

# Руководство FreeBSD по созданию портов



# Содержание

1. Введение .....	10
2. Как самому сделать новый порт .....	11
3. Быстрое портирование .....	12
3.1. Создание файла Makefile .....	12
3.2. Создание информационных файлов .....	13
3.3. Создание файла с контрольной суммой .....	15
3.4. Тестирование порта .....	15
3.5. Проверка вашего порта утилитой <code>portlint</code> .....	15
3.6. Посылка нового порта .....	16
4. Медленное портирование .....	18
4.1. Как всё это работает .....	18
4.2. Получение исходного кода .....	19
4.3. Модификация порта .....	20
4.4. Работа с патчами .....	20
4.5. Конфигурирование .....	25
4.6. Обработка пользовательского ввода .....	25
5. Настройка Makefile .....	26
5.1. Оригинальный исходный код .....	26
5.2. Именованье .....	26
5.3. Категоризация .....	36
5.4. Файлы дистрибутива .....	45
5.5. <code>MAINTAINER</code> .....	71
5.6. <code>COMMENT</code> .....	73
5.7. Веб-сайт проекта .....	73
5.8. Лицензии .....	74
5.9. <code>PORTSCOUT</code> .....	86
5.10. Зависимости .....	87
5.11. Подчиненные порты и <code>MASTERDIR</code> .....	94
5.12. Страницы Справочника .....	95
5.13. Файлы информации .....	95
5.14. Параметры Makefile .....	96
5.15. Указание рабочего каталога .....	116
5.16. Обработка конфликтов .....	117
5.17. Установка файлов .....	119
5.18. Используйте <code>BINARY_ALIAS</code> для переименования команд вместо исправления сборки .....	123
6. Особые соглашения .....	125
6.1. Разделение длинных файлов .....	125

6.2. Staging	125
6.3. Встроенные библиотеки	127
6.4. Динамические библиотеки	128
6.5. Порты с ограничениями на распространение или с правовым обременением	129
6.6. Механизмы построения	131
6.7. Использование GNU Autotools	149
6.8. Использование GNU <code>gettext</code>	149
6.9. Использование Perl	151
6.10. Использование X11	154
6.11. Использование GNOME	156
6.12. Компоненты GNOME	158
6.13. Использование Qt	164
6.14. Использование KDE	170
6.15. Использование LXQt	179
6.16. Использование Java	179
6.17. Веб-приложения, Apache и PHP	183
6.18. Использование Python	186
6.19. Использование Tcl/Tk	189
6.20. Использование SDL	190
6.21. Использование wxWidgets	191
6.22. Использование Lua	195
6.23. Использование Guile	199
6.24. Использование <code>iconv</code>	203
6.25. Использование Xfce	205
6.26. Использование Budgie	207
6.27. Использование баз данных	207
6.28. Запуск и остановка служб (скрипты <code>rc</code> )	208
6.29. Добавление пользователей и групп	212
6.30. Порты, зависящие от исходных кодов ядра	212
6.31. Библиотеки Go	212
6.32. Библиотеки Haskell	213
6.33. Файлы завершения командной оболочки	213
7. Флейворы	214
7.1. Введение в флейворы (Flavors)	214
7.2. Использование FLAVORS	214
7.3. <code>USES=php</code> и флейворы	216
7.4. <code>USES=python</code> и флейворы	218
7.5. <code>USES=lua</code> и флейворы	219
7.6. <code>USES=guile</code> и флейворы	219
8. Продвинутое использование pkg-plist	220
8.1. Изменение содержимого pkg-plist в зависимости от make-переменных	220

8.2. Пустые каталоги	221
8.3. Файлы конфигурации	222
8.4. Динамический или статический список упаковки	223
8.5. Автоматическое создание списка упаковки	223
8.6. Расширение списка пакетов используя ключевые слова	225
9. pkg-*	233
9.1. pkg-message	233
9.2. pkg-install, pkg-pre-install и pkg-post-install	236
9.3. pkg-deinstall, pkg-pre-deinstall и pkg-post-deinstall	236
9.4. Изменение имён файлов pkg-*	237
9.5. Использование <code>SUB_FILES</code> и <code>SUB_LIST</code>	237
10. Тестирование порта	239
10.1. Запуск <code>make describe</code>	239
10.2. Запуск <code>make test</code>	239
10.3. Portclippy / Portfmt	240
10.4. Portlint	240
10.5. Инструменты для работы с портами	240
10.6. <code>PREFIX</code> и <code>DESTDIR</code>	240
10.7. poudriere	242
10.8. Отладка портов	250
11. Обновление отдельного порта	252
11.1. Использование Git для создания патчей	253
11.2. <code>UPDATING</code> и <code>MOVED</code>	255
12. Безопасность	258
12.1. Почему безопасность так важна	258
12.2. Исправление уязвимостей безопасности	258
12.3. Обеспечение сообщества информацией	259
13. Что делать нужно, и что делать нельзя	265
13.1. Введение	265
13.2. <code>WRKDIR</code>	265
13.3. <code>WRKDIRPREFIX</code>	265
13.4. Различение операционных систем и версий ОС	265
13.5. Написание чего-либо после <code>bsd.port.mk</code>	266
13.6. Использование выражения <code>exec</code> в сценариях обёртках	267
13.7. Поступайте разумно	267
13.8. Относитесь внимательно как к <code>CC</code> , так и <code>CXX</code>	267
13.9. Относитесь внимательно к <code>CFLAGS</code>	268
13.10. Подробные логи сборки	269
13.11. Обратная связь	269
13.12. <code>README.html</code>	269
13.13. Пометка неустанавливаемого порта как <code>BROKEN</code> , <code>FORBIDDEN</code> или <code>IGNORE</code>	270

13.14. Архитектурные соображения	271
13.15. Пометка порта на удаление с <code>DEPRECATED</code> или <code>EXPIRATION_DATE</code>	273
13.16. Избегайте использования конструкции <code>.err</code>	273
13.17. Использование <code>sysctl</code>	274
13.18. Меняющиеся дистрибутивные файлы	274
13.19. Используйте стандарты POSIX	274
13.20. Разное	275
14. Примерный Makefile	276
15. Порядок переменных в Makefile портов	279
15.1. Блок <code>PORTNAME</code>	279
15.2. Блок <code>PATCHFILES</code>	280
15.3. Блок <code>MAINTAINER</code>	280
15.4. Блок <code>LICENSE</code>	280
15.5. Общие сообщения <code>BROKEN/IGNORE/DEPRECATED</code>	280
15.6. Блок зависимостей	281
15.7. Флейворы	281
15.8. <code>USES</code> и <code>USE_x</code>	282
15.9. Стандартные переменные <code>bsd.port.mk</code>	282
15.10. Параметры и помощники	282
15.11. Остальные переменные	283
15.12. Цели	283
16. Актуализация	284
16.1. FreshPorts	284
16.2. Web-интерфейс к хранилищу исходных текстов	284
16.3. Список рассылки FreeBSD, посвящённый портам	284
16.4. Кластер построения портов FreeBSD	285
16.5. Portscout: сканер дистрибутивных файлов портов FreeBSD	285
17. Использование макроса <code>USES</code>	286
17.1. Введение в <code>USES</code>	286
17.2. <code>7z</code>	286
17.3. <code>ada</code>	287
17.4. <code>angr</code>	287
17.5. <code>ansible</code>	287
17.6. <code>apache</code>	288
17.7. <code>autoreconf</code>	289
17.8. <code>azurepy</code>	289
17.9. <code>blaslapack</code>	290
17.10. <code>bdb</code>	290
17.11. <code>bison</code>	290
17.12. <code>budgie</code>	291
17.13. <code>cabal</code>	291

17.14. cargo	292
17.15. charsetfix	293
17.16. cl	293
17.17. cmake	294
17.18. compiler	294
17.19. cpe	295
17.20. cran	296
17.21. desktop-file-utils	296
17.22. desthack	296
17.23. display	296
17.24. dos2unix	297
17.25. drupal	297
17.26. ebur128	297
17.27. eigen	297
17.28. electronfix	297
17.29. elfctl	298
17.30. elixir	298
17.31. emacs	300
17.32. erlang	301
17.33. fakeroot	301
17.34. fam	302
17.35. firebird	302
17.36. fonts	302
17.37. fortran	302
17.38. fpc	302
17.39. fuse	302
17.40. gem	303
17.41. gettext	303
17.42. gettext-runtime	303
17.43. gettext-tools	303
17.44. ghostscript	303
17.45. gl	303
17.46. gmake	304
17.47. gnome	304
17.48. go	307
17.49. gperf	308
17.50. grantlee	308
17.51. groff	308
17.52. gssapi	308
17.53. gstreamer	309
17.54. guile	312

17.55. horde	313
17.56. iconv	313
17.57. imake	313
17.58. java	313
17.59. jpeg	315
17.60. kde	316
17.61. kmod	316
17.62. kodi	316
17.63. lazarus	317
17.64. ldap	317
17.65. lha	318
17.66. libarchive	318
17.67. libedit	318
17.68. libtool	318
17.69. linux	319
17.70. llvm	321
17.71. localbase	321
17.72. lua	321
17.73. luajit	322
17.74. lxqt	322
17.75. magick	322
17.76. makeinfo	323
17.77. makeself	323
17.78. mate	323
17.79. meson	324
17.80. metaport	324
17.81. minizip	324
17.82. mlt	324
17.83. mysql	324
17.84. mono	325
17.85. motif	325
17.86. mpi	325
17.87. ncurses	326
17.88. nextcloud	326
17.89. ninja	326
17.90. nodejs	326
17.91. objc	327
17.92. ocaml	327
17.93. octave	328
17.94. openal	328
17.95. pathfix	328

17.96. pear	328
17.97. perl5	329
17.98. pgsql	329
17.99. php	329
17.100. pkgconfig	332
17.101. pure	332
17.102. pyqt	332
17.103. pytest	333
17.104. python	334
17.105. qmail	334
17.106. qmake	334
17.107. qt	334
17.108. qt-dist	335
17.109. readline	336
17.110. ruby	336
17.111. samba	337
17.112. scons	337
17.113. sdl	338
17.114. shared-mime-info	338
17.115. shebangfix	339
17.116. sqlite	341
17.117. sbrk	341
17.118. ssl	341
17.119. tar	342
17.120. tcl	342
17.121. terminfo	343
17.122. tex	343
17.123. tk	344
17.124. trigger	344
17.125. uidfix	345
17.126. uniquefiles	345
17.127. vala	345
17.128. varnish	345
17.129. waf	345
17.130. webplugin	346
17.131. xfce	346
17.132. xorg	347
17.133. xorg-cat	348
17.134. zip	349
18. Значения <code>__FreeBSD_version</code>	350
18.1. Версии FreeBSD 15	350

18.2. Версии FreeBSD 14 .....	355
18.3. FreeBSD 13 Версии .....	368
18.4. Версии FreeBSD 12 .....	393
18.5. FreeBSD 11 Версии .....	412
18.6. Версии FreeBSD 10 .....	436
18.7. Версии FreeBSD 9 .....	453
18.8. Версии FreeBSD 8 .....	463
18.9. Версии FreeBSD 7 .....	483
18.10. Версии FreeBSD 6 .....	493
18.11. Версии FreeBSD 5 .....	501
18.12. Версии FreeBSD 4 .....	514
18.13. Версии FreeBSD 3 .....	520
18.14. Версии FreeBSD 2.2 .....	521
18.15. FreeBSD 2 Версии до 2.2-RELEASE .....	522

# Глава 1. Введение

Коллекция портов FreeBSD является способом, используемым практически каждым для установки приложений ("портов") на FreeBSD. Как и почти всё остальное во FreeBSD, эта система в основном является добровольно поддерживаемым начинанием. Важно иметь это в виду при чтении данного документа.

Во FreeBSD каждый может прислать новый порт либо изъявить желание поддерживать существующий порт, если его никто ещё никто не поддерживает-вам не нужно иметь никаких особых привилегий на внесение изменений, чтобы это делать.

## Глава 2. Как самому сделать новый порт

Итак, вы интересуетесь, как создать собственный порт или обновить существующий? Великолепно!

Ниже находятся некоторые указания по созданию нового порта для FreeBSD. Если вы хотите обновить существующий порт, вы должны прочесть их, а затем [Обновление отдельного порта](#).

Если этот документ недостаточно подробен, вы должны обратиться к файлу `/usr/ports/Mk/bsd.port.mk`, который включается в make-файл каждого порта. Он хорошо прокомментирован, и даже если вы не занимаетесь хакингом make-файлов ежедневно, из него вы сможете узнать много нового. Кроме того, конкретные вопросы можно задать, послав письмо на адрес [Список рассылки, посвящённый Портам FreeBSD](#).



Только часть переменных (`VAR`), которые могут быть переопределены, описаны в этом документе. Большинство (если не все) описаны в начале файла `/usr/ports/Mk/bsd.port.mk`; остальные, скорее всего, тоже там описаны. Заметьте, что в этом файле используется нестандартная настройка шага табуляции: Emacs и Vim должны распознать это при загрузке файла. Как `vi(1)`, так и `ex(1)` могут быть настроены на использование правильного значения выдачей команды `:set tabstop=4` после загрузки файла.

Ищете, с чего бы начать попроще? Посмотрите на [перечень запрошенных портов](#), есть ли там такие, над которыми вы можете работать.

# Глава 3. Быстрое портирование

В этом разделе описано, как быстро создать новый порт. Для случаев, когда этот быстрый метод не подходит, полный процесс "Медленного портирования" описан в разделе [Медленное портирование](#).

Во-первых, скачайте оригинальный tar-файл и поместите его в каталог `DISTDIR`, который по умолчанию есть не что иное, как `/usr/ports/distfiles`.



Здесь предполагается, что программное обеспечение компилируется без проблем как есть, то есть для работы приложения на вашей системе FreeBSD не потребовалось абсолютно никаких изменений. Если требовалось что-то изменить, то вам придется обратиться также и к следующему разделу — [Медленное портирование](#).

Перед началом портирования рекомендуется установить переменную `make(1) DEVELOPER` в `/etc/make.conf`.



```
# echo DEVELOPER=yes >> /etc/make.conf
```

Эта настройка включает "режим разработчика", в котором отображаются предупреждения при использовании устаревших конструкций и задействуются некоторые дополнительные проверки при вызове команды `make`.

## 3.1. Создание файла Makefile

Минимальный Makefile будет выглядеть примерно так:

```
PORTNAME=  oneko
DISTVERSION=  1.1b
CATEGORIES=  games
MASTER_SITES=  ftp://ftp.rediris.es/sites/ftp.freebsd.org/pub/FreeBSD/

MAINTAINER=  youremail@example.com
COMMENT=  Cat chasing a mouse all over the screen
WWW=  http://www.daidouji.com/oneko/

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

Посмотрим, сможете ли вы его понять. Более подробный пример приведен в секции [пример Makefile](#).

## 3.2. Создание информационных файлов

Имеется два информационных файла, которые требуются для любого порта, вне зависимости от того, является ли он пакетом или нет. Это `pkg-descr` и `pkg-plist`. Префикс `pkg-` отличает их от других файлов.

### 3.2.1. `pkg-descr`

Это более подробное краткое описание порта. От одного до нескольких абзацев, кратко описывающих, что представляет собой порт, будет достаточно.



Это *не* руководство и не подробное описание того, как использовать или компилировать порт! *Пожалуйста, будьте осторожны при копировании из README или manpage.* Очень часто они не содержат краткого описания порта или имеют неудобный формат. Например, `manpages` используют выравнивание по ширине, что особенно плохо выглядит с моноширинными шрифтами.

С другой стороны, содержимое файла `pkg-descr` должно быть длиннее, чем строка `COMMENT` из `Makefile`. Оно должно более подробно объяснять, что собой представляет данный порт.

Хорошо составленный `pkg-descr` описывает порт достаточно полно, чтобы пользователю не приходилось сверяться с документацией или посещать вебсайт для понимания того, что делает данное программное обеспечение, чем оно может быть полезно или какие хорошие функции у него имеются. Упоминание про определённые требования, такие как используемый графический инструментарий, тяжёлые зависимости, окружение для запуска или используемый язык программирования помогут пользователям определиться, будет ли этот порт для них работать.



URL, который ранее включался в последнюю строку файла `pkg-descr`, был перемещен в `Makefile`.

### 3.2.2. `pkg-plist`

Здесь перечисляются все файлы, устанавливаемые портом. Его также называют "списком для упаковки", потому что пакет генерируется упаковкой файлов, которые здесь указаны. Имена путей указываются относительно установочного префикса (обычно `/usr/local`).

Вот маленький пример:

```
bin/oneko
man/man1/oneko.1.gz
lib/X11/app-defaults/Oneko
lib/X11/oneko/cat1.xpm
lib/X11/oneko/cat2.xpm
lib/X11/oneko/mouse.xpm
```

Обратитесь к странице справочной системы по команде `pkg-create(8)` с подробным описанием формата списка упаковки.



Рекомендуется, чтобы имена файлов в этом списке были отсортированы в алфавитном порядке. Это позволит значительно облегчить сверку изменений при обновлении порта. Фреймворк делает это корректно, когда список пакета [сгенерирован автоматически](#).



Создание списка упаковки вручную может оказаться весьма трудоёмкой задачей. Если порт устанавливает большое количество файлов, раздел об [автоматическом построении списка упаковки](#) может помочь сэкономить время.

Существует только одно исключение, когда у порта может отсутствовать `pkg-plist`. Если порт устанавливает лишь несколько файлов, а возможно, и каталогов, то они могут быть перечислены в переменных `PLIST_FILES` и `PLIST_DIRS`, соответственно, внутри файла `Makefile` порта. К примеру, мы можем обойтись без файла `pkg-plist` у приведённого выше порта `oneko`, добавив следующие строки в `Makefile`:

```
PLIST_FILES=  bin/oneko \
              man/man1/oneko.1.gz \
              lib/X11/app-defaults/Oneko \
              lib/X11/oneko/cat1.xpm \
              lib/X11/oneko/cat2.xpm \
              lib/X11/oneko/mouse.xpm
```



Использование `PLIST_FILES` не должно быть чрезмерным. При поиске происхождения файла обычно просматривают файлы `pkg-plist` в дереве портов с помощью `grep`. Указание файлов в `PLIST_FILES` в `Makefile` усложняет этот поиск.



Если порту требуется создать пустой каталог или он создает каталоги вне `${PREFIX}` во время установки, обратитесь к разделу [Очистка пустых каталогов](#) для получения дополнительной информации.



Поскольку `PLIST_FILES` является переменной `make(1)`, любая запись с пробелами должна быть заключена в кавычки. Например, при использовании ключевых слов, описанных в `pkg-create(8)` и [Расширение списка пакетов с помощью ключевых слов](#), запись должна быть заключена в кавычки.

```
PLIST_FILES=  "@sample ${ETCDIR}/oneko.conf.sample"
```

Позже мы увидим, как `pkg-plist` и `PLIST_FILES` могут использоваться для выполнения [более сложных задач](#).

## 3.3. Создание файла с контрольной суммой

Просто введите команду `make makesum`. Правила утилиты `make` автоматически сгенерируют файл `distinfo`. Не пытайтесь создавать этот файл вручную.

## 3.4. Тестирование порта

Вы должны удостовериться, что правила построения порта выполняют именно то, что вы хотите, включая создание пакета для порта. Вот те важные вещи, которые вы должны проверить:

- `pkg-plist` не содержит ничего сверх того, что устанавливается портом.
- `pkg-plist` содержит абсолютно все, что устанавливается портом.
- Порт может быть установлен с помощью указания цели `install`. Это позволяет убедиться в правильной работе сценария установки.
- Порт может быть правильным образом удалён с помощью указания цели `deinstall`. Это позволяет убедиться в правильной работе сценария удаления.
- Порт имеет доступ к сетевым ресурсам только во время фазы цели `fetch`. Это важно для сборщиков пакетов, таких как `ports-mgmt/poudriere`.
- Убедитесь, что команду `make package` можно выполнить от имени обычного пользователя (то есть не от `root`). Если это не работает, возможно, потребуется исправить программное обеспечение. См. также `fakeroot` и `uidfix`.

### Procedure: Рекомендуемый порядок проверки

1. `make stage`
2. `make check-orphans`
3. `make package`
4. `make install`
5. `make deinstall`
6. `make package` (как пользователь)

Убедитесь, что на любом из этапов не выдается никаких предупреждений.

Тщательное автоматизированное тестирование можно выполнить с помощью `ports-mgmt/poudriere` из коллекции портов, дополнительную информацию см. в `poudriere`. Он поддерживает `клетки`, в которых можно протестировать все указанные выше шаги без воздействия на состояние основной системы.

## 3.5. Проверка вашего порта утилитой `portlint`

Будьте добры, пользуйтесь утилитой `portlint` для проверки того, что ваш порт соответствует нашим рекомендациям. Программа `ports-mgmt/portlint` является частью Коллекции Портов. В частности, вы можете захотеть проверить, правильно ли сформирован файл `Makefile` и соответствующим ли образом именован `пакет`.



Не следуйте слепо выводу `portlint`. Это статический инструмент для проверки, и иногда он ошибается.

## 3.6. Посылка нового порта

Перед посылкой нового порта прочитайте раздел о том, что [можно и нельзя](#) делать.

Когда вы наконец довольны своим первым портом, единственное, что осталось сделать, это включить его в основное дерево портов FreeBSD и осчастливить этим всех остальных.



Нам не нужны каталог `work` или пакет `pkgname.txz`, поэтому их можно удалить.

Далее создайте файл `patch(1)`. Предположим, что порт называется `oneko` и находится в категории `games`.

*Пример 1. Создание `.diff` для нового порта*

Добавьте все файлы с помощью `git add .`, затем просмотрите изменения с помощью `git diff`. Например:

```
% git add .
% git diff --staged
```

Убедитесь, что все необходимые файлы включены, затем зафиксируйте изменение в вашей локальной ветке и создайте патч с помощью `git format-patch`

```
% git commit
% git format-patch origin/main
```

Патч, созданный с помощью `git format-patch`, будет содержать идентификатор автора и адреса электронной почты, что упрощает применение разработчиками (с помощью `git am`) и правильное указание авторства.

Чтобы упростить применение патча для коммиттеров в их рабочей копии дерева портов, пожалуйста, создайте файл `.diff` из корня вашего дерева портов.

Отправьте файл `oneko.diff` через [форму отправки отчётов об ошибках](#). Укажите продукт "Ports & Packages", компонент "Individual Port(s)" и следуйте приведённым там инструкциям. Добавьте краткое описание программы в поле Description PR (например, сокращённую версию `COMMENT`) и не забудьте прикрепить файл `oneko.diff`.



Хорошее описание в заголовке сообщения о проблеме значительно облегчает работу коммиттеров портов. Для новых портов мы предпочитаем

нечто вроде "[NEW PORT] категория/название\_порта краткое описание порта". Следование этой схеме упрощает и ускоряет начало работы по добавлению нового порта.

После отправки порта, пожалуйста, потерпите. Время, необходимое для включения нового порта во FreeBSD, может занимать от нескольких дней до нескольких месяцев. [Здесь](#) можно увидеть список ожидающих PR для портов.

Чтобы получить список *открытых* PR для портов, выберите *Open* и *Ports & Packages* в форме поиска, затем нажмите **[ Search ]**.

После ознакомления с новым портом мы ответим, если это необходимо, и добавим его в дерево. Имя отправителя также будет добавлено в список [Дополнительных участников FreeBSD](#) и другие файлы.



Ранее можно было отправлять исправления для новых портов, используя файл `shar(1)`; однако с развитием `git(1)` это больше не актуально. Коммиттеры больше не принимают файлы `shar(1)`, так как их использование чревато ошибками и не добавляет соответствующую запись в Makefile категории.

# Глава 4. Медленное портирование

Итак, все оказалось не так уж и просто, и порт потребовал некоторых модификаций для того, чтобы заставить его работать. В этом разделе мы расскажем, шаг за шагом, как его модифицировать, чтобы он работал с нашей системой портов.

## 4.1. Как всё это работает

Во-первых, когда пользователь дает в своем каталоге с портом команду `make`, происходит целая череда событий. Во время чтения этого текста может оказаться полезным иметь файл `bsd.port.mk` открытым в другом окне, что сильно поможет его понять.

Но не волнуйтесь сильно, если вы не до конца понимаете, что делается в `bsd.port.mk`, не так уж много людей его понимает... :-)

1. Запускается цель `fetch`. Цель `fetch` отвечает за то, что архив исходных текстов имеется в наличии локально в каталоге `DISTDIR`. Если цель `fetch` не может найти требуемые файлы в каталоге `DISTDIR`, то они будут искаться по указателю URL `MASTER_SITES`, который устанавливается в `Makefile`, а также на наших FTP зеркалах, куда мы по возможности помещаем дистрибутивные файлы для архива. Затем она попытается сгрузить указанный файл с помощью `FETCH`, полагая, что запрашивающая машина имеет прямое подключение к Интернет. Если файл скачается удачно, то он будет помещен в каталог `DISTDIR` для последующего использования и обработки.
2. Выполняется цель `extract`. Она ищет дистрибутивный файл порта (как правило, tar-архив `gzip`) в каталоге `DISTDIR` и распаковывает его во временный каталог, задаваемый переменной `WRKDIR` (по умолчанию `work`).
3. Выполняется цель `patch`. Во-первых, применяются все патчи, заданные переменной `PATCHFILES`. Во-вторых, если какие-либо файлы с патчами, носящие имена `patch-*`, имеются в подкаталоге `PATCHDIR` (по умолчанию это каталог `files`), то они применяются в этот момент в алфавитном порядке.
4. Запускается цель `configure`. Здесь может выполняться любая из многих различных вещей.
  - a. Если он существует, запускается `scripts/configure`.
  - b. Если установлены `HAS_CONFIGURE` или `GNU_CONFIGURE`, запускается `WRKSRC/configure`.
5. Выполняется цель `build`. Она отвечает за переход в собственный рабочий каталог порта (`WRKSRC`) и его построение.
6. Выполняется цель `stage`. Конечный набор построенных файлов помещается во временный каталог (`STAGEDIR`, смотрите [Staging](#)). Иерархия этого каталога отражает иерархию каталогов системы, в которую данный пакет будет устанавливаться.
7. Выполняется цель `package`. При этом создается пакет с использованием файлов из временного каталога, созданного во время выполнения цели `stage`, и файла `pkg-plist` порта.
8. Выполняется цель `install`. Это устанавливает пакет, созданный во время цели `package`, в хост-систему.

Выше перечислены стандартные действия. Кроме того, вы сами можете определить цели `pre-что-то` или `post-что-то`, или создать скрипты с такими именами в подкаталоге `scripts`, и они будут запущены до или после выполнения действий по умолчанию.

Например, если у вас есть цель `post-extract`, определённая в вашем файле `Makefile` и файл `pre-build` в подкаталоге `scripts`, то после выполнения обычных действий по распаковке, будет вызвана цель `post-extract` а скрипт `pre-build` будет выполнен перед запуском стандартных правил построения. Рекомендуется использовать цели из `Makefile`, если действия достаточно просты, потому что в дальнейшем будет проще определить, какие нестандартные действия требуют порт.

Действия по умолчанию выполняются целями `do-что-то` из `bsd.port.mk`. Например, команды для распаковки порта находятся в цели `do-extract`. Если вам не хватает цели по умолчанию, вы можете ее исправить, переопределив цель `do-something` в вашем файле `Makefile`.



"Основные" цели (к примеру, `extract`, `configure` и так далее) не делают ничего больше, чем проверяют успешность завершения всех предыдущих шагов и вызывают настоящие цели или скрипты, и их не нужно менять. Если вам нужно изменить распаковку, исправляйте `do-extract`, но никогда не меняйте способ работы `extract`! Кроме того, цель `post-deinstall` является недействительной и не выполняется инфраструктурой портов.

Теперь, когда вы представляете, что происходит, когда пользователь набирает команду `make install`, давайте пройдемся через шаги, рекомендуемые для создания настоящего порта.

## 4.2. Получение исходного кода

Получите оригинальные исходные тексты (обычно) в виде упакованного `tar`-архива (`foo.tar.gz` или `foo.tar.bz2`) и скопируйте его в каталог `DISTDIR`. Всегда используйте исходные тексты *основной ветки разработки* везде, где это возможно.

Вам потребуется задать значение переменной `MASTER_SITES` так, чтобы оно указывало на местоположение оригинального `tar`-архива. В файле `bsd.sites.mk` вы найдёте краткие обозначения для большинства популярных сайтов. Пожалуйста, используйте эти сайты-и соответствующие определения-везде, где это возможно, чтобы избежать проблем повторения одной и той же информации в базе источников. Так как эти сайты со временем меняются, для всех причастных поддержка становится настоящим кошмаром. Для подробностей смотрите `MASTER_SITES`.

Если вы не можете найти FTP/HTTP сайт с хорошим подключением к сети, или находите только сайты, которые имеют раздражающе нестандартные форматы, то можете захотеть поместить копию на надёжный сервер FTP или HTTP, который вам доступен (например, ваша домашняя страница).

Если вы не можете найти доступного и надёжного места для помещения дистрибутивного файла, то мы сами сможем разместить его на сервере `ftp.FreeBSD.org`; однако это наименее рекомендуемое решение. Дистрибутивный файл должен быть помещён в каталог `~/public_distfiles/` одного из пользователей машины `freefall`. Попросите того, кто коммитил

ваш порт, сделать это. Этот человек также задаст переменной `MASTER_SITES` значение `MASTER_SITE_LOCAL`, а в переменной `MASTER_SITE_SUBDIR` укажет логин кластера FreeBSD.

Если дистрибутивные файлы вашего порта постоянно меняются по неизвестным причинам без изменения версий со стороны автора, остаётся только поместить дистрибутив на вашу домашнюю Web-страницу и указать её первой в списке `MASTER_SITES`. Если можете, попытайтесь договориться с автором порта об этом; это действительно помогает в достижении некоторого управления исходным кодом. Размещение собственной версии поможет избежать появления ошибок у пользователей типа `checksum mismatch`, а также уменьшит нагрузку на людей, сопровождающих наш FTP-сервер. Также, если у порта имеется только один основной сервер, то рекомендуется поместить архивную копию на свой сайт и указать его в списке `MASTER_SITES` вторым.

Если вашему порту требуются дополнительные патчи, доступные в Интернет, скачайте также и их, поместив в каталог `DISTDIR`. Не волнуйтесь, если они находятся не на том же сайте, откуда взят дистрибутивный архив, мы умеем обрабатывать такие ситуации (смотрите описание `PATCHFILES` ниже).

### 4.3. Модификация порта

Распакуйте копию дистрибутивного файла в отдельный каталог и внесите изменения, которые необходимы для того, чтобы порт компилировался нормально в текущей версии FreeBSD. *Тщательно отслеживайте* все, что вы делаете, этот процесс вам предстоит автоматизировать. Все, включая удаление, добавление или модификацию в файлах должны будут выполняться автоматически с помощью скриптов или файлов патчей, когда вы завершите работу над портом.

Если вашему порту во время компиляции, установки и настройки требуется довольно много взаимодействовать с пользователем, то посмотрите на один из классических скриптов Configure Лэрри Уолла (Larry Wall) и сделайте сами что-либо подобное. Предназначение новой коллекции портов - это сделать каждое приложение в стиле "plug-and-play" настолько, насколько это вообще возможно для конечного пользователя при минимальном использовании дискового пространства.



Если явно не указано обратное, то патчи, скрипты и другие файлы, которые вы создали и предоставили для Коллекции Портов FreeBSD, неявно подпадают под стандартные условия лицензии BSD.

### 4.4. Работа с патчами

Файлы, которые добавлялись или изменялись в процессе создания порта, могут быть выявлены программой `diff(1)`, а результат работы этой программы может быть в дальнейшем передан программе `patch(1)`. Такое действие с обычным файлом подразумевает сохранение копии файла с первоначальным содержимым перед внесением каких-либо изменений.

