is `199506` for the 4.4BSD-Lite2 code base. This is stated for informational purposes only. It should not be used to distinguish between versions of FreeBSD based only on 4.4-Lite vs. versions that have merged in changes from 4.4-Lite2. The `__FreeBSD__` macro should be used instead.

Use sparingly:

- `__FreeBSD__` is defined in all versions of FreeBSD. Use it if the change you are making *only* affects FreeBSD. Porting gotchas like the use of `sys_errlist[]` vs `strerror()` are Berkeley-isms, not FreeBSD changes.
- In FreeBSD 2.x, `__FreeBSD__` is defined to be 2. In earlier versions, it is 1. Later versions always bump it to match their major version number.
- If you need to tell the difference between a FreeBSD 1.x system and a FreeBSD 2.x or above system, usually the right answer is to use the `BSD` macros described above. If there actually is a FreeBSD specific change (such as special shared library options when

using `ld`) then it is OK to use `__FreeBSD__` and `#if __FreeBSD__ > 1` to detect a FreeBSD 2.x and later system. If you need more granularity in detecting FreeBSD systems since 2.0-RELEASE you can use the following:

```
#if __FreeBSD__ >= 2
#include <osreldate.h>
#    if __FreeBSD_version >= 199504
  /* 2.0.5+ release specific code here */
#    endif
#endif
```

In the hundreds of ports that have been done, there have only been one or two cases where `__FreeBSD__` should have been used. Just because an earlier port screwed up and used it in the wrong place does not mean you should do so too.

12.5. FreeBSD #####(__FreeBSD_version)

sys/param.h ## `__FreeBSD_version`

12.1. __FreeBSD_version values

Release	**__FreeBSD_version**
2.0-RELEASE	119411
2.1-CURRENT	199501, 199503
2.0.5-RELEASE	199504
2.2-CURRENT before 2.1	199508
2.1.0-RELEASE	199511
2.2-CURRENT before 2.1.5	199512
2.1.5-RELEASE	199607
2.2-CURRENT before 2.1.6	199608
2.1.6-RELEASE	199612
2.1.7-RELEASE	199612
2.2-RELEASE	220000
2.2.1-RELEASE	220000 (no change)
2.2-STABLE after 2.2.1-RELEASE	220000 (no change)
2.2-STABLE after texinfo-3.9	221001
2.2-STABLE after top	221002
2.2.2-RELEASE	222000
2.2-STABLE after 2.2.2-RELEASE	222001

Release	__FreeBSD_version
2.2.5-RELEASE	225000
2.2-STABLE after 2.2.5-RELEASE	225001
2.2-STABLE after ldconfig -R merge	225002
2.2.6-RELEASE	226000
2.2.7-RELEASE	227000
2.2-STABLE after 2.2.7-RELEASE	227001
2.2-STABLE after semctl(2) change	227002
2.2.8-RELEASE	228000
2.2-STABLE after 2.2.8-RELEASE	228001
3.0-CURRENT before mount(2) change	300000
3.0-CURRENT after mount(2) change	300001
3.0-CURRENT after semctl(2) change	300002
3.0-CURRENT after ioctl arg changes	300003
3.0-CURRENT after ELF conversion	300004
3.0-RELEASE	300005
3.0-CURRENT after 3.0-RELEASE	300006
3.0-STABLE after 3/4 branch	300007
3.1-RELEASE	310000
3.1-STABLE after 3.1-RELEASE	310001
3.1-STABLE after C++ constructor/ destructor order change	310002
3.2-RELEASE	320000
3.2-STABLE	320001
3.2-STABLE after binary-incompatible IPFW and socket changes	320002
3.3-RELEASE	330000
3.3-STABLE	330001
3.3-STABLE after adding mkstemp(3) to libc	330002
3.4-RELEASE	340000
3.4-STABLE	340001
3.5-RELEASE	350000

Release	`__FreeBSD_version`
3.5-STABLE	350001
4.0-CURRENT after 3.4 branch	400000
4.0-CURRENT after change in dynamic linker handling	400001
4.0-CURRENT after C++ constructor/destructor order change	400002
4.0-CURRENT after functioning dladdr(3)	400003
4.0-CURRENT after __deregister_frame_info dynamic linker bug fix (also 4.0-CURRENT after EGCS 1.1.2 integration)	400004
4.0-CURRENT after suser(9) API change (also 4.0-CURRENT after newbus)	400005
4.0-CURRENT after cdevsw registration change	400006
4.0-CURRENT after the addition of so_cred for socket level credentials	400007
4.0-CURRENT after the addition of a poll syscall wrapper to libc_r	400008
4.0-CURRENT after the change of the kernel's `dev_t` type to `struct specinfo` pointer	400009
4.0-CURRENT after fixing a hole in jail(2)	400010
4.0-CURRENT after the `sigset_t` datatype change	400011
4.0-CURRENT after the cutover to the GCC 2.95.2 compiler	400012
4.0-CURRENT after adding pluggable linux-mode ioctl handlers	400013
4.0-CURRENT after importing OpenSSL	400014
4.0-CURRENT after the C++ ABI change in GCC 2.95.2 from -fvtable-thunks to -fno-vtable-thunks by default	400015
4.0-CURRENT after importing OpenSSH	400016
4.0-RELEASE	400017
4.0-STABLE after 4.0-RELEASE	400018

Release	__FreeBSD_version
4.0-STABLE after the introduction of delayed checksums.	400019
4.0-STABLE after merging libxpg4 code into libc.	400020
4.0-STABLE after upgrading Binutils to 2.10.0, ELF branding changes, and tcsh in the base system.	400021
4.1-RELEASE	410000
4.1-STABLE after 4.1-RELEASE	410001
4.1-STABLE after setproctitle(3) moved from libutil to libc.	410002
4.1.1-RELEASE	411000
4.1.1-STABLE after 4.1.1-RELEASE	411001
4.2-RELEASE	420000
4.2-STABLE after combining libgcc.a and libgcc_r.a, and associated GCC linkage changes.	420001
4.3-RELEASE	430000
4.3-STABLE after wint_t introduction.	430001
4.3-STABLE after PCI powerstate API merge.	430002
4.4-RELEASE	440000
4.4-STABLE after d_thread_t introduction.	440001
4.4-STABLE after mount structure changes (affects filesystem klds).	440002
4.4-STABLE after the userland components of smbfs were imported.	440003
4.5-RELEASE	450000
4.5-STABLE after the usb structure element rename.	450001
4.5-STABLE after the `sendmail_enable` rc.conf(5) variable was made to take the value `NONE`.	450004
4.5-STABLE after moving to XFree86 4 by default for package builds.	450005