```
% cp file file.orig
```

Патчи сохраняются в виде файлов с именем `patch-*`, где `*` обозначает путь к файлу, к которому применяется патч, такой как `patch-Imakefile` или `patch-src-config.h`.



Используйте `BINARY_ALIAS` для замены жёстко заданных команд во время сборки и избежания исправлений в файлах сборки. Подробнее см. в [Использование BINARY\\_ALIAS для переименования команд вместо исправления сборки](#).

#### 4.4.1. Общие правила для установки патчей

Файлы патчей хранятся в `PATCHDIR`, обычно это `files/`, откуда они будут автоматически применены. Все исправления должны быть относительно к `WRKSRС`. Обычно `WRKSRС` является подкаталогом `WRKDIR`, каталога, в котором распаковывается `distfile`. Используйте `make -V WRKSRС` для просмотра фактического пути. Имена файлов патчей должны соответствовать следующим правилам:

- Избегайте ситуации, когда несколько патчей изменяют один и тот же файл. Например, если и `patch-foobar.c`, и `patch-foobar.c2` вносят изменения в `${WRKSRС}/foobar.c`, это делает их хрупкими и затрудняет отладку.
- При создании имен для файлов исправлений заменяйте каждое подчеркивание (`_`) на два подчеркивания (`__`) и каждый слэш (`/`) на одно подчеркивание (`_`). Например, чтобы исправить файл с именем `src/freelut_joystick.c`, назовите соответствующий исправление `patch-src_freelut__joystick.c`. Не называйте исправления как `patch-aa` или `patch-ab`. Всегда используйте путь и имя файла в названиях исправлений. Использование `make makepatch` автоматически генерирует правильные имена.
- Патч может изменять несколько файлов, если изменения связаны между собой и патч назван соответствующим образом. Например, `patch-add-missing-stdlib.h`.
- Используйте только символы `[-+._a-zA-Z0-9]` для именования патчей. В частности, *не используйте `::` как разделитель путей*, вместо этого используйте `_`.

Минимизируйте количество нефункциональных изменений пробелов в патчах. В мире открытого исходного кода распространена практика, когда проекты используют обширные части кодовой базы, но следуют разным правилам стиля и отступов. При переносе работоспособного функционала из одного проекта для исправления аналогичных участков в другом будьте внимательны: итоговый патч может быть переполнен нефункциональными изменениями. Это не только увеличивает размер репозитория портов, но и затрудняет понимание того, что именно вызвало проблему и какие изменения были внесены.

Если файл необходимо удалить, сделайте это в цели `post-extract`, а не как часть исправления.

## 4.4.2. Ручное создание патчей



Ручное создание патчей обычно не требуется. Предпочтительным методом является автоматическая генерация патчей, как описано ранее в этом разделе. Однако иногда может потребоваться ручное исправление.

Патчи сохраняются в файлы с именами `patch-*`, где `*` указывает на путь к файлу, который патчится, например `patch-Imakefile` или `patch-src-config.h`. Патчи с именами файлов, не начинающимися с `patch-`, не будут применены автоматически.

После изменения файла используется `diff(1)` для записи различий между оригинальной и изменённой версиями. `-u` заставляет `diff(1)` выводить различия файлов в "унифицированном" формате (unified diffs), которые являются предпочтительным форматом.

```
% diff -u file.orig file > patch-pathname-file
```

Для порождения патчей для новых добавляемых файлов используется параметр `-N`, который заставляет `diff(1)` трактовать несуществующие прежде файлы как если бы они существовали, но имели пустое содержимое:

```
% diff -u -N newfile.orig newfile > patch-pathname-newfile
```

Использование опции рекурсии (`-r`) в `diff(1)` для создания патчей допустимо, но пожалуйста, проверяйте полученные патчи, чтобы убедиться в отсутствии ненужных данных. В частности, различия между резервными файлами, Makefile, когда порт использует `Imake` или GNU `configure`, и т.д., являются избыточными и должны быть удалены. Если потребовалось отредактировать `configure.in` и запустить `autoconf` для регенерации `configure`, не включайте различия в `configure` (его объем часто достигает нескольких тысяч строк!). Вместо этого определите `USES=autoreconf` и возьмите различия для `configure.in`.

## 4.4.3. Простая автоматическая замена

Простые замены могут быть выполнены напрямую из Makefile порта, используя режим редактирования на месте утилиты `sed(1)`. Это полезно, когда изменения используют значение переменной:

```
post-patch:
    @${REINPLACE_CMD} -e 's|/usr/local|${PREFIX}|g' ${WRKSRC}/Makefile
```



Используйте `sed(1)` только для замены изменяемого содержимого. Для замены статического содержимого необходимо использовать файлы исправлений вместо `sed(1)`.

Довольно часто портируемое программное обеспечение использует соглашение CR/LF в

исходных файлах. Это может вызвать проблемы с дальнейшим наложением патчей, предупреждениями компилятора или выполнением скриптов (например, `/bin/sh^M не найден`). Для быстрого преобразования всех файлов из CR/LF в просто LF добавьте следующую запись в Makefile порта:

```
USES= dos2unix
```

Список конкретных файлов для преобразования может быть указан:

```
USES= dos2unix
DOS2UNIX_FILES= util.c util.h
```

Используйте `DOS2UNIX_REGEX` для преобразования группы файлов во вложенных каталогах. Его аргумент — это совместимое с `find(1)` регулярное выражение. Подробнее о формате можно узнать в `re_format(7)`. Эта опция полезна для преобразования всех файлов с заданным расширением. Например, преобразовать все исходные файлы кода, оставив двоичные файлы без изменений:

```
USES= dos2unix
DOS2UNIX_REGEX= .*\.([ch]|cpp)
```

Аналогичной опцией является `DOS2UNIX_GLOB`, которая запускает `find` для каждого указанного в ней элемента.

```
USES= dos2unix
DOS2UNIX_GLOB= *.c *.cpp *.h
```

Базовый каталог для преобразования может быть установлен. Это полезно, когда имеется несколько distfiles и в нескольких из них содержатся файлы, требующие преобразования окончаний строк.

```
USES= dos2unix
DOS2UNIX_WKRSRC= ${WRKDIR}
```

#### 4.4.4. Внесение исправлений при условии

Некоторые порты требуют патчей, которые применяются только для определённых версий FreeBSD или при включении или отключении конкретной опции. Условные патчи указываются путём размещения полных путей к файлам патчей в `EXTRA_PATCHES`. Имена файлов условных патчей обычно начинаются с `extra-`, хотя это и не обязательно. Однако их имена *не должны* начинаться с `patch-`. Если это произойдёт, они будут применены безусловно фреймворком, что нежелательно для условных патчей.

### Пример 2. Применение патча для конкретной версии FreeBSD

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

# Patch in the iconv const qualifier before this
.if ${OPSYS} == FreeBSD && ${OSVERSION} < 1100069
EXTRA_PATCHES= ${PATCHDIR}/extra-patch-fbsd10
.endif

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

### Пример 3. Опциональное применение патча

Когда для **опции** требуется патч, используйте `opt_EXTRA_PATCHES` и `opt_EXTRA_PATCHES_OFF`, чтобы сделать исправление зависимым от опции `opt`. Дополнительные сведения см. в [Generic Variables Replacement](#).

```
OPTIONS_DEFINE=  FOO BAR
FOO_EXTRA_PATCHES=  ${PATCHDIR}/extra-patch-foo
BAR_EXTRA_PATCHES_OFF=  ${PATCHDIR}/extra-patch-bar.c \
                        ${PATCHDIR}/extra-patch-bar.h
```

### Пример 4. Использование EXTRA\_PATCHES с директорией

Иногда для функции требуется множество патчей, в таком случае можно указать `EXTRA_PATCHES` на директорию, и все файлы с именем `patch-*` в ней будут применены автоматически.

Создайте подкаталог в `${PATCHDIR}` и переместите в него патчи. Например:

```
% ls -l files/foo-patches
-rw-r--r--  1 root  wheel   350 Jan 16 01:27 patch-Makefile.in
-rw-r--r--  1 root  wheel  3084 Jan 18 15:37 patch-configure.ac
```

Затем добавьте это в Makefile:

```
OPTIONS_DEFINE= FOO
FOO_EXTRA_PATCHES=  ${PATCHDIR}/foo-patches
```

Затем фреймворк использует все файлы с именем `patch-*` в этом каталоге.

## 4.5. Конфигурирование

Поместите все дополнительные команды, требуемые для настройки, в ваш скрипт `configure` и сохраните его в подкаталоге `scripts`. Как отмечено выше, вы можете сделать это целями в файле `Makefile` и/или скриптами с именами `pre-configure` или `post-configure`.

## 4.6. Обработка пользовательского ввода

Если для построения, конфигурации или установки вашего порта требуется некоторый ввод со стороны пользователя, то вы должны задать переменную `IS_INTERACTIVE` в вашем файле `Makefile`. В случае "ночного построения" это позволит пропустить ваш порт, если пользователь в своем окружении задал переменную `BATCH` (и если пользователь установил переменную `INTERACTIVE`, то будут строиться *только* порты, которые требуют взаимодействия с пользователем. Это сэкономит значительное количество времени на части машин, которые постоянно строят порты (смотрите ниже).

При наличии разумных ответов на задаваемые вопросы, подходящих по умолчанию, также рекомендуется проверять переменную `PACKAGE_BUILDING` и выключать интерактивный скрипт, если он есть. Это позволит нам строить пакеты для помещения на компакт-диски и FTP-серверы.

# Глава 5. Настройка Makefile

Настройка Makefile довольно проста, и мы снова рекомендуем изучить существующие примеры перед началом. Также в этом руководстве есть [пример Makefile](#), поэтому ознакомьтесь с ним и, пожалуйста, соблюдайте порядок переменных и разделов в этом шаблоне, чтобы порт был удобнее для чтения другими.

Рассмотрите эти проблемы последовательно при разработке нового Makefile:

## 5.1. Оригинальный исходный код

Находится ли он в `DISTDIR` в виде стандартного архива `gzip` с именем вроде `foozolix-1.2.tar.gz`? Если да, переходите к следующему шагу. Если нет, возможно, для формата имени файла дистрибутива потребуется переопределить одну или несколько переменных: `DISTVERSION`, `DISTNAME`, `EXTRACT_CMD`, `EXTRACT_BEFORE_ARGS`, `EXTRACT_AFTER_ARGS`, `EXTRACT_SUFFIX` или `DISTFILES`.

В худшем случае создайте пользовательскую цель `do-extract`, чтобы переопределить стандартную. Это редко, если вообще когда-либо, необходимо.

## 5.2. Именованное

Первая часть Makefile порта указывает его название, описывает номер версии и помещает его в соответствующую категорию.

### 5.2.1. PORTNAME

Установите `PORTNAME` как базовое имя программы. Оно используется в качестве основы для пакета FreeBSD и для `DISTNAME`.



Название пакета должно быть уникальным во всём дереве портов. Убедитесь, что `PORTNAME` ещё не используется существующим портом и что никакой другой порт уже не имеет такой же `PKGBASE`. Если имя уже занято, добавьте либо `PKGNAMEPREFIX`.

### 5.2.2. Версии, DISTVERSION или PORTVERSION

Установите `DISTVERSION` в номер версии программы.

`PORTVERSION` — это версия, используемая для пакета FreeBSD. Она будет автоматически вычислена из `DISTVERSION` в соответствии со схемой версионирования пакетов FreeBSD. Если версия содержит буквы, может потребоваться задать `PORTVERSION` вместо `DISTVERSION`.



Только одна из переменных `PORTVERSION` и `DISTVERSION` может быть установлена одновременно.

Время от времени некоторые программы используют схему версионирования, которая несовместима с тем, как `DISTVERSION` преобразуется в `PORTVERSION`.



При обновлении порта можно использовать аргумент `-t` утилиты `pkg-version(8)`, чтобы проверить, является ли новая версия больше или меньше предыдущей. Смотрите ниже, как использовать `pkg-version(8)` для сравнения версий.

Пример 5. Использование `pkg-version(8)` для сравнения версий

`pkg version -t` принимает две версии в качестве аргументов и возвращает `<`, `=` или `>`, если первая версия меньше, равна или больше второй версии соответственно.

```
% pkg version -t 1.2 1.3
< ①
% pkg version -t 1.2 1.2
= ②
% pkg version -t 1.2 1.2.0
= ③
% pkg version -t 1.2 1.2.p1 > <.> % pkg version -t 1.2.a1 1.2.b1 < <.> % pkg
version -t 1.2 1.2p1 < ④
```

- ① 1.2 идёт перед 1.3.
- ② 1.2 и 1.2 равны, так как имеют одинаковую версию.
- ③ 1.2 и 1.2.0 равны, так как ноль ничего не значит.
- ④ 1.2 идёт после 1.2.p1, так как .p1 означает «pre-release 1» (предрелизная версия 1).
- ⑤ 1.2.a1 предшествует 1.2.b1, представьте "alpha" и "beta", где a идёт перед b.
- ⑥ 1.2 находится перед 1.2p1, так же как 2p1 (читается как "2, уровень исправления 1") — это версия, следующая после любой 2.X, но перед 3.

Здесь `a`, `b` и `p` используются так, как если бы они означали "альфа", "бета" или "пре-релиз" и "уровень патча", но на самом деле это просто буквы, которые сортируются в алфавитном порядке, поэтому можно использовать любую букву, и они будут отсортированы соответствующим образом.

Таблица 1. Примеры `DISTVERSION` и производной `PORTVERSION`

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

### Пример 6. Использование `DISTVERSION`

Если версия содержит только числа, разделённые точками, тире или подчёркиваниями, используйте `DISTVERSION`.

```
PORTNAME= nekoto
DISTVERSION= 1.2-4
```

Это сгенерирует `PORTVERSION` равный `1.2.4`.

### Пример 7. Использование `DISTVERSION` когда версия начинается с буквы или префикса

Когда версия начинается или заканчивается буквой, или префиксом, или суффиксом, которые не являются частью версии, используйте `DISTVERSIONPREFIX`, `DISTVERSION` и `DISTVERSIONSUFFIX`.

Если версия `v1.2-4`:

```
PORTNAME= nekoto
DISTVERSIONPREFIX= v
DISTVERSION= 1_2_4
```

Некоторые проекты, использующие GitHub, могут включать своё название в версии. Например, версия может выглядеть как `nekoto-1.2-4`:

```
PORTNAME= nekoto
DISTVERSIONPREFIX= nekoto-
DISTVERSION= 1.2_4
```

Эти проекты также иногда используют строку в конце версии, например, `1.2-4_RELEASE`:

```
PORTNAME= nekoto
DISTVERSION= 1.2-4
DISTVERSIONSUFFIX= _RELEASE
```

Или они делают и то, и другое, например, `nekoto-1.2-4_RELEASE`:

```
PORTNAME= nekoto
DISTVERSIONPREFIX= nekoto-
DISTVERSION= 1.2-4
DISTVERSIONSUFFIX= _RELEASE
```

`DISTVERSIONPREFIX` и `DISTVERSIONSUFFIX` не будут использоваться при формировании `PORTVERSION`, а только в `DISTNAME`.

Все сгенерируют `PORTVERSION` равный `1.2.4`.

*Пример 8. Использование `DISTVERSION`, когда версия содержит буквы, означающие "alpha", "beta" или "pre-release"*

Если версия содержит числа, разделённые точками, тире или подчёркиваниями, а буквы используются для обозначения "альфа", "бета" или "предварительной версии" (то есть до версии без букв), используйте `DISTVERSION`.

```
PORTNAME= nekoto
DISTVERSION= 1.2-pre4
```

```
PORTNAME= nekoto
DISTVERSION= 1.2p4
```

Оба варианта создадут `PORTVERSION` равную `1.2.p4`, что предшествует версии 1.2. Для проверки этого факта можно использовать `pkg-version(8)`:

```
% pkg version -t 1.2.p4 1.2
<
```

*Пример 9. Не использовать `DISTVERSION`, если версия содержит буквы, означающие "уровень патча"*

Если версия содержит буквы, которые не означают "alpha", "beta" или "pre", а скорее указывают на "уровень исправления" и следуют после версии без букв, используйте `PORTVERSION`.

```
PORTNAME= nekoto
PORTVERSION= 1.2p4
```

В данном случае использование `DISTVERSION` невозможно, так как это приведёт к генерации версии `1.2.p4`, которая будет считаться более ранней, чем `1.2`, а не более поздней. `pkg-version(8)` подтвердит это:

```
% pkg version -t 1.2 1.2.p4
> ①
% pkg version -t 1.2 1.2p4
< ②
```

① `1.2` идёт после `1.2.p4`, что в данном случае неверно.

② `1.2` находится перед `1.2p4`, что и требовалось.

Для более сложных примеров настройки `PORTVERSION`, когда версия программного обеспечения действительно несовместима с FreeBSD, или `DISTNAME`, когда файл дистрибутива не содержит саму версию, см. `DISTNAME`.

### 5.2.3. `PORTREVISION` и `PORTPOCH`

#### 5.2.3.1. `PORTREVISION`

`PORTREVISION` — это монотонно возрастающее значение, которое сбрасывается в 0 при каждом увеличении `DISTVERSION`, обычно при каждом новом официальном выпуске от поставщика. Если `PORTREVISION` не равен нулю, его значение добавляется к имени пакета. Изменения `PORTREVISION` используются автоматизированными инструментами, такими как `pkg-version(8)`, для определения доступности нового пакета.

`PORTREVISION` должен быть увеличен каждый раз, когда в порт вносятся изменения, которые так или иначе влияют на сгенерированный пакет. Это включает изменения, затрагивающие только пакеты, собранные с нестандартными `опциями`.

Примеры случаев, когда необходимо увеличить `PORTREVISION`:

- Добавление исправлений для устранения уязвимостей безопасности, ошибок или для добавления новой функциональности в порт.
- Изменения в `Makefile` порта для включения или отключения параметров сборки в пакете.
- Изменения в списке файлов пакета или в поведении во время установки. Например, изменение скрипта, который генерирует начальные данные для пакета, такие как ключи хоста `ssh(1)`.
- Увеличение версии зависимости порта от общей библиотеки (в данном случае, попытка установить старый пакет после установки более новой версии зависимости завершится неудачей, так как будет искаться старая версия `libfoo.x` вместо `libfoo.(x+1)`).
- Тихие изменения в дистрибутивном файле порта, которые имеют значительные функциональные отличия. Например, изменения в дистрибутивном файле, требующие корректировки файла `distinfo` без соответствующего изменения `DISTVERSION`, когда сравнение `diff -ru` старой и новой версий показывает нетривиальные изменения в коде.
- Изменения в `MAINTAINER`.

Примеры изменений, которые не требуют увеличения `PORTREVISION`:

- Стилиевые изменения в каркасе портов без функциональных изменений в итоговом пакете.
- Изменения в `MASTER_SITES` или другие функциональные изменения порта, которые не влияют на итоговый пакет.
- Тривиальные исправления в дистрибутивном файле, такие как исправление опечаток, которые не настолько важны, чтобы пользователи пакета были вынуждены тратить время на обновление.

- Исправления сборки, которые позволяют пакету компилироваться там, где ранее это не удавалось. При условии, что изменения не вносят функциональных изменений на других платформах, где порт ранее собирался. Поскольку `PORTREVISION` отражает содержимое пакета, если пакет ранее не мог быть собран, то нет необходимости увеличивать `PORTREVISION` для обозначения изменений.

Эмпирическое правило заключается в том, чтобы решить, является ли изменение, внесённое в порт, чем-то, что принесёт пользу *некоторым* пользователям. Будь то улучшение, исправление или просто факт, что новый пакет вообще будет работать. Затем необходимо сопоставить это с тем, что всем, кто регулярно обновляет своё дерево портов, придётся выполнить обновление. Если ответ положительный, необходимо увеличить `PORTREVISION`.



Пользователи бинарных пакетов *никогда* не увидят обновления, если `PORTREVISION` не увеличен. Без увеличения `PORTREVISION` сборщики пакетов не могут обнаружить изменение и, следовательно, не пересоберут пакет.

### 5.2.3.2. `PORTPOCH`

Время от времени разработчики программного обеспечения или сопровождающие портов FreeBSD совершают ошибку и выпускают версию своего ПО, которая фактически имеет меньший номер по сравнению с предыдущей. Примером может служить переход с `foo-20000801` на `foo-1.0` (первая версия будет ошибочно считаться более новой, поскольку число `20000801` больше, чем `1.0`).

Результаты сравнения номеров версий не всегда очевидны. Команда `pkg version` (см. [pkg-version\(8\)](#)) может быть использована для проверки сравнения двух строк с номерами версий. Например:



```
% pkg version -t 0.031 0.29
>
```

Символ `>` в выводе указывает, что версия `0.031` считается больше, чем версия `0.29`, что могло быть неочевидно для сопровождающего.

В таких ситуациях необходимо увеличить `PORTPOCH`. Если `PORTPOCH` не равен нулю, он добавляется к имени пакета, как описано в разделе 0 выше. `PORTPOCH` никогда не должен уменьшаться или сбрасываться до нуля, потому что это приведёт к ошибке при сравнении с пакетом из более ранней эпохи. Например, пакет не будет обнаружен как устаревший. Новый номер версии, `1.0,1` в приведённом выше примере, всё ещё численно меньше предыдущей версии, `20000801`, но суффикс `,1` обрабатывается автоматизированными инструментами особым образом и считается большим, чем подразумеваемый суффикс `,0` у предыдущего пакета.

Неправильное удаление или сброс `PORTPOCH` приводит к бесконечным проблемам. Если приведённое выше объяснение недостаточно ясно, обратитесь к [Список рассылки, посвящённый Портам FreeBSD](#).

Ожидается, что **PORTEPOCH** не будет использоваться для большинства портов, и что разумное использование **DISTVERSION** или аккуратное применение **PORTVERSION** часто может предотвратить необходимость его использования, если будущий выпуск программного обеспечения изменит структуру версий. Однако разработчикам портов FreeBSD следует быть осторожными, когда вендор выпускает релиз без официального номера версии — например, релиз в виде "снимка" кода. Возникает соблазн обозначить такой релиз датой выпуска, что вызовет проблемы, как в примере выше, когда будет сделан новый "официальный" релиз.

Например, если снимок выпущен на дату **20000917**, а предыдущая версия программного обеспечения была **1.2**, не следует использовать **20000917** для **DISTVERSION**. Правильным будет указать **DISTVERSION** как **1.2.20000917** или подобное, чтобы следующая версия, например **1.3**, оставалась численно большей.

### 5.2.3.3. Пример использования **PORTREVISION** и **PORTEPOCH**

Порт **gtkumble** версии **0.10** добавлен в коллекцию портов:

```
PORTNAME=  gtkumble
DISTVERSION=  0.10
```

**PKGNAME** становится **gtkumble-0.10**.

Обнаружена уязвимость в безопасности, требующая локального исправления FreeBSD. **PORTREVISION** соответствующим образом увеличивается.

```
PORTNAME=  gtkumble
DISTVERSION=  0.10
PORTREVISION=  1
```

**PKGNAME** принимает значение **gtkumble-0.10\_1**

Выпущена новая версия от поставщика под номером **0.2** (оказывается, автор изначально подразумевал, что **0.10** на самом деле означает **0.1.0**, а не "то, что идет после 0.9" — увы, теперь уже поздно). Поскольку новая минорная версия **2** численно меньше предыдущей версии **10**, необходимо увеличить **PORTEPOCH**, чтобы вручную заставить систему распознавать новый пакет как "более новый". Так как это новый релиз кода от поставщика, **PORTREVISION** сбрасывается до 0 (или удаляется из Makefile).

```
PORTNAME=  gtkumble
DISTVERSION=  0.2
PORTEPOCH=  1
```

**PKGNAME** становится **gtkumble-0.2,1**

Следующий выпуск — **0.3**. Поскольку **PORTEPOCH** никогда не уменьшается, переменные версий теперь выглядят так:

```
PORTNAME= gtkmumble
DISTVERSION= 0.3
PORTEPOCH= 1
```

**PKGNAME** принимает значение `gtkmumble-0.3,1`



Если бы **PORTEPOCH** был сброшен в `0` при этом обновлении, пользователь, установивший пакет `gtkmumble-0.10_1`, не увидел бы пакет `gtkmumble-0.3` как более новый, поскольку `3` всё ещё численно меньше, чем `10`. Помните, в этом и заключается вся суть **PORTEPOCH** изначально.

#### 5.2.4. **PKGNAMEPREFIX** и **PKGNAME\_SUFFIX**

Две необязательные переменные, **PKGNAMEPREFIX** и **PKGNAME\_SUFFIX**, объединяются с **PORTNAME** и **PORTVERSION** для формирования **PKGNAME** в виде `${PKGNAMEPREFIX}${PORTNAME}${PKGNAME_SUFFIX}-${PORTVERSION}`. Убедитесь, что это соответствует нашим [рекомендациям по именованию пакетов](#). В частности, использование дефиса (-) в **PORTVERSION** не допускается. Кроме того, если имя пакета содержит часть *language-* или *-compiled.specifcs* (см. ниже), используйте **PKGNAMEPREFIX** и **PKGNAME\_SUFFIX** соответственно. Не включайте их в **PORTNAME**.

#### 5.2.5. **Соглашения о наименовании пакетов**

Вот соглашения, которым следует придерживаться при наименовании пакетов. Это сделано для того, чтобы каталог пакетов было легко просматривать, поскольку там уже тысячи пакетов, и пользователи могут отказаться от их использования, если это будет напрягать их глаза!

Имена пакетов имеют формат `language_region-name-compiled.specifcs-version.numbers`.

Имя пакета определяется как `${PKGNAMEPREFIX}${PORTNAME}${PKGNAME_SUFFIX}-${PORTVERSION}`. Убедитесь, что переменные заданы в соответствии с этим форматом.

##### **language\_region-**

FreeBSD стремится поддерживать родной язык своих пользователей. Часть *language-* представляет собой двухбуквенное сокращение естественного языка, определённое стандартом ISO-639, когда порт относится к определённому языку. Примерами являются `ja` для японского, `ru` для русского, `vi` для вьетнамского, `zh` для китайского, `ko` для корейского и `de` для немецкого.

Если порт относится к определённому региону в языковой зоне, добавьте также двухбуквенный код страны. Например, `en_US` для американского английского и `fr_CH` для швейцарского французского.

Часть *language-* задается в **PKGNAMEPREFIX**.

##### **name**

Убедитесь, что название порта и его версия четко разделены и указаны в **PORTNAME** и **DISTVERSION**. Единственная причина, по которой **PORTNAME** может содержать часть версии,

— это если вышестоящее распространяемое ПО действительно так названо, как в портах `textproc/libxml2` или `japanese/kinput2-freewnn`. В противном случае `PORTNAME` не может содержать информацию о версии. Довольно нормально, когда несколько портов имеют одинаковый `PORTNAME`, как это делают порты `www/apache*`; в таком случае разные версии (и разные записи в индексе) различаются значениями `PKGNAMEPREFIX` и `PKGNAME_SUFFIX`.

Существует традиция называть модули Perl 5, добавляя префикс `p5-` и заменяя разделитель в виде двойного двоеточия на дефис. Например, модуль `Data::Dumper` становится `p5-Data-Dumper`.

### **-compiled.specifcs**

Если порт может быть собран с различными **жестко заданными значениями по умолчанию** (обычно это часть имени каталога в семействе портов), часть `-compiled.specifcs` указывает скомпилированные значения по умолчанию. Дефис является необязательным. Примерами могут служить размер бумаги и единицы измерения шрифтов.

Часть `-compiled.specifcs` задаётся в `PKGNAME_SUFFIX`.

### **-version.numbers**

Строка версии следует после тире (-) и представляет собой разделённый точками список целых чисел и строчных букв латинского алфавита. В частности, не допускается использование дополнительных тире внутри строки версии. Единственное исключение — строка `p1` (означающая "уровень исправления"), которую можно использовать *только* в случае отсутствия у программного обеспечения номеров основной и дополнительной версий. Если в версии программного обеспечения встречаются строки типа "alpha", "beta", "rc" или "pre", следует взять первую букву и поместить её сразу после точки. Если после таких названий строка версии продолжается, числа следуют за буквой без дополнительной точки между ними (например, `1.0b2`).

Идея заключается в упрощении сортировки портов за счёт анализа строки версии. В частности, необходимо убедиться, что компоненты номера версии всегда разделены точкой, а если дата является частью строки, использовать формат `dyyyy.mm.dd`, а не `dd.mm.yyyy` или не соответствующий стандарту Y2K формат `yy.mm.dd`. Важно добавлять перед версией букву, в данном случае `d` (от слова "дата"), на случай, если будет выпущена версия с фактическим номером, который численно окажется меньше `yyyy`.



Название пакета должно быть уникальным среди всех портов в дереве. Убедитесь, что порт с таким же `PORTNAME` ещё не существует, и если он есть, добавьте один из `PKGNAMEPREFIX` или `PKGNAME_SUFFIX`.

Вот несколько (реальных) примеров преобразования названия, указанного авторами программного обеспечения, в подходящее имя пакета. В каждой строке указана только одна из переменных `DISTVERSION` или `PORTVERSION`, в зависимости от того, какая используется в Makefile порта:

Таблица 2. Примеры наименования пакетов

Имя дистрибутива	PKGNAMEPREFIX	PORTNAME	PKGNAME_SUFFIX	DISTVERSION	.PORTVERSION	Причина или комментарий
mule-2.2.2	(пусто)	mule	(пусто)	2.2.2		Никаких изменений не требуется
mule-1.0.1	(пусто)	mule	1	1.0.1		Это версия 1 mule, а версия 2 уже существует
EmiClock-1.0.2	(пусто)	emiclock	(пусто)	1.0.2		Нет имен в верхнем регистре для отдельных программ
rdist-1.3alpha	(пусто)	rdist	(пусто)	1.3alpha		Версия будет <b>1.3.a</b>
es-0.9-beta1	(пусто)	es	(пусто)	0.9-beta1		Версия будет <b>0.9.b1</b>
mailman-2.0rc3	(пусто)	mailman	(пусто)	2.0rc3		Версия будет <b>2.0.rc3</b>
v3.3beta021.src	(пусто)	tiff	(пусто)		3.3	Что это вообще было?
tvtwm	(пусто)	tvtwm	(пусто)		p11	Нет версии в имени файла, используйте то, что указано в исходном коде
piewm	(пусто)	piewm	(пусто)	1.0		Нет версии в имени файла, используйте то, что указано в исходном коде

Имя дистрибутива	PKGNAMEPREFIX	PORTNAME	PKGNAME_SUFFIX	DISTVERSION	.PORTVERSION	Причина или комментарий
xvgr-2.10pl1	(пусто)	xvgr	(пусто)		2.10.pl1	В таком случае, <b>pl1</b> означает уровень патча, поэтому использование DISTVERSION невозможно.
gawk-2.15.6	ja-	gawk	(пусто)	2.15.6		Японская языковая версия
psutils-1.13	(пусто)	psutils	-letter	1.13		Размер бумаги жестко задан во время сборки пакета
pkfonts	(пусто)	pkfonts	300	1.0		Пакет для шрифтов с разрешением 300dpi

Если в исходном источнике полностью отсутствует информация о версии и маловероятно, что автор когда-либо выпустит новую версию, просто укажите строку версии как **1.0** (как в примере с **riewm** выше). В противном случае, спросите автора или используйте дату выпуска исходного файла в формате **дyyyy.mm.dd** или **дyyyymmdd** в качестве версии.



Используйте любую букву. Здесь **d** означает дату, если источник — это репозиторий Git, часто используется **g** с последующей датой коммита, также распространено использование **s** для снимка.

## 5.3. Категоризация

### 5.3.1. CATEGORIES

При создании пакета он помещается в `/usr/ports/packages/All`, и ссылки на него создаются в

одной или нескольких поддиректориях /usr/ports/packages. Имена этих поддиректорий задаются переменной **CATEGORIES**. Это предназначено для облегчения поиска пакетов пользователем при просмотре большого количества пакетов на FTP-сайте или CDROM. Пожалуйста, ознакомьтесь с [текущим списком категорий](#) и выберите подходящие для данного порта.

Этот список также определяет, где в дереве портов будет размещён порт. Если здесь указано несколько категорий, файлы порта должны быть помещены в подкаталог с названием первой категории. Дополнительные сведения о выборе подходящих категорий см. в [ниже](#).

### 5.3.2. Текущий список категорий

Вот текущий список категорий портов. Категории, помеченные звёздочкой (\*), являются *виртуальными* — они не имеют соответствующего подкаталога в дереве портов. Они используются только как вторичные категории и исключительно для целей поиска.



Для неvirtуальных категорий в **COMMENT** в Makefile соответствующего подкаталога содержится однострочное описание.

Категория	Описание	Заметки
accessibility	Порты для помощи пользователям с ограниченными возможностями.	
afterstep*	Порты для поддержки оконного менеджера <a href="#">AfterStep</a> .	
arabic	Поддержка арабского языка.	
archivers	Инструменты для архивирования.	
astro	Астрономические порты.	
audio	Поддержка звука.	
benchmarks	Утилиты для тестирования производительности.	
biology	Программное обеспечение, связанное с биологией.	
cad	Компьютерные средства автоматизированного проектирования.	
chinese	Поддержка китайского языка.	
comms	Программное обеспечение для связи.	В основном программное обеспечение для работы с последовательным портом.

Категория	Описание	Заметки
converters	Преобразователи символьных кодировок.	
databases	Базы данных.	
deskutils	Вещи, которые раньше находились на рабочем столе до изобретения компьютеров.	
devel	Средства разработки.	Не размещайте библиотеки здесь только потому, что они являются библиотеками. Они <i>не</i> должны быть в этой категории, если только они действительно не подходят никуда больше.
dns	Программное обеспечение, связанное с DNS.	
docs*	Мета-порты для документации FreeBSD.	
editors	Общие редакторы.	Специализированные редакторы помещаются в раздел соответствующих инструментов. Например, редактор математических формул будет помещён в math, а editors будет для него второй категорией.
education*	Программное обеспечение для образования.	Это включает приложения, утилиты или игры, разработанные в первую очередь или в значительной степени для помощи пользователю в изучении конкретной темы или обучении в целом. Также сюда входят приложения для создания курсов, приложения для предоставления курсов и приложения для управления классом или школой
elisp*	Порты Emacs-lisp.	

Категория	Описание	Заметки
emulators	Эмуляторы других операционных систем.	Терминальные эмуляторы <i>не</i> относятся сюда. Основанные на X идут в x11, а текстовые — либо в comms, либо в misc, в зависимости от конкретной функциональности.
enlightenment*	Порты, связанные с оконным менеджером Enlightenment.	
filesystems	Файловые системы и связанные утилиты.	
finance	Монетарные, финансовые и связанные с ними приложения.	
french	Поддержка французского языка.	
ftp	Клиентские и серверные утилиты FTP.	Если порт поддерживает как FTP, так и HTTP, поместите его в ftp с дополнительной категорией www.
games	Игры.	
география*	Программное обеспечение, связанное с географией.	
german	Поддержка немецкого языка.	
gnome*	Порты из проекта <a href="#">GNOME</a> .	
gnustep*	Программное обеспечение, связанное со средой рабочего стола GNUstep.	
graphics	Графические утилиты.	
hamradio*	Программное обеспечение для радиолюбителей.	
haskell*	Программное обеспечение, связанное с языком Haskell.	
hebrew	Поддержка иврита.	
hungarian	Венгерская языковая поддержка.	
irc	Утилиты Internet Relay Chat.	
japanese	Поддержка японского языка.	

Категория	Описание	Заметки
java	Программное обеспечение, связанное с языком Java™.	Категория java не должна быть единственной для порта. За исключением портов, непосредственно связанных с языком Java, разработчикам также рекомендуется не использовать java в качестве основной категории для порта.
kde*	Порты проекта KDE (общие).	
kde-приложения*	Приложения от проекта KDE.	
kde-frameworks*	Дополнительные библиотеки от проекта KDE для программирования с использованием Qt.	
kde-plasma*	Рабочий стол от проекта KDE.	
kld*	Загружаемые модули ядра.	
korean	Поддержка корейского языка.	
lang	Языки программирования.	
linux*	Приложения и вспомогательные утилиты Linux.	
lisp*	Программное обеспечение, связанное с языком Lisp.	
mail	Почтовое программное обеспечение.	
mate*	Порты, связанные с окружением рабочего стола MATE, форком GNOME 2.	
math	Численные расчеты и другие математические утилиты.	
mbone*	Приложения MBone.	
misc	Различные утилиты	Вещи, которые не подходят никуда больше. По возможности, попытайтесь найти для порта категорию лучше, чем misc, так как порты здесь часто остаются без внимания.

Категория	Описание	Заметки
multimedia	Мультимедийное программное обеспечение.	
net	Различное сетевое программное обеспечение.	
net-im	Программное обеспечение для обмена мгновенными сообщениями.	
net-mgmt	Программное обеспечение для управления сетями.	
net-p2p	Одноранговые сетевые приложения.	
сеть-vpn*	Виртуальные частные сети.	
news	Программное обеспечение для USENET-новостей.	
parallel*	Приложения, работающие с параллелизмом в вычислениях.	
pear*	Порты, связанные с PHP-фреймворком Pear.	
perl5*	Порты, требующие Perl версии 5 для работы.	
plan9*	Различные программы с <a href="#">Plan9</a> .	
polish	Поддержка польского языка.	
ports-mgmt	Порты для управления, установки и разработки портов и пакетов FreeBSD.	
portuguese	Поддержка португальского языка.	
print	Программное обеспечение для печати.	Инструменты для настольных издательских систем (превьюеры и т. д.) также относятся сюда.
python*	Программное обеспечение, связанное с языком <a href="#">Python</a> .	
ruby*	Программное обеспечение, связанное с языком <a href="#">Ruby</a> .	
rubygems*	Порты пакетов <a href="#">RubyGems</a> .	
russian	Поддержка русского языка.	

Категория	Описание	Заметки
scheme*	Программное обеспечение, связанное с языком Scheme.	
science	Научные порты, которые не входят в другие категории, такие как astro, biology и math.	
security	Средства обеспечения безопасности.	
shells	Командные оболочки.	
spanish*	Поддержка испанского языка.	
sysutils	Системные утилиты.	
tcl*	Порты, использующие Tcl для запуска.	
textproc	Средства обработки текста.	Он не включает инструменты для настольных издательских систем, которые помещаются в print.
tk*	Порты, использующие Tk для работы.	
ukrainian	Поддержка украинского языка.	
vietnamese	Поддержка вьетнамского языка.	
wayland*	Порты для поддержки сервера дисплея Wayland.	
windowmaker*	Порты для поддержки оконного менеджера Window Maker.	
www	Программное обеспечение, связанное с Всемирной паутиной.	Поддержка языка HTML также относится сюда.
x11	Система X Window и связанные компоненты.	Эта категория предназначена только для программного обеспечения, которое напрямую поддерживает оконную систему. Не помещайте сюда обычные X-приложения. Большинство из них относятся к другим категориям x11-* (см. ниже).

Категория	Описание	Заметки
x11-clocks	Часы X11.	
x11-drivers	Драйверы X11.	
x11-fm	Менеджеры файлов X11.	
x11-fonts	Шрифты и утилиты для работы со шрифтами в X11.	
x11-servers	Серверы X11.	
x11-themes	Темы X11.	
x11-toolkits	Инструментарии X11.	
x11-wm	Оконные менеджеры X11.	
xfce*	Порты, связанные с окружением рабочего стола <a href="#">Xfce</a> .	
zope*	<a href="#">Zope</a> поддержка.	

### 5.3.3. Выбор подходящей категории

Поскольку многие категории пересекаются, выбор основной категории для порта может быть утомительным. Существует несколько правил, регулирующих этот вопрос. Вот список приоритетов в порядке убывания важности:

- Первая категория должна быть физической (см. [выше](#)). Это необходимо для работы упаковки. Виртуальные категории и физические категории могут чередоваться после этого.
- Языковые категории всегда указываются первыми. Например, если порт устанавливает японские шрифты для X11, то строка **CATEGORIES** будет выглядеть так: `japanese x11-fonts`.
- Конкретные категории перечислены перед менее специфичными. Например, HTML-редактор указывается как `www editors`, а не наоборот. Также не следует указывать `net`, если порт принадлежит к любой из категорий `irc`, `mail`, `news`, `security` или `www`, так как `net` подразумевается автоматически.
- `x11` используется как вторичная категория только в случае, когда основной категорией указан естественный язык. В частности, не указывайте `x11` в строке категории для X-приложений.
- Режимы Emacs размещаются в той же категории портов, что и приложение, поддерживаемое данным режимом, а не в `editors`. Например, режим Emacs для редактирования исходных файлов какого-либо языка программирования попадает в `lang`.
- Порты, устанавливающие загружаемые модули ядра, также имеют виртуальную категорию `kld` в строке **CATEGORIES**. Это одна из вещей, автоматически обрабатываемых при добавлении **USES=kmod**.
- `misc` не встречается вместе с другими неvirtуальными категориями. Если `misc` указан

вместе с чем-то еще в **CATEGORIES**, это означает, что **misc** можно безопасно удалить, а порт разместить только в другом подкаталоге.

- Если порт действительно не подходит никуда больше, поместите его в **misc**.

Если категория не определена четко, пожалуйста, укажите это в комментарии при [отправке порта](#) в баг-трекере, чтобы мы могли обсудить её перед импортом. Как коммиттер, отправьте сообщение в рассылку [Список рассылки, посвящённый Портам FreeBSD](#), чтобы мы сначала обсудили это. Слишком часто новые порты импортируются в неправильную категорию, после чего их сразу же приходится перемещать.

### 5.3.4. Предложение новой категории

По мере роста Коллекции портов со временем были введены различные новые категории. Новые категории могут быть *виртуальными* — те, у которых нет соответствующего подкаталога в дереве портов, или *физическими* — те, у которых он есть. В этом разделе обсуждаются вопросы, связанные с созданием новой физической категории. Внимательно ознакомьтесь с ним, прежде чем предлагать новую.

Наша текущая практика заключается в том, чтобы избегать создания новой физической категории, если только либо большое количество портов логически принадлежит к ней, либо порты, которые к ней относятся, представляют собой логически обособленную группу, представляющую ограниченный общий интерес (например, категории, связанные с разговорными человеческими языками), или, желательно, оба условия одновременно.

Обоснование этого заключается в том, что такое изменение создает **значительный объем работы** как для коммиттеров, так и для всех пользователей, которые отслеживают изменения в Коллекции портов. Кроме того, предлагаемые изменения категорий, как правило, вызывают споры. (Возможно, это связано с отсутствием четкого консенсуса относительно того, когда категория становится «слишком большой», а также относительно того, должны ли категории способствовать удобству просмотра (и, следовательно, какое количество категорий было бы идеальным), и так далее.)

Вот процедура:

1. Предложите новую категорию на [Список рассылки, посвящённый Портам FreeBSD](#). Включите подробное обоснование для новой категории, объясните, почему существующие категории недостаточны, и укажите список существующих портов, предлагаемых к перемещению. (Если в Bugzilla есть ожидающие рассмотрения новые порты, которые подходят под эту категорию, также перечислите их.) Если вы являетесь сопровождающим и/или подающим предложение, укажите это, так как это может помочь в рассмотрении.
2. Участвуйте в обсуждении.
3. Если кажется, что идея находит поддержку, оформите PR, включающий как обоснование, так и список существующих портов, которые необходимо переместить. В идеале, этот PR также должен содержать следующие исправления:
  - Makefile для новых портов после копирования их репозитория
  - Makefile для новой категории

- Makefile для старых категорий портов
  - Makefile для портов, зависящих от старых портов
  - (для дополнительной оценки включите другие файлы, которые необходимо изменить, в соответствии с процедурой, описанной в Руководстве коммиттера.)
4. Поскольку это затрагивает инфраструктуру портов и включает перемещение и исправление многих портов, а также, возможно, проведение регрессионных тестов на сборочном кластере, назначьте PR для Группа Менеджеров Дерева Портов FreeBSD <portmgr@FreeBSD.org>.
  5. Если этот PR будет одобрен, коммиттер должен будет выполнить оставшуюся часть процедуры, описанной в Руководстве коммиттера.

Предложение новой виртуальной категории аналогично описанному выше, но гораздо менее трудоёмко, так как фактически не потребуются перемещать порты. В этом случае единственные патчи, которые нужно включить в PR, — это добавление новой категории в `CATEGORIES` затронутых портов.

### 5.3.5. Предложение о реорганизации всех категорий

Иногда кто-то предлагает реорганизовать категории, используя либо двухуровневую структуру, либо какую-либо другую структуру ключевых слов. На сегодняшний день ни одно из этих предложений не было реализовано, потому что, хотя их очень легко выдвинуть, усилия, необходимые для переработки всей существующей коллекции портов в рамках любой реорганизации, пугают, мягко говоря. Пожалуйста, ознакомьтесь с историей этих предложений в архивах списка рассылки, прежде чем публиковать эту идею. Более того, будьте готовы к тому, что вас попросят предоставить рабочий прототип.

## 5.4. Файлы дистрибутива

Вторая часть Makefile описывает файлы, которые необходимо загрузить для сборки порта, и места, откуда их можно скачать.

### 5.4.1. DISTNAME

`DISTNAME` — это имя порта, используемое авторами программного обеспечения. По умолчанию `DISTNAME` имеет значение `_${PORTNAME}-${DISTVERSIONPREFIX}${DISTVERSION}${DISTVERSIONSUFFIX}`, а если не задано, `DISTVERSION` по умолчанию принимает значение `_${PORTVERSION}`, поэтому переопределяйте `DISTNAME` только при необходимости. `DISTNAME` используется только в двух случаях. Во-первых, список файлов дистрибутива (`DISTFILES`) по умолчанию имеет значение `_${DISTNAME}${EXTRACT_SUFFIX}`. Во-вторых, ожидается, что файл дистрибутива распакуется в подкаталог с именем `WRKSRCS`, который по умолчанию равен `work/${DISTNAME}`.

Некоторые названия дистрибутивов от поставщиков, которые не соответствуют схеме `_${PORTNAME}-${PORTVERSION}`, могут обрабатываться автоматически путем установки `DISTVERSIONPREFIX`, `DISTVERSION` и `DISTVERSIONSUFFIX`. `PORTVERSION` будет автоматически вычисляться из `DISTVERSION`.



Только одна из переменных `PORTVERSION` и `DISTVERSION` может быть установлена одновременно. Если `DISTVERSION` не определяет корректную `PORTVERSION`, не используйте `DISTVERSION`.

Если схема версий исходного проекта может быть преобразована в схему, совместимую с портами, установите некоторую переменную в версию исходного проекта, *не используйте* имя переменной `DISTVERSION`. Установите `PORTVERSION` в вычисленную версию на основе созданной вами переменной и задайте `DISTNAME` соответствующим образом.

Если схема версионирования вышестоящего проекта не может быть легко преобразована в значение, совместимое с портами, установите `PORTVERSION` в разумное значение и задайте `DISTNAME` как `PORTNAME` с дословной версией вышестоящего проекта.

#### Пример 10. Получение `PORTVERSION` вручную

BIND9 использует схему версионирования, несовместимую с версиями портов (в версиях используется `-`), и её нельзя получить с помощью `DISTVERSION`, так как после выпуска 9.9.9 выходят «уровни исправлений» в формате 9.9.9-P1. `DISTVERSION` преобразует это в 9.9.9.p1, что в схеме версионирования портов означает 9.9.9 pre-release 1, то есть версию, предшествующую 9.9.9, а не следующую за ней. Поэтому `PORTVERSION` вручную формируется из переменной `ISCVERSION`, чтобы получить 9.9.9p1.

Порядок, в котором система портов и `pkg` будут сортировать версии, проверяется с помощью аргумента `-t` из `pkg-version(8)`:

```
% pkg version -t 9.9.9 9.9.9.p1
> ①
% pkg version -t 9.9.9 9.9.9p1
< ②
```

- ① Знак `>` означает, что первый аргумент, переданный в `-t`, больше второго аргумента. 9.9.9 находится после 9.9.9.p1.
- ② Знак `<` означает, что первый аргумент, переданный в `-t`, меньше второго аргумента. 9.9.9 находится перед 9.9.9p1.

В файле Makefile порта, например `dns/bind99`, это достигается с помощью:

```
PORTNAME= bind
PORTVERSION= ${ISCVERSION:S/-P/P/:S/b/.b/:S/a/.a/:S/rc/.rc/}
CATEGORIES= dns net
MASTER_SITES= ISC/bind9/${ISCVERSION}
PKGNAME_SUFFIX= 99
DISTNAME= ${PORTNAME}-${ISCVERSION}

MAINTAINER= mat@FreeBSD.org
COMMENT= BIND DNS suite with updated DNSSEC and DNS64
WWW= https://www.isc.org/bind/
```

```
LICENSE= ISCL
```

```
# ISC releases things like 9.8.0-P1 or 9.8.1rc1, which our versioning does not  
like  
ISCVERSION= 9.9.9-P6
```

Определите версию вышестоящего пакета в `ISCVERSION`, с комментарием, объясняющим, почему это необходимо. Используйте `ISCVERSION` для получения совместимого с портами `PORTVERSION`. Используйте `ISCVERSION` напрямую для получения правильного URL для загрузки файла дистрибутива. Используйте `ISCVERSION` напрямую для именования дистрибутивного файла.

#### Пример 11. Получить `DISTNAME` из `PORTVERSION`

Время от времени имя файла дистрибутива имеет мало отношения или вообще никакого отношения к версии программного обеспечения.

В пакете `comms/kermit`, в файле дистрибутива присутствует только последний элемент версии:

```
PORTNAME= kermit  
PORTVERSION= 9.0.304  
CATEGORIES= comms ftp net  
MASTER_SITES= ftp://ftp.kermitproject.org/kermit/test/tar/  
DISTNAME= cku${PORTVERSION:E}-dev20
```

Модификатор `:E` `make(1)` возвращает суффикс переменной, в данном случае `304`. Файл дистрибутива корректно создаётся как `cku304-dev20.tar.gz`.

#### Пример 12. Экзотический случай 1

Иногда нет связи между названием программы, её версией и файлом дистрибутива, в котором она распространяется.

Из пакета `audio/libworkman`:

```
PORTNAME= libworkman  
PORTVERSION= 1.4  
CATEGORIES= audio  
MASTER_SITES= LOCAL/jim  
DISTNAME= ${PORTNAME}-1999-06-20
```

#### Пример 13. Экзотический случай 2

В пакете `comms/librs232` файл дистрибутива не имеет версии, поэтому необходимо

ИСПОЛЬЗОВАТЬ `DIST_SUBDIR`:

```
PORTNAME=      librs232
PORTVERSION=   20160710
CATEGORIES=    comms
MASTER_SITES=  http://www.teuniz.net/RS-232/
DISTNAME=      RS-232
DIST_SUBDIR=   ${PORTNAME}-${PORTVERSION}
```



`PKGNAMEPREFIX` и `PKGNAME_SUFFIX` не влияют на `DISTNAME`. Также обратите внимание, что если `WRKSRC` равно `${WRKDIR}/${DISTNAME}`, а исходный архив с исходным кодом называется иначе, чем `${PORTNAME}-${PORTVERSION}${EXTRACT_SUFFIX}`, оставьте `DISTNAME` без изменений — определение только `DISTFILES` проще, чем определение и `DISTNAME`, и `WRKSRC` (а возможно, и `EXTRACT_SUFFIX`).

## 5.4.2. MASTER\_SITES

Запишите именем каталога из FTP/HTTP-URL, указывающего на исходный tarball, в `MASTER_SITES`. Не забудьте завершающий слэш (!)

Макросы `make` будут пытаться использовать эту спецификацию для загрузки файла дистрибутива с помощью `FETCH`, если не смогут найти его уже в системе.

Рекомендуется включать в этот список несколько сайтов, желательно с разных континентов. Это обеспечит защиту от проблем в глобальной сети.



`MASTER_SITES` не должен быть пустым. Он должен указывать на реальный сайт, где размещены файлы дистрибутива. Он не может указывать на веб-архивы или кэшированные сайты с файлами дистрибутива FreeBSD. Единственное исключение из этого правила — порты, у которых нет файлов дистрибутива. Например, мета-порты не имеют файлов дистрибутива, поэтому `MASTER_SITES` не нужно задавать.

### 5.4.2.1. Использование переменных `MASTER_SITE_*`

Для популярных архивов, таких как SourceForge (`SOURCEFORGE`), GNU (`GNU`) или Perl CPAN (`PERL_CPAN`), доступны сокращённые обозначения. `MASTER_SITES` может использовать их напрямую:

```
MASTER_SITES=  GNU/make
```

Старый расширенный формат по-прежнему работает, но все порты были преобразованы в компактный формат. Расширенный формат выглядит следующим образом:

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

Эти значения и переменные определены в [Mk/bsd.sites.mk](#). Новые записи добавляются часто, поэтому обязательно проверяйте последнюю версию этого файла перед отправкой порта.



Для любой переменной `MASTER_SITE_FOO` можно использовать сокращение `FOO`. Например, используйте:

```
MASTER_SITES=  FOO
```

Если требуется `MASTER_SITE_SUBDIR`, используйте следующее:

```
MASTER_SITES=  FOO/bar
```

Некоторые имена `MASTER_SITE_*` довольно длинные, и для удобства использования были определены сокращения:

Таблица 3. Сокращения для макросов `MASTER_SITE_*`



Макрос	Сокращение
<code>PERL_CPAN</code>	<code>CPAN</code>
<code>GITHUB</code>	<code>GH</code>
<code>GITHUB_CLOUD</code>	<code>GHC</code>
<code>LIBREOFFICE_DEV</code>	<code>LODEV</code>
<code>NETLIB</code>	<code>NL</code>
<code>RUBYGEMS</code>	<code>RG</code>
<code>SOURCEFORGE</code>	<code>SF</code>

#### 5.4.2.2. Волшебные макросы `MASTER_SITES`

Существует несколько "волшебных" макросов для популярных сайтов с предсказуемой структурой каталогов. Для них достаточно использовать сокращение, и система автоматически выберет подкаталог. Например, для порта с именем `Stardict`, версии `1.2.3`, размещенного на SourceForge, добавьте следующую строку:

```
MASTER_SITES=  SF
```

подразумевает подкаталог с именем `/project/stardict/stardict/1.2.3`. Если подразумеваемый каталог указан неверно, его можно переопределить:

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

Это также можно записать как

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

Таблица 4. Волшебные макросы `MASTER_SITES`

Макрос	Предполагаемая поддиректория
<code>APACHE_COMMONS_BINARIES</code>	<code>\${PORTNAME:S,commons-,}</code>
<code>APACHE_COMMONS_SOURCE</code>	<code>\${PORTNAME:S,commons-,}</code>
<code>APACHE_JAKARTA</code>	<code>\${PORTNAME:S,-,/}/source</code>
<code>BERLIOS</code>	<code>\${PORTNAME:t1}.berlios</code>
<code>CHEESESHOP</code>	<code>source/\${DISTNAME:C/(.)*\1}/\${DISTNAME:C/(.)*-[0-9]*\1/}</code>
<code>CPAN</code>	<code>\${PORTNAME:C/-.*//}</code>
<code>DEBIAN</code>	<code>pool/main/\${PORTNAME:C/^(lib)?.*\1}/\${PORTNAME}</code>
<code>FARSIGHT</code>	<code>\${PORTNAME}</code>
<code>FESTIVAL</code>	<code>\${PORTREVISION}</code>
<code>GCC</code>	<code>releases/\${DISTNAME}</code>
<code>GENTOO</code>	<code>distfiles</code>
<code>GIMP</code>	<code>\${PORTNAME}/\${PORTVERSION:R}/</code>
<code>GH</code>	<code>\${GH_ACCOUNT}/\${GH_PROJECT}/tar.gz/\${GH_TAGNAME}?dummy=/</code>
<code>GHC</code>	<code>\${GH_ACCOUNT}/\${GH_PROJECT}/</code>
<code>GNOME</code>	<code>sources/\${PORTNAME}/\${PORTVERSION:C/^(\[0-9])*\1/}</code>
<code>GNU</code>	<code>\${PORTNAME}</code>
<code>GNUPG</code>	<code>\${PORTNAME}</code>
<code>GNU_ALPHA</code>	<code>\${PORTNAME}</code>
<code>HORDE</code>	<code>\${PORTNAME}</code>
<code>LODEV</code>	<code>\${PORTNAME}</code>
<code>MATE</code>	<code>\${PORTVERSION:C/^(\[0-9])*\1/}</code>
<code>MOZDEV</code>	<code>\${PORTNAME:t1}</code>
<code>NL</code>	<code>\${PORTNAME}</code>
<code>QT</code>	<code>archive/qt/\${PORTVERSION:R}</code>
<code>SAMBA</code>	<code>\${PORTNAME}</code>
<code>SAVANNAH</code>	<code>\${PORTNAME:t1}</code>
<code>SF</code>	<code>\${PORTNAME:t1}/\${PORTNAME:t1}/\${PORTVERSION}</code>

### 5.4.3. USE\_GITHUB

Если файл дистрибутива получен из определённого коммита или тега на [GitHub](#), для которого нет официально выпущенного файла, существует простой способ автоматически установить правильные значения `DISTNAME` и `MASTER_SITES`.



По состоянию на 2023-02-21 [GitHub](#) объявили, что загрузки исходного кода будут стабильными в течение года. Пожалуйста, переключитесь на ресурсы выпусков (release assets), а если они недоступны, запросите их создание у вышестоящих разработчиков.

Доступны следующие переменные:

Таблица 5. `USE_GITHUB` Описание

Переменная	Описание	По умолчанию
<code>GH_ACCOUNT</code>	Имя учётной записи пользователя GitHub, который размещает проект	<code>\${PORTNAME}</code>
<code>GH_PROJECT</code>	Название проекта на GitHub	<code>\${PORTNAME}</code>
<code>GH_TAGNAME</code>	Имя тега для загрузки (2.0.1, хэш, ...) Использование имени ветки здесь некорректно. Также можно использовать хэш идентификатора коммита для создания снимка состояния.	<code>\${DISTVERSIONPREFIX}\${DISTVERSION}\${DISTVERSIONSUFFIX}</code>
<code>GH_SUBDIR</code>	Когда программному обеспечению требуется дополнительный файл дистрибутива для извлечения в <code>\${WRKSR}</code> , можно использовать эту переменную. Примеры можно найти в <a href="#">Загрузка нескольких файлов из GitHub</a> для получения дополнительной информации.	(отсутствует)

Переменная	Описание	По умолчанию
<code>GH_TUPLE</code>	<code>GH_TUPLE</code> позволяет объединить <code>GH_ACCOUNT</code> , <code>GH_PROJECT</code> , <code>GH_TAGNAME</code> и <code>GH_SUBDIR</code> в одну переменную. Формат следующий: <code>account`:`project`:`tagname`:`group`/`subdir</code> . Часть <code>`/`subdir</code> является необязательной. Это полезно, когда требуется получить несколько проектов с GitHub.	



Не используйте `GH_TUPLE` для файла дистрибутива по умолчанию, так как у него нет значения по умолчанию.

#### Пример 14. Простое использование `USE_GITHUB`

При попытке создать порт для версии `1.2.7` `pkg` от пользователя FreeBSD на github, по адресу <https://github.com/freebsd/pkg/>, файл Makefile в итоге будет выглядеть следующим образом (незначительно сокращено для примера):

```
PORTNAME=  pkg
DISTVERSION=  1.2.7

USE_GITHUB= yes
GH_ACCOUNT= freebsd
```

Он автоматически получит `MASTER_SITES` установленным в `GH` и `WRKSRC` в `${WRKDIR}/pkg-1.2.7`.

#### Пример 15. Более полное использование `USE_GITHUB`

При попытке создать порт для самой последней версии `pkg` от пользователя FreeBSD на github, по адресу <https://github.com/freebsd/pkg/>, файл Makefile в итоге выглядит следующим образом (незначительно сокращено для примера):

```
PORTNAME=  pkg-devel
DISTVERSION=  1.3.0.a.20140411

USE_GITHUB= yes
GH_ACCOUNT= freebsd
GH_PROJECT= pkg
GH_TAGNAME= 6dbb17b
```

Он автоматически получит `MASTER_SITES` со значением `GH` и `WRKSRC` со значением `${WRKDIR}/pkg-6dbb17b`.

`20140411` — это дата коммита, указанного в `GH_TAGNAME`, а не дата редактирования файла Makefile или дата создания коммита.

#### Пример 16. Использование `USE_GITHUB` с `DISTVERSIONPREFIX`

Время от времени `GH_TAGNAME` немного отличается от `DISTVERSION`. Например, если версия `1.0.2`, то тег будет `v1.0.2`. В таких случаях можно использовать `DISTVERSIONPREFIX` или `DISTVERSIONSUFFIX`:

```
PORTNAME=  foo
DISTVERSIONPREFIX=  v
DISTVERSION=  1.0.2

USE_GITHUB=  yes
```

Он автоматически установит `GH_TAGNAME` в `v1.0.2`, в то время как `WRKSRC` останется `${WRKDIR}/foo-1.0.2`.

#### Пример 17. Использование `USE_GITHUB` при отсутствии версий у исходного проекта

Если никогда не было версии вышестоящего репозитория, не изобретайте её, например `0.1` или `1.0`. Создайте порт с `DISTVERSION` в формате `gYYYYMMDD`, где `g` означает Git, а `YYYYMMDD` представляет дату коммита, указанного в `GH_TAGNAME`.

```
PORTNAME=  bar
DISTVERSION=  g20140411

USE_GITHUB=  yes
GH_TAGNAME=  c472d66b
```

Это создаёт схему версионирования, которая увеличивается со временем и всё ещё находится до версии `0`. Подробности об использовании `pkg-version(8)` для сравнения версий смотрите в [этой секции](#):

```
% pkg version -t g20140411 0
<
```

Что означает, что использование `PORTPOCH` не потребуется, если вышестоящий проект решит сократить версии в будущем.

Пример 18. Использование `USE_GITHUB` для доступа к коммиту между двумя версиями

Если текущая версия программного обеспечения использует тег Git, и порт необходимо обновить до более новой промежуточной версии без тега, используйте `git-describe(1)`, чтобы определить версию для использования:

```
% git describe --tags f0038b1
v0.7.3-14-gf0038b1
```

`v0.7.3-14-gf0038b1` можно разделить на три части:

#### `v0.7.3`

Это последний тег Git, который появляется в истории коммитов перед запрошенным коммитом.

#### `-14`

Это означает, что запрошенный коммит `f0038b1` является 14-м коммитом после тега `v0.7.3`.

#### `-gf0038b1`

`-g` означает "Git", а `f0038b1` — это хеш коммита, на который указывает данная ссылка.

```
PORTNAME=  bar
DISTVERSIONPREFIX=  v
DISTVERSION=  0.7.3-14
DISTVERSIONSUFFIX=  -gf0038b1

USE_GITHUB=  yes
```

Это создаёт схему версионирования, которая увеличивается со временем (точнее, с коммитами) и не конфликтует с созданием версии `0.7.4`. Подробности об использовании `pkg-version(8)` для сравнения версий смотрите в [этой секции](#) :

```
% pkg version -t 0.7.3 0.7.3.14
<
% pkg version -t 0.7.3.14 0.7.4
<
```

Если запрошенный коммит совпадает с тегом, по умолчанию отображается более короткое описание. Полная версия эквивалентна:

```
% git describe --tags c66c71d
v0.7.3

% git describe --tags --long c66c71d
```

### 5.4.3.1. Извлечение нескольких файлов из GitHub

Фреймворк `USE_GITHUB` также поддерживает загрузку нескольких файлов дистрибутива из разных мест в GitHub. Он работает очень похоже на [Файлы дистрибуции или патчей из нескольких мест](#).

В `GH_ACCOUNT`, `GH_PROJECT` и `GH_TAGNAME` добавляются несколько значений. Каждому различному значению присваивается группа. Основное значение может не иметь группы или принадлежать группе `:DEFAULT`. Значение может быть опущено, если оно совпадает со значением по умолчанию, указанным в [описании USE\\_GITHUB](#).

`GH_TUPLE` также можно использовать, когда имеется множество файлов дистрибутива. Это помогает сохранять учётные данные, проект, имя тега и информацию о группе в одном месте.

Для каждой группы создаётся вспомогательная переменная `_${WRKSRC_group}`, содержащая каталог, в который был извлечён файл. Переменные `_${WRKSRC_group}` могут использоваться для перемещения каталогов во время `post-extract`, добавления в `CONFIGURE_ARGS` или любых других действий, необходимых для корректной сборки программного обеспечения.



Часть `:group` должна использоваться *только для одного* файла дистрибутива. Она служит уникальным ключом, и её повторное использование приведёт к перезаписи предыдущих значений.



Поскольку это всего лишь синтаксический сахар над `DISTFILES` и `MASTER_SITES`, имена групп должны соответствовать ограничениям на имена групп, описанным в [Файлы дистрибутивов или патчей из нескольких источников](#)

При получении нескольких файлов из GitHub иногда файл дистрибутива по умолчанию не загружается из GitHub. Чтобы отключить загрузку файла дистрибутива по умолчанию, установите:

```
USE_GITHUB= nodefault
```



При использовании `USE_GITHUB=nodefault` в Makefile необходимо указать `DISTFILES` в его [верхнем блоке](#). Определение должно быть следующим:

```
DISTFILES=    ${DISTNAME}${EXTRACT_SUFFIX}
```

*Пример 19. Использование `USE_GITHUB` с несколькими файлами дистрибутива*

Время от времени возникает необходимость загрузить более одного файла

дистрибутива. Например, когда вышестоящий репозиторий `git` использует подмодули. Это можно легко сделать с помощью групп в переменных `GH_*`:

```
PORTNAME=  foo
DISTVERSION=  1.0.2

USE_GITHUB= yes
GH_ACCOUNT= bar:icons,contrib
GH_PROJECT= foo-icons:icons foo-contrib:contrib
GH_TAGNAME= 1.0:icons fa579bc:contrib
GH_SUBDIR=  ext/icons:icons

CONFIGURE_ARGS= --with-contrib=${WRKSRC_contrib}
```

Это загрузит три файла дистрибутива с github. Стандартный берется из `foo/foo` и имеет версию `1.0.2`. Второй, с группой `icons`, берется из `bar/foo-icons` и имеет версию `1.0`. Третий берется из `bar/foo-contrib` и использует Git-коммит `fa579bc`. Файлы дистрибутива называются `foo-foo-1.0.2_GH0.tar.gz`, `bar-foo-icons-1.0_GH0.tar.gz` и `bar-foo-contrib-fa579bc_GH0.tar.gz`.

Все файлы дистрибутива извлекаются в `${WRKDIR}` в соответствующих подкаталогах. Основной файл по-прежнему извлекается в `${WRKSRC}`, в данном случае, `${WRKDIR}/foo-1.0.2`. Каждый дополнительный файл дистрибутива извлекается в `${WRKSRC_group}`. Здесь, для группы `icons`, он называется `${WRKSRC_icons}` и содержит `${WRKDIR}/foo-icons-1.0`. Файл с группой `contrib` называется `${WRKSRC_contrib}` и содержит `${WRKDIR}/foo-contrib-fa579bc`.

Система сборки программы ожидает найти иконки в подкаталоге `ext/icons` в её исходниках, поэтому используется `GH_SUBDIR`. `GH_SUBDIR` гарантирует, что `ext` существует, но `ext/icons` ещё не существует. Затем он выполняет следующее:

```
post-extract:
    @${MV} ${WRKSRC_icons} ${WRKSRC}/ext/icons
```

*Пример 20. Использование `USE_GITHUB` с несколькими файлами дистрибутива с помощью `GH_TUPLE`*

Это функционально эквивалентно [Использованию `USE\_GITHUB` с несколькими файлами дистрибутива](#), но с использованием `GH_TUPLE`:

```
PORTNAME=  foo
DISTVERSION=  1.0.2

USE_GITHUB= yes
GH_TUPLE=  bar:foo-icons:1.0:icons/ext/icons \
           bar:foo-contrib:fa579bc:contrib
```

```
CONFIGURE_ARGS= --with-contrib=${WRKSRC_contrib}
```

В предыдущем примере использовалась группировка с `bar:icons,contrib`. В `GH_TUPLE` присутствует избыточная информация, так как группировка невозможна.

*Пример 21. Как использовать `USE_GITHUB` с подмодулями Git?*

Порты, использующие GitHub в качестве вышестоящего репозитория, иногда применяют подмодули. Подробнее см. [git-submodule\(1\)](#).

Проблема с подмодулями заключается в том, что каждый из них является отдельным репозиторием. Таким образом, каждый из них должен быть загружен отдельно.

В качестве примера используем пакет [finance/moneymanagereX](#), его репозиторий на GitHub находится по адресу <https://github.com/moneymanagereX/moneymanagereX/>. В корне репозитория есть файл `.gitmodules`. Этот файл описывает все подмодули, используемые в данном репозитории, и перечисляет дополнительные необходимые репозитории. Этот файл покажет, какие дополнительные репозитории требуются:

```
[submodule "lib/wxsqlite3"]
  path = lib/wxsqlite3
  url = https://github.com/utelle/wxsqlite3.git
[submodule "3rd/mongoose"]
  path = 3rd/mongoose
  url = https://github.com/cesanta/mongoose.git
[submodule "3rd/LuaGlue"]
  path = 3rd/LuaGlue
  url = https://github.com/moneymanagereX/LuaGlue.git
[submodule "3rd/cgitemplate"]
  path = 3rd/cgitemplate
  url = https://github.com/moneymanagereX/html-template.git
[...]
```

Единственная информация, отсутствующая в этом файле, — это хэш коммита или тег, который следует использовать в качестве версии. Эта информация находится после клонирования репозитория:

```
% git clone --recurse-submodules
https://github.com/moneymanagereX/moneymanagereX.git
Cloning into 'moneymanagereX'...
remote: Counting objects: 32387, done.
[...]
Submodule '3rd/LuaGlue' (https://github.com/moneymanagereX/LuaGlue.git) registered
for path '3rd/LuaGlue'
Submodule '3rd/cgitemplate' (https://github.com/moneymanagereX/html-template.git)
registered for path '3rd/cgitemplate'
Submodule '3rd/mongoose' (https://github.com/cesanta/mongoose.git) registered for
```

```

path '3rd/mongoose'
Submodule 'lib/wxsqlite3' (https://github.com/utelle/wxsqlite3.git) registered for
path 'lib/wxsqlite3'
[...]
Cloning into
'/home/mat/work/freebsd/ports/finance/moneymanagerex/moneymanagerex/3rd/LuaGlue'..
.
Cloning into
'/home/mat/work/freebsd/ports/finance/moneymanagerex/moneymanagerex/3rd/cgitemplat
e'...
Cloning into
'/home/mat/work/freebsd/ports/finance/moneymanagerex/moneymanagerex/3rd/mongoose'.
..
Cloning into
'/home/mat/work/freebsd/ports/finance/moneymanagerex/moneymanagerex/lib/wxsqlite3'
...
[...]
Submodule path '3rd/LuaGlue': checked out
'c51d11a247ee4d1e9817dfa2a8da8d9e2f97ae3b'
Submodule path '3rd/cgitemplate': checked out
'cd434eeeb35904ebcd3d718ba29c281a649b192c'
Submodule path '3rd/mongoose': checked out
'2140e5992ab9a3a9a34ce9a281abf57f00f95cda'
Submodule path 'lib/wxsqlite3': checked out
'fb66eb230d8aed21dec273b38c7c054dcb7d6b51'
[...]
% cd moneymanagerex
% git submodule status
c51d11a247ee4d1e9817dfa2a8da8d9e2f97ae3b 3rd/LuaGlue (heads/master)
cd434eeeb35904ebcd3d718ba29c281a649b192c 3rd/cgitemplate (cd434ee)
2140e5992ab9a3a9a34ce9a281abf57f00f95cda 3rd/mongoose (6.2-138-g2140e59)
fb66eb230d8aed21dec273b38c7c054dcb7d6b51 lib/wxsqlite3 (v3.4.0)
[...]

```

Это также можно найти на GitHub. Каждый подкаталог, который является подмодулем, отображается как *директория @ хэш*, например, *mongoose @ 2140e59*.

Хотя получение информации из GitHub кажется более простым, данные, полученные с помощью `git submodule status`, будут более информативными. Например, здесь хеш коммита `lib/wxsqlite3 fb66eb2` соответствует `v3.4.0`. Оба варианта можно использовать взаимозаменяемо, но если доступен тег, предпочтительнее использовать его.

Теперь, когда вся необходимая информация собрана, можно написать Makefile (показаны только строки, связанные с GitHub):

```

PORTNAME= moneymanagerex
DISTVERSIONPREFIX= v

```

```

DISTVERSION= 1.3.0

USE_GITHUB= yes
GH_TUPLE= utelle:wxsqLite3:v3.4.0:wxsqLite3/lib/wxsqLite3 \
           moneymanagerex:LuaGlue:c51d11a:lua_glue/3rd/LuaGlue \
           moneymanagerex:html-template:cd434ee:html_template/3rd/cgitemplate \
           cesanta:mongoose:2140e59:mongoose/3rd/mongoose \
           [...]

```

#### 5.4.4. USE\_GITLAB

Подобно GitHub, если файл дистрибутива поставляется с [gitlab.com](https://gitlab.com) или использует программное обеспечение GitLab, эти переменные доступны для использования и могут потребовать установки.

Таблица 6. Описание `USE_GITLAB`

Переменная	Описание	По умолчанию
<code>GL_SITE</code>	Название сайта, на котором размещен проект GitLab	<a href="https://gitlab.com/">https://gitlab.com/</a>
<code>GL_ACCOUNT</code>	Имя учётной записи пользователя GitLab, размещающего проект	<code>\${PORTNAME}</code>
<code>GL_PROJECT</code>	Название проекта на GitLab	<code>\${PORTNAME}</code>
<code>GL_COMMIT</code>	Хэш коммита для загрузки. Должен быть полным 160-битным, 40-символьным шестнадцатеричным хэшем sha1. Это обязательная переменная для GitLab.	(нет)
<code>GL_SUBDIR</code>	Когда программному обеспечению требуется дополнительный файл дистрибутива для извлечения в <code>\${WRKSR}</code> , можно использовать эту переменную. Примеры можно найти в <a href="#">Загрузка нескольких файлов из GitLab</a> для получения дополнительной информации.	(отсутствует)

Переменная	Описание	По умолчанию
<code>GL_TUPLE</code>	<code>GL_TUPLE</code> позволяет объединить <code>GL_SITE</code> , <code>GL_ACCOUNT</code> , <code>GL_PROJECT</code> , <code>GL_COMMIT</code> и <code>GL_SUBDIR</code> в одну переменную. Формат записи: <code>сайт`:`учётная запись`:`проект`:`коммит`:`группа`/поддиректория</code> . Части <code>сайт`:`</code> и <code>`/поддиректория</code> являются необязательными. Это полезно, когда требуется загрузить данные из нескольких проектов GitLab.	

*Пример 22. Простое использование `USE_GITLAB`*

Пытаясь создать порт для версии `1.14` библиотеки `libsignon-glib` от пользователя `accounts-sso` на `gitlab.com`, по адресу <https://gitlab.com/accounts-sso/libsignon-glib/>, файл `Makefile` будет выглядеть следующим образом для загрузки дистрибутивных файлов:

```
PORTNAME= libsignon-glib
DISTVERSION= 1.14

USE_GITLAB= yes
GL_ACCOUNT= accounts-sso
GL_COMMIT= e90302e342bfd27bc8c9132ab9d0ea3d8723fd03
```

Он автоматически получит `MASTER_SITES`, установленный на `gitlab.com`, и `WRKSRC` на `${WRKDIR}/libsignon-glib-e90302e342bfd27bc8c9132ab9d0ea3d8723fd03-e90302e342bfd27bc8c9132ab9d0ea3d8723fd03`.

*Пример 23. Более полное использование `USE_GITLAB`*

Более полный пример использования вышеописанного, если порт не имеет версионирования и `foobar` принадлежит пользователю `foo` в проекте `bar` на самостоятельно размещенном сайте GitLab <https://gitlab.example.com/>, тогда `Makefile` будет выглядеть следующим образом для загрузки дистрибутивных файлов:

```
PORTNAME= foobar
DISTVERSION= g20170906

USE_GITLAB= yes
GL_SITE= https://gitlab.example.com
GL_ACCOUNT= foo
GL_PROJECT= bar
```

```
GL_COMMIT= 9c1669ce60c3f4f5eb43df874d7314483fb3f8a6
```

В нем будет установлено `MASTER_SITES` в `"https://gitlab.example.com"` и `WRKSRC` в `${WRKDIR}/bar-9c1669ce60c3f4f5eb43df874d7314483fb3f8a6-9c1669ce60c3f4f5eb43df874d7314483fb3f8a6`.



`20170906` — это дата коммита, указанного в `GL_COMMIT`, а не дата редактирования файла `Makefile` или дата коммита в дерево портов FreeBSD.



Протокол, порт и корневая директория веб-сервера `GL_SITE` могут быть изменены в той же переменной.

#### 5.4.4.1. Извлечение нескольких файлов из GitLab

Фреймворк `USE_GITLAB` также поддерживает загрузку нескольких файлов дистрибутивов из различных мест GitLab и сайтов, размещённых на GitLab. Он работает очень похоже на [Несколько файлов дистрибутивов или патчей из разных местоположений](#) и [Загрузка нескольких файлов из GitLab](#).

В `GL_SITE`, `GL_ACCOUNT`, `GL_PROJECT` и `GL_COMMIT` добавляются множественные значения. Каждое уникальное значение назначается группе. [Описание USE\\_GITLAB](#).

`GL_TUPLE` также может использоваться, когда имеется множество файлов дистрибутива. Это помогает хранить информацию о сайте, учётной записи, проекте, коммите и группе в одном месте.

Для каждой группы создаётся вспомогательная переменная `${WRKSRC_group}`, содержащая каталог, в который был извлечён файл. Переменные `${WRKSRC_group}` могут использоваться для перемещения каталогов во время `post-extract`, добавления в `CONFIGURE_ARGS` или любых других действий, необходимых для корректной сборки программного обеспечения.



Часть `:group` должна использоваться *только* для одного файла дистрибутива. Она служит уникальным ключом, и её повторное использование приведёт к перезаписи предыдущих значений.



Поскольку это всего лишь синтаксический сахар над `DISTFILES` и `MASTER_SITES`, имена групп должны соответствовать ограничениям на имена групп, описанным в [Файлы дистрибутивов или патчей из нескольких источников](#)

При получении нескольких файлов с использованием GitLab иногда файл дистрибутива по умолчанию не загружается с сайта GitLab. Чтобы отключить загрузку файла дистрибутива по умолчанию, установите:

```
USE_GITLAB= nodefault
```



При использовании `USE_GITLAB=nodefault`, Makefile должен устанавливать `DISTFILES` в своем [верхнем блоке](#). Определение должно быть следующим:

```
DISTFILES=    ${DISTNAME}${EXTRACT_SUFFIX}
```

*Пример 24. Использование `USE_GITLAB` с несколькими файлами дистрибутива*

Время от времени возникает необходимость загрузить более одного файла дистрибутива. Например, когда вышестоящий git-репозиторий использует подмодули. Это можно легко сделать с помощью групп в переменных `GL_*`:

```
PORTNAME=    foo
DISTVERSION= 1.0.2

USE_GITLAB=  yes
GL_SITE=     https://gitlab.example.com:9434/gitlab:icons
GL_ACCOUNT=  bar:icons,contrib
GL_PROJECT=  foo-icons:icons foo-contrib:contrib
GL_COMMIT=   c189207a55da45305c884fe2b50e086fcad4724b
             ae7368cab1ca7ca754b38d49da064df87968ffe4:icons
             9e4dd76ad9b38f33fdb417a4c01935958d5acd2a:contrib
GL_SUBDIR=   ext/icons:icons

CONFIGURE_ARGS= --with-contrib=${WRKSRC_contrib}
```

Это загрузит два файла дистрибутива с `gitlab.com` и один с `gitlab.example.com`, где размещается GitLab. По умолчанию файл берется из <https://gitlab.com/foo/foo>, а коммит — `c189207a55da45305c884fe2b50e086fcad4724b`. Второй файл, из группы `icons`, берется из <https://gitlab.example.com:9434/gitlab/bar/foo-icons>, а коммит — `ae7368cab1ca7ca754b38d49da064df87968ffe4`. Третий файл берется из <https://gitlab.com/bar/foo-contrib>, а коммит — `9e4dd76ad9b38f33fdb417a4c01935958d5acd2a`. Файлы дистрибутива называются `foo-foo-c189207a55da45305c884fe2b50e086fcad4724b_GL0.tar.gz`, `bar-foo-icons-ae7368cab1ca7ca754b38d49da064df87968ffe4_GL0.tar.gz` и `bar-foo-contrib-9e4dd76ad9b38f33fdb417a4c01935958d5acd2a_GL0.tar.gz`.

Все файлы дистрибутива извлекаются в `${WRKDIR}` в соответствующих подкаталогах. Основной файл по-прежнему извлекается в `${WRKSRC}`, в данном случае это `${WRKDIR}/foo-c189207a55da45305c884fe2b50e086fcad4724b-c189207a55da45305c884fe2b50e086fcad4724b`. Каждый дополнительный файл дистрибутива извлекается в `${WRKSRC_group}`. Здесь для группы `icons` он называется `${WRKSRC_icons}` и содержит `${WRKDIR}/foo-icons-ae7368cab1ca7ca754b38d49da064df87968ffe4-ae7368cab1ca7ca754b38d49da064df87968ffe4`. Файл группы `contrib` называется `${WRKSRC_contrib}` и содержит `${WRKDIR}/foo-contrib-9e4dd76ad9b38f33fdb417a4c01935958d5acd2a-9e4dd76ad9b38f33fdb417a4c01935958d5acd2a`.

Система сборки программного обеспечения ожидает найти иконки в подкаталоге `ext/icons` в своих исходниках, поэтому используется `GL_SUBDIR`. `GL_SUBDIR` гарантирует, что

ext существует, но ext/icons ещё не существует. Затем она выполняет следующее:

```
post-extract:
    @${MV} ${WRKSRC_icons} ${WRKSRC}/ext/icons
```

Пример 25. Использование `USE_GITLAB` с несколькими файлами дистрибуции с помощью `GL_TUPLE`

Это функционально эквивалентно [Использование `USE\_GITLAB` с несколькими файлами дистрибуции](#), но с использованием `GL_TUPLE`:

```
PORTNAME=  foo
DISTVERSION=  1.0.2

USE_GITLAB=  yes
GL_COMMIT=  c189207a55da45305c884fe2b50e086fcad4724b
GL_TUPLE=  https://gitlab.example.com:9434/gitlab:bar:foo-
icons:ae7368cab1ca7ca754b38d49da064df87968ffe4:icons/ext/icons \
        bar:foo-contrib:9e4dd76ad9b38f33fdb417a4c01935958d5acd2a:contrib

CONFIGURE_ARGS=  --with-contrib=${WRKSRC_contrib}
```

В предыдущем примере использовалась группировка с `bar:icons,contrib`. Некоторую избыточную информацию приходится указывать с `GL_TUPLE`, так как группировка невозможна.

### 5.4.5. `EXTRACT_SUFIX`

Если имеется один файл дистрибутива, и он использует нестандартное суффикс для указания механизма сжатия, установите `EXTRACT_SUFIX`.

Например, если файл дистрибутива был назван `foo.tar.gzip` вместо более привычного `foo.tar.gz`, напишите:

```
DISTNAME=  foo
EXTRACT_SUFIX=  .tar.gzip
```

`USES=tar[:xxx]`, `USES=lha` или `USES=zip` автоматически устанавливают `EXTRACT_SUFIX` в наиболее распространённые расширения архивов при необходимости, подробнее см. [Использование макросов `USES`](#). Если ни один из них не задан, `EXTRACT_SUFIX` по умолчанию принимает значение `.tar.gz`.



Как `EXTRACT_SUFIX` используется только в `DISTFILES`, следует задавать только один из них.

### 5.4.6. DISTFILES

Иногда названия файлов для загрузки не имеют ничего общего с именем порта. Например, файл может называться `source.tar.gz` или подобным образом. В других случаях исходный код приложения может быть разбит на несколько различных архивов, все из которых необходимо загрузить.

Если это так, установите `DISTFILES` как список разделённых пробелами файлов, которые необходимо загрузить.

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

Если явно не задано, `DISTFILES` по умолчанию равно `${DISTNAME}${EXTRACT_SUFFIX}`.

### 5.4.7. EXTRACT\_ONLY

Если необходимо извлечь только некоторые из `DISTFILES` — например, один из них является исходным кодом, а другой — сжатым документом — укажите имена файлов, которые нужно извлечь, в `EXTRACT_ONLY`.

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

Если ни один из `DISTFILES` не требует распаковки, установите `EXTRACT_ONLY` в пустую строку.

```
EXTRACT_ONLY=
```

### 5.4.8. PATCHFILES

Если порт требует дополнительных исправлений, доступных через FTP или HTTP, установите `PATCHFILES` в имена файлов, а `PATCH_SITES` — в URL каталога, содержащего их (формат такой же, как у `MASTER_SITES`).

Если патч не относится к корню исходного дерева (то есть к `WRKSRCS`), потому что содержит дополнительные пути, установите `PATCH_DIST_STRIP` соответствующим образом. Например, если все пути в патче имеют дополнительный префикс `foozolic-1.0/` перед именами файлов, задайте `PATCH_DIST_STRIP=-p1`.

Не беспокойтесь, если патчи сжаты; они будут автоматически распакованы, если их имена заканчиваются на `.Z`, `.gz`, `.bz2` или `.xz`.

Если патч распространяется вместе с другими файлами, такими как документация, в сжатом tarball, использование `PATCHFILES` невозможно. В таком случае добавьте имя и расположение tarball с патчами в `DISTFILES` и `MASTER_SITES`. Затем используйте `EXTRA_PATCHES`, чтобы указать на эти файлы, и `bsd.port.mk` автоматически применит их. В частности, *не* копируйте файлы патчей в `${PATCHDIR}`. Этот каталог может быть недоступен для записи.

Если есть несколько патчей и для них требуются разные значения параметра `strip`, его можно добавить рядом с именем патча в `PATCHFILES`, например:



```
PATCHFILES= patch1 patch2:-p1
```

Это не конфликтует с функцией группировки мастер-сайтов, добавление группы также работает:

```
PATCHFILES= patch2:-p1:source2
```



Tarball уже будет распакован вместе с обычными исходными кодами, поэтому нет необходимости явно его распаковывать, если это обычный сжатый tarball. Будьте особенно осторожны, чтобы не перезаписать существующие файлы в этом каталоге при ручной распаковке. Также не забудьте добавить команду для удаления скопированного патча в цель `pre-clean`.

### 5.4.9. Несколько файлов дистрибутивов или исправлений из нескольких местоположений

(Считайте, что это несколько «продвинутая тема»; тем, кто впервые читает этот документ, возможно, стоит сначала пропустить этот раздел).

Этот раздел содержит информацию о механизме загрузки, известном как `MASTER_SITES:n` и `MASTER_SITES_NN`. Мы будем называть этот механизм `MASTER_SITES:n`.

Небольшая предыстория. В OpenBSD есть удобная функция внутри `DISTFILES` и `PATCHFILES`, которая позволяет добавлять постфикс `:n` к файлам и патчам. Здесь `n` может быть любым словом, содержащим `[0-9a-zA-Z_]`, и обозначать группу. Например:

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

В OpenBSD файл дистрибутива `alpha` будет связан с переменной `MASTER_SITES0`, а не с нашей общей `MASTER_SITES`, а `beta` — с `MASTER_SITES1`.

Это очень интересная функция, которая может сократить бесконечные поиски нужного сайта для загрузки.

Представьте 2 файла в `DISTFILES` и 20 сайтов в `MASTER_SITES`, причём сайты медленные как черепаха, где `beta` есть на всех сайтах из `MASTER_SITES`, а `alpha` можно найти только на 20-м сайте. Было бы так обидно проверять их все, если бы сопровождающий знал это заранее, не так ли? Не самое лучшее начало для чудесных выходов!

Теперь, когда вы поняли идею, представьте больше `DISTFILES` и больше `MASTER_SITES`.

Безусловно, наш "мастер по исследованию distfiles" оценил бы снижение нагрузки на сеть, которое это принесло бы.

В следующих разделах будет приведена информация о реализации этой идеи в FreeBSD. Мы немного улучшили концепцию OpenBSD.



Имена групп не могут содержать дефисы (-), более того, они не могут содержать любые символы вне диапазона [a-zA-Z0-9\_]. Это связано с тем, что, хотя `make(1)` допускает использование имён переменных с дефисами, `sh(1)` — нет.

#### 5.4.9.1. Упрощенная информация

В этом разделе объясняется, как быстро настроить детализированное получение нескольких файлов дистрибутивов и патчей с разных сайтов и подкаталогов. Здесь описывается случай упрощённого использования `MASTER_SITES:n`. Этого будет достаточно для большинства сценариев. Более подробная информация доступна в [Подробная Информация](#).

Некоторые приложения состоят из нескольких распространяемых файлов, которые необходимо загрузить с различных сайтов. Например, Ghostscript включает основную часть программы и множество драйверов, используемых в зависимости от принтера пользователя. Некоторые из этих драйверов поставляются вместе с основной частью, но многие другие необходимо загружать с различных сайтов.

Для поддержки этого, каждая запись в `DISTFILES` может сопровождаться двоеточием и "именем группы". Затем каждый сайт, указанный в `MASTER_SITES`, сопровождается двоеточием и группой, которая указывает, какие файлы дистрибутива загружаются с данного сайта.

Например, рассмотрим приложение, исходный код которого разделён на две части: `source1.tar.gz` и `source2.tar.gz`, которые необходимо загрузить с двух разных сайтов. В Makefile порта будут присутствовать строки, подобные [Упрощённое использование MASTER\\_SITES:n с одним файлом на сайт](#).

*Пример 26. Упрощённое использование `MASTER_SITES:n` с одним файлом на сайт*

```
MASTER_SITES= ftp://ftp1.example.com/:source1 \  
              http://www.example.com/:source2  
DISTFILES= source1.tar.gz:source1 \  
           source2.tar.gz:source2
```

Несколько файлов дистрибутивов могут принадлежать одной группе. Продолжая предыдущий пример, предположим, что существует третий файл дистрибутива `source3.tar.gz`, который загружается с `ftp.example2.com`. Тогда Makefile будет записан, как показано в [Упрощённое использование MASTER\\_SITES:n с несколькими файлами на один сайт](#).

```
MASTER_SITES= ftp://ftp.example.com/:source1 \  
               http://www.example.com/:source2  
DISTFILES= source1.tar.gz:source1 \  
            source2.tar.gz:source2 \  
            source3.tar.gz:source2
```

### 5.4.9.2. Подробная информация

Хорошо, значит, предыдущий пример не отражал потребности нового порта? В этом разделе мы подробно объясним, как работает механизм детализированного получения `MASTER_SITES:n` и как его можно использовать.

1. Элементы могут иметь постфикс `:n`, где `n` — это ```, то есть `_n_` концептуально может быть любой буквенно-цифровой строкой, но пока мы ограничим её ``[a-zA-Z][0-9a-zA-Z_]``.

Более того, сравнение строк чувствительно к регистру; то есть, `n` отличается от `N`.

Однако эти слова не могут использоваться для постфиксных целей, так как имеют специальное значение: `default`, `all` и `ALL` (они используются внутри системы, см. [ii](#)). Кроме того, `DEFAULT` является словом специального назначения (проверьте пункт [3](#)).

2. Элементы с постфиксом `:n` принадлежат группе `n`, `:m` — группе `m` и так далее.
3. Элементы без постфикса не принадлежат к группам, все они относятся к специальной группе `DEFAULT`. Элементы с постфиксом `DEFAULT` избыточны, за исключением случаев, когда элемент одновременно принадлежит и к `DEFAULT`, и к другим группам (см. пункт [5](#)).

Эти примеры эквивалентны, но первый предпочтительнее:

```
MASTER_SITES= alpha
```

```
MASTER_SITES= alpha:DEFAULT
```

4. Группы не являются исключительными, элемент может принадлежать нескольким разным группам одновременно, а группа может содержать несколько разных элементов или не содержать их вовсе.
5. Когда элемент принадлежит нескольким группам одновременно, используйте оператор запятой `(,)`.

Вместо повторения несколько раз, каждый раз с разным постфиксом, мы можем перечислить несколько групп сразу в одном постфиксе. Например, `:m,n,o` обозначает элемент, принадлежащий группам `m`, `n` и `o`.

Все эти примеры эквивалентны, но последний является предпочтительным:

```
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. Все сайты в заданной группе сортируются согласно `MASTER_SORT_AWK`. Все группы в `MASTER_SITES` и `PATCH_SITES` также сортируются.
7. Семантика групп может использоваться в любых переменных `MASTER_SITES`, `PATCH_SITES`, `MASTER_SITE_SUBDIR`, `PATCH_SITE_SUBDIR`, `DISTFILES` и `PATCHFILES` согласно следующему синтаксису:
  - а. Все элементы `MASTER_SITES`, `PATCH_SITES`, `MASTER_SITE_SUBDIR` и `PATCH_SITE_SUBDIR` должны заканчиваться символом дробной черты `/`. Если элементы принадлежат к какой-либо группе, постфикс группы `:n` должен следовать сразу после завершающего символа `/`. Механизм `MASTER_SITES:n` полагается на наличие завершающего символа `/`, чтобы избежать путаницы между элементами, где `:n` является допустимой частью элемента, и случаями, где `:n` обозначает группу `n`. В целях совместимости, поскольку ранее завершающий символ `/` не требовался в элементах `MASTER_SITE_SUBDIR` и `PATCH_SITE_SUBDIR`, если символ, непосредственно предшествующий постфиксу, не является `/`, то `:n` будет считаться допустимой частью элемента, а не постфиксом группы, даже если элемент оканчивается на `:n`. См. оба раздела [Подробное использование MASTER\\_SITES:n в MASTER\\_SITE\\_SUBDIR](#) и [Подробное использование MASTER\\_SITES:n с оператором запятой](#).

*Пример 28. Подробное использование `MASTER_SITES:n` в `MASTER_SITE_SUBDIR`*

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

- Каталоги в группе `DEFAULT` → `old:n`
- Каталоги в группе `NEW` → `new`

*Пример 29. Подробное использование `MASTER_SITES:n` с оператором запятой, несколькими файлами, сайтами и подкаталогами*

```
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/:group8
```

```
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
```

Предыдущий пример приводит к такой детализированной загрузке файлов. Сайты перечислены в точном порядке их использования.

- file1 будет загружен из
  - **MASTER\_SITE\_OVERRIDE**
  - <http://site1/directory-trial:1/>
  - <http://site1/directory-one/>
  - <http://site1/directory/>
  - <http://site2/>
  - <http://site7/>
  - **MASTER\_SITE\_BACKUP**
- file2 будет загружен точно так же, как file1, поскольку они оба принадлежат к одной и той же группе
  - **MASTER\_SITE\_OVERRIDE**
  - <http://site1/directory-trial:1/>
  - <http://site1/directory-one/>
  - <http://site1/directory/>
  - <http://site2/>
  - <http://site7/>
  - **MASTER\_SITE\_BACKUP**
- file3 будет загружен из
  - **MASTER\_SITE\_OVERRIDE**
  - <http://site3/>
  - **MASTER\_SITE\_BACKUP**
- file4 будет загружен из
  - **MASTER\_SITE\_OVERRIDE**
  - <http://site4/>
  - <http://site5/>
  - <http://site6/>

- <http://site7/>
- <http://site8/directory-one/>
- MASTER\_SITE\_BACKUP
- file5 будет загружен из
  - MASTER\_SITE\_OVERRIDE
  - MASTER\_SITE\_BACKUP
- file6 будет получен из
  - MASTER\_SITE\_OVERRIDE
  - <http://site8/>
  - MASTER\_SITE\_BACKUP

8. Как сгруппировать один из специальных макросов из `bsd.sites.mk`, например, `SourceForge (SF)`?

Это максимально упрощено. См. [Подробное использование MASTER\\_SITES:n с SourceForge \(SF\)](#).

*Пример 30. Подробное использование MASTER\_SITES:n с SourceForge (SF)*

```
MASTER_SITES= http://site1/ SF/something/1.0:sourceforge,TEST
DISTFILES= something.tar.gz:sourceforge
```

`something.tar.gz` будет загружен со всех сайтов в пределах SourceForge.

9. Как использовать это с `PATCH*`?

Все примеры были выполнены с `MASTER*`, но они работают точно так же для `PATCH*`, как можно увидеть в [Упрощённое использование MASTER\\_SITES:n с PATCH\\_SITES](#).

*Пример 31. Упрощённое использование MASTER\_SITES:n с PATCH\_SITES*

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

### 5.4.9.3. Что меняется для портов? Что остается неизменным?

- i. Все текущие порты остаются без изменений. Функция `MASTER_SITES:n` активируется только при наличии элементов с постфиксом `:n`, соответствующих указанному выше синтаксическим правилам, в частности, как показано в пункте 7.
- ii. Порты сохраняют те же цели: `checksum`, `makesum`, `patch`, `configure`, `build` и т.д., за исключением очевидных случаев: `do-fetch`, `fetch-list`, `master-sites` и `patch-sites`.

- `do-fetch`: развертывает новую группировку с постфиксом `DISTFILES` и `PATCHFILES` с соответствующими групповыми элементами в `MASTER_SITES` и `PATCH_SITES`, которые используют соответствующие групповые элементы в `MASTER_SITE_SUBDIR` и `PATCH_SITE_SUBDIR`. Проверьте [Подробное использование MASTER\\_SITES:n с оператором запятой](#).
- `fetch-list`: работает как старый `fetch-list`, за исключением того, что группировка происходит так же, как в `do-fetch`.
- `master-sites` и `patch-sites`: (несовместимо с более старыми версиями) возвращают только элементы группы `DEFAULT`; фактически они выполняют цели `master-sites-default` и `patch-sites-default` соответственно.