Release	__FreeBSD_version
4.5-STABLE after accept filtering was fixed so that is no longer susceptible to an easy DoS.	450006
4.6-RELEASE	460000
4.6-STABLE sendfile(2) fixed to comply with documentation, not to count any headers sent against the amount of data to be sent from the file.	460001
4.6.2-RELEASE	460002
4.6-STABLE	460100
4.6-STABLE after MFC of `sed -i'.	460101
4.6-STABLE after MFC of many new pkg_install features from the HEAD.	460102
4.7-RELEASE	470000
4.7-STABLE	470100
Start generated __std{in,out,err}p references rather than __sF. This changes std{in,out,err} from a compile time expression to a runtime one.	470101
4.7-STABLE after MFC of mbuf changes to replace m_aux mbufs by m_tag's	470102
4.7-STABLE gets OpenSSL 0.9.7	470103
4.8-RELEASE	480000
4.8-STABLE	480100
4.8-STABLE after realpath(3) has been made thread-safe	480101
4.8-STABLE 3ware API changes to twe.	480102
4.9-RELEASE	490000
4.9-STABLE	490100
4.9-STABLE after e_sid was added to struct kinfo_eproc.	490101
4.9-STABLE after MFC of libmap functionality for rtld.	490102
4.10-RELEASE	491000

Release	__FreeBSD_version
4.10-STABLE	491100
4.10-STABLE after MFC of revision 20040629 of the package tools	491101
4.10-STABLE after VM fix dealing with unwiring of fictitious pages	491102
4.11-RELEASE	492000
4.11-STABLE	492100
4.11-STABLE after adding libdata/ldconfig directories to mtree files.	492101
5.0-CURRENT	500000
5.0-CURRENT after adding addition ELF header fields, and changing our ELF binary branding method.	500001
5.0-CURRENT after kld metadata changes.	500002
5.0-CURRENT after buf/bio changes.	500003
5.0-CURRENT after binutils upgrade.	500004
5.0-CURRENT after merging libxpg4 code into libc and after TASKQ interface introduction.	500005
5.0-CURRENT after the addition of AGP interfaces.	500006
5.0-CURRENT after Perl upgrade to 5.6.0	500007
5.0-CURRENT after the update of KAME code to 2000/07 sources.	500008
5.0-CURRENT after ether_ifattach() and ether_ifdetach() changes.	500009
5.0-CURRENT after changing mtree defaults back to original variant, adding -L to follow symlinks.	500010
5.0-CURRENT after kqueue API changed.	500011
5.0-CURRENT after setproctitle(3) moved from libutil to libc.	500012
5.0-CURRENT after the first SMPng commit.	500013
5.0-CURRENT after <sys/select.h> moved to <sys/selinfo.h>.	500014

Release	__FreeBSD_version
5.0-CURRENT after combining libgcc.a and libgcc_r.a, and associated GCC linkage changes.	500015
5.0-CURRENT after change allowing libc and libc_r to be linked together, deprecating -pthread option.	500016
5.0-CURRENT after switch from struct ucred to struct xucred to stabilize kernel-exported API for mountd et al.	500017
5.0-CURRENT after addition of CPUTYPE make variable for controlling CPU-specific optimizations.	500018
5.0-CURRENT after moving machine/ioctl_fd.h to sys/fdcio.h	500019
5.0-CURRENT after locale names renaming.	500020
5.0-CURRENT after Bzip2 import. Also signifies removal of S/Key.	500021
5.0-CURRENT after SSE support.	500022
5.0-CURRENT after KSE Milestone 2.	500023
5.0-CURRENT after d_thread_t, and moving UUCP to ports.	500024
5.0-CURRENT after ABI change for descriptor and creds passing on 64 bit platforms.	500025
5.0-CURRENT after moving to XFree86 4 by default for package builds, and after the new libc strnstr() function was added.	500026
5.0-CURRENT after the new libc strcasestr() function was added.	500027
5.0-CURRENT after the userland components of smbfs were imported.	500028
5.0-CURRENT after the new C99 specific-width integer types were added.	(Not incremented.)
5.0-CURRENT after a change was made in the return value of sendfile(2).	500029

Release	__FreeBSD_version
5.0-CURRENT after the introduction of the type `fflags_t`, which is the appropriate size for file flags.	500030
5.0-CURRENT after the usb structure element rename.	500031
5.0-CURRENT after the introduction of Perl 5.6.1.	500032
5.0-CURRENT after the `sendmail_enable` rc.conf(5) variable was made to take the value `NONE`.	500033
5.0-CURRENT after mtx_init() grew a third argument.	500034
5.0-CURRENT with Gcc 3.1.	500035
5.0-CURRENT without Perl in /usr/src	500036
5.0-CURRENT after the addition of dlfunc(3)	500037
5.0-CURRENT after the types of some struct sockbuf members were changed and the structure was reordered.	500038
5.0-CURRENT after GCC 3.2.1 import. Also after headers stopped using _BSD_FOO_T_ and started using _FOO_T_DECLARED. This value can also be used as a conservative estimate of the start of bzip2(1) package support.	500039
5.0-CURRENT after various changes to disk functions were made in the name of removing dependency on disklabel structure internals.	500040
5.0-CURRENT after the addition of getopt_long(3) to libc.	500041
5.0-CURRENT after Binutils 2.13 upgrade, which included new FreeBSD emulation, vec, and output format.	500042
5.0-CURRENT after adding weak pthread_XXX stubs to libc, obsoleting libXThrStub.so. 5.0-RELEASE.	500043

Release	`__FreeBSD_version`
5.0-CURRENT after branching for RELENG_5_0	500100
<sys/dkstat.h> is empty and should not be included.	500101
5.0-CURRENT after the d_mmap_t interface change.	500102
5.0-CURRENT after taskqueue_swi changed to run without Giant, and taskqueue_swi_giant added to run with Giant.	500103
cdevsw_add() and cdevsw_remove() no longer exists. Appearance of MAJOR_AUTO allocation facility.	500104
5.0-CURRENT after new cdevsw initialization method.	500105
devstat_add_entry() has been replaced by devstat_new_entry()	500106
Devstat interface change; see sys/sys/param.h 1.149	500107
Token-Ring interface changes.	500108
Addition of vm_paddr_t.	500109
5.0-CURRENT after realpath(3) has been made thread-safe	500110
5.0-CURRENT after usbhid(3) has been synced with NetBSD	500111
5.0-CURRENT after new NSS implementation and addition of POSIX.1 getpw*_r, getgr*_r functions	500112
5.0-CURRENT after removal of the old rc system.	500113
5.1-RELEASE.	501000
5.1-CURRENT after branching for RELENG_5_1.	501100
5.1-CURRENT after correcting the semantics of sigtimedwait(2) and sigwaitinfo(2).	501101

Release	__FreeBSD_version
5.1-CURRENT after adding the lockfunc and lockfuncarg fields to bus_dma_tag_create(9).	501102
5.1-CURRENT after GCC 3.3.1-pre 20030711 snapshot integration.	501103
5.1-CURRENT 3ware API changes to twe.	501104
5.1-CURRENT dynamically-linked /bin and /sbin support and movement of libraries to /lib.	501105
5.1-CURRENT after adding kernel support for Coda 6.x.	501106
5.1-CURRENT after 16550 UART constants moved from <dev/sio/sioreg.h> to <dev/ic/ns16550.h> . Also when libmap functionality was unconditionally supported by rtld.	501107
5.1-CURRENT after PFIL_HOOKS API update	501108
5.1-CURRENT after adding kiconv(3)	501109
5.1-CURRENT after changing default operations for open and close in cdevsw	501110
5.1-CURRENT after changed layout of cdevsw	501111
5.1-CURRENT after adding kobj multiple inheritance	501112
5.1-CURRENT after the if_xname change in struct ifnet	501113
5.1-CURRENT after changing /bin and /sbin to be dynamically linked	501114
5.2-RELEASE	502000
5.2.1-RELEASE	502010
5.2-CURRENT after branching for RELENG_5_2	502100
5.2-CURRENT after __cxa_atexit/ __cxa_finalize functions were added to libc.	502101
5.2-CURRENT after change of default thread library from libc_r to libpthread.	502102