Кроме того, предпочтительнее использовать цель `master-sites-all` или `patch-sites-all`, чем напрямую проверять `MASTER_SITES` или `PATCH_SITES`. Кроме того, прямая проверка не гарантирует работу в будущих версиях. Для получения дополнительной информации об этих новых целях портов см. пункт [B](#).

### iii. Новые цели портов

- Существуют цели `master-sites-n` и `patch-sites-n`, которые будут выводить элементы соответствующей группы `n` в `MASTER_SITES` и `PATCH_SITES` соответственно. Например, и `master-sites-DEFAULT`, и `patch-sites-DEFAULT` вернут элементы группы `DEFAULT`, `master-sites-test` и `patch-sites-test` — группы `test`, и так далее.
- Существуют новые цели `master-sites-all` и `patch-sites-all`, которые выполняют работу старых `master-sites` и `patch-sites`. Они возвращают элементы всех групп, как если бы они все принадлежали одной группе, с оговоркой, что перечисляется столько же `MASTER_SITE_BACKUP` и `MASTER_SITE_OVERRIDE`, сколько определено групп в `DISTFILES` или `PATCHFILES`; соответственно для `master-sites-all` и `patch-sites-all`.

#### 5.4.10. DIST\_SUBDIR

Не допускайте захламления портом каталога `/usr/ports/distfiles`. Если порт требует загрузки большого количества файлов или содержит файл с именем, которое может конфликтовать с другими портами (например, `Makefile`), установите `DIST_SUBDIR` в имя порта (подойдут `${PORTNAME}` или `${PKGNAMEPREFIX}${PORTNAME}`). Это изменит `DISTDIR` со значения по умолчанию `/usr/ports/distfiles` на `/usr/ports/distfiles/${DIST_SUBDIR}`, фактически помещая все необходимые для порта файлы в этот подкаталог.

Также будет проверяться подкаталог с тем же именем на основном резервном сайте по адресу <http://distcache.FreeBSD.org> (Явное указание `DISTDIR` в `Makefile` не решит эту задачу, поэтому используйте `DIST_SUBDIR`.)



Это не влияет на сайты в `MASTER_SITES`, определённые в `Makefile`.

## 5.5. MAINTAINER

Установите здесь свой адрес электронной почты. Пожалуйста. :-)

Только один адрес без комментария допускается в качестве значения `MAINTAINER`.

Используемый формат: `user@hostname.domain`. Пожалуйста, не включайте в эту запись описательный текст, например, настоящее имя. Это только вносит путаницу в инфраструктуру Ports и большинство инструментов, которые её используют.

Ответственный за поддержку порта обязан поддерживать порт в актуальном состоянии и обеспечивать его корректную работу. Подробное описание обязанностей ответственного за поддержку порта приведено в разделе [Задача для сопровождающих портов](#).



Сопровождающий добровольно поддерживает порт в рабочем состоянии. Сопровождающие несут основную ответственность за свои порты, но не имеют исключительных прав на них. Порты существуют для пользы сообщества и, по сути, принадлежат сообществу. Это означает, что люди, не являющиеся сопровождающими, также могут вносить изменения в порт. Крупные изменения в коллекции портов могут потребовать правок во многих портах. Команда управления портами FreeBSD или члены других команд могут изменять порты для исправления проблем с зависимостями или других проблем, таких как обновление версии динамической библиотеки.

Некоторые типы исправлений имеют "автоматическое согласование" от Группы Менеджеров Древа Портов FreeBSD [<portmgr@FreeBSD.org>](mailto:portmgr@FreeBSD.org), что позволяет любому коммиттеру исправлять эти категории проблем в любом порте. Такие исправления не требуют одобрения от сопровождающего.

Автоматическое согласование для большинства портов применяется к исправлениям, таким как изменения инфраструктуры, или тривиальным и *проверенным* исправлениям сборки и выполнения. Текущий список доступен в [разделе Портов Руководства коммиттера](#).

Другие изменения в порте будут отправлены сопровождающему на проверку и утверждение перед внесением. Если сопровождающий не отвечает на запрос об обновлении в течение двух недель (за исключением основных государственных праздников), это считается превышением времени ожидания сопровождающего, и обновление может быть внесено без его явного одобрения. Если сопровождающий не отвечает в течение трех месяцев или если произошло три последовательных превышения времени ожидания, то сопровождающий считается отсутствующим без уведомления, и все его порты могут быть возвращены в общий пул. Исключениями являются порты, сопровождаемые Группой Менеджеров Древа Портов FreeBSD [<portmgr@FreeBSD.org>](mailto:portmgr@FreeBSD.org) или Группой Офицеров Безопасности [<security-officer@FreeBSD.org>](mailto:security-officer@FreeBSD.org). Никакие несанкционированные изменения не могут быть внесены в порты, сопровождаемые этими группами.

Мы оставляем за собой право изменять представленные сопровождающим материалы, чтобы лучше соответствовать существующим политикам и стилю Коллекции портов, без явного одобрения отправителя или сопровождающего. Кроме того, масштабные инфраструктурные изменения могут привести к модификации порта без согласия сопровождающего. Подобные изменения никогда не повлияют на функциональность порта.

Группа Менеджеров Древа Портов FreeBSD [<portmgr@FreeBSD.org>](mailto:portmgr@FreeBSD.org) оставляет за собой

право отозвать или изменить права сопровождающего по любой причине, а Группа Офицеров Безопасности <[security-officer@FreeBSD.org](mailto:security-officer@FreeBSD.org)> оставляет за собой право отозвать или изменить права сопровождающего по соображениям безопасности.

## 5.6. COMMENT

Комментарий — это однострочное описание порта, отображаемое командой `pkg info`. При составлении придерживайтесь следующих правил:

1. Строка COMMENT должна быть не длиннее 70 символов.
2. Не включайте название пакета или номер версии программного обеспечения.
3. Комментарий должен начинаться с заглавной буквы и заканчиваться без точки.
4. Не начинайте с неопределённого артикля (то есть A или An).
5. Пишите названия с заглавной буквы, например: Apache, JavaScript или Perl.
6. Используйте запятую для списков слов: "green, red, and blue."
7. Проверяйте на наличие орфографических ошибок.

Вот пример:

```
COMMENT=    Cat chasing a mouse all over the screen
```

Переменная COMMENT следует сразу за переменной MAINTAINER в файле Makefile.

## 5.7. Веб-сайт проекта

Каждый порт должен указывать на веб-сайт, предоставляющий дополнительную информацию о программном обеспечении.

Везде, где это возможно, следует использовать официальный сайт проекта, поддерживаемый разработчиками программного обеспечения.

```
WWW=        https://ffmpeg.org/
```

Но это также может быть каталог или ресурс в репозитории исходного кода:

```
WWW=        https://sourceforge.net/projects/mpd/
```

Переменная WWW следует сразу за переменной COMMENT в файле Makefile.

Если один и тот же контент доступен по HTTP и HTTPS, следует использовать URL, начинающийся с `https://`. Если URI является корнем веб-сайта или директории, он должен заканчиваться косой чертой.

Эта информация ранее размещалась в последней строке файла pkg-descr. Она была перенесена в Makefile для удобства обслуживания и обработки. Наличие строки WWW: в конце файла pkg-descr считается устаревшим.

## 5.8. Лицензии

Каждый порт должен содержать документацию о лицензии, под которой он распространяется. Если лицензия не одобрена OSI, необходимо также указать любые ограничения на распространение.

### 5.8.1. LICENSE

Краткое название лицензии или лицензий, если применяется более одной лицензии.

Если это одна из лицензий, перечисленных в [Предопределенный список лицензий](#), можно задать только переменные LICENSE\_FILE и LICENSE\_DISTFILES.

Если это лицензия, которая не определена в рамках портов (см. [Список предопределённых лицензий](#)), необходимо задать LICENSE\_PERMS и LICENSE\_NAME, а также LICENSE\_FILE или LICENSE\_TEXT. Также можно задать LICENSE\_DISTFILES и LICENSE\_GROUPS, но это не обязательно.

Предопределенные лицензии показаны в [Список предопределенных лицензий](#). Текущий список всегда доступен в Mk/bsd.licenses.db.mk.

*Пример 32. Простейшее использование, предопределённые лицензии*

Когда в файле README какого-либо программного обеспечения указано: «Данное программное обеспечение распространяется на условиях GNU Lesser General Public License, опубликованной Free Software Foundation; либо версии 2.1 Лицензии, либо (по вашему выбору) любой более поздней версии», но сам файл лицензии не предоставлен, используйте следующее:

```
LICENSE=    LGPL21+
```

Когда программное обеспечение предоставляет файл лицензии, используйте это:

```
LICENSE=    LGPL21+
LICENSE_FILE=  ${WRKSRC}/COPYING
```

Для предопределённых лицензий права по умолчанию: dist-mirror dist-sell pkg-mirror pkg-sell auto-accept.

*Таблица 7. Предопределенный список лицензий*

Короткое имя	Имя	Группа	Разрешения
AGPLv3	Универсальная общественная лицензия GNU Affero версии 3	FSF GPL OSI	(по умолчанию)
AGPLv3+	Универсальная общественная лицензия GNU Affero версии 3 (или позднее)	FSF GPL OSI	(по умолчанию)
APACHE10	Apache License 1.0	FSF	(по умолчанию)
APACHE11	Apache License 1.1	FSF OSI	(по умолчанию)
APACHE20	Apache License 2.0	FSF OSI	(по умолчанию)
ART10	Художественная лицензия версия 1.0	OSI	(по умолчанию)
ART20	Художественная лицензия версии 2.0	FSF GPL OSI	(по умолчанию)
ARTPERL10	Художественная лицензия (perl) версия 1.0	OSI	(по умолчанию)
BSD	Лицензия BSD, общая версия (устарела)	FSF OSI COPYFREE	(по умолчанию)
BSD2CLAUSE	BSD 2-пунктная лицензия "Упрощенная"	FSF OSI COPYFREE	(по умолчанию)
BSD3CLAUSE	BSD 3-пунктная лицензия "Новая" или "Пересмотренная"	FSF OSI COPYFREE	(по умолчанию)
BSD4CLAUSE	BSD 4-пунктная лицензия "Оригинальная" или "Старая"	FSF	(по умолчанию)
BSL	Лицензия программного обеспечения Boost	FSF OSI COPYFREE	(по умолчанию)
CC-BY-1.0	Creative Commons с указанием авторства 1.0		(по умолчанию)
CC-BY-2.0	Creative Commons с указанием авторства 2.0		(по умолчанию)

Короткое имя	Имя	Группа	Разрешения
CC-BY-2.5	Creative Commons с указанием авторства 2.5		(по умолчанию)
CC-BY-3.0	Creative Commons с указанием авторства 3.0		(по умолчанию)
CC-BY-4.0	Creative Commons с указанием авторства 4.0		(по умолчанию)
CC-BY-NC-1.0	Creative Commons с указанием авторства – некоммерческая 1.0		dist-mirrorpkg-mirrorauto-accept
CC-BY-NC-2.0	Creative Commons с указанием авторства – некоммерческая 2.0		dist-mirrorpkg-mirrorauto-accept
CC-BY-NC-2.5	Creative Commons с указанием авторства – некоммерческая 2.5		dist-mirrorpkg-mirrorauto-accept
CC-BY-NC-3.0	Creative Commons с указанием авторства – некоммерческая 3.0		dist-mirrorpkg-mirrorauto-accept
CC-BY-NC-4.0	Creative Commons с указанием авторства – некоммерческая 4.0		dist-mirrorpkg-mirrorauto-accept
CC-BY-NC-ND-1.0	Creative Commons с указанием авторства – некоммерческая – без производных 1.0		dist-mirrorpkg-mirrorauto-accept
CC-BY-NC-ND-2.0	Creative Commons с указанием авторства – некоммерческая – без производных 2.0		dist-mirrorpkg-mirrorauto-accept
CC-BY-NC-ND-2.5	Creative Commons с указанием авторства – некоммерческая – без производных 2.5		dist-mirrorpkg-mirrorauto-accept
CC-BY-NC-ND-3.0	Creative Commons с указанием авторства – некоммерческая – без производных 3.0		dist-mirrorpkg-mirrorauto-accept

Короткое имя	Имя	Группа	Разрешения
CC-BY-NC-ND-4.0	Creative Commons с указанием авторства – некоммерческая – без производных 4.0		dist-mirrorpkg-mirrorauto-accept
CC-BY-NC-SA-1.0	Creative Commons с указанием авторства – некоммерческая – на тех же условиях 1.0		dist-mirrorpkg-mirrorauto-accept
CC-BY-NC-SA-2.0	Creative Commons с указанием авторства – некоммерческая – на тех же условиях 2.0		dist-mirrorpkg-mirrorauto-accept
CC-BY-NC-SA-2.5	Creative Commons с указанием авторства – некоммерческая – на тех же условиях 2.5		dist-mirrorpkg-mirrorauto-accept
CC-BY-NC-SA-3.0	Creative Commons с указанием авторства – некоммерческая – на тех же условиях 3.0		dist-mirrorpkg-mirrorauto-accept
CC-BY-NC-SA-4.0	Creative Commons с указанием авторства – некоммерческая – на тех же условиях 4.0		dist-mirrorpkg-mirrorauto-accept
CC-BY-ND-1.0	Creative Commons с указанием авторства – без производных 1.0		(по умолчанию)
CC-BY-ND-2.0	Creative Commons с указанием авторства – без производных 2.0		(по умолчанию)
CC-BY-ND-2.5	Creative Commons с указанием авторства – без производных 2.5		(по умолчанию)
CC-BY-ND-3.0	Creative Commons с указанием авторства – без производных 3.0		(по умолчанию)
CC-BY-ND-4.0	Creative Commons с указанием авторства – без производных 4.0		(по умолчанию)

Короткое имя	Имя	Группа	Разрешения
CC-BY-SA-1.0	Creative Commons с указанием авторства – на тех же условиях 1.0		(по умолчанию)
CC-BY-SA-2.0	Creative Commons с указанием авторства – на тех же условиях 2.0		(по умолчанию)
CC-BY-SA-2.5	Creative Commons с указанием авторства – на тех же условиях 2.5		(по умолчанию)
CC-BY-SA-3.0	Creative Commons с указанием авторства – на тех же условиях 3.0		(по умолчанию)
CC-BY-SA-4.0	Creative Commons с указанием авторства – на тех же условиях 4.0		(по умолчанию)
CC0-1.0	Creative Commons Zero v1.0 Universal (Отказ от прав 1.0 Универсальная)	FSF GPL COPYFREE	(по умолчанию)
CDDL	Лицензия на совместную разработку и распространение	FSF OSI	(по умолчанию)
CPAL-1.0	Публичная лицензия общего распространения с указанием авторства	FSF OSI	(по умолчанию)
ClArtistic	Уточнённая художественная лицензия	FSF GPL OSI	(по умолчанию)
EPL	Публичная лицензия Eclipse	FSF OSI	(по умолчанию)
GFDL	GNU Свободная лицензия на документацию	FSF	(по умолчанию)

Короткое имя	Имя	Группа	Разрешения
GMGPL	Модифицированная Общедоступная лицензия GNAT	FSF GPL OSI	(по умолчанию)
GPLv1	Универсальная общественная лицензия GNU версии 1	FSF GPL OSI	(по умолчанию)
GPLv1+	Универсальная общественная лицензия GNU версии 1 (или более поздняя)	FSF GPL OSI	(по умолчанию)
GPLv2	Универсальная общественная лицензия GNU версии 2	FSF GPL OSI	(по умолчанию)
GPLv2+	Универсальная общественная лицензия GNU версии 2 (или более поздняя)	FSF GPL OSI	(по умолчанию)
GPLv3	Универсальная общественная лицензия GNU версии 3	FSF GPL OSI	(по умолчанию)
GPLv3+	Универсальная общественная лицензия GNU версии 3 (или более поздняя)	FSF GPL OSI	(по умолчанию)
GPLv3RLE	Исключение для библиотеки времени выполнения GNU GPL версии 3	FSF GPL OSI	(по умолчанию)
GPLv3RLE+	Исключение для библиотеки времени выполнения GNU GPL версии 3 (или более поздняя)	FSF GPL OSI	(по умолчанию)
ISCL	Лицензия Internet Systems Consortium	FSF GPL OSI COPYFREE	(по умолчанию)
LGPL20	Общедоступная лицензия GNU для библиотек, версия 2.0	FSF GPL OSI	(по умолчанию)

Короткое имя	Имя	Группа	Разрешения
LGPL20+	Общедоступная лицензия GNU для библиотек, версия 2.0 (или более поздняя)	FSF GPL OSI	(по умолчанию)
LGPL21	Универсальная общественная лицензия GNU ограниченного применения, версия 2.1	FSF GPL OSI	(по умолчанию)
LGPL21+	Универсальная общественная лицензия GNU ограниченного применения, версия 2.1 (или более поздняя)	FSF GPL OSI	(по умолчанию)
LGPL3	Универсальная общественная лицензия GNU ограниченного применения, версия 3	FSF GPL OSI	(по умолчанию)
LGPL3+	Универсальная общественная лицензия GNU ограниченного применения, версия 3 (или более поздней)	FSF GPL OSI	(по умолчанию)
LPPL10	Публичная лицензия проекта LaTeX, версия 1.0	FSF OSI	dist-mirror dist-sell
LPPL11	Публичная лицензия проекта LaTeX, версия 1.1	FSF OSI	dist-mirror dist-sell
LPPL12	Публичная лицензия проекта LaTeX, версия 1.2	FSF OSI	dist-mirror dist-sell
LPPL13	Публичная лицензия проекта LaTeX, версия 1.3	FSF OSI	dist-mirror dist-sell

Короткое имя	Имя	Группа	Разрешения
LPPL13a	Публичная лицензия проекта LaTeX, версия 1.3a	FSF OSI	dist-mirror dist-sell
LPPL13b	Публичная лицензия проекта LaTeX, версия 1.3b	FSF OSI	dist-mirror dist-sell
LPPL13c	Публичная лицензия проекта LaTeX, версия 1.3c	FSF OSI	dist-mirror dist-sell
MIT	Лицензия MIT / Лицензия X11	COPYFREE FSF GPL OSI	(по умолчанию)
MPL10	Публичная лицензия Mozilla, версия 1.0	FSF OSI	(по умолчанию)
MPL11	Публичная лицензия Mozilla, версия 1.1	FSF OSI	(по умолчанию)
MPL20	Публичная лицензия Mozilla, версия 2.0	FSF OSI	(по умолчанию)
NCSA	Открытая лицензия Университета Иллинойса/NCSA	COPYFREE FSF GPL OSI	(по умолчанию)
NONE	Лицензия не указана		none
OFL10	Лицензия SIL Open Font версия 1.0 ( <a href="https://scripts.sil.org/OFL/">https://scripts.sil.org/OFL/</a> )	FONTS	(по умолчанию)
OFL11	Лицензия SIL Open Font версия 1.1 ( <a href="https://scripts.sil.org/OFL/">https://scripts.sil.org/OFL/</a> )	FONTS	(по умолчанию)
OWL	Лицензия Открытых Произведений (owl.apotheon.org)	COPYFREE	(по умолчанию)
OpenSSL	Лицензия OpenSSL	FSF	(по умолчанию)
PD	Общественное достояние	GPL COPYFREE	(по умолчанию)
PHP202	Лицензия PHP версии 2.02	FSF OSI	(по умолчанию)
PHP30	Лицензия PHP версии 3.0	FSF OSI	(по умолчанию)

Короткое имя	Имя	Группа	Разрешения
PHP301	Лицензия PHP версии 3.01	FSF OSI	(по умолчанию)
PSFL	Лицензия Python Software Foundation	FSF GPL OSI	(по умолчанию)
PostgreSQL	Лицензия PostgreSQL	FSF GPL OSI COPYFREE	(по умолчанию)
RUBY	Лицензия Ruby	FSF	(по умолчанию)
UNLICENSE	Отказ от лицензии (The Unlicense)	COPYFREE FSF GPL	(по умолчанию)
WTFPL	Публичная лицензия "Делай что хочешь" версия 2	GPL FSF COPYFREE	(по умолчанию)
WTFPL1	Публичная лицензия "Делай что хочешь" версия 1	GPL FSF COPYFREE	(по умолчанию)
ZLIB	Лицензия zlib	GPL FSF OSI	(по умолчанию)
ZPL21	Публичная лицензия Zope версия 2.1	GPL OSI	(по умолчанию)

### 5.8.2. LICENSE\_PERMS и LICENSE\_PERMS\_NAME\_

Разрешения. Используйте `none`, если пусто.

*Список разрешений лицензии*

#### **dist-mirror**

Разрешается распространение дистрибутивных файлов. Дистрибутивные файлы будут добавлены в CDN `MASTER_SITE_BACKUP` FreeBSD.

#### **no-dist-mirror**

Распространение дистрибутивных файлов запрещено. Это эквивалентно установке `RESTRICTED`. Дистрибутивные файлы *не* будут добавлены в CDN `MASTER_SITE_BACKUP` FreeBSD.

#### **dist-sell**

Продажа файлов дистрибутива разрешена. Файлы дистрибутива будут присутствовать на образах установщика.

#### **no-dist-sell**

Продажа файлов дистрибутива запрещена. Это эквивалентно установке `NO_CDROM`.

#### **pkg-mirror**

Свободное распространение пакета разрешено. Пакет будет распространяться через CDN пакетов FreeBSD <https://pkg.freebsd.org/>.

### no-pkg-mirror

Свободное распространение пакета запрещено. Эквивалентно установке `NO_PACKAGE`. Пакет *не* будет распространяться через FreeBSD CDN для пакетов <https://pkg.freebsd.org/>.

### pkg-sell

Продажа пакета разрешена. Пакет будет присутствовать на образах установщика.

### no-pkg-sell

Продажа пакета запрещена. Это эквивалентно установке `NO_CDROM`. Пакет *не* будет присутствовать на образах установщика.

### auto-accept

Лицензия принимается по умолчанию. Запросы на принятие лицензии не отображаются, если пользователь не определил `LICENSES_ASK`. Используйте это, если в лицензии не указано, что пользователь должен принять условия лицензии.

### no-auto-accept

Лицензия не принимается по умолчанию. Пользователь всегда будет запрошен на подтверждение принятия данной лицензии. Это должно использоваться, если лицензия требует, чтобы пользователь принял её условия.

Когда присутствуют и `permission`, и `no-permission`, то `no-permission` отменяет `permission`.

Когда `permission` отсутствует, это считается как `no-permission`.



Некоторые отсутствующие разрешения могут сделать порт (и все зависящие от него порты) непригодными для использования пользователями пакетов:

Порт без разрешения `auto-accept` никогда не будет собран, и все зависящие от него порты будут проигнорированы.

Порт без разрешения `pkg-mirror`, а также любые порты, зависящие от него, будут удалены после сборки, что гарантирует их отсутствие в дистрибуции.

### Пример 33. Нестандартная лицензия

Прочитайте условия лицензии и переведите их, используя доступные разрешения.

```
LICENSE= UNKNOWN
LICENSE_NAME= unknown
LICENSE_TEXT= This program is NOT in public domain.\
              It can be freely distributed for non-commercial purposes only.
LICENSE_PERMS= dist-mirror no-dist-sell pkg-mirror no-pkg-sell auto-accept
```

### Пример 34. Стандартные и нестандартные лицензии

Прочитайте условия лицензии и укажите их, используя доступные разрешения. В

случае сомнений обратитесь за разъяснениями на [Список рассылки, посвящённый Портam FreeBSD](#).

```
LICENSE=      WARSOW GPLv2
LICENSE_COMB= multi
LICENSE_NAME_WARSOW=  Warsaw Content License
LICENSE_FILE_WARSOW=  ${WRKSRС}/docs/license.txt
LICENSE_PERMS_WARSOW= dist-mirror pkg-mirror auto-accept
```

Когда разрешения лицензий GPLv2 и UNKNOWN смешиваются, порт получает `dist-mirror dist-sell pkg-mirror pkg-sell auto-accept dist-mirror no-dist-sell pkg-mirror no-pkg-sell auto-accept`. Опции *no-*разрешения отменяют соответствующие *разрешения*. Итоговый список разрешений: `dist-mirror pkg-mirror auto-accept`. Файлы дистрибутива и пакеты не будут доступны в образах установщика.

### 5.8.3. LICENSE\_GROUPS и LICENSE\_GROUPS\_NAME

Группы, к которым принадлежит лицензия.

*Список predeterminedных групп лицензий*

#### FSF

Одобрено Free Software Foundation, см. [Команда по лицензированию и соответствию FSF](#).

#### GPL

Совместимые с GPL

#### OSI

Одобрено OSI, см. страницу [Открытых лицензий](#).

#### COPYFREE

Соответствует определению стандарта Copyleft, см. страницу [лицензий Copyleft](#).

#### FONTS

Лицензии на шрифты

### 5.8.4. LICENSE\_NAME и LICENSE\_NAME\_NAME

Полное название лицензии.

*Пример 35. LICENSE\_NAME*

```
LICENSE=      UNRAR
LICENSE_NAME=  UnRAR License
LICENSE_FILE=  ${WRKSRС}/license.txt
LICENSE_PERMS= dist-mirror dist-sell pkg-mirror pkg-sell auto-accept
```

### 5.8.5. LICENSE\_FILE и LICENSE\_FILE\_NAME

Полный путь к файлу, содержащему текст лицензии, обычно `${WRKSRC}/some/file`. Если файл отсутствует в дистрибутиве, а его содержимое слишком длинное для размещения в `LICENSE_TEXT`, поместите его в новый файл в `${FILESDIR}`.

Пример 36. `LICENSE_FILE`

```
LICENSE=    GPLv3+
LICENSE_FILE=  ${WRKSRC}/COPYING
```

### 5.8.6. LICENSE\_TEXT и LICENSE\_TEXT\_NAME

Текст для использования в качестве лицензии. Полезно, когда лицензия отсутствует в файлах дистрибутива и её текст краток.

Пример 37. `LICENSE_TEXT`

```
LICENSE=    UNKNOWN
LICENSE_NAME=  unknown
LICENSE_TEXT=  This program is NOT in public domain.\
               It can be freely distributed for non-commercial purposes only,\
               and THERE IS NO WARRANTY FOR THIS PROGRAM.
LICENSE_PERMS=  dist-mirror no-dist-sell pkg-mirror no-pkg-sell auto-accept
```

### 5.8.7. LICENSE\_DISTFILES и LICENSE\_DISTFILES\_NAME

Файлы дистрибутива, к которым применяются лицензии. По умолчанию — все файлы дистрибутива.

Пример 38. `LICENSE_DISTFILES`

Используется, когда файлы дистрибутива имеют разные лицензии. Например, один файл имеет лицензию на код, а другой содержит некоторые произведения искусства, которые нельзя распространять:

```
MASTER_SITES=  SF/some-game
DISTFILES=     ${DISTNAME}${EXTRACT_SUFX} artwork.zip

LICENSE=       BSD3CLAUSE ARTWORK
LICENSE_COMB=  dual
LICENSE_NAME_ARTWORK=  The game artwork license
LICENSE_TEXT_ARTWORK=  The README says that the files cannot be redistributed
LICENSE_PERMS_ARTWORK=  pkg-mirror pkg-sell auto-accept
LICENSE_DISTFILES_BSD3CLAUSE=  ${DISTNAME}${EXTRACT_SUFX}
```

```
LICENSE_DISTFILES_ARTWORK= artwork.zip
```

### 5.8.8. LICENSE\_COMB

Установите значение **multi**, если применяются все лицензии. Установите значение **dual**, если применяется любая из лицензий. По умолчанию используется значение **single**.

*Пример 39. Двойные лицензии*

Когда порт содержит указание «Это программное обеспечение может распространяться под GNU General Public License или Artistic License», это означает, что можно использовать любую из этих лицензий. Используйте следующее:

```
LICENSE=    ART10 GPLv1
LICENSE_COMB= dual
```

Если предоставлены файлы лицензий, используйте это:

```
LICENSE=    ART10 GPLv1
LICENSE_COMB= dual
LICENSE_FILE_ART10=    ${WRKSRC}/Artistic
LICENSE_FILE_GPLv1=    ${WRKSRC}/Copying
```

*Пример 40. Множественные лицензии*

Если часть порта имеет одну лицензию, а другая часть — другую, используйте **multi**:

```
LICENSE=    GPLv2 LGPL21+
LICENSE_COMB= multi
```

## 5.9. PORTSCOUT

Portscout — это автоматизированная утилита проверки distfile для Коллекции портов FreeBSD, подробно описанная в [Portscout: сканирование distfile портов FreeBSD](#).

**PORTSCOUT** определяет специальные условия, в рамках которых работа сканера дистрибутивных файлов Portscout ограничена.

Ситуации, когда установлена переменная **PORTSCOUT**, включают:

- Когда необходимо игнорировать distfiles для определённых версий. Например, чтобы исключить версию 8.2 и версию 8.3 из проверок версий distfiles, так как известно, что они неработоспособны, добавьте:

```
PORTSCOUT= skipv:8.2,8.3
```

- Когда проверки версий distfile необходимо полностью отключить. Например, если порт больше не будет обновляться, добавьте:

```
PORTSCOUT= ignore:1
```

- Когда необходимо проверять конкретные версии или определенные мажорные и минорные редакции distfile. Например, если нужно отслеживать только версию 0.6.4, потому что более новые версии имеют проблемы совместимости с FreeBSD, добавьте:

```
PORTSCOUT= limit:^0\.6\4
```

- Когда URL-адреса, перечисляющие доступные версии, отличаются от URL-адресов загрузки. Например, чтобы ограничить проверку версий distfile страницей загрузки для пакета: [databases/pgtune](http://databases/pgtune) добавьте:

```
PORTSCOUT= site:http://www.renpy.org/dl/release/
```

## 5.10. Зависимости

Многие порты зависят от других портов. Это очень удобная особенность большинства Unix-подобных операционных систем, включая FreeBSD. Несколько портов могут использовать общую зависимость вместо того, чтобы включать эту зависимость в каждый порт или пакет, который в ней нуждается. Существует семь переменных, которые можно использовать для обеспечения наличия всех необходимых компонентов на машине пользователя. Также есть предопределенные переменные зависимостей для распространенных случаев и несколько дополнительных для управления поведением зависимостей.



Когда у программного обеспечения есть дополнительные зависимости, предоставляющие дополнительные возможности, основные зависимости, перечисленные в `*_DEPENDS`, должны включать те дополнительные зависимости, которые будут полезны большинству пользователей. Основные зависимости никогда не должны быть "минимальным" набором зависимостей. Цель состоит не в том, чтобы включить все возможные зависимости. Включайте только те, которые будут полезны большинству людей.

### 5.10.1. LIB\_DEPENDS

Эта переменная определяет разделяемые библиотеки, от которых зависит данный порт. Это список кортежей вида `lib:dir`, где `lib` — имя разделяемой библиотеки, а `dir` — директория, в

которой её следует искать, если она недоступна. Например,

```
LIB_DEPENDS= libjpeg.so:graphics/jpeg
```

проверит наличие общей библиотеки jpeg с любой версией и перейдет в подкаталог graphics/jpeg дерева портов, чтобы собрать и установить её, если она не найдена.

Зависимость проверяется дважды: один раз внутри цели `build` и затем внутри цели `install`. Также имя зависимости добавляется в пакет, чтобы `pkg install` (см. `pkg-install(8)`) автоматически установил её, если её нет в системе пользователя.

### 5.10.2. RUN\_DEPENDS

Эта переменная определяет исполняемые файлы или файлы, от которых зависит порт во время выполнения. Это список кортежей `path:dir[:target]`, где `path` — это имя исполняемого файла или файла, `dir` — директория, в которой его следует искать, если он недоступен, а `target` — цель, которую нужно вызвать в этой директории. Если `path` начинается с косой черты (`/`), он считается файлом, и его существование проверяется с помощью `test -e`; в противном случае предполагается, что это исполняемый файл, и `which -s` используется для проверки наличия программы в пути поиска.

Например,

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

проверит, существует ли файл или каталог `/usr/local/news/bin/innd`, и соберет и установит его из подкаталога `news/inn` дерева портов, если он не найден. Также будет проверено, находится ли исполняемый файл `xmlcatmgr` в пути поиска, и если он не найден, будет выполнен переход в `textproc/xmlcatmgr` для сборки и установки.



В этом случае `innd` является исполняемым файлом; если исполняемый файл находится в месте, которое не ожидается в пути поиска, используйте полный путь.



Официальный путь поиска `PATH`, используемый в кластере сборки портов

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

Зависимость проверяется внутри цели `install`. Также имя зависимости добавляется в пакет, чтобы команда `pkg install` (см. `pkg-install(8)`) автоматически установила её, если она отсутствует в системе пользователя. Часть `target` может быть опущена, если она совпадает с `DEPENDS_TARGET`.

Довольно распространённая ситуация, когда `RUN_DEPENDS` буквально совпадает с `BUILD_DEPENDS`,

особенно если портируемое программное обеспечение написано на скриптовом языке или требует одинаковой среды для сборки и выполнения. В этом случае возникает соблазн и интуитивное желание напрямую присвоить одно другому:

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

Однако такое присваивание может загрязнить зависимости во время выполнения записями, не определёнными в оригинальном `BUILD_DEPENDS` порта. Это происходит из-за ленивого вычисления присваивания переменных в `make(1)`. Рассмотрим Makefile с `USE_*`, которые обрабатываются `ports/Mk/bsd.*.mk` для добавления начальных зависимостей сборки. Например, `USES= gmake` добавляет `devel/gmake` в `BUILD_DEPENDS`. Чтобы предотвратить попадание таких дополнительных зависимостей в `RUN_DEPENDS`, создайте другую переменную с текущим содержимым `BUILD_DEPENDS` и присвойте её как `BUILD_DEPENDS`, так и `RUN_DEPENDS`:

```
MY_DEPENDS= some:devel/some \  
            other:lang/other  
BUILD_DEPENDS=  ${MY_DEPENDS}  
RUN_DEPENDS=    ${MY_DEPENDS}
```



Не используйте `:=` для присваивания `BUILD_DEPENDS` в `RUN_DEPENDS` или наоборот. Все переменные раскрываются немедленно, что является совершенно неправильным и почти всегда приводит к ошибке.

### 5.10.3. BUILD\_DEPENDS

Эта переменная указывает исполняемые файлы или файлы, необходимые для сборки данного порта. Как и `RUN_DEPENDS`, это список кортежей `path:dir[:target]`. Например,

```
BUILD_DEPENDS= unzip:archivers/unzip
```

проверит наличие исполняемого файла с именем `unzip` и перейдет в подкаталог `archivers/unzip` дерева портов, чтобы собрать и установить его, если он не будет найден.



"build" здесь означает все процессы от извлечения до компиляции. Зависимость проверяется внутри цели `extract`. Часть `target` может быть опущена, если она совпадает с `DEPENDS_TARGET`

### 5.10.4. FETCH\_DEPENDS

Эта переменная определяет исполняемые файлы или файлы, необходимые для загрузки этого порта. Как и предыдущие две, это список кортежей `path:dir[:target]`. Например,

```
FETCH_DEPENDS= ncftp2:net/ncftp2
```

проверит наличие исполняемого файла с именем `ncftp2` и перейдет в подкаталог `net/ncftp2` дерева портов для сборки и установки, если файл не будет найден.

Зависимость проверяется внутри цели `fetch`. Часть `target` может быть опущена, если она совпадает с `DEPENDS_TARGET`.

### 5.10.5. EXTRACT\_DEPENDS

Эта переменная указывает исполняемые файлы или файлы, которые требуются для извлечения данного порта. Как и предыдущая, это список кортежей `path:dir[:target]`. Например,

```
EXTRACT_DEPENDS= unzip:archivers/unzip
```

проверит наличие исполняемого файла с именем `unzip` и перейдет в подкаталог `archivers/unzip` дерева портов, чтобы собрать и установить его, если он не будет найден.

Зависимость проверяется внутри цели `extract`. Часть `target` может быть опущена, если она совпадает с `DEPENDS_TARGET`.



Используйте эту переменную только если извлечение уже не работает (по умолчанию предполагается `tar`) и не может быть исправлено с помощью `USES=tar`, `USES=lha` или `USES=zip`, как описано в [Использование макросов USES](#).

### 5.10.6. PATCH\_DEPENDS

Эта переменная указывает исполняемые файлы или файлы, которые требуются этому порту для применения патчей. Как и предыдущая, это список кортежей `path:dir[:target]`. Например,

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

будет спускаться в подкаталог `java/jfc` дерева портов для его извлечения.

Зависимость проверяется в рамках цели `patch`. Часть `target` может быть опущена, если она совпадает с `DEPENDS_TARGET`.

### 5.10.7. USES

Параметры могут быть добавлены для определения различных функций и зависимостей, используемых портом. Они указываются путем добавления этой строки в `Makefile`:

```
USES= feature[:arguments]
```

Для полного списка значений обратитесь к [Использование макросов USES](#).



`USES` нельзя назначать после включения `bsd.port.pre.mk`.

### 5.10.8. `USE_*`

Существует несколько переменных для определения общих зависимостей, используемых многими портами. Их использование необязательно, но помогает сократить многословность Makefile портов. Каждая из них оформлена как `USE_*`. Эти переменные могут использоваться только в Makefile портов и `ports/Mk/bsd.*.mk`. Они не предназначены для настраиваемых пользователем опций — для этой цели используйте `PORT_OPTIONS`.

Всегда неправильно устанавливать любые `USE_*` в `/etc/make.conf`. Например, установка



```
USE_GCC=X.Y
```

(где X.Y — номер версии) добавит зависимость от `gccXY` для каждого порта, включая сам `lang/gccXY`!

Таблица 8. `USE_*`

## Переменная

## Значение

`USE_GCC`

Порт требует GCC (`gcc` или `g++`) для сборки. Некоторые порты нуждаются в определённой, старой версии GCC, другие требуют современных, актуальных версий. Обычно устанавливается в `yes` (означает всегда использовать стабильную, современную версию GCC из портов, согласно `GCC_DEFAULT` в `Mk/bsd.default-versions.mk`). Это также значение по умолчанию. Точная версия также может быть указана, например, значением `10`. GCC из базовой системы используется, если он удовлетворяет запрашиваемой версии, в противном случае подходящий компилятор собирается из портов, а `CC` и `CXX` корректируются соответствующим образом. Аргумент `:build`, следующий за указанием версии, добавляет только зависимость во время сборки порта.

Например:

```
USE_GCC=yes      # порт требует
                  текущей версии GCC
USE_GCC=11:build # порт требует
                  GCC 11 только во время сборки
```



`USE_GCC=any` устарел и не должен использоваться в новых портах

Переменные, связанные с `gmake` и `configure`, описаны в [Механизмы сборки](#), тогда как `autoconf`, `automake` и `libtool` описаны в [Использование GNU Autotools](#). Переменные, связанные с `Perl`, описаны в [Использование Perl](#). Переменные X11 перечислены в [Использование X11](#). [Использование GNOME](#) посвящено GNOME, а [Использование KDE](#) — переменным, связанным с KDE. [Использование Java](#) документирует переменные Java, тогда как [Веб-приложения](#) содержит информацию о модулях Apache, PHP и PEAR. Python обсуждается в [Использование Python](#), а Ruby — в [Использование Ruby](#). [Использование SDL](#) предоставляет переменные, используемые для приложений SDL, и, наконец, [Использование Xfce](#) содержит информацию о Xfce.

### 5.10.9. Минимальная версия зависимого пакета

Минимальная версия зависимого пакета может быть указана в любом `*_DEPENDS`, кроме

`LIB_DEPENDS`, используя следующий синтаксис:

```
p5-Spiffy>=0.26:devel/p5-Spiffy
```

Первое поле содержит имя зависимого пакета, которое должно соответствовать записи в базе данных пакетов, знак сравнения и версию пакета. Зависимость считается удовлетворённой, если на машине установлен `p5-Spiffy-0.26` или новее.

### 5.10.10. Заметки о зависимостях

Как упомянуто выше, цель по умолчанию, вызываемая при необходимости зависимости, — это `DEPENDS_TARGET`. По умолчанию она установлена в `install`. Это пользовательская переменная; она никогда не определяется в `Makefile` порта. Если порту требуется особый способ обработки зависимости, используйте часть `:target` в `*_DEPENDS` вместо переопределения `DEPENDS_TARGET`.

При выполнении `make clean` зависимости портов также автоматически очищаются. Если это нежелательно, определите переменную `NOCLEANDEPENDS` в окружении. Это может быть особенно полезно, если среди зависимостей порта есть что-то, что требует много времени для пересборки, например KDE, GNOME или Mozilla.

Для безусловной зависимости от другого порта используйте переменную `_${NONEXISTENT}` в качестве первого поля `BUILD_DEPENDS` или `RUN_DEPENDS`. Используйте это только в случае, когда необходим исходный код другого порта. Время компиляции можно сократить, указав также цель. Например

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

всегда будет переходить к порту `jpeg` и извлекать его.

### 5.10.11. Циклические зависимости фатальны



Не создавайте циклических зависимостей в дереве портов!

Технология сборки портов не допускает циклических зависимостей. Если такая зависимость будет добавлена, у кого-то в мире почти сразу окажется сломанной установка FreeBSD, а за этим последуют многие другие. Подобные проблемы бывает очень сложно обнаружить. Если есть сомнения, перед внесением изменений обязательно выполните: `cd /usr/ports; make index`. Этот процесс может быть довольно медленным на старых машинах, но он способен избавить множество людей, включая вас, от серьёзных проблем.

### 5.10.12. Проблемы, вызванные автоматическими зависимостями

Зависимости должны быть объявлены явно или с использованием `OPTIONS framework`. Использование других методов, таких как автоматическое обнаружение, усложняет индексацию, что вызывает проблемы для управления портами и пакетами.

*Пример 41. Неправильное объявление необязательной зависимости*

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

.if exists(${LOCALBASE}/bin/foo)
LIB_DEPENDS=  libbar.so:foo/bar
.endif
```

Проблема с попыткой автоматического добавления зависимостей заключается в том, что файлы и настройки за пределами отдельного порта могут измениться в любой момент. Например: индекс строится, затем устанавливается группа портов. Но один из портов устанавливает проверяемый файл. Теперь индекс неверен, потому что у установленного порта неожиданно появилась новая зависимость. Индекс может оставаться неверным даже после пересборки, если другие порты также определяют свою потребность в зависимостях на основе существования других файлов.

*Пример 42. Правильное объявление необязательной зависимости*

```
OPTIONS_DEFINE= BAR
BAR_DESC=  Calling cellphones via bar

BAR_LIB_DEPENDS=  libbar.so:foo/bar
```

Проверка переменных опций является правильным методом. Это не вызовет несоответствий в индексе группы портов, при условии что опции были определены до сборки индекса. Затем можно использовать простые скрипты для автоматизации сборки, установки и обновления этих портов и их пакетов.

## 5.11. Подчиненные порты и **MASTERDIR**

Если порту необходимо собирать немного разные версии пакетов, используя переменную (например, разрешение или размер бумаги) с разными значениями, создайте по одному подкаталогу для каждого пакета, чтобы пользователям было проще понять, что делать, но старайтесь максимально использовать общие файлы между портами. Обычно, при грамотном использовании переменных, во всех каталогах, кроме одного, требуется лишь очень короткий Makefile. В единственном Makefile укажите директорию с остальными файлами с помощью **MASTERDIR**. Также используйте переменную как часть **PKGNAME\_SUFFIX**, чтобы пакеты имели разные имена.

Это лучше всего продемонстрировать на примере. Это часть файла `print/pkfonts300/Makefile`;

```
PORTNAME=  pkfonts${RESOLUTION}
PORTVERSION=  1.0
DISTFILES=  pk${RESOLUTION}.tar.gz
```

```

PLIST=      ${PKGDIR}/pkg-plist.${RESOLUTION}

.if !defined(RESOLUTION)
RESOLUTION= 300
.else
.if ${RESOLUTION} != 118 && ${RESOLUTION} != 240 && \
   ${RESOLUTION} != 300 && ${RESOLUTION} != 360 && \
   ${RESOLUTION} != 400 && ${RESOLUTION} != 600
.BEGIN:
  @${ECHO_MSG} "Error: invalid value for RESOLUTION: \"${RESOLUTION}\""
  @${ECHO_MSG} "Possible values are: 118, 240, 300, 360, 400 and 600."
  @${FALSE}
.endif
.endif

```

Пакет `print/pkfonts300` также содержит все обычные исправления, файлы пакетов и т.д. При запуске `make` в этом месте будет взято значение разрешения по умолчанию (300), и порт будет собран в обычном режиме.

Что касается других разрешений, это *полный* `print/pkfonts360/Makefile`:

```

RESOLUTION= 360
MASTERDIR=  ${CURDIR}/../pkfonts300

.include     "${MASTERDIR}/Makefile"

```

(`print/pkfonts118/Makefile`, `print/pkfonts600/Makefile` и все остальные аналогичны). Определение `MASTERDIR` указывает `bsd.port.mk`, что стандартный набор подкаталогов, таких как `FILESDIR` и `SCRIPTDIR`, следует искать в `pkfonts300`. Строка `RESOLUTION=360` переопределяет строку `RESOLUTION=300` в `pkfonts300/Makefile`, и порт будет собран с разрешением, установленным на 360.

## 5.12. Страницы Справочника

Если порт размещает дерево `man` в другом месте, отличном от `PREFIX`, используйте `MANDIRS` для указания этих каталогов. Обратите внимание, что файлы, соответствующие страницам руководства, должны быть добавлены в `pkg-plist` вместе с остальными файлами. Назначение `MANDIRS` — обеспечить автоматическое сжатие страниц руководства, поэтому имена файлов имеют суффикс `.gz`.

## 5.13. Файлы информации

Если пакету требуется установить файлы GNU info, перечислите их в `INFO` (без завершающего `.info`), по одному документу на строку. Предполагается, что эти файлы будут установлены в `PREFIX/INFO_PATH`. Измените `INFO_PATH`, если пакет использует другое расположение. Однако это не рекомендуется. Эти записи содержат только путь

относительно PREFIX/INFO\_PATH. Например, пакет `lang/gcc34` устанавливает файлы `info` в `PREFIX/INFO_PATH/gcc34`, и `INFO` будет выглядеть примерно так:

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

Соответствующий код установки/удаления будет автоматически добавлен во временный файл `pkg-plist` перед регистрацией пакета.

## 5.14. Параметры Makefile

Многие приложения могут быть собраны с дополнительными или различными конфигурациями. Примеры включают выбор естественного (человеческого) языка, графический интерфейс или командная строка, тип поддерживаемой базы данных. Пользователям может потребоваться конфигурация, отличная от стандартной, поэтому система портов предоставляет хуки, которые автор порта может использовать для управления вариантом сборки. Правильная поддержка этих опций сделает пользователей счастливыми и эффективно предоставит два или более порта по цене одного.

### 5.14.1. OPTIONS

#### 5.14.1.1. Пояснения

`OPTIONS_*` предоставляют пользователю, устанавливающему порт, диалоговое окно с доступными опциями, после чего сохраняют выбранные опции в `${PORT_DBDIR}/${OPTIONS_NAME}/options`. При следующей сборке порта эти опции будут использованы повторно. `PORT_DBDIR` по умолчанию имеет значение `/var/db/ports`. `OPTIONS_NAME` соответствует имени порта (`origin`) с заменой разделителя на подчёркивания, например, для `dns/bind99` это будет `dns_bind99`.

Когда пользователь запускает `make config` (или впервые запускает `make build`), система проверяет наличие файла `${PORT_DBDIR}/${OPTIONS_NAME}/options`. Если этот файл не существует, используются значения `OPTIONS_*`, и отображается диалоговое окно, где можно включить или отключить опции. Затем файл `options` сохраняется, а настроенные переменные используются при сборке порта.

Если новая версия порта добавляет новые `OPTIONS`, пользователю будет показан диалог с сохранёнными значениями старых `OPTIONS`, заполненными заранее.

`make showconfig` показывает сохранённую конфигурацию. Используйте `make rmconfig` для удаления сохранённой конфигурации.

#### 5.14.1.2. Синтаксис

`OPTIONS_DEFINE` содержит список `OPTIONS`, которые будут использоваться. Они независимы друг от друга и не сгруппированы:

```
OPTIONS_DEFINE= OPT1 OPT2
```

После определения **OPTIONS** описываются (необязательно, но настоятельно рекомендуется):

```
OPT1_DESC= Describe OPT1
OPT2_DESC= Describe OPT2
OPT3_DESC= Describe OPT3
OPT4_DESC= Describe OPT4
OPT5_DESC= Describe OPT5
OPT6_DESC= Describe OPT6
```

ports/Mk/bsd.options.desc.mk содержит описания для многих распространённых **OPTIONS**. Хотя они часто полезны, переопределите их, если описание недостаточно для порта.



При описании параметров рассматривайте их с точки зрения пользователя: «Какую функциональность это изменяет?» и «Зачем мне включать этот параметр?» Не просто повторяйте название. Например, описание параметра **NLS** как «включить поддержку NLS» не помогает пользователю, который уже видит название параметра, но может не знать, что оно означает. Описание вроде «Поддержка родного языка с помощью утилиты gettext» гораздо полезнее.



Названия параметров всегда пишутся в верхнем регистре. Они не могут использовать смешанный регистр или нижний регистр.

**OPTIONS** могут быть сгруппированы как переключаемые варианты, где допускается только один выбор из каждой группы:

```
OPTIONS_SINGLE= SG1
OPTIONS_SINGLE_SG1= OPT3 OPT4
```



В каждый момент времени *должна* быть выбрана одна опция из каждой группы **OPTIONS\_SINGLE**, чтобы параметры были действительными. Один вариант из каждой группы *должен* быть добавлен в **OPTIONS\_DEFAULT**.

**OPTIONS** могут быть сгруппированы как переключаемые варианты, где ни один или только один вариант из каждой группы разрешён:

```
OPTIONS_RADIO= RG1
OPTIONS_RADIO_RG1= OPT7 OPT8
```

**OPTIONS** также могут быть сгруппированы в виде списков "множественного выбора", где *хотя бы одна* опция должна быть включена:

```
OPTIONS_MULTI= MG1
OPTIONS_MULTI_MG1= OPT5 OPT6
```

**OPTIONS** также могут быть сгруппированы в виде списков "множественного выбора", где ни одна или любые опции могут быть включены:

```
OPTIONS_GROUP=      GG1
OPTIONS_GROUP_GG1=  OPT9 OPT10
```

**OPTIONS** по умолчанию не установлены, если они не перечислены в **OPTIONS\_DEFAULT**:

```
OPTIONS_DEFAULT=    OPT1 OPT3 OPT6
```

Определения **OPTIONS** должны быть указаны до включения файла `bsd.port.options.mk`. Значения **PORT\_OPTIONS** можно проверять только после включения `bsd.port.options.mk`. Включение `bsd.port.pre.mk` также может использоваться и до сих пор широко применяется в портах, написанных до введения `bsd.port.options.mk`. Однако следует учитывать, что некоторые переменные не будут работать как ожидается после включения `bsd.port.pre.mk`, обычно это некоторые флаги **USE\_\***.

*Пример 43. Простое использование **OPTIONS***

```
OPTIONS_DEFINE= FOO BAR
OPTIONS_DEFAULT=FOO

FOO_DESC=      Option foo support
BAR_DESC=      Feature bar support

# Will add --with-foo / --without-foo
FOO_CONFIGURE_WITH= foo
BAR_RUN_DEPENDS=  bar:bar/bar

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

*Пример 44. Проверка неустановленных **OPTIONS** порта*

```
.if ! ${PORT_OPTIONS:MEXAMPLES}
CONFIGURE_ARGS+=--without-examples
.endif
```

Приведённая выше форма не рекомендуется. Предпочтительный метод — использование параметра `configure` для фактического включения и отключения функции в соответствии с опцией:

```
# Will add --with-examples / --without-examples
EXAMPLES_CONFIGURE_WITH=  examples
```

```
OPTIONS_DEFINE=      EXAMPLES
OPTIONS_DEFAULT=    PGSQL LDAP SSL

OPTIONS_SINGLE=     BACKEND
OPTIONS_SINGLE_BACKEND= MYSQL PGSQL BDB

OPTIONS_MULTI=      AUTH
OPTIONS_MULTI_AUTH= LDAP PAM SSL

EXAMPLES_DESC=      Install extra examples
MYSQL_DESC=         Use MySQL as backend
PGSQL_DESC=         Use PostgreSQL as backend
BDB_DESC=           Use Berkeley DB as backend
LDAP_DESC=          Build with LDAP authentication support
PAM_DESC=           Build with PAM support
SSL_DESC=           Build with OpenSSL support

# Will add USE_PGSQL=yes
PGSQL_USE=          postgresql=yes
# Will add --enable-postgres / --disable-postgres
PGSQL_CONFIGURE_ENABLE= postgres

ICU_LIB_DEPENDS=    libicuuc.so:devel/icu

# Will add --with-examples / --without-examples
EXAMPLES_CONFIGURE_WITH= examples

# Check other OPTIONS

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

### 5.14.1.3. Опции по умолчанию

Эти опции всегда включены по умолчанию.

- **DOCS** — сборка и установка документации.
- **NLS** — Поддержка родного языка.
- **EXAMPLES** — сборка и установка примеров.
- **IPV6** — Поддержка протокола IPv6.



Нет необходимости добавлять их в **OPTIONS\_DEFAULT**. Однако, чтобы они были активны и отображались в диалоге выбора опций, их необходимо добавить в **OPTIONS\_DEFINE**.

## 5.14.2. Автоматическая активация функций

При использовании скрипта GNU configure следите за тем, какие дополнительные функции активируются автоматическим определением. Явно отключите ненужные дополнительные функции, добавив `--without-xxx` или `--disable-xxx` в `CONFIGURE_ARGS`.

Пример 46. Неправильная обработка опции

```
.if ${PORT_OPTIONS:MFOO}
LIB_DEPENDS+=      libfoo.so:devel/foo
CONFIGURE_ARGS+=   --enable-foo
.endif
```

В приведённом выше примере представьте, что библиотека `libfoo` установлена в системе. Пользователь не хочет, чтобы это приложение использовало `libfoo`, поэтому он отключил соответствующую опцию в диалоге `make config`. Однако скрипт `configure` приложения обнаруживает библиотеку в системе и включает её поддержку в итоговом исполняемом файле. Теперь, когда пользователь решает удалить `libfoo` из системы, система портов не протестует (зависимость от `libfoo` не была записана), но приложение перестает работать.

Пример 47. Правильная обработка опции

```
FOO_LIB_DEPENDS=   libfoo.so:devel/foo
# Will add --enable-foo / --disable-foo
FOO_CONFIGURE_ENABLE=  foo
```

В некоторых случаях сокращенный синтаксис условных выражений может вызывать проблемы со сложными конструкциями. Ошибки обычно имеют вид `Malformed conditional`, тогда можно использовать альтернативный синтаксис.



```
.if !empty(VARIABLE:MVALUE)
```

в качестве альтернативы

```
.if ${VARIABLE:MVALUE}
```

## 5.14.3. Помощники параметров

Существуют макросы, которые помогают упростить условные значения, различающиеся в зависимости от установленных опций. Для удобства приведён полный список:

## PLIST\_SUB, SUB\_LIST

Для автоматической генерации `%%OPT%%` и `%%NOOPT%%` см. `OPTIONS_SUB`.

Для более сложных случаев использования см. [Замена общих переменных](#).

## CONFIGURE\_ARGS

Для информации о `--enable-x` и `--disable-x` см. `OPT_CONFIGURE_ENABLE`.

О `--with-x` и `--without-x` см. `OPT_CONFIGURE_WITH`.

Во всех остальных случаях см. `OPT_CONFIGURE_ON` и `OPT_CONFIGURE_OFF`.

## CMAKE\_ARGS

Для аргументов, которые являются булевыми значениями (`on`, `off`, `true`, `false`, `0`, `1`), см. `OPT_CMAKE_BOOL` и `OPT_CMAKE_BOOL_OFF`.

Для всех остальных случаев см. `OPT_CMAKE_ON` и `OPT_CMAKE_OFF`.

## MESON\_ARGS

Для аргументов, принимающих `true` или `false`, см. `OPT_MESON_TRUE` и `OPT_MESON_FALSE`.

Для аргументов, принимающих `yes` или `no`, используйте `OPT_MESON_YES` и `OPT_MESON_NO`.

Для аргументов, принимающих `enabled` или `disabled`, см. `OPT_MESON_ENABLED` и `OPT_MESON_DISABLED`.

Во всех остальных случаях используйте `OPT_MESON_ON` и `OPT_MESON_OFF`.

## QMAKE\_ARGS

См. `OPT_QMAKE_ON` и `OPT_QMAKE_OFF`.

## USE\_\*

См. `OPT_USE` и `OPT_USE_OFF`.

## \*\_DEPENDS

См. [Зависимости](#).

## \* (Любая переменная)

Наиболее используемые переменные имеют своих помощников, см. [Замена Общих Переменных](#).

Для любой переменной без специального помощника см. `OPT_VARS` и `OPT_VARS_OFF`.

## Зависимости параметров

Когда для работы опции требуется другая опция, см. `OPT_IMPLIES`.

## Конфликты опций

Когда опция не может работать, если включена другая, см. `OPT_PREVENTS` и `OPT_PREVENTS_MSG`.

## Цели сборки

Когда для опции требуется дополнительная обработка, см. [Дополнительные цели сборки](#).

### 5.14.3.1. OPTIONS\_SUB

Если `OPTIONS_SUB` установлен в `yes`, то каждая из опций, добавленных в `OPTIONS_DEFINE`, будет добавлена в `PLIST_SUB` и `SUB_LIST`, например:

```
OPTIONS_DEFINE= OPT1
OPTIONS_SUB=    yes
```

эквивалентно:

```
OPTIONS_DEFINE= OPT1

.include <bsd.port.options.mk>

.if ${PORT_OPTIONS:MOPT1}
PLIST_SUB+= OPT1="" NO_OPT1="@comment "
SUB_LIST+= OPT1="" NO_OPT1="@comment "
.else
PLIST_SUB+= OPT1="@comment " NO_OPT1=""
SUB_LIST+= OPT1="@comment " NO_OPT1=""
.endif
```



Значение `OPTIONS_SUB` игнорируется. Установка любого значения добавит записи `PLIST_SUB` и `SUB_LIST` для *всех* опций.

### 5.14.3.2. OPT\_USE и OPT\_USE\_OFF

Когда выбрана опция `OPT`, для каждой пары `ключ=значение` в `OPT_USE`, `значение` добавляется к соответствующему `USE_KEY`. Если `значение` содержит пробелы, замените их запятыми, и они будут преобразованы обратно в пробелы во время обработки. `OPT_USE_OFF` работает аналогично, но когда `OPT` не выбрана. Например:

```
OPTIONS_DEFINE= OPT1
OPT1_USES=    xorg
OPT1_USE=     mysql=yes xorg=x11,xextproto,xext,xrandr
OPT1_USE_OFF= openssl=yes
```

эквивалентно:

```
OPTIONS_DEFINE= OPT1

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

```
.if ${PORT_OPTIONS:MOPT1}
USE_MYSQL= yes
USES+=     xorg
USE_XORG=  x11 xextproto xext xrandr
.else
USE_OPENSSL=  yes
.endif
```

### 5.14.3.3. Помощники `CONFIGURE_ARGS`

#### 5.14.3.3.1. `OPT_CONFIGURE_ENABLE`

Когда выбрана опция `OPT`, для каждого элемента в `OPT_CONFIGURE_ENABLE` к `CONFIGURE_ARGS` добавляется `--enable-элемент`. Если опция `OPT` не выбрана, к `CONFIGURE_ARGS` добавляется `--disable-элемент`. Необязательный аргумент может быть указан с помощью символа `=`. Этот аргумент добавляется только к опции конфигурации `--enable-элемент`. Например:

```
OPTIONS_DEFINE= OPT1 OPT2
OPT1_CONFIGURE_ENABLE= test1 test2
OPT2_CONFIGURE_ENABLE= test2=exhaustive
```

эквивалентно:

```
OPTIONS_DEFINE= OPT1

.include <bsd.port.options.mk>

.if ${PORT_OPTIONS:MOPT1}
CONFIGURE_ARGS+= --enable-test1 --enable-test2
.else
CONFIGURE_ARGS+= --disable-test1 --disable-test2
.endif

.if ${PORT_OPTIONS:MOPT2}
CONFIGURE_ARGS+= --enable-test2=exhaustive
.else
CONFIGURE_ARGS+= --disable-test2
.endif
```

#### 5.14.3.3.2. `OPT_CONFIGURE_WITH`

Когда выбрана опция `OPT`, для каждого элемента в `OPT_CONFIGURE_WITH` к `CONFIGURE_ARGS` добавляется `--with-элемент`. Если опция `OPT` не выбрана, к `CONFIGURE_ARGS` добавляется `--without-элемент`. Необязательный аргумент можно указать с помощью символа `=`. Этот аргумент добавляется только к опции конфигурации `--with-элемент`. Например:

```
OPTIONS_DEFINE= OPT1 OPT2
```

```
OPT1_CONFIGURE_WITH= test1
OPT2_CONFIGURE_WITH= test2=exhaustive
```

ЭКВИВАЛЕНТНО:

```
OPTIONS_DEFINE= OPT1 OPT2

.include <bsd.port.options.mk>

.if ${PORT_OPTIONS:MOPT1}
CONFIGURE_ARGS+= --with-test1
.else
CONFIGURE_ARGS+= --without-test1
.endif

.if ${PORT_OPTIONS:MOPT2}
CONFIGURE_ARGS+= --with-test2=exhaustive
.else
CONFIGURE_ARGS+= --without-test2
.endif
```

#### 5.14.3.3.3. `OPT_CONFIGURE_ON` и `OPT_CONFIGURE_OFF`

Когда выбрана опция `OPT`, значение `OPT_CONFIGURE_ON`, если оно определено, добавляется к `CONFIGURE_ARGS`. `OPT_CONFIGURE_OFF` работает аналогично, но когда `OPT` не выбрана. Например:

```
OPTIONS_DEFINE= OPT1
OPT1_CONFIGURE_ON= --add-test
OPT1_CONFIGURE_OFF= --no-test
```

ЭКВИВАЛЕНТНО:

```
OPTIONS_DEFINE= OPT1

.include <bsd.port.options.mk>

.if ${PORT_OPTIONS:MOPT1}
CONFIGURE_ARGS+= --add-test
.else
CONFIGURE_ARGS+= --no-test
.endif
```



В большинстве случаев помощники `OPT_CONFIGURE_ENABLE` и `OPT_CONFIGURE_WITH` предоставляют более короткий и понятный функционал.

### 5.14.3.4. Помощники CMAKE\_ARGS

#### 5.14.3.4.1. OPT\_CMAKE\_ON и OPT\_CMAKE\_OFF

Когда выбрана опция *OPT*, значение *OPT\_CMAKE\_ON*, если оно определено, добавляется к *CMAKE\_ARGS*. *OPT\_CMAKE\_OFF* работает аналогично, но когда *OPT* не выбрана. Например:

```
OPTIONS_DEFINE= OPT1
OPT1_CMAKE_ON=  -DTEST:BOOL=true -DDEBUG:BOOL=true
OPT1_CMAKE_OFF= -DOPTIMIZE:BOOL=true
```

эквивалентно:

```
OPTIONS_DEFINE= OPT1

.include <bsd.port.options.mk>

.if ${PORT_OPTIONS:MOP1}
CMAKE_ARGS+=  -DTEST:BOOL=true -DDEBUG:BOOL=true
.else
CMAKE_ARGS+=  -DOPTIMIZE:BOOL=true
.endif
```



См. *OPT\_CMAKE\_BOOL* и *OPT\_CMAKE\_BOOL\_OFF* для более краткой записи, когда значение является булевым.