Release	__FreeBSD_version
5.2-CURRENT after device driver API megapatch.	502103
5.2-CURRENT after getopt_long_only() addition.	502104
5.2-CURRENT after NULL is made into ((void *)0) for C, creating more warnings.	502105
5.2-CURRENT after pf is linked to the build and install.	502106
5.2-CURRENT after time_t is changed to a 64-bit value on sparc64.	502107
5.2-CURRENT after Intel C/C++ compiler support in some headers and execve(2) changes to be more strictly conforming to POSIX.	502108
5.2-CURRENT after the introduction of the bus_alloc_resource_any API	502109
5.2-CURRENT after the addition of UTF-8 locales	502110
5.2-CURRENT after the removal of the getvfsent(3) API	502111
5.2-CURRENT after the addition of the .warning directive for make.	502112
5.2-CURRENT after ttyioctl() was made mandatory for serial drivers.	502113
5.2-CURRENT after import of the ALTQ framework.	502114
5.2-CURRENT after changing sema_timedwait(9) to return 0 on success and a non-zero error code on failure.	502115
5.2-CURRENT after changing kernel dev_t to be pointer to struct cdev *.	502116
5.2-CURRENT after changing kernel udev_t to dev_t.	502117
5.2-CURRENT after adding support for CLOCK_VIRTUAL and CLOCK_PROF to clock_gettime(2) and clock_getres(2).	502118

Release	__FreeBSD_version
5.2-CURRENT after changing network interface cloning overhaul.	502119
5.2-CURRENT after the update of the package tools to revision 20040629.	502120
5.2-CURRENT after marking Bluetooth code as non-i386 specific.	502121
5.2-CURRENT after the introduction of the KDB debugger framework, the conversion of DDB into a backend and the introduction of the GDB backend.	502122
5.2-CURRENT after change to make VFS_ROOT take a struct thread argument as does vflush. Struct kinfo_proc now has a user data pointer. The switch of the default X implementation to `xorg` was also made at this time.	502123
5.2-CURRENT after the change to separate the way ports rc.d and legacy scripts are started.	502124
5.2-CURRENT after the backout of the previous change.	502125
5.2-CURRENT after the removal of kmem_alloc_pageable() and the import of gcc 3.4.2.	502126
5.2-CURRENT after changing the UMA kernel API to allow ctors/inits to fail.	502127
5.2-CURRENT after the change of the vfs_mount signature as well as global replacement of PRISON_ROOT with SUSER_ALLOWJAIL for the suser(9) API.	502128
5.3-BETA/RC before the pfil API change	503000
5.3-RELEASE	503001
5.3-STABLE after branching for RELENG_5_3	503100
5.3-STABLE after addition of glibc style strftime(3) padding options.	503101
5.3-STABLE after OpenBSD's nc(1) import MFC.	503102

Release	`__FreeBSD_version`
5.4-PRERELEASE after the MFC of the fixes in `<src/include/stdbool.h>` and `<src/sys/i386/include/_types.h>` for using the GCC-compatibility of the Intel C/C++ compiler.	503103
5.4-PRERELEASE after the MFC of the change of ifi_epoch from wall clock time to uptime.	503104
5.4-PRERELEASE after the MFC of the fix of EOVERFLOW check in vswprintf(3).	503105
5.4-RELEASE.	504000
5.4-STABLE after branching for RELENG_5_4	504100
5.4-STABLE after increasing the default thread stacksizes	504101
5.4-STABLE after the addition of sha256	504102
5.4-STABLE after the MFC of if_bridge	504103
5.4-STABLE after the MFC of bsdiff and portsnap	504104
5.4-STABLE after MFC of ldconfig_local_dirs change.	504105
5.5-RELEASE.	505000
5.5-STABLE after branching for RELENG_5_5	505100
6.0-CURRENT	600000
6.0-CURRENT after permanently enabling PFIL_HOOKS in the kernel.	600001
6.0-CURRENT after initial addition of ifi_epoch to struct if_data. Backed out after a few days. Do not use this value.	600002
6.0-CURRENT after the re-addition of the ifi_epoch member of struct if_data.	600003
6.0-CURRENT after addition of the struct inpcb argument to the pfil API.	600004
6.0-CURRENT after addition of the "-d DESTDIR" argument to newsyslog.	600005
6.0-CURRENT after addition of glibc style strftime(3) padding options.	600006

Release	__FreeBSD_version
6.0-CURRENT after addition of 802.11 framework updates.	600007
6.0-CURRENT after changes to VOP_*VOBJECT() functions and introduction of MNTK_MPSAFE flag for Giantfree filesystems.	600008
6.0-CURRENT after addition of the cpufreq framework and drivers.	600009
6.0-CURRENT after importing OpenBSD's nc(1).	600010
6.0-CURRENT after removing semblance of SVID2 `matherr()` support.	600011
6.0-CURRENT after increase of default thread stacks' size.	600012
6.0-CURRENT after fixes in `<src/include/stdbool.h>` and `<src/sys/i386/include/_types.h>` for using the GCC-compatibility of the Intel C/C++ compiler.	600013
6.0-CURRENT after EOVERFLOW checks in vswprintf(3) fixed.	600014
6.0-CURRENT after changing the struct if_data member, ifi_epoch, from wall clock time to uptime.	600015
6.0-CURRENT after LC_CTYPE disk format changed.	600016
6.0-CURRENT after NLS catalogs disk format changed.	600017
6.0-CURRENT after LC_COLLATE disk format changed.	600018
Installation of acpica includes into /usr/include.	600019
Addition of MSG_NOSIGNAL flag to send(2) API.	600020
Addition of fields to cdevsw	600021
Removed gtar from base system.	600022

Release	__FreeBSD_version
LOCAL_CREDS, LOCAL_CONNWAIT socket options added to unix(4).	600023
hwpmc(4) and related tools added to 6.0-CURRENT.	600024
struct icmphdr added to 6.0-CURRENT.	600025
pf updated to 3.7.	600026
Kernel libalias and ng_nat introduced.	600027
POSIX ttyname_r(3) made available through unistd.h and libc.	600028
6.0-CURRENT after libpcap updated to v0.9.1 alpha 096.	600029
6.0-CURRENT after importing NetBSD's if_bridge(4).	600030
6.0-CURRENT after struct ifnet was broken out of the driver softcs.	600031
6.0-CURRENT after the import of libpcap v0.9.1.	600032
6.0-STABLE after bump of all shared library versions that had not been changed since RELENG_5.	600033
6.0-STABLE after credential argument is added to dev_clone event handler. 6.0-RELEASE.	600034
6.0-STABLE after 6.0-RELEASE	600100
6.0-STABLE after incorporating scripts from the local_startup directories into the base rcorder(8).	600101
6.0-STABLE after updating the ELF types and constants.	600102
6.0-STABLE after MFC of pidfile(3) API.	600103
6.0-STABLE after MFC of ldconfig_local_dirs change.	600104
6.0-STABLE after NLS catalog support of csh(1).	600105
6.1-RELEASE	601000