#### 5.14.3.4.2. OPT\_CMAKE\_BOOL и OPT\_CMAKE\_BOOL\_OFF

Когда выбрана опция *OPT*, для каждого элемента в *OPT\_CMAKE\_BOOL* добавляется *-D\_элемент\_:BOOL=true* к *CMAKE\_ARGS*. Если опция *OPT* не выбрана, *-D\_элемент\_:BOOL=false* добавляется к *CONFIGURE\_ARGS*. *OPT\_CMAKE\_BOOL\_OFF* работает наоборот: *-D\_элемент\_:BOOL=false* добавляется к *CMAKE\_ARGS*, когда опция выбрана, и *-D\_элемент\_:BOOL=true*, когда опция не выбрана. Например:

```
OPTIONS_DEFINE= OPT1
OPT1_CMAKE_BOOL=  TEST DEBUG
OPT1_CMAKE_BOOL_OFF=  OPTIMIZE
```

эквивалентно:

```
OPTIONS_DEFINE= OPT1

.include <bsd.port.options.mk>

.if ${PORT_OPTIONS:MOP1}
CMAKE_ARGS+=  -DTEST:BOOL=true -DDEBUG:BOOL=true \
```

```

        -DOPTIMIZE:BOOL=false
    .else
    CMAKE_ARGS+=    -DTEST:BOOL=false -DDEBUG:BOOL=false \
        -DOPTIMIZE:BOOL=true
    .endif

```

### 5.14.3.5. Помощники `MESON_ARGS`

#### 5.14.3.5.1. `OPT_MESON_ON` и `OPT_MESON_OFF`

Когда выбрана опция `OPT`, значение `OPT_MESON_ON`, если оно определено, добавляется к `MESON_ARGS`. `OPT_MESON_OFF` работает аналогичным образом, но когда `OPT` не выбрана. Например:

```

OPTIONS_DEFINE= OPT1
OPT1_MESON_ON=  -Dopt=1
OPT1_MESON_OFF= -Dopt=2

```

ЭКВИВАЛЕНТНО:

```

OPTIONS_DEFINE= OPT1

.include <bsd.port.options.mk>

.if ${PORT_OPTIONS:MOPT1}
MESON_ARGS+=    -Dopt=1
.else
MESON_ARGS+=    -Dopt=2
.endif

```

#### 5.14.3.5.2. `OPT_MESON_TRUE` и `OPT_MESON_FALSE`

Когда выбрана опция `OPT`, для каждого элемента в `OPT_MESON_TRUE` добавляется `-D_элемент_=true` в `MESON_ARGS`. Если опция `OPT` не выбрана, добавляется `-D_элемент_=false` в `MESON_ARGS`. `OPT_MESON_FALSE` работает противоположным образом: `-D_элемент_=false` добавляется в `MESON_ARGS`, когда опция выбрана, и `-D_элемент_=true`, когда опция не выбрана. Например:

```

OPTIONS_DEFINE= OPT1
OPT1_MESON_TRUE=  test debug
OPT1_MESON_FALSE= optimize

```

ЭКВИВАЛЕНТНО:

```

OPTIONS_DEFINE= OPT1

.include <bsd.port.options.mk>

```

```
.if ${PORT_OPTIONS:MOPT1}
MESON_ARGS+= -Dtest=true -Ddebug=true \
             -Doptimize=false
.else
MESON_ARGS+= -Dtest=false -Ddebug=false \
             -Doptimize=true
.endif
```

#### 5.14.3.5.3. OPT\_MESON\_YES и OPT\_MESON\_NO

Когда выбрана опция *OPT*, для каждого элемента в *OPT\_MESON\_YES* добавляется *-D\_элемент\_=yes* к *MESON\_ARGS*. Если опция *OPT* не выбрана, добавляется *-D\_элемент\_=no* к *MESON\_ARGS*. *OPT\_MESON\_NO* работает противоположным образом: *-D\_элемент\_=no* добавляется к *MESON\_ARGS*, когда опция выбрана, и *-D\_элемент\_=yes*, когда опция не выбрана. Например:

```
OPTIONS_DEFINE= OPT1
OPT1_MESON_YES= test debug
OPT1_MESON_NO=  optimize
```

ЭКВИВАЛЕНТНО:

```
OPTIONS_DEFINE= OPT1

.include <bsd.port.options.mk>

.if ${PORT_OPTIONS:MOPT1}
MESON_ARGS+= -Dtest=yes -Ddebug=yes \
             -Doptimize=no
.else
MESON_ARGS+= -Dtest=no -Ddebug=no \
             -Doptimize=yes
.endif
```

#### 5.14.3.5.4. OPT\_MESON\_ENABLED и OPT\_MESON\_DISABLED

Когда выбрана опция *OPT*, для каждого элемента в *OPT\_MESON\_ENABLED* добавляется *-D\_элемент\_=enabled* к *MESON\_ARGS*. Когда опция *OPT* не выбрана, добавляется *-D\_элемент\_=disabled* к *MESON\_ARGS*. *OPT\_MESON\_DISABLED* работает противоположным образом: *-D\_элемент\_=disabled* добавляется к *MESON\_ARGS*, когда опция выбрана, и *-D\_элемент\_=enabled*, когда опция не выбрана. Например:

```
OPTIONS_DEFINE= OPT1
OPT1_MESON_ENABLED= test
OPT1_MESON_DISABLED=  debug
```

ЭКВИВАЛЕНТНО:

```
OPTIONS_DEFINE= OPT1

.include <bsd.port.options.mk>

.if ${PORT_OPTIONS:MOPT1}
MESON_ARGS+= -Dtest=enabled -Ddebug=disabled
.else
MESON_ARGS+= -Dtest=disabled -Ddebug=enabled
.endif
```

#### 5.14.3.6. OPT\_QMAKE\_ON и OPT\_QMAKE\_OFF

Когда выбрана опция *OPT*, значение *OPT\_QMAKE\_ON*, если оно определено, добавляется к *QMAKE\_ARGS*. *OPT\_QMAKE\_OFF* работает аналогичным образом, но когда *OPT* не выбрана. Например:

```
OPTIONS_DEFINE= OPT1
OPT1_QMAKE_ON= -DTEST:BOOL=true
OPT1_QMAKE_OFF= -DPRODUCTION:BOOL=true
```

ЭКВИВАЛЕНТНО:

```
OPTIONS_DEFINE= OPT1

.include <bsd.port.options.mk>

.if ${PORT_OPTIONS:MOPT1}
QMAKE_ARGS+= -DTEST:BOOL=true
.else
QMAKE_ARGS+= -DPRODUCTION:BOOL=true
.endif
```

#### 5.14.3.7. OPT\_IMPLIES

Предоставляет способ добавления зависимостей между опциями.

При выборе *OPT* все перечисленные в этой переменной опции также будут выбраны. В качестве примера можно использовать описанный ранее *OPT\_CONFIGURE\_ENABLE*:

```
OPTIONS_DEFINE= OPT1 OPT2
OPT1_IMPLIES= OPT2

OPT1_CONFIGURE_ENABLE= opt1
OPT2_CONFIGURE_ENABLE= opt2
```

Эквивалентно:

```
OPTIONS_DEFINE= OPT1 OPT2

.include <bsd.port.options.mk>

.if ${PORT_OPTIONS:MOPT1}
CONFIGURE_ARGS+= --enable-opt1
.else
CONFIGURE_ARGS+= --disable-opt1
.endif

.if ${PORT_OPTIONS:MOPT2} || ${PORT_OPTIONS:MOPT1}
CONFIGURE_ARGS+= --enable-opt2
.else
CONFIGURE_ARGS+= --disable-opt2
.endif
```

Пример 48. Простое использование `OPT_IMPLIES`

Этот порт имеет опцию `X11` и опцию `GNOME`, для сборки которой необходимо выбрать опцию `X11`.

```
OPTIONS_DEFINE= X11 GNOME
OPTIONS_DEFAULT= X11

X11_USES= xorg
X11_USE= xorg=xi,xextproto
GNOME_USE= gnome=gtk30
GNOME_IMPLIES= X11
```

#### 5.14.3.8. `OPT_PREVENTS` и `OPT_PREVENTS_MSG`

Предоставляет способ добавления конфликтов между опциями.

Когда выбрана `OPT`, все опции, перечисленные в `OPT_PREVENTS`, должны быть сняты. Если задано `OPT_PREVENTS_MSG` и возникает конфликт, его содержимое будет показано с объяснением причины конфликта. Например:

```
OPTIONS_DEFINE= OPT1 OPT2
OPT1_PREVENTS= OPT2
OPT1_PREVENTS_MSG= OPT1 and OPT2 enable conflicting options
```

Примерно эквивалентно:

```
OPTIONS_DEFINE= OPT1 OPT2
```

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

.if ${PORT_OPTIONS:MOPT2} && ${PORT_OPTIONS:MOPT1}
BROKEN= Option OPT1 conflicts with OPT2 (select only one)
.endif
```

Единственное отличие заключается в том, что первый вариант выведет ошибку после выполнения `make config`, предлагая изменить выбранные настройки.

#### Пример 49. Простое использование `OPT_PREVENTS`

Этот порт имеет опции `X509` и `SCTP`. Обе опции добавляют патчи, но патчи конфликтуют друг с другом, поэтому их нельзя выбрать одновременно.

```
OPTIONS_DEFINE= X509 SCTP

SCTP_PATCHFILES=  ${PORTNAME}-6.8p1-sctp-2573.patch.gz:-p1
SCTP_CONFIGURE_WITH=  sctp

X509_PATCH_SITES=  http://www.roumenpetrov.info/openssh/x509/:x509
X509_PATCHFILES=  ${PORTNAME}-7.0p1+x509-8.5.diff.gz:-p1:x509
X509_PREVENTS=    SCTP
X509_PREVENTS_MSG= X509 and SCTP patches conflict
```

#### 5.14.3.9. `OPT_VARS` и `OPT_VARS_OFF`

Предоставляет универсальный способ установки и добавления значений переменным.



Перед использованием `OPT_VARS` и `OPT_VARS_OFF` проверьте, доступен ли более специфичный вспомогательный инструмент в [Универсальная замена переменных](#).

Когда выбрана опция `OPT` и определены `OPT_VARS`, пары `key=value` и `key+=value` обрабатываются из `OPT_VARS`. Оператор `=` приводит к перезаписи существующего значения `KEY`, а `+=` добавляет к значению. `OPT_VARS_OFF` работает аналогично, но когда `OPT` не выбрана.

```
OPTIONS_DEFINE= OPT1 OPT2 OPT3
OPT1_VARS=  also_build+=bin1
OPT2_VARS=  also_build+=bin2
OPT3_VARS=  bin3_build=yes
OPT3_VARS_OFF=  bin3_build=no

MAKE_ARGS=  ALSO_BUILD="${ALSO_BUILD}" BIN3_BUILD="${BIN3_BUILD}"
```

эквивалентно:

```

OPTIONS_DEFINE= OPT1 OPT2

MAKE_ARGS=  ALSO_BUILD="${ALSO_BUILD}" BIN3_BUILD="${BIN3_BUILD}"

.include <bsd.port.options.mk>

.if ${PORT_OPTIONS:MOPT1}
ALSO_BUILD+=  bin1
.endif

.if ${PORT_OPTIONS:MOPT2}
ALSO_BUILD+=  bin2
.endif

.if ${PORT_OPTIONS:MOPT2}
BIN3_BUILD= yes
.else
BIN3_BUILD= no
.endif

```

Значения, содержащие пробелы, должны быть заключены в кавычки:

```
OPT_VARS=  foo="bar baz"
```



Это связано с тем, как `make(1)` обрабатывает пробелы при раскрытии переменных. Когда `OPT_VARS= foo=bar baz` раскрывается, переменная в итоге содержит две строки: `foo=bar` и `baz`. Однако отправитель, вероятно, предполагал, что должна быть только одна строка — `foo=bar baz`. Заключение значения в кавычки предотвращает использование пробела в качестве разделителя.

Также *не* добавляйте лишние пробелы после знака `var=` и перед значением, это также разобьёт строку на две части. *Это не работает:*

```
OPT_VARS=  foo=  bar
```

### 5.14.3.10. Зависимости, `OPT_DEPTYPE` и `OPT_DEPTYPE_OFF`

Для любого из этих типов зависимостей:

- `PKG_DEPENDS`
- `EXTRACT_DEPENDS`
- `PATCH_DEPENDS`
- `FETCH_DEPENDS`
- `BUILD_DEPENDS`

- LIB\_DEPENDS
- RUN\_DEPENDS

Когда выбрана опция *OPT*, значение *OPT\_DEPTYPE*, если оно определено, добавляется к *DEPTYPE*. *OPT\_DEPTYPE\_OFF* работает аналогично, но когда *не* выбрана *OPT*. Например:

```
OPTIONS_DEFINE= OPT1
OPT1_LIB_DEPENDS=  liba.so:devel/a
OPT1_LIB_DEPENDS_OFF=  libb.so:devel/b
```

эквивалентно:

```
OPTIONS_DEFINE= OPT1

.include <bsd.port.options.mk>

.if ${PORT_OPTIONS:MOP1}
LIB_DEPENDS+=  liba.so:devel/a
.else
LIB_DEPENDS+=  libb.so:devel/b
.endif
```

#### 5.14.3.11. Универсальная замена переменных, *OPT\_VARIABLE* и *OPT\_VARIABLE\_OFF*

Для любой из этих переменных:

- ALL\_TARGET
- BINARY\_ALIAS
- BROKEN
- CATEGORIES
- CFLAGS
- CONFIGURE\_ENV
- CONFLICTS
- CONFLICTS\_BUILD
- CONFLICTS\_INSTALL
- CPPFLAGS
- CXXFLAGS
- DESKTOP\_ENTRIES
- DISTFILES
- EXTRACT\_ONLY
- EXTRA\_PATCHES

- GH\_ACCOUNT
- GH\_PROJECT
- GH\_SUBDIR
- GH\_TAGNAME
- GH\_TUPLE
- GL\_ACCOUNT
- GL\_COMMIT
- GL\_PROJECT
- GL\_SITE
- GL\_SUBDIR
- GL\_TUPLE
- IGNORE
- INFO
- INSTALL\_TARGET
- LDFLAGS
- LIBS
- MAKE\_ARGS
- MAKE\_ENV
- MASTER\_SITES
- PATCHFILES
- PATCH\_SITES
- PLIST\_DIRS
- PLIST\_FILES
- PLIST\_SUB
- PORTDOCS
- PORTEXAMPLES
- SUB\_FILES
- SUB\_LIST
- TEST\_TARGET
- USES

Когда выбрана опция *OPT*, значение `OPT_ABOVEVARIABLE`, если оно определено, добавляется к `ABOVEVARIABLE`. `OPT_ABOVEVARIABLE_OFF` работает аналогично, но когда *OPT* не выбрана. Например:

```
OPTIONS_DEFINE= OPT1
OPT1_USES= gmake
```

```
OPT1_CFLAGS_OFF= -DTEST
```

ЭКВИВАЛЕНТНО:

```
OPTIONS_DEFINE= OPT1

.include <bsd.port.options.mk>

.if ${PORT_OPTIONS:MOPT1}
USES+=      gmake
.else
CFLAGS+=    -DTEST
.endif
```



Некоторые переменные отсутствуют в этом списке, в частности **PKGNAMEPREFIX** и **PKGNAME\_SUFFIX**. Это сделано намеренно. Порт *не должен* изменять своё имя при изменении набора опций.

Некоторые из этих переменных, по крайней мере **ALL\_TARGET**, **DISTFILES** и **INSTALL\_TARGET**, получают свои значения по умолчанию *после* обработки опций.

С такими строками в Makefile:

```
ALL_TARGET= all

DOCS_ALL_TARGET= doc
```



Если опция **DOCS** включена, **ALL\_TARGET** будет иметь конечное значение **all doc**; если опция отключена, значение будет **all**.

Только со строкой помощника опций в Makefile:

```
DOCS_ALL_TARGET= doc
```

Если опция **DOCS** включена, **ALL\_TARGET** будет иметь окончательное значение **doc**; если опция отключена, значение будет **all**.

#### 5.14.3.12. Дополнительные цели сборки, **target-OPT-on** и **target-OPT-off**

Эти цели в Makefile могут принимать дополнительные опциональные цели сборки:

- **pre-fetch**
- **do-fetch**
- **post-fetch**

- pre-extract
- do-extract
- post-extract
- pre-patch
- do-patch
- post-patch
- pre-configure
- do-configure
- post-configure
- pre-build
- do-build
- post-build
- pre-install
- do-install
- post-install
- post-stage
- pre-package
- do-package
- post-package

Когда выбрана опция *OPT*, цель *TARGET-OPT-on*, если она определена, выполняется после *TARGET*. *TARGET-OPT-off* работает аналогично, но когда *OPT* не выбрана. Например:

```

OPTIONS_DEFINE= OPT1

post-patch-OPT1-on:
    @${REINPLACE_CMD} -e '/opt1/s|usr/bin|${EXAMPLESDIR}/|' ${WRKSRC}/Makefile

post-patch-OPT1-off:
    @${REINPLACE_CMD} -e '/opt1/s|usr/bin|${PREFIX}/bin/|' ${WRKSRC}/Makefile

```

ЭКВИВАЛЕНТНО:

```

OPTIONS_DEFINE= OPT1

.include <bsd.port.options.mk>

post-patch:
    .if ${PORT_OPTIONS:MOPT1}
        @${REINPLACE_CMD} -e '/opt1/s|usr/bin|${EXAMPLESDIR}/|' ${WRKSRC}/Makefile
    .else

```

```
@${REINPLACE_CMD} -e '/opt1/s|usr/bin|${PREFIX}/bin|' ${WRKSRC}/Makefile
.endif
```

## 5.15. Указание рабочего каталога

Каждый порт извлекается в рабочий каталог, который должен быть доступен для записи. Система портов по умолчанию распаковывает **DISTFILES** в каталог с именем **\${DISTNAME}**. Другими словами, если в Makefile указано:

```
PORTNAME=  foo
DISTVERSION=  1.0
```

то файлы дистрибутива порта содержат каталог верхнего уровня foo-1.0, и остальные файлы находятся в этом каталоге.

Если нужно расположение файлов в других каталогах, можно переопределить ряд переменных.

### 5.15.1. WRKSRC

Переменная указывает имя каталога, который создается при распаковке distfiles приложения. Чтобы в нашем предыдущем примере распаковка происходила в каталог с именем foo (а не foo-1.0), напишите:

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

или можно

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

### 5.15.2. WRKSRC\_SUBDIR

Если исходные файлы, необходимые для порта, находятся в подкаталоге распакованного дистрибутива, присвойте **WRKSRC\_SUBDIR** имя этого каталога.

```
WRKSRC_SUBDIR=  src
```

### 5.15.3. NO\_WRKSUBDIR

Если порт не распаковывается в подкаталог вообще, установите **NO\_WRKSUBDIR**, чтобы указать это.

```
NO_WRKSUBDIR= yes
```



Поскольку **WRKDIR** является единственной директорией, которая должна быть доступна для записи во время сборки, и используется для хранения многих файлов, фиксирующих состояние сборки, извлечение порта будет принудительно выполнено в поддиректорию.

## 5.16. Обработка конфликтов

Существует три различные переменные для регистрации конфликтов между пакетами и портами: **CONFLICTS**, **CONFLICTS\_INSTALL** и **CONFLICTS\_BUILD**.



Эти переменные автоматически устанавливают переменную **IGNORE**, более подробно описанную в [Пометка порта как неустанавливаемого с помощью \*\*BROKEN\*\*](#).

При удалении одного из нескольких конфликтующих портов рекомендуется оставлять **CONFLICTS** в тех других портах на несколько месяцев, чтобы учесть пользователей, которые обновляются лишь время от времени.

### **CONFLICTS\_INSTALL**

Если пакет не может сосуществовать с другими пакетами (из-за конфликтов файлов, несовместимости во время выполнения и т.д.). Проверка **CONFLICTS\_INSTALL** выполняется после этапа сборки и перед этапом установки.

### **CONFLICTS\_BUILD**

Если порт не может быть собран, когда уже установлены другие определённые порты. Конфликты сборки не фиксируются в результирующем пакете.

### **CONFLICTS**

Если порт не может быть собран, когда определённый порт уже установлен и итоговый пакет не может сосуществовать с другим пакетом. Проверка **CONFLICTS** выполняется до этапа сборки и до этапа установки.

Каждый элемент, разделённый пробелами, в значениях переменных **CONFLICTS\*** сопоставляется с пакетами(кроме того, который собирается) с использованием правил раскрытия шаблонов имен файлов в оболочке shell. Это позволяет перечислить все варианты порта в списке конфликтов вместо необходимости исключать собираемый вариант из этого списка. Например, если установлен git-lite, **CONFLICTS\_INSTALL=git git-lite** позволит выполнить:

```
% make -C devel/git FLAVOR=lite all deinstall install
```

Но следующая команда сообщит о конфликте, так как установленное имя базового пакета — **git-lite**, а **git** будет собран, но не может быть установлен вместе с **git-lite**:

```
% make -C devel/git FLAVOR=default all deinstall install
```

Без этой функции Makefile потребовал бы по одному `_flavor__CONFLICTS_INSTALL` для каждого варианта, перечисляя все остальные варианты.

Наиболее распространённым содержимым одной из этих переменных является база пакета другого порта. База пакета — это имя пакета без указания версии, её можно получить, выполнив команду `make -V PKGBASE`.

*Пример 50. Простой пример использования `CONFLICTS*`*

Пакет `dns/bind99` не может быть установлен, если присутствует пакет `dns/bind910`, так как они устанавливают одинаковые файлы. Сначала соберите базовый пакет для использования:

```
% make -C dns/bind99 -V PKGBASE
bind99
% make -C dns/bind910 -V PKGBASE
bind910
```

Затем добавьте в Makefile пакета `dns/bind99`:

```
CONFLICTS_INSTALL= bind910
```

И добавьте в Makefile пакета `dns/bind910`:

```
CONFLICTS_INSTALL= bind99
```

Иногда только определенные версии другого порта несовместимы. В этом случае используйте полное имя пакета, включая версию. При необходимости используйте подстановочные символы шаблонов имён файлов оболочки, такие как `*` и `?`, чтобы охватить все необходимые версии.

*Пример 51. Использование `CONFLICTS*` с шаблонами имён файлов.*

В версиях с 2.0 по 2.4.1\_2 пакет `deskutils/gnotime` устанавливал встроенную версию пакета `databases/qof`.

Чтобы отразить это прошлое, Makefile пакета `databases/qof` содержит:

```
CONFLICTS_INSTALL= gnotime-2.[0-3]* \
                   gnotime-2.4.0* gnotime-2.4.1 \
                   gnotime-2.4.1_[12]
```

Первый элемент соответствует версиям 2.0–2.3, второй — всем редакциям 2.4.0, третий — точно версии 2.4.1, а последний — первой и второй редакциям версии 2.4.1.

`deskutils/gnotime` не имеет строки конфликтов, потому что его текущая версия не конфликтует ни с чем другим.

Переменная `DISABLE_CONFLICTS` может быть временно установлена при выполнении целей, на которые не влияют конфликты. Эту переменную не следует устанавливать в Makefiles портов.

```
% make -DDISABLE_CONFLICTS patch
```

## 5.17. Установка файлов



Фаза `install` очень важна для конечного пользователя, так как она добавляет файлы в его систему. Все дополнительные команды, выполняемые в целях `*-install` Makefile порта, должны выводиться на экран. Не заглушайте эти команды с помощью `@` или `.SILENT`.

### 5.17.1. Макросы `INSTALL_*`

Используйте макросы, предоставленные в `bsd.port.mk`, чтобы обеспечить корректные режимы файлов в целях `*-install` порта. Устанавливайте владельца напрямую в `pkg-plist` в соответствующих записях, таких как `@(владелец, группа,)`, `@owner` владелец и `@group` группа. Эти операторы действуют до переопределения или до конца `pkg-plist`, поэтому не забудьте сбросить их, когда они больше не нужны. Владелец по умолчанию — `root:wheel`. Дополнительную информацию см. в [Базовые Ключевые Слова](#).

- `INSTALL_PROGRAM` — команда для установки бинарных исполняемых файлов.
- `INSTALL_SCRIPT` — команда для установки исполняемых скриптов.
- `INSTALL_LIB` — это команда для установки общих библиотек (но не статических библиотек).
- `INSTALL_KLD` — это команда для установки загружаемых модулей ядра. Некоторые архитектуры не поддерживают удаление символов из модулей, поэтому используйте эту команду вместо `INSTALL_PROGRAM`.
- `INSTALL_DATA` — это команда для установки общих данных, включая статические библиотеки.
- `INSTALL_MAN` — это команда для установки map-страниц и другой документации (она ничего не сжимает).

Эти переменные передаются команде `install(1)` с соответствующими флагами для каждой ситуации.



Не используйте `INSTALL_LIB` для установки статических библиотек, так как

их удаление делает их бесполезными. Вместо этого используйте `INSTALL_DATA`.

### 5.17.2. Удаление символов из бинарных файлов и разделяемых библиотек

Установленные бинарные файлы должны быть очищены от отладочной информации. Не очищайте бинарные файлы вручную, если это не является абсолютно необходимым. Макрос `INSTALL_PROGRAM` устанавливает и очищает бинарный файл одновременно. Макрос `INSTALL_LIB` делает то же самое с разделяемыми библиотеками.

Когда файл необходимо очистить, но ни макросы `INSTALL_PROGRAM`, ни `INSTALL_LIB` не подходят, `strip` очищает программу или разделяемую библиотеку. Обычно это делается в цели `post-install`. Например:

```
post-install:
    ${STRIP_CMD} ${STAGEDIR}${PREFIX}/bin/xd1
```

Когда необходимо удалить отладочную информацию из нескольких файлов:

```
post-install:
    .for l in geometry media body track world
        ${STRIP_CMD} ${STAGEDIR}${PREFIX}/lib/lib${PORTNAME}-${l}.so.0
    .endfor
```

Используйте `file(1)` для файла, чтобы определить, был ли он подвергнут удалению символов. `file(1)` сообщает, что бинарные файлы либо `stripped` (удалены символы), либо `not stripped` (символы не удалены). Кроме того, `strip(1)` обнаружит программы, которые уже были подвергнуты удалению символов, и завершит работу без ошибок.

Когда определён `WITH_DEBUG`, elf-файлы *не должны* быть очищены.

Переменные (`STRIP_CMD`, `INSTALL_PROGRAM`, `INSTALL_LIB`, ...) и `USES`, предоставляемые фреймворком, обрабатывают это автоматически.



Некоторое программное обеспечение добавляет `-s` к своим `LDFLAGS`. В этом случае либо удалите `-s`, если установлен `WITH_DEBUG`, либо удалите его безусловно и используйте `STRIP_CMD` в `post-install`.

### 5.17.3. Установка целого дерева файлов

Иногда необходимо установить большое количество файлов с сохранением их иерархической структуры. Например, копирование всего дерева каталогов из `WRKSRC` в целевой каталог под `PREFIX`. Обратите внимание, что `PREFIX`, `EXAMPLESDIR`, `DATADIR` и другие переменные путей всегда должны предваряться `STAGEDIR` для соблюдения процедуры промежуточной установки (см. [Промежуточная установка](#)).

Для этой ситуации существуют два макроса. Преимущество использования этих макросов

вместо `ср` заключается в том, что они гарантируют целевым файлам правильные значения владельца и разрешений. Первый макрос, `COPYTREE_BIN`, устанавливает все установленные файлы как исполняемые, что делает его подходящим для установки в `PREFIX/bin`. Вторым макросом, `COPYTREE_SHARE#`, не устанавливаются исполняемые разрешения для файлов и, следовательно, подходит для установки файлов в `PREFIX/share`.

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

Этот пример установит содержимое каталога `examples` из дистрибутива вендора в соответствующее расположение примеров порта.

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

И этот пример установит данные летних месяцев в подкаталог `summer` каталога `DATADIR`.

Дополнительные аргументы `find` могут быть переданы через третий аргумент макросов `COPYTREE_*`. Например, чтобы установить все файлы из первого примера, кроме `Makefiles`, можно использовать следующие команды.

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

Эти макросы не добавляют установленные файлы в `pkg-plist`. Их необходимо добавлять вручную. Для дополнительной документации (`PORTDOCS`, см. [Установка дополнительной документации](#)) и примеров (`PORTEXAMPLES`), префиксы `%%PORTDOCS%` или `%%PORTEXAMPLES%` должны быть добавлены в `pkg-plist`.

#### 5.17.4. Установка дополнительной документации

Если у программного обеспечения есть документация, помимо стандартных страниц `man` и `info`, которая может быть полезна пользователю, установите её в `DOCSDIR`. Это можно сделать, как и в предыдущем пункте, в цели `post-install`.

Создайте новый каталог для порта. Имя каталога — `DOCSDIR`. Обычно оно равно `PORTNAME`. Однако, если пользователю может потребоваться установка разных версий порта одновременно, можно использовать полное имя `PKGNAME`.

Поскольку устанавливаются только файлы, перечисленные в `pkg-plist`, можно безопасно всегда устанавливать документацию в `STAGEDIR` (см. [Staging](#)). Поэтому блоки `.if` требуются

только в тех случаях, когда устанавливаемые файлы достаточно велики, чтобы вызвать значительные накладные расходы на ввод-вывод.