Release	__FreeBSD_version
6.1-STABLE after 6.1-RELEASE.	601100
6.1-STABLE after the import of csup.	601101
6.1-STABLE after the iwi(4) update.	601102
6.1-STABLE after the resolver update to BIND9, and exposure of reentrant version of netdb functions.	601103
6.1-STABLE after DSO (dynamic shared objects) support has been enabled in OpenSSL.	601104
6.1-STABLE after 802.11 fixups changed the api for the IEEE80211_IOC_STA_INFO ioctl.	601105
6.2-RELEASE	602000
6.2-STABLE after 6.2-RELEASE.	602100
6.2-STABLE after the addition of Wi-Spy quirk.	602101
6.2-STABLE after pci_find_extcap() addition.	602102
6.2-STABLE after MFC of dlsym change to look for a requested symbol both in specified dso and its implicit dependencies.	602103
6.2-STABLE after MFC of ng_deflate(4) and ng_pred1(4) netgraph nodes and new compression and encryption modes for ng_ppp(4) node.	602104
6.2-STABLE after MFC of BSD licensed version of gzip(1) ported from NetBSD.	602105
6.2-STABLE after MFC of PCI MSI and MSI-X support.	602106
6.2-STABLE after MFC of ncurses 5.6 and wide character support.	602107
6.2-STABLE after MFC of CAM 'SG' peripheral device, which implements a subset of Linux SCSI SG passthrough device API.	602108
6.2-STABLE after MFC of readline 5.2 patchset 002.	602109

Release	`__FreeBSD_version`
6.2-STABLE after MFC of pmap_invalidate_cache(), pmap_change_attr(), pmap_mapbios(), pmap_mapdev_attr(), and pmap_unmapbios() for amd64 and i386.	602110
6.2-STABLE after MFC of BOP_BDFLUSH and caused breakage of the filesystem modules KBI.	602111
6.2-STABLE after libutil(3) MFC's.	602112
6.2-STABLE after MFC of wide and single byte ctype separation. Newly compiled binary that references to ctype.h may require a new symbol, __mb_sb_limit, which is not available on older systems.	602113
6.2-STABLE after ctype ABI forward compatibility restored.	602114
6.2-STABLE after back out of wide and single byte ctype separation.	602115
6.3-RELEASE	603000
6.3-STABLE after 6.3-RELEASE.	603100
6.3-STABLE after fixing multibyte type support in bit macro.	603101
6.3-STABLE after adding l_sysid to struct flock.	603102
6.3-STABLE after MFC of the `memrchr` function.	603103
6.3-STABLE after MFC of support for `:u` variable modifier in make(1).	603104
6.4-RELEASE	604000
6.4-STABLE after 6.4-RELEASE.	604100
7.0-CURRENT.	700000
7.0-CURRENT after bump of all shared library versions that had not been changed since RELENG_5.	700001
7.0-CURRENT after credential argument is added to dev_clone event handler.	700002

Release	__FreeBSD_version
7.0-CURRENT after memmem(3) is added to libc.	700003
7.0-CURRENT after solisten(9) kernel arguments are modified to accept a backlog parameter.	700004
7.0-CURRENT after IFP2ENADDR() was changed to return a pointer to IF_LLADDR().	700005
7.0-CURRENT after addition of `if_addr` member to `struct ifnet` and IFP2ENADDR() removal.	700006
7.0-CURRENT after incorporating scripts from the local_startup directories into the base rcorder(8).	700007
7.0-CURRENT after removal of MNT_NODEV mount option.	700008
7.0-CURRENT after ELF-64 type changes and symbol versioning.	700009
7.0-CURRENT after addition of hostb and vgapci drivers, addition of pci_find_extcap(), and changing the AGP drivers to no longer map the aperture.	700010
7.0-CURRENT after tv_sec was made time_t on all platforms but Alpha.	700011
7.0-CURRENT after ldconfig_local_dirs change.	700012
7.0-CURRENT after changes to `/etc/rc.d/abi` to support `/compat/linux/etc/ld.so.cache` being a symlink in a readonly filesystem.	700013
7.0-CURRENT after pts import.	700014
7.0-CURRENT after the introduction of version 2 of hwpmc(4)'s ABI.	700015
7.0-CURRENT after addition of fcloseall(3) to libc.	700016
7.0-CURRENT after removal of ip6fw.	700017
7.0-CURRENT after import of snd_emu10kx.	700018

Release	__FreeBSD_version
7.0-CURRENT after import of OpenSSL 0.9.8b.	700019
7.0-CURRENT after addition of bus_dma_get_tag function	700020
7.0-CURRENT after libpcap 0.9.4 and tcpdump 3.9.4 import.	700021
7.0-CURRENT after dlsym change to look for a requested symbol both in specified dso and its implicit dependencies.	700022
7.0-CURRENT after adding new sound IOCTLs.	700023
7.0-CURRENT after import of OpenSSL 0.9.8d.	700024
7.0-CURRENT after the addition of libelf.	700025
7.0-CURRENT after major changes on sound sysctls.	700026
7.0-CURRENT after the addition of Wi-Spy quirk.	700027
7.0-CURRENT after the addition of sctp calls to libc	700028
7.0-CURRENT after the GNU gzip(1) implementation was replaced with a BSD licensed version ported from NetBSD.	700029
7.0-CURRENT after the removal of IPIP tunnel encapsulation (VIFF_TUNNEL) from the IPv4 multicast forwarding code.	700030
7.0-CURRENT after the modification of bus_setup_intr() (newbus).	700031
7.0-CURRENT after the inclusion of ipw(4) and iwi(4) firmwares.	700032
7.0-CURRENT after the inclusion of ncurses wide character support.	700033
7.0-CURRENT after changes to how insmntque(), getnewvnode(), and vfs_hash_insert() work.	700034

Release	__FreeBSD_version
7.0-CURRENT after addition of a notify mechanism for CPU frequency changes.	700035
7.0-CURRENT after import of the ZFS filesystem.	700036
7.0-CURRENT after addition of CAM 'SG' peripheral device, which implements a subset of Linux SCSI SG passthrough device API.	700037
7.0-CURRENT after changing getenv(3), putenv(3), setenv(3) and unsetenv(3) to be POSIX conformant.	700038
7.0-CURRENT after the changes in 700038 were backed out.	700039
7.0-CURRENT after the addition of flopen(3) to libutil.	700040
7.0-CURRENT after enabling symbol versioning, and changing the default thread library to libthr.	700041
7.0-CURRENT after the import of gcc 4.2.0.	700042
7.0-CURRENT after bump of all shared library versions that had not been changed since RELENG_6.	700043
7.0-CURRENT after changing the argument for vn_open()/VOP_OPEN() from filedescriptor index to the struct file *.	700044
7.0-CURRENT after changing pam_nologin(8) to provide an account management function instead of an authentication function to the PAM framework.	700045
7.0-CURRENT after updated 802.11 wireless support.	700046
7.0-CURRENT after adding TCP LRO interface capabilities.	700047
7.0-CURRENT after RFC 3678 API support added to the IPv4 stack. Legacy RFC 1724 behaviour of the IP_MULTICAST_IF ioctl has now been removed; 0.0.0.0/8 may no longer	700048