```
post-install:
  ${MKDIR} ${STAGEDIR}${DOCSDIR}
  ${INSTALL_DATA} ${WRKSRC}/docs/xvdocs.ps ${STAGEDIR}${DOCSDIR}
```

С другой стороны, если в порте есть опция DOCS, установите документацию в цели `post-install-DOCS-on`. Эти цели описаны в [Дополнительные цели сборки](#).

Вот несколько полезных переменных и их стандартное раскрытие при использовании в Makefile:

- `DATADIR` раскрывается в `PREFIX/share/PORTNAME`.
- `DATADIR_REL` раскрывается в `share/PORTNAME`.
- `DOCSDIR` раскрывается в `PREFIX/share/doc/PORTNAME`.
- `DOCSDIR_REL` раскрывается в `share/doc/PORTNAME`.
- `EXAMPLESDIR` раскрывается в `PREFIX/share/examples/PORTNAME`.
- `EXAMPLESDIR_REL` раскрывается в `share/examples/PORTNAME`.



Опция `DOCS` управляет только дополнительной документацией, устанавливаемой в `DOCSDIR`. Она не применяется к стандартным man-страницам и info-страницам. Содержимое, устанавливаемое в `EXAMPLESDIR`, контролируется опцией `EXAMPLES`.

Эти переменные экспортируются в `PLIST_SUB`. Их значения будут представлены там в виде путей относительно `PREFIX`, если это возможно. То есть, `share/doc/PORTNAME` будет заменено на `%%DOCSDIR%%` в списке упаковки по умолчанию и так далее. (Подробнее о подстановках в `pkg-plist` см. [здесь](#).)

Все условно устанавливаемые файлы и каталоги документации включаются в `pkg-plist` с префиксом `%%PORTDOCS%%`, например:

```
%%PORTDOCS%%DOCSDIR%/AUTHORS
%%PORTDOCS%%DOCSDIR%/CONTACT
```

В качестве альтернативы перечислению файлов документации в `pkg-plist`, порт может установить переменную `PORTDOCS` в список имён файлов и шаблонов имен файлов shell для добавления в итоговый список упаковки. Имена будут относительно `DOCSDIR`. Поэтому порт, использующий `PORTDOCS` и нестандартное расположение документации, должен соответствующим образом установить `DOCSDIR`. Если в `PORTDOCS` указан каталог или он соответствует шаблону из этой переменной, всё поддерево содержащихся файлов и каталогов будет зарегистрировано в итоговом списке упаковки. Если опция `DOCS` отключена, файлы и каталоги, перечисленные в `PORTDOCS`, не будут установлены или добавлены в список упаковки порта. Установка документации в `PORTDOCS`, как показано выше, остаётся на

усмотрение самого порта. Типичный пример использования **PORTDOCS**:

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



Эквивалентами **PORTDOCS** для файлов, установленных в **DATADIR** и **EXAMPLESDIR**, являются **PORTDATA** и **PORTEXAMPLES** соответственно.

Содержимое файла `pkg-message` отображается при установке. Подробности см. в разделе [использование файла pkg-message](#). Файл `pkg-message` не нужно добавлять в `pkg-plist`.

### 5.17.5. Подкаталоги в **PREFIX**

Попробуйте сделать так, чтобы порт размещал файлы в правильных подкаталогах **PREFIX**. Некоторые порты собирают всё в кучу и помещают в подкаталог с именем порта, что неверно. Также многие порты размещают все файлы, кроме бинарников, заголовочных файлов и страниц руководства, в подкаталоге `lib`, что плохо согласуется с парадигмой BSD. Многие из этих файлов должны быть перемещены в один из следующих каталогов: `etc` (файлы настройки/конфигурации), `libexec` (исполняемые файлы для внутреннего использования), `sbin` (исполняемые файлы для суперпользователей/администраторов), `info` (документация для браузера `info`) или `share` (архитектурно-независимые файлы). Подробности см. в [hier\(7\)](#); правила, действующие для `/usr`, в основном применимы и к `/usr/local`. Исключение составляют порты, связанные с USENET "news". Они могут использовать `PREFIX/news` в качестве места назначения для своих файлов.

## 5.18. Используйте **BINARY\_ALIAS** для переименования команд вместо исправления сборки

Когда определена переменная **BINARY\_ALIAS**, будут созданы символьные ссылки на указанные команды в каталоге, который будет добавлен в начало переменной **PATH**.

Используйте это для замены жёстко заданных команд, от которых зависит этап сборки, без необходимости исправлять какие-либо файлы сборки.

*Пример 52. Использование **BINARY\_ALIAS** для предоставления `gsed` в качестве `sed`*

Некоторые порты ожидают, что `sed` будет вести себя как GNU `sed` и используют возможности, которые `sed(1)` не предоставляет. GNU `sed` доступен в пакете [textproc/gsed](#) на FreeBSD.

Используйте **BINARY\_ALIAS** для замены `sed` на `gsed` на время сборки:

```
BUILD_DEPENDS=  gsed:textproc/gsed
...
BINARY_ALIAS=  sed=gsed
```

Пример 53. Использование `BINARY_ALIAS` для создания псевдонимов жестко заданных команд `python3`

Порт, в котором есть жёсткая ссылка на `python3` в скриптах сборки, требует его наличия в `PATH` во время сборки. Используйте `BINARY_ALIAS` для создания псевдонима, указывающего на нужный бинарный файл Python 3:

```
USES= python:3.4+,build
...
BINARY_ALIAS= python3=${PYTHON_CMD}
```

См. [Использование Python](#) для получения дополнительной информации о `USES=python`.



Бинарные псевдонимы создаются после обработки зависимостей, указанных через `BUILD_DEPENDS` и `LIB_DEPENDS`, но до цели `configure`. Это приводит к различным ограничениям. Например, программы, установленные через `TEST_DEPENDS`, нельзя использовать для создания бинарного псевдонима, так как тестовые зависимости, указанные таким образом, обрабатываются после создания бинарных псевдонимов.

# Глава 6. Особые соглашения

Имеется ещё несколько вещей, которые вы должны иметь в виду при создании порта. Этот раздел описывает наиболее часто встречающиеся из них.

## 6.1. Разделение длинных файлов

Иногда Makefiles могут быть очень длинными. Например, порты `rust` могут содержать очень длинный список `CARGO_CRATES`. В других случаях Makefile может содержать код, который варьируется в зависимости от архитектуры. В таких ситуациях может быть удобно разделить исходный Makefile на несколько файлов. `bsd.port.mk` автоматически включает некоторые типы Makefiles в основной Makefile порта.

Вот файлы, которые система обрабатывает автоматически, если они обнаружены:

- `Makefile.crates`. Пример можно найти в пакете [audio/ebur128](#).
- `Makefile.inc`. Пример можно найти в пакете [net/ntp](#).
- `Makefile.${ARCH}-${OPYSYS}`
- `Makefile.${OPYSYS}`. Пример можно найти в пакете [net/cvsup-static](#).
- `Makefile.${ARCH}`
- `Makefile.local`

Также распространённой практикой является разделение списка пакетов порта на несколько файлов, если список сильно различается в зависимости от архитектуры или выбранного флейвора. В этом случае файл `pkg-plist` для каждой архитектуры именуется по шаблону `pkg-plist.${ARCH}` или `pkg-plist.${FLAVOR}`. Фреймворк не создаёт список пакетов автоматически, если существует несколько файлов `pkg-plist`. Ответственность за выбор подходящего файла `pkg-plist` и его присвоение переменной `PLIST` лежит на того, кто делает порт. Примеры работы с этим можно найти в портах [audio/logitechmediaserver](#) и [deskutils/libportal](#).

## 6.2. Staging

`bsd.port.mk` ожидает, что порты будут работать с "директорией стадии". Это означает, что порт не должен устанавливать файлы напрямую в обычные целевые директории (например, под `PREFIX`), а вместо этого в отдельную директорию, из которой затем собирается пакет. Во многих случаях это не требует прав `root`, что позволяет собирать пакеты от имени непривилегированного пользователя. При использовании стадии порт собирается и устанавливается в директорию стадии `STAGEDIR`. Пакет создаётся из директории стадии и затем устанавливается в систему. Инструменты Automake называют эту концепцию `DESTDIR`, но в FreeBSD `DESTDIR` имеет другое значение (см. [PREFIX](#) и [DESTDIR](#)).



Ни один порт *на самом деле* не должен работать от `root`. Этого в большинстве случаев можно избежать, используя `USES=uidfix`. Если порт всё ещё выполняет команды, такие как `chown(8)`, `chgrp(1)`, или принудительно

устанавливает владельца или группу с помощью `install(1)`, то используйте `USES=fakeroot`, чтобы подделывать эти вызовы. Потребуется некоторая правка Makefiles порта.

Метапорты, то есть порты, которые не устанавливают файлы непосредственно, а только зависят от других портов, должны по возможности избегать распаковки `mtree(8)` в каталог сборки. Это основная иерархия каталогов пакета, и эти пустые каталоги будут выглядеть лишними. Для предотвращения распаковки `mtree(8)` добавьте эту строку:

```
NO_MTREE= yes
```



Метапорты должны использовать `USES=metaport`. Это устанавливает значения по умолчанию для портов, которые не загружают, не собирают и не устанавливают ничего.

Этап подготовки включается путем добавления `STAGEDIR` к путям, используемым в целях `pre-install`, `do-install` и `post-install` (см. примеры в книге). Обычно это включает `PREFIX`, `ETCDIR`, `DATADIR`, `EXAMPLESDIR`, `DOCSDIR` и т. д. Каталоги должны создаваться как часть цели `post-install`. По возможности избегайте использования абсолютных путей.



Порты, которые устанавливают модули ядра, должны добавлять `STAGEDIR` к пути назначения, по умолчанию это `/boot/modules`.

### 6.2.1. Обработка символических ссылок

При создании символической ссылки настоятельно рекомендуется использовать относительные пути. Используйте `${RLN}` для создания относительных символических ссылок. Эта команда использует `install(1)` для автоматического определения относительной ссылки, которую нужно создать.

*Пример 54. Автоматическое создание относительных символических ссылок*

`${RLN}` использует относительную символическую ссылку из `install(1)`, что освобождает сборщика порта от вычисления относительного пути.

```
${RLN} ${STAGEDIR}${PREFIX}/lib/libfoo.so.42 ${STAGEDIR}${PREFIX}/lib/libfoo.so
${RLN} ${STAGEDIR}${PREFIX}/libexec/foo/bar ${STAGEDIR}${PREFIX}/bin/bar
${RLN} ${STAGEDIR}/var/cache/foo ${STAGEDIR}${PREFIX}/share/foo
```

Будет создано:

```
% ls -lF ${STAGEDIR}${PREFIX}/lib
lrwxr-xr-x 1 nobody nobody 181 Aug 3 11:27 libfoo.so@ -> libfoo.so.42
-rwxr-xr-x 1 nobody nobody 15 Aug 3 11:24 libfoo.so.42*
% ls -lF ${STAGEDIR}${PREFIX}/bin
lrwxr-xr-x 1 nobody nobody 181 Aug 3 11:27 bar@ -> ../libexec/foo/bar
```

```
% ls -lF ${STAGEDIRDIR}${PREFIX}/share  
lrwxr-xr-x 1 nobody nobody 181 Aug 3 11:27 foo@ -> ../../../../var/cache/foo
```

## 6.3. Встроенные библиотеки

Этот раздел объясняет, почему встроенные зависимости считаются плохими и что с этим делать.

### 6.3.1. Почему встроенные библиотеки — это плохо

Некоторое программное обеспечение требует от упаковщика найти сторонние библиотеки и добавить необходимые зависимости в порт. Другое ПО включает все необходимые библиотеки в дистрибутивный файл. Второй подход кажется проще на первый взгляд, но имеет серьёзные недостатки:

Этот список частично основан на вики [Fedora](#) и [Gentoo](#), оба распространяются по лицензии [CC-BY-SA 3.0](#).

#### Безопасность

Если уязвимости обнаружены в вышестоящей библиотеке и исправлены там, они могут остаться неисправленными в библиотеке, поставляемой с портом. Одной из причин может быть то, что автор не знает о проблеме. Это означает, что портировщик должен исправить их или обновить до не уязвимой версии и отправить исправление автору. Всё это требует времени, что приводит к тому, что программное обеспечение остаётся уязвимым дольше, чем необходимо. Это, в свою очередь, затрудняет координацию исправления без неоправданного раскрытия информации об уязвимости.

#### Ошибки

Эта проблема аналогична проблеме с безопасностью в последнем абзаце, но в целом менее серьёзная.

#### Ветвление

Автору проще создать форк upstream-библиотеки после её включения в дистрибутив. Хотя на первый взгляд это кажется удобным, такой подход приводит к расхождению кода с исходным репозиторием, что усложняет исправление уязвимостей и других проблем в ПО. Одна из причин — затруднённое применение патчей.

Ещё одна проблема форкинга заключается в том, что из-за расхождения кода с основной веткой, ошибки исправляются снова и снова, вместо того чтобы быть исправленными один раз в центральном месте. Это противоречит самой идее открытого программного обеспечения.

#### Конфликты символов

Когда библиотека установлена в системе, может возникнуть конфликт с встроенной в порт версией. Это может привести к немедленным ошибкам во время компиляции или компоновки. Также могут возникать ошибки при запуске программы, которые сложнее отследить. Последняя проблема может быть вызвана несовместимостью версий двух

библиотек.

## Лицензирование

При объединении проектов из различных источников могут возникать проблемы с лицензиями, особенно если лицензии несовместимы.

## Растрата ресурсов

Библиотеки, поставляемые в комплекте, растрачивают ресурсы на нескольких уровнях. Сборка самого приложения занимает больше времени, особенно если эти библиотеки уже присутствуют в системе. Во время выполнения они могут занимать дополнительную память, когда общесистемная библиотека уже загружена одной программой, а входящая в комплект библиотека загружена другой программой.

## Пустая трата усилий

Когда библиотеке требуются патчи для FreeBSD, эти патчи приходится дублировать в составе библиотеки. Это приводит к потере времени разработчиков, поскольку патчи могут применяться некорректно. Кроме того, бывает сложно сразу заметить, что эти патчи вообще необходимы.

### 6.3.2. Что делать со встроенными библиотеками

По возможности используйте независимую версию библиотеки, добавив `LIB_DEPENDS` в порт. Если такого порта ещё не существует, рассмотрите возможность его создания.

Используйте встроенные библиотеки только в том случае, если разработчик имеет хорошую репутацию в вопросах безопасности, а использование внешних версий приводит к излишне сложным исправлениям.



В некоторых особых случаях, например, для эмуляторов, таких как Wine, порт должен включать библиотеки, потому что они предназначены для другой архитектуры или были изменены для использования в данном программном обеспечении. В таком случае эти библиотеки не должны быть доступны для связывания с другими портами. Добавьте `BUNDLE_LIBS=yes` в Makefile порта. Это укажет `pkg(8)` не учитывать предоставляемые библиотеки. Всегда спрашивайте Группа Менеджеров Деревя Портов FreeBSD [portmgr@FreeBSD.org](mailto:portmgr@FreeBSD.org), прежде чем добавлять это в порт.

## 6.4. Динамические библиотеки

Если ваш порт устанавливает одну или несколько динамических библиотек, определите переменную `USE_LDCONFIG`, которая приведёт к запуску из `bsd.port.mk` команды `${LDCONFIG} -m` относительно каталога, в который устанавливается новая библиотека (как правило, это `PREFIX/lib`), во время выполнения цели `post-install` для её регистрации в кэше динамических библиотек. Эта переменная, если она определена, также приведёт к добавлению соответствующей пары команд `@exec /sbin/ldconfig -m` и `@unexec /sbin/ldconfig -R` в ваш файл `pkg-plist`, так что пользователь, устанавливающий пакет, сможет сразу же использовать динамическую библиотеку, а удаление пакета не приведёт к тому, что система будет предполагать, что библиотека всё ещё имеется в наличии.

```
USE_LDCONFIG= yes
```

Если нужно, вы можете переопределить каталог по умолчанию, задав значение `USE_LDCONFIG`, в котором должны быть перечислены каталоги, в которые устанавливаются динамические библиотеки. Например, если ваш порт устанавливает динамические библиотеки в каталоги `PREFIX/lib/foo` и `PREFIX/lib/bar`, то вы можете в файле `Makefile` указать следующее:

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

Будьте добры перепроверить, т.к. часто это вовсе не является необходимым и может быть решено иначе с помощью `-rpath` или установки `LD_RUN_PATH` во время компоновки (для примера смотрите [lang/moscow\\_ml](#)), или с помощью сценария-обёртки, который выставляет `LD_LIBRARY_PATH` перед запуском исполняемого файла как это делает [www/seamonkey](#).

При установке 32-разрядных библиотек на 64-разрядной системе используйте вместо этого `USE_LDCONFIG32`.

Если программное обеспечение использует `autotools`, в частности `libtool`, добавьте `USES=libtool`.

Если при обновлении порта увеличивается старший номер версии библиотеки, то для всех портов, компонуемых с затронутой библиотекой, следует увеличить значение `PORTREVISION` для форсирования перекомпиляции с новой версией библиотеки.

## 6.5. Порты с ограничениями на распространение или с правовым обременением

Лицензии бывают разных видов, и некоторые накладывают ограничение на то, как приложение может быть оформлено в виде пакета, может ли оно продаваться для извлечения коммерческой выгоды, и так далее.



На вас, как на человека, портирующего приложение, ложится обязанность прочесть лицензионные соглашения на программное обеспечение и удостовериться, что проект FreeBSD не будет являться их нарушителем, если будет заниматься распространением исходного кода или в бинарном виде по FTP/HTTP или на CD-ROM. Если у вас возникли сомнения, то, пожалуйста, обратитесь в [Список рассылки, посвящённый Портam FreeBSD](#).

В подобных ситуациях можно использовать переменные, описываемые в последующих разделах.

### 6.5.1. NO\_PACKAGE

Эта переменная указывает, что мы не можем создавать для приложения двоичный пакет. К примеру, лицензия не позволяет бинарное распространение или она может запрещать

распространение пакетов, созданных из изменённых исходников.

Однако файлы **DISTFILES** могут свободно зеркалироваться по FTP/HTTP. Они также могут распространяться, используя CD-ROM (или на похожих носителях), если не установлена переменная **NO\_CDROM**.

**NO\_PACKAGE** должна также использоваться, если двоичный пакет, как правило, бесполезен, а приложение должно всегда компилироваться из исходного кода. К примеру, если в приложение во время компиляции жёстко включается конфигурационная информация, привязанная к конкретной системе, то задайте переменную **NO\_PACKAGE**.

Значением переменной **NO\_PACKAGE** должна быть строка, описывающая причину, по которой пакет не должен создаваться.

### 6.5.2. **NO\_CDROM**

Эта переменная указывает на то, что, хотя мы имеем право создавать бинарные пакеты, мы не можем помещать эти пакеты или файлы **DISTFILES** порта на CD-ROM (или на похожие носители) для перепродажи. Однако бинарные пакеты и файлы **DISTFILES** порта будут оставаться доступными посредством FTP/HTTP.

Если эта переменная устанавливается вместе с **NO\_PACKAGE**, то только файлы порта **DISTFILES** будут доступны, и только посредством FTP/HTTP.

В качестве значения **NO\_CDROM** должна указываться строка, описывающая причины, по которым порт не может распространяться на CD-ROM. К примеру, это применяется, если лицензионное соглашение приложения предполагает только его "некоммерческое" использование.

### 6.5.3. **NOFETCHFILES**

Файлы, определенные в переменной **NOFETCHFILES**, не будут извлекаться ни из одного из **MASTER\_SITES**. Примером такого файла является файл, поставляемый на CD-ROM.

Инструменты, проверяющие доступность этих файлов на **MASTER\_SITES**, должны игнорировать эти файлы и не сообщать о них.

### 6.5.4. **RESTRICTED**

Задайте эту переменную, если лицензия на приложение не позволяет ни зеркалировать файлы **DISTFILES**, ни распространять бинарный пакет через FTP/HTTP или на CD-ROM.

Ни **NO\_CDROM**, ни **NO\_PACKAGE** не стоит устанавливать вместе с **RESTRICTED**, так как последняя переменная подразумевает первые две.

В качестве значения **RESTRICTED** должна указываться строка, описывающая причины, по которым порт нельзя распространять. Обычно это означает, что порт использует закрытое программное обеспечение, а пользователь должен вручную скачать файлы **DISTFILES**, возможно, после заполнения регистрационной формы или подтверждения соглашения с условиями EULA.

### 6.5.5. RESTRICTED\_FILES

Если заданы `RESTRICTED` или `NO_CDROM`, то значение этой переменной по умолчанию соответствует `${DISTFILES} ${PATCHFILES}`, в противном случае она пуста. Если ограничены в распространении лишь некоторые из дистрибутивных файлов, то в этой переменной задаётся их список.

### 6.5.6. LEGAL\_TEXT

Если порт имеет правовое обременение, которое не покрывается перечисленными выше переменными, то переменной `LEGAL_TEXT` следует присвоить строку с описанием данного обременения. Например, если было получено особое разрешение для FreeBSD на распространение двоичного файла, то эта переменная должна содержать соответствующее указание.

### 6.5.7. /usr/ports/LEGAL и LEGAL

Порт, содержащий любую из перечисленных выше переменных, также должен быть добавлен в `/usr/ports/LEGAL`. Первый столбец содержит шаблон совпадения с дистрибутивными файлами, имеющими ограничения на распространение. Второй столбец содержит корень порта. Третий столбец содержит вывод `make -VLEGAL`.

### 6.5.8. Примеры

Предпочтительным способом реализации утверждения "архивы исходных текстов для этого порта должны загружаться самостоятельно" является следующее:

```
.if !exists(${DISTDIR}/${DISTNAME}${EXTRACT_SUFFIX})
IGNORE= may not be redistributed because of licensing reasons. Please visit some-
website to accept their license and download ${DISTFILES} into ${DISTDIR}
.endif
```

Это одновременно и информирует пользователя, и устанавливает нужные метаданные на пользовательской машине для использования автоматическими программами.

Обратите внимание, что данная кляуза должна предшествовать подключению файла `bsd.port.pre.mk`.

## 6.6. Механизмы построения

### 6.6.1. Параллельное построение портов

Инфраструктура портов FreeBSD поддерживает параллельное построение с использованием множественных подпроцессов `make`, что позволяет системам SMP задействовать всю доступную мощность CPU, тем самым делая построение портов более быстрым и эффективным.

Это достигается путём передачи флага `-jX` команде `make(1)`. Такое построение портов

является поведением по умолчанию. К сожалению, не все порты поддерживают параллельную сборку достаточно хорошо, и поэтому может потребоваться выключить этот механизм явным образом путём добавления переменной `MAKE_JOBS_UNSAFE=yes`. Эта переменная используется в случае, когда известно, что порт ломается с `-jX`.



При установке `MAKE_JOBS_UNSAFE` очень важно объяснить либо комментарием в Makefile, либо хотя бы в сообщении коммита, *почему* порт не собирается при включении. В противном случае практически невозможно ни исправить проблему, ни проверить, была ли она исправлена при коммите обновления в дальнейшем.

### 6.6.2. `make`, `gmake` и `imake`

Существует несколько различных реализаций `make`. Переносимому программному обеспечению часто требуется конкретная реализация, например GNU `make`, известная в FreeBSD как `gmake`.

Если порт использует GNU `make`, добавьте `gmake` в `USES`.

`MAKE_CMD` может использоваться для ссылки на конкретную команду, настроенную параметром `USES` в Makefile порта. Используйте `MAKE_CMD` только внутри Makefile приложения в `WRKSRC` для вызова реализации `make`, ожидаемой портируемым программным обеспечением.

Если ваш порт является приложением X, которое создает файлы Makefile из Imakefile, используя `imake`, то установите `USES= imake`. Это заставит стадию конфигурирования автоматически выполнить `xmkmf -a`. Если флаг `-a` представляет для вашего порта проблему, то установите `XMKMF=xmkmf`. Если порт использует `imake`, но не понимает цель `install.man`, то следует установить `NO_INSTALL_MANPAGES=yes`.

Если исходный Makefile вашего порта имеет что-нибудь помимо `all` в качестве основной цели построения, то задайте соответствующее значение `ALL_TARGET`. То же касается `install` и `INSTALL_TARGET`.

### 6.6.3. Сценарий `configure`

Если ваш порт использует сценарий `configure` для получения файлов Makefile из файлов `Makefile.in`, то установите `GNU_CONFIGURE=yes`. Если вы хотите дать дополнительные параметры сценарию `configure` (аргументом по умолчанию является `--prefix=${PREFIX} --infodir=${PREFIX}/${INFO_PATH} --mandir=${MANPREFIX}/man --build=${CONFIGURE_TARGET}`), установите эти параметры в `CONFIGURE_ARGS`. Дополнительные переменные окружения можно передать, используя переменную `CONFIGURE_ENV`.

Таблица 9. Переменные для портов, использующих `configure`

Переменная	Значение
<code>GNU_CONFIGURE</code>	Порт использует сценарий <code>configure</code> для подготовки построения.

Переменная	Значение
HAS_CONFIGURE	То же, что и GNU_CONFIGURE, кроме того, что цель configure по умолчанию не добавляется в CONFIGURE_ARGS.
CONFIGURE_ARGS	Дополнительные параметры, передаваемые сценарию configure.
CONFIGURE_ENV	Дополнительные переменные окружения, задаваемые для запуска сценария configure.
CONFIGURE_TARGET	Переопределить цель configure по умолчанию. Значением по умолчанию является <code>\${MACHINE_ARCH}-portbld-freebsd\${OSREL}</code> .

#### 6.6.4. Использование cmake

Если порт использует CMake, определите `USES= cmake`.

Таблица 10. Переменные для портов, использующих cmake

Переменная	Значение
CMAKE_ARGS	Специфичные для порта флаги CMake, передаваемые в бинарный файл cmake.
CMAKE_ON	Для каждой записи в CMAKE_ON добавляется булево значение "включено" в CMAKE_ARGS. См. CMAKE_ON и CMAKE_OFF.
CMAKE_OFF	Для каждой записи в CMAKE_OFF в CMAKE_ARGS добавляется отключенное булево значение. См. CMAKE_ON и CMAKE_OFF.
CMAKE_BUILD_TYPE	Тип сборки (предопределённые профили сборки CMake). По умолчанию Release, или Debug, если установлен WITH_DEBUG.
CMAKE_SOURCE_PATH	Путь к исходному каталогу. По умолчанию <code>\${WRKSRC}</code> .
CONFIGURE_ENV	Дополнительные переменные окружения, которые должны быть установлены для бинарного файла cmake.

Таблица 11. Переменные, которые пользователи могут определить для сборок cmake

Переменная	Значение
CMAKE_NOCOLOR	Запрещает цветной вывод сообщений при построении. Значение по умолчанию не задано, если не заданы BATCH или PACKAGE_BUILDING.

CMake поддерживает следующие профили построения: `Debug`, `Release`, `RelWithDebInfo` и `MinSizeRel`. Профили `Debug` и `Release` учитывают системные флаги `*FLAGS`; `RelWithDebInfo` и `MinSizeRel` соответственно определяют `CFLAGS` со значением `-O2 -g` и `-Os -DNDEBUG`. Значение `CMAKE_BUILD_TYPE` экспортируется в нижнем регистре в `PLIST_SUB` и должно использоваться, если порт устанавливает файлы `*.cmake` в зависимости от типа построения (для примера посмотрите на [deskutils/strigi](#)). Следует учитывать, что некоторые проекты могут определять собственные профили построения и/или форсировать конкретный тип построения через установку `CMAKE_BUILD_TYPE` в файлах `CMakeLists.txt`. Для того чтобы порт для такого проекта учитывал `CFLAGS` и `WITH_DEBUG`, из этих файлов должны быть удалены значения `CMAKE_BUILD_TYPE`.

Большинство проектов, основанных на CMake, поддерживают метод внешнего (out-of-source) построения. Для порта внешнее построение можно запросить с использованием суффикса `:outsource`. В этом случае `CONFIGURE_WKSR`, `BUILD_WKSR` и `INSTALL_WKSR` будут иметь значение `${WRKDIR}/.build` для каталога, содержащего файлы, получаемые на этапах конфигурации и построения; при этом каталог с исходным кодом будет оставаться без изменений.

*Пример 55. Пример для USES= cmake*

Следующий отрывок демонстрирует использование CMake для порта. `CMAKE_SOURCE_PATH` обычно не требуется, но может быть установлен, когда исходный код не находится в верхнем каталоге или если порт используется для построения части проекта.

```
USES=          cmake
CMAKE_SOURCE_PATH=  ${WRKSR}/subproject
```

*Пример 56. CMAKE\_ON и CMAKE\_OFF*

При добавлении логических значений в `CMAKE_ARGS` проще использовать переменные `CMAKE_ON` и `CMAKE_OFF` вместо этого. Это:

```
CMAKE_ON=  VAR1 VAR2
CMAKE_OFF= VAR3
```

Эквивалентно:

```
CMAKE_ARGS= -DVAR1:BOOL=TRUE -DVAR2:BOOL=TRUE -DVAR3:BOOL=FALSE
```



Это относится только к значениям по умолчанию в `CMAKE_ARGS`. Вспомогательные функции, описанные в `OPT_CMAKE_BOOL` и `OPT_CMAKE_BOOL_OFF`, используют ту же семантику, но для опциональных значений.

### 6.6.5. Использование `scons`

Если порт использует SCons, определите `USES=scons`.

Чтобы сторонний SConstruct учитывал все, что передается в SCons через окружение (то есть, что наиболее важно, `CC/CXX/CFLAGS/CXXFLAGS`), исправьте SConstruct, чтобы сборка `Environment` создавалась следующим образом:

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

Он может быть изменён с помощью `env.Append` и `env.Replace`.

### 6.6.6. Сборка приложений на Rust с помощью `cargo`

Для портов, использующих Cargo, определите `USES=cargo`.

Таблица 12. Переменные, которые пользователи могут определить для сборок `cargo`

Переменная	По умолчанию	Описание
<code>CARGO_CRATES</code>		Список ящиков (crates), от которых зависит порт. Каждая запись должна быть в формате <code>имя_ящика-семвер</code> , например, <code>libc-0.2.40</code> . Поддерживающие порт могут сгенерировать этот список из файла <code>Cargo.lock</code> с помощью команды <code>make cargo-crates</code> . Вручную обновлять версии ящиков возможно, но следует учитывать транзитивные зависимости. Если список, сгенерированный <code>make cargo-crates</code> , слишком велик, его можно разместить в файле <code>Makefile.crates</code> в корневом каталоге порта. Если такой файл присутствует, фреймворк портов автоматически загружает его. Это помогает сохранять основной <code>Makefile</code> порта в удобном для работы размере.

Переменная	По умолчанию	Описание
CARGO_FEATURES		Список функций приложения для сборки (список через пробел). Чтобы отключить все функции по умолчанию, добавьте специальный токен <code>--no-default-features</code> в CARGO_FEATURES. Вручную передавать его в CARGO_BUILD_ARGS, CARGO_INSTALL_ARGS и CARGO_TEST_ARGS не требуется.
CARGO_CARGOTOML	<code>\${WRKSRС}/Cargo.toml</code>	Путь к файлу Cargo.toml, который следует использовать.
CARGO_CARGOLOCK	<code>\${WRKSRС}/Cargo.lock</code>	Путь к файлу Cargo.lock, используемому для <code>make cargo-crates</code> . Можно указать более одного файла блокировки, если это необходимо.
CARGO_ENV		Список переменных окружения, передаваемых в Cargo, аналогично MAKE_ENV.
RUSTFLAGS		Флаги для передачи компилятору Rust.
CARGO_CONFIGURE	yes	Используйте стандартный <code>do-configure</code> .
CARGO_UPDATE_ARGS		Дополнительные аргументы для передачи Cargo во время фазы настройки. Допустимые аргументы можно посмотреть с помощью <code>cargo update --help</code> .
CARGO_BUILDDEP	yes	Добавить зависимость сборки на <code>lang/rust</code> .
CARGO_CARGO_BIN	<code>\${LOCALBASE}/bin/cargo</code>	Расположение бинарного файла <code>cargo</code> .
CARGO_BUILD	yes	Используйте значение по умолчанию <code>do-build</code> .

Переменная	По умолчанию	Описание
<code>CARGO_BUILD_ARGS</code>		Дополнительные аргументы для передачи Cargo во время фазы сборки. Допустимые аргументы можно посмотреть с помощью <code>cargo build --help</code> .
<code>CARGO_INSTALL</code>	<code>yes</code>	Используйте настройку <code>do-install</code> по умолчанию.
<code>CARGO_INSTALL_ARGS</code>		Дополнительные аргументы для передачи Cargo во время фазы установки. Допустимые аргументы можно посмотреть с помощью <code>cargo install --help</code> .
<code>CARGO_INSTALL_PATH</code>	<code>.</code>	Путь к пакету для установки. Этот аргумент передается в <code>cargo install</code> через параметр <code>--path</code> . Если указано несколько путей, <code>cargo install</code> запускается несколько раз.
<code>CARGO_TEST</code>	<code>yes</code>	Используйте значение по умолчанию <code>do-test</code> .
<code>CARGO_TEST_ARGS</code>		Дополнительные аргументы для передачи Cargo во время тестовой фазы. Допустимые аргументы можно посмотреть с помощью <code>cargo test --help</code> .
<code>CARGO_TARGET_DIR</code>	<code>\${WRKDIR}/target</code>	Расположение выходного каталога cargo.
<code>CARGO_DIST_SUBDIR</code>	<code>rust/crates</code>	Каталог относительно <code>DISTDIR</code> , в котором будут храниться файлы дистрибутивов пакетов (crates).
<code>CARGO_VENDOR_DIR</code>	<code>\${WRKSRC}/cargo-crates</code>	Расположение каталога сторонних поставщиков ПО, в который будут извлечены все крейты. Старайтесь держать его в пределах <code>PATCH_WRKSRC</code> , чтобы упростить применение патчей.

Переменная	По умолчанию	Описание
<code>CARGO_USE_GITHUB</code>	no	Включить загрузку крейтов, привязанных к определённым коммитам Git на GitHub, с помощью <code>GH_TUPLE</code> . Это попытается исправить все файлы <code>Cargo.toml</code> в <code>WRKDIR</code> , чтобы они ссылались на автономные источники вместо загрузки из репозитория Git во время сборки.
<code>CARGO_USE_GITLAB</code>	no	То же, что и <code>CARGO_USE_GITHUB</code> , но для экземпляров GitLab и <code>GL_TUPLE</code> .

*Пример 57. Создание порта для простого приложения на Rust*

Создание порта на основе Cargo — это процесс из трёх этапов. Сначала необходимо предоставить шаблон портов, который загружает дистрибутивный файл приложения:

```

PORTNAME=  tokei
DISTVERSIONPREFIX=  v
DISTVERSION=  7.0.2
CATEGORIES=  devel

MAINTAINER=  tobik@FreeBSD.org
COMMENT=  Display statistics about your code
WWW=  https://github.com/XAMPPRocky/tokei/

USES=  cargo
USE_GITHUB=  yes
GH_ACCOUNT=  Aaronpower

.include <bsd.port.mk>

```

Сгенерировать первоначальный distinfo:

```

% make makesum
=> Aaronpower-tokei-v7.0.2_GH0.tar.gz doesn't seem to exist in
/usr/ports/distfiles/.
=> Attempting to fetch
https://codeload.github.com/Aaronpower/tokei/tar.gz/v7.0.2?dummy=/Aaronpower-
tokei-v7.0.2_GH0.tar.gz
fetch:
https://codeload.github.com/Aaronpower/tokei/tar.gz/v7.0.2?dummy=/Aaronpower-
tokei-v7.0.2_GH0.tar.gz: size of remote file is not known

```

Теперь файл дистрибутива готов к использованию, и мы можем продолжить, извлекая зависимости пакета из встроенного файла Cargo.lock:

```
% make cargo-crates
CARGO_CRATES=  aho-corasick-0.6.4 \
                ansi_term-0.11.0 \
                arrayvec-0.4.7 \
                atty-0.2.9 \
                bitflags-1.0.1 \
                byteorder-1.2.2 \
                [...]
```

Вывод этой команды необходимо вставить напрямую в Makefile:

```
PORTNAME=  tokei
DISTVERSIONPREFIX=  v
DISTVERSION=  7.0.2
CATEGORIES=  devel

MAINTAINER=  tobik@FreeBSD.org
COMMENT=  Display statistics about your code
WWW=  https://github.com/XAMPPRocky/tokei/

USES=  cargo
USE_GITHUB=  yes
GH_ACCOUNT=  Aaronpower

CARGO_CRATES=  aho-corasick-0.6.4 \
                ansi_term-0.11.0 \
                arrayvec-0.4.7 \
                atty-0.2.9 \
                bitflags-1.0.1 \
                byteorder-1.2.2 \
                [...]
```

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

distinfo необходимо регенерировать, чтобы включить все дистрибутивные файлы крейтов:

```
% make makesum
=> rust/crates/aho-corasick-0.6.4.tar.gz doesn't seem to exist in
/usr/ports/distfiles/.
=> Attempting to fetch https://crates.io/api/v1/crates/aho-
corasick/0.6.4/download?dummy=/rust/crates/aho-corasick-0.6.4.tar.gz
```

```

rust/crates/aho-corasick-0.6.4.tar.gz      100% of  24 kB