Release	`__FreeBSD_version`
be used to specify an interface index. struct ipmreqn should be used instead.	
7.0-CURRENT after importing pf from OpenBSD 4.1	700049
7.0-CURRENT after adding IPv6 support for FAST_IPSEC, deleting KAME IPSEC, and renaming FAST_IPSEC to IPSEC.	(not changed)
7.0-CURRENT after converting setenv/putenv/etc. calls from traditional BSD to POSIX.	700050
7.0-CURRENT after adding new mmap/lseek/etc syscalls.	700051
7.0-CURRENT after moving I4B headers to include/i4b.	700052
7.0-CURRENT after the addition of support for PCI domains	700053
7.0-CURRENT after MFC of wide and single byte ctype separation.	700054
7.0-RELEASE, and 7.0-CURRENT after ABI backwards compatibility to the FreeBSD 4/5/6 versions of the PCIOCGETCONF, PCIOCREAD and PCIOCWRITE IOCTLs was MFC'ed, which required the ABI of the PCIOCGETCONF IOCTL to be broken again	700055
7.0-STABLE after 7.0-RELEASE	700100
7.0-STABLE after the MFC of m_collapse().	700101
7.0-STABLE after the MFC of kdb_enter_why().	700102
7.0-STABLE after adding l_sysid to struct flock.	700103
7.0-STABLE after the MFC of procstat(1).	700104
7.0-STABLE after the MFC of umtx features.	700105
7.0-STABLE after the MFC of write(2) support to psm(4).	700106
7.0-STABLE after the MFC of F_DUP2FD command to fcntl(2).	700107

Release	__FreeBSD_version
7.0-STABLE after some lockmgr(9) changes, which makes it necessary to include sys/lock.h in order to use lockmgr(9).	700108
7.0-STABLE after MFC of the memrchr function.	700109
7.0-STABLE after MFC of kernel NFS lockd client.	700110
7.0-STABLE after addition of physically contiguous jumbo frame support.	700111
7.0-STABLE after MFC of kernel DTrace support.	700112
7.1-RELEASE	701000
7.1-STABLE after 7.1-RELEASE.	701100
8.0-CURRENT. Separating wide and single byte ctype.	800000
8.0-CURRENT after libpcap 0.9.8 and tcpdump 3.9.8 import.	800001
8.0-CURRENT after renaming kthread_create() and friends to kproc_create() etc.	800002
8.0-CURRENT after ABI backwards compatibility to the FreeBSD 4/5/6 versions of the PCIOCGETCONF, PCIOCREAD and PCIOCWRITE IOCTLs was added, which required the ABI of the PCIOCGETCONF IOCTL to be broken again	800003
8.0-CURRENT after agp(4) driver moved from src/sys/pci to src/sys/dev/agp	800004
8.0-CURRENT after changes to the jumbo frame allocator.	800005
8.0-CURRENT after the addition of callgraph capture functionality to hwpmc(4).	800006
8.0-CURRENT after kdb_enter() gains a "why" argument.	800007
8.0-CURRENT after LK_EXCLUPGRADE option removal.	800008

Release	__FreeBSD_version
8.0-CURRENT after introduction of lockmgr_disown(9)	800009
8.0-CURRENT after the vn_lock(9) prototype change.	800010
8.0-CURRENT after the VOP_LOCK(9) and VOP_UNLOCK(9) prototype changes.	800011
8.0-CURRENT after introduction of lockmgr_recursed(9), BUF_RECURSED(9) and BUF_ISLOCKED(9) and the removal of BUF_REFCNT() .	800012
8.0-CURRENT after introduction of the "ASCII" encoding.	800013
8.0-CURRENT after changing the prototype of lockmgr(9) and removal of lockcount() and LOCKMGR_ASSERT() .	800014
8.0-CURRENT after extending the types of the fts(3) structures.	800015
8.0-CURRENT after adding an argumentt to MEXTADD(9)	800016
8.0-CURRENT after the introduction of LK_NODUP and LK_NOWITNESS options in the lockmgr(9) space.	800017
8.0-CURRENT after the addition of m_collapse.	800018
8.0-CURRENT after the addition of current working directory, root directory, and jail directory support to the kern.proc.filedesc sysctl.	800019
8.0-CURRENT after introduction of lockmgr_assert(9) and BUF_ASSERT functions.	800020
8.0-CURRENT after introduction of lockmgr_args(9) and LK_INTERNAL flag removal.	800021
8.0-CURRENT after changing the default system ar to BSD ar(1).	800022

Release	__FreeBSD_version
8.0-CURRENT after changing the prototypes of lockstatus(9) and VOP_ISLOCKED(9), more specifically retiring the `struct thread` argument.	800023
8.0-CURRENT after axing out the `lockwaiters` and `BUF_LOCKWAITERS` functions, changing the return value fo `brelvp` from void to int and introducing new flags for lockinit(9).	800024
8.0-CURRENT after adding F_DUP2FD command to fcntl(2).	800025
8.0-CURRENT after changing the priority parameter to cv_broadcastpri such that 0 means no priority.	800026
8.0-CURRENT after changing the bpf monitoring ABI when zerocopy bpf buffers were added.	800027
8.0-CURRENT after adding l_sysid to struct flock.	800028
8.0-CURRENT after reintegration of the `BUF_LOCKWAITERS` function and the addition of lockmgr_waiters(9).	800029
8.0-CURRENT after the introduction of the rw_try_rlock(9) and rw_try_wlock(9) functions.	800030
8.0-CURRENT after the introduction of the `lockmgr_rw` and `lockmgr_args_rw` functions.	800031
8.0-CURRENT after the implementation of the openat and related syscalls, introduction of the O_EXEC flag for the open(2), and providing the corresponding linux compatibility syscalls.	800032
8.0-CURRENT after added write(2) support for psm(4) in native operation level. Now arbitrary commands can be written to `/dev/psm%d` and status can be read back from it.	800033

Release	`__FreeBSD_version`
8.0-CURRENT after introduction of the `memrchr` function.	800034
8.0-CURRENT after introduction of the `fdopendir` function.	800035
8.0-CURRENT after switchover of 802.11 wireless to multi-bss support (aka vaps).	800036
8.0-CURRENT after addition of multi routing table support (a.k.a. setfib(1), setfib(2)).	800037
8.0-CURRENT after removal of netatm and ISDN4BSD.	800038
8.0-CURRENT after removal of sgtty.	800039
8.0-CURRENT with kernel NFS lockd client.	800040
8.0-CURRENT after addition of arc4random_buf(3) and arc4random_uniform(3).	800041
8.0-CURRENT after addition of cpuctl(4).	800042
8.0-CURRENT after changing bpf(4) to use a single device node, instead of device cloning.	800043
8.0-CURRENT after the commit of the first step of the vimage project renaming global variables to be virtualized with a V_ prefix with macros to map them back to their global names.	800044
8.0-CURRENT after the integration of the MPSAFE TTY layer, including changes to various drivers and utilities that interact with it.	800045
8.0-CURRENT after the separation of the GDT per CPU on amd64 architecture.	800046
8.0-CURRENT after removal of VSVTX, VSGID and VSUID.	800047
8.0-CURRENT after converting the kernel NFS mount code to accept individual mount options in the nmount() iovec, not just one big struct nfs_args.	800048

Release	__FreeBSD_version
8.0-CURRENT after the removal of suser(9) and suser_cred(9).	800049

##

Note that 2.2-STABLE sometimes identifies itself as "2.2.5-STABLE" after the 2.2.5-RELEASE. The pattern used to be year followed by the month, but we decided to change it to a more straightforward major/minor system starting from 2.2. This is because the parallel development on several branches made it infeasible to classify the releases simply by their real release dates. If you are making a port now, you do not have to worry about old -CURRENTs; they are listed here just for your reference.

12.6. Writing something after bsd.port.mk

Do not write anything after the .include <bsd.port.mk> line. It usually can be avoided by including bsd.port.pre.mk somewhere in the middle of your Makefile and bsd.port.post.mk at the end.

##

You need to include either the bsd.port.pre.mk / bsd.port.post.mk pair or bsd.port.mk only; do not mix these two usages.

bsd.port.pre.mk only defines a few variables, which can be used in tests in the Makefile, bsd.port.post.mk defines the rest.

Here are some important variables defined in bsd.port.pre.mk (this is not the complete list, please read bsd.port.mk for the complete list).

Variable	Description
ARCH	The architecture as returned by uname -m (e.g., i386)
OPSYS	The operating system type, as returned by uname -s (e.g., FreeBSD)

Variable	Description
OSREL	The release version of the operating system (e.g., 2.1.5 or 2.2.7)
OSVERSION	The numeric version of the operating system; the same as __FreeBSD_version .
PORTOBJFORMAT	The object format of the system (elf or aout; note that for "modern" versions of FreeBSD, aout is deprecated.)
LOCALBASE	The base of the "local" tree (e.g., /usr/local/)
PREFIX	Where the port installs itself (see more on PREFIX).

##

If you have to define the variables USE_IMAKE , USE_X_PREFIX , or MASTERDIR , do so before including bsd.port.pre.mk .

Here are some examples of things you can write after bsd.port.pre.mk :

```
# no need to compile lang/perl5 if perl5 is already in system
.if ${OSVERSION} > 300003
BROKEN= perl is in system
.endif

# only one shlib version number for ELF
.if ${PORTOBJFORMAT} == "elf"
TCL_LIB_FILE=  ${TCL_LIB}.${SHLIB_MAJOR}
.else
TCL_LIB_FILE=  ${TCL_LIB}.${SHLIB_MAJOR}.${SHLIB_MINOR}
.endif

# software already makes link for ELF, but not for a.out
post-install:
.if ${PORTOBJFORMAT} == "aout"
       ${LN} -sf liblinpack.so.1.0 ${PREFIX}/lib/liblinpack.so
.endif
```

You did remember to use tab instead of spaces after BROKEN= and TCL_LIB_FILE= , did you not? :-).

12.7. # wrapper scripts ### exec

port ############## shell script# ######### script ############## script ## exec ## #statement#######

```
#!/bin/sh
exec %%LOCALBASE%%/bin/java -jar %%DATADIR%%/foo.jar "$@"
```

exec ############# shell #### ### exec #### shell ## ############################

12.8. UIDs # GIDs

The current list of reserved UIDs and GIDs can be found in `ports/UIDs` and `ports/GIDs`.

If your port requires a certain user to be on the installed system, let the `pkg-install` script call `pw` to create it automatically. Look at net/cvsup-mirror for an example. Please note that this is strongly discouraged, please register user/group ID numbers as stated below.

If your port must use the same user/group ID number when it is installed as a binary package as when it was compiled, then you must choose a free UID from 50 to 999 and register it either in `ports/UIDs` (for users) or in `ports/GIDs` (for groups). Look at japanese/Wnn6 for an example.

Make sure you do not use a UID already used by the system or other ports.

Please include a patch against these two files when you require a new user or group to be created for your port.

12.9. Do things rationally

The `Makefile` should do things simply and reasonably. If you can make it a couple of lines shorter or more readable, then do so. Examples include using a make `.if` construct instead of a shell `if` construct, not redefining `do-extract` if you can redefine `EXTRACT*` instead, and using `GNU_CONFIGURE` instead of `CONFIGURE_ARGS += --prefix= ${PREFIX}`.

If you find yourself having to write a lot of new code to try to do something, please go back and review `bsd.port.mk` to see if it contains an existing implementation of what you are trying to do. While hard to read, there are a great many seemingly-hard problems for which `bsd.port.mk` already provides a shorthand solution.

12.10. Respect both CC and CXX

The port should respect both CC and CXX variables. What we mean by this is that the port should not set the values of these variables absolutely, overriding existing values; instead, it should append whatever values it needs to the existing values. This is so that build options that affect all ports can be set globally.

If the port does not respect these variables, please add NO_PACKAGE=ignores either cc or cxx to the Makefile.

An example of a Makefile respecting both CC and CXX variables follows. Note the ?=:

```
CC?= gcc
```

```
CXX?= g++
```

Here is an example which respects neither CC nor CXX variables:

```
CC= gcc
```

```
CXX= g++
```

Both CC and CXX variables can be defined on FreeBSD systems in /etc/make.conf. The first example defines a value if it was not previously set in /etc/make.conf, preserving any system-wide definitions. The second example clobbers anything previously defined.

12.11. Respect CFLAGS

The port should respect the CFLAGS variable. What we mean by this is that the port should not set the value of this variable absolutely, overriding the existing value; instead, it should append whatever values it needs to the existing value. This is so that build options that affect all ports can be set globally.

If it does not, please add NO_PACKAGE=ignores cflags to the Makefile.

An example of a Makefile respecting the CFLAGS variable follows. Note the +=:

```
CFLAGS+= -Wall -Werror
```

Here is an example which does not respect the CFLAGS variable:

```
CFLAGS= -Wall -Werror
```

The CFLAGS variable is defined on FreeBSD systems in /etc/make.conf. The first example appends additional flags to the CFLAGS variable, preserving any system-wide definitions. The second example clobbers anything previously defined.

You should remove optimization flags from the third party Makefiles. System CFLAGS contains system-wide optimization flags. An example from an unmodified Makefile:

```
CFLAGS= -O3 -funroll-loops -DHAVE_SOUND
```

Using system optimization flags, the `Makefile` would look similar to the following example:

```
CFLAGS+= -DHAVE_SOUND
```

12.12. Threading libraries

The threading library must be linked to the binaries using a special linker flag `-pthread` on FreeBSD. If a port insists on linking `-lpthread` or `-lc_r` directly, patch it to use `PTHREAD_LIBS` variable provided by the ports framework. This variable usually has the value of `-pthread`, but on certain architectures and FreeBSD versions it can have different values, so do not just hardcode `-pthread` into patches and always use `PTHREAD_LIBS`.

##

If building the port errors out with `unrecognized option '-pthread'` when setting `PTHREAD_LIBS`, it may be desirable to use `gcc` as linker by setting `CONFIGURE_ENV` to `LD=${CC}`. The `-pthread` option is not supported by `ld` directly.

12.13. Feedback

Do send applicable changes/patches to the original author/maintainer for inclusion in next release of the code. This will only make your job that much easier for the next release.

12.14. `README.html`

Do not include the `README.html` file. This file is not part of the cvs collection but is generated using the `make readme` command.

12.15. Marking a port not installable with `BROKEN`, `FORBIDDEN`, or `IGNORE`

In certain cases users should be prevented from installing a port. To tell a user that a port should not be installed, there are several `make` variables that can be used in a port's `Makefile`. The value of the following `make` variables will be the reason that is given back to users for why the port refuses to install itself. Please use the correct `make` variable as

each make variable conveys radically different meanings to both users, and to automated systems that depend on the `Makefile`s, such as the ports build cluster, FreshPorts, and portsmon.

12.15.1. Variables

- `BROKEN` is reserved for ports that currently do not compile, install, or deinstall correctly. It should be used for ports where the problem is believed to be temporary.

  If instructed, the build cluster will still attempt to try to build them to see if the underlying problem has been resolved. (However, in general, the cluster is run without this.)

  For instance, use `BROKEN` when a port:

  - does not compile
  - fails its configuration or installation process
  - installs files outside of `${LOCALBASE}`
  - does not remove all its files cleanly upon deinstall (however, it may be acceptable, and desirable, for the port to leave user-modified files behind)

- `FORBIDDEN` is used for ports that do contain a security vulnerability or induce grave concern regarding the security of a FreeBSD system with a given port installed (ex: a reputably insecure program or a program that provides easily exploitable services). Ports should be marked as `FORBIDDEN` as soon as a particular piece of software has a vulnerability and there is no released upgrade. Ideally ports should be upgraded as soon as possible when a security vulnerability is discovered so as to reduce the number of vulnerable FreeBSD hosts (we like being known for being secure), however sometimes there is a noticeable time gap between disclosure of a vulnerability and an updated release of the vulnerable software. Do not mark a port `FORBIDDEN` for any reason other than security.

- `IGNORE` is reserved for ports that should not be built for some other reason. It should be used for ports where the problem is believed to be structural. The build cluster will not, under any circumstances, build ports marked as `IGNORE`. For instance, use `IGNORE` when a port:

  - compiles but does not run properly
  - does not work on the installed version of FreeBSD
  - requires FreeBSD kernel sources to build, but the user does not have them installed
  - has a distfile which may not be automatically fetched due to licensing restrictions

- does not work with some other currently installed port (for instance, the port depends on www/apache21 but www/apache13 is installed)

##

If a port would conflict with a currently installed port (for example, if they install a file in the same place that perfoms a different function), use `CONFLICTS` instead. `CONFLICTS` will set `IGNORE` by itself.

- If a port should be marked `IGNORE` only on certain architectures, there are two other convenience variables that will automatically set `IGNORE` for you: `ONLY_FOR_ARCHS` and `NOT_FOR_ARCHS` . Examples:

```
ONLY_FOR_ARCHS= i386 amd64
```

```
NOT_FOR_ARCHS= alpha ia64 sparc64
```

A custom `IGNORE` message can be set using `ONLY_FOR_ARCHS_REASON` and `NOT_FOR_ARCHS_REASON` . Per architecture entries are possible with `ONLY_FOR_ARCHS_REASON_` *ARCH* and `NOT_FOR_ARCHS_REASON_` *ARCH*.

- If a port fetches i386 binaries and installs them, `IA32_BINARY_PORT` should be set. If this variable is set, it will be checked whether the `/usr/lib32` directory is available for IA32 versions of libraries and whether the kernel has IA32 compatibility compiled in. If one of these two dependencies is not satisfied, `IGNORE` will be set automatically.

12.15.2. Implementation Notes

The strings should not be quoted. Also, the wording of the string should be somewhat different due to the way the information is shown to the user. Examples:

```
BROKEN= this port is unsupported on FreeBSD 5.x
```

```
IGNORE= is unsupported on FreeBSD 5.x
```

resulting in the following output from `make describe` :

```
===>  foobar-0.1 is marked as broken: this port is unsupported on ↺
FreeBSD 5.x.
```

```
===>  foobar-0.1 is unsupported on FreeBSD 5.x.
```

12.16. Marking a port for removal with `DEPRECATED` or `EXPIRATION_DATE`

Do remember that `BROKEN` and `FORBIDDEN` are to be used as a temporary resort if a port is not working. Permanently broken ports should be removed from the tree entirely.

When it makes sense to do so, users can be warned about a pending port removal with `DEPRECATED` and `EXPIRATION_DATE`. The former is simply a string stating why the port is scheduled for removal; the latter is a string in ISO 8601 format (YYYY-MM-DD). Both will be shown to the user.

It is possible to set `DEPRECATED` without an `EXPIRATION_DATE` (for instance, recommending a newer version of the port), but the converse does not make any sense.

There is no set policy on how much notice to give. Current practice seems to be one month for security-related issues and two months for build issues. This also gives any interested committers a little time to fix the problems.

12.17. Avoid use of the `.error` construct

The correct way for a `Makefile` to signal that the port can not be installed due to some external factor (for instance, the user has specified an illegal combination of build options) is to set a nonblank value to `IGNORE`. This value will be formatted and shown to the user by `make install`.

It is a common mistake to use `.error` for this purpose. The problem with this is that many automated tools that work with the ports tree will fail in this situation. The most common occurrence of this is seen when trying to build `/usr/ports/INDEX` (see # 9.1, "Running `make describe`"). However, even more trivial commands such as `make -V maintainer` also fail in this scenario. This is not acceptable.

12.1. How to avoid using `.error`

Assume that someone has the line

```
USE_POINTYHAT=yes
```

in `make.conf`. The first of the next two `Makefile` snippets will cause `make index` to fail, while the second one will not:

```
.if USE_POINTYHAT
.error "POINTYHAT is not supported"
.endif
```

```
.if USE_POINTYHAT
IGNORE=POINTYHAT is not supported
.endif
```

12.18. sysctl

sysctl ### targets ########## ###### makevar ######################## make index ########## sysctl #

sysctl(8) ##### SYSCTL ######################

12.19. Rerolling distfiles

Sometimes the authors of software change the content of released distfiles without changing the file's name. You have to verify that the changes are official and have been performed by the author. It has happened in the past that the distfile was silently altered on the download servers with the intent to cause harm or compromise end user security.

Put the old distfile aside, download the new one, unpack them and compare the content with diff(1). If you see nothing suspicious, you can update `distinfo`. Be sure to summarize the differences in your PR or commit log, so that other people know that you have taken care to ensure that nothing bad has happened.

You might also want to contact the authors of the software and confirm the changes with them.

12.20. Necessary workarounds

Sometimes it is necessary to work around bugs in software included with older versions of FreeBSD.

- Some versions of make(1) were broken on at least 4.8 and 5.0 with respect to handling comparisons based on `OSVERSION`. This would often lead to failures during `make describe` (and thus, the overall ports `make index`). The workaround is to enclose the conditional comparison in spaces, e.g.:

  ```
  if ( ${OSVERSION} > 500023 )
  ```

  Be aware that test-installing a port on 4.9 or 5.2 will *not* detect this problem.

12.21. Miscellanea

The files `pkg-descr` and `pkg-plist` should each be double-checked. If you are reviewing a port and feel they can be worded better, do so.

Do not copy more copies of the GNU General Public License into our system, please.

Please be careful to note any legal issues! Do not let us illegally distribute software!

13. A Sample `Makefile`

Here is a sample `Makefile` that you can use to create a new port. Make sure you remove all the extra comments (ones between brackets)!

It is recommended that you follow this format (ordering of variables, empty lines between sections, etc.). This format is designed so that the most important information is easy to locate. We recommend that you use portlint to check the `Makefile`.

```
[the header...just to make it easier for us to identify the ports.-]
# New ports collection makefile for:   xdvi
[the "version required" line is only needed when the PORTVERSION
 variable is not specific enough to describe the port.-]
# Date created:               26 May 1995
[this is the person who did the original port to FreeBSD, in ↺
particular, the
person who wrote the first version of this Makefile.  Remember, ↺
this should
not be changed when upgrading the port later.-]
# Whom:                       Satoshi Asami <asami@FreeBSD.org>
#
# $FreeBSD$
[ ^^^^^^^^^ This will be automatically replaced with RCS ID string ↺
by CVS
when it is committed to our repository.  If upgrading a port, do ↺
not alter
this line back to "$FreeBSD$".  CVS deals with it automatically.-]
#

[section to describe the port itself and the master site - PORTNAME
 and PORTVERSION are always first, followed by CATEGORIES,
 and then MASTER_SITES, which can be followed by MASTER_SITE_SUBDIR.
 PKGNAMEPREFIX and PKGNAMESUFFIX, if needed, will be after that.
 Then comes DISTNAME, EXTRACT_SUFX and/or DISTFILES, and then
 EXTRACT_ONLY, as necessary.-]
PORTNAME=      xdvi
PORTVERSION=   18.2
CATEGORIES=    print
[do not forget the trailing slash ("/")!
 if you are not using MASTER_SITE_* macros]
MASTER_SITES=  ${MASTER_SITE_XCONTRIB}
MASTER_SITE_SUBDIR= applications
PKGNAMEPREFIX= ja-
DISTNAME=      xdvi-pl18
[set this if the source is not in the standard ".tar.gz" form]
EXTRACT_SUFX=  .tar.Z

[section for distributed patches -- can be empty]
PATCH_SITES=   ftp://ftp.sra.co.jp/pub/X11/japanese/
PATCHFILES=    xdvi-18.patch1.gz xdvi-18.patch2.gz
```

```
[maintainer; *mandatory*!  This is the person who is volunteering to
 handle port updates, build breakages, and to whom a users can direct
 questions and bug reports.  To keep the quality of the Ports ↺
Collection
 as high as possible, we no longer accept new ports that are ↺
assigned to
 "ports@FreeBSD.org".-]
MAINTAINER=    asami@FreeBSD.org
COMMENT=       A DVI Previewer for the X Window System

[dependencies -- can be empty]
RUN_DEPENDS=   gs:${PORTSDIR}/print/ghostscript
LIB_DEPENDS=   Xpm.5:${PORTSDIR}/graphics/xpm

[this section is for other standard bsd.port.mk variables that do not
 belong to any of the above]
[If it asks questions during configure, build, install...-]
IS_INTERACTIVE=        yes
[If it extracts to a directory other than ${DISTNAME}...-]
WRKSRC=                ${WRKDIR}/xdvi-new
[If the distributed patches were not made relative to ${WRKSRC}, you
 may need to tweak this]
PATCH_DIST_STRIP=      -p1
[If it requires a "configure" script generated by GNU autoconf to ↺
be run]
GNU_CONFIGURE= yes
[If it requires GNU make, not /usr/bin/make, to build...-]
USE_GMAKE=     yes
[If it is an X application and requires "xmkmf -a" to be run...-]
USE_IMAKE=     yes
[et cetera.-]

[non-standard variables to be used in the rules below]
MY_FAVORITE_RESPONSE=  "yeah, right"

[then the special rules, in the order they are called]
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

[and then the epilogue]
.include <bsd.port.mk>
```

14. Keeping Up

The FreeBSD Ports Collection is constantly changing. Here is some information on how to keep up.

14.1. FreshPorts

One of the easiest ways to learn about updates that have already been committed is by subscribing to FreshPorts. You can select multiple ports to monitor. Maintainers are strongly encouraged to subscribe, because they will receive notification of not only their own changes, but also any changes that any other FreeBSD committer has made. (These are often necessary to keep up with changes in the underlying ports framework—— although it would be most polite to receive an advance heads-up from those committing such changes, sometimes this is overlooked or just simply impractical. Also, in some cases, the changes are very minor in nature. We expect everyone to use their best judgement in these cases.)

If you wish to use FreshPorts, all you need is an account. If your registered email address is `@FreeBSD.org` , you will see the opt-in link on the right hand side of the webpages. For those of you who already have a FreshPorts account, but are not using your `@FreeBSD.org` email address, just change your email to `@FreeBSD.org` , subscribe, then change it back again.

FreshPorts also has a sanity test feature which automatically tests each commit to the FreeBSD ports tree. If subscribed to this service, you will be notified of any errors which FreshPorts detects during sanity testing of your commits.

14.2. The Web Interface to the Source Repository

It is possible to browse the files in the source repository by using a web interface. Changes that affect the entire port system are now documented in the CHANGES file. Changes that affect individual ports are now documented in the UPDATING file. However, the definitive answer to any question is undoubtedly to read the source code of bsd.port.mk, and associated files.

14.3. The FreeBSD Ports Mailing List

If you maintain ports, you should consider following the FreeBSD ports ####. Important changes to the way ports work will be announced there, and then committed to `CHANGES` .

14.4. The FreeBSD Port Building Cluster on pointyhat.FreeBSD.org

One of the least-publicized strengths of FreeBSD is that an entire cluster of machines is dedicated to continually building the Ports Collection, for each of the major OS releases and for each Tier-1 architecture. You can find the results of these builds at package building logs and errors.

Individual ports are built unless they are specifically marked with `IGNORE`. Ports that are marked with `BROKEN` will still be attempted, to see if the underlying problem has been resolved. (This is done by passing `TRYBROKEN` to the port's `Makefile`.)

14.5. The FreeBSD Port Distfile Survey

The build cluster is dedicated to building the latest release of each port with distfiles that have already been fetched. However, as the Internet continually changes, distfiles can quickly go missing. The FreeBSD Ports distfiles survey attempts to query every download site for every port to find out if each distfile is still currently available. Maintainers are asked to check this report periodically, not only to speed up the building process for users, but to help avoid wasting bandwidth of the sites that volunteer to host all these distfiles.

14.6. The FreeBSD Ports Monitoring System

Another handy resource is the FreeBSD Ports Monitoring System (also known as `portsmon`). This system comprises a database that processes information from several sources and allows its to be browsed via a web interface. Currently, the ports Problem Reports (PRs), the error logs from the build cluster, and individual files from the ports collection are used. In the future, this will be expanded to include the distfile survey, as well as other sources.

To get started, you can view all information about a particular port by using the Overview of One Port.

As of this writing, this is the only resource available that maps GNATS PR entries to portnames. (PR submitters do not always include the portname in their Synopsis, although we would prefer that they did.) So, `portsmon` is a good place to start if you want to find out whether an existing port has any PRs filed against it and/or any build errors; or, to find out if a new port that you may be thinking about creating has already been submitted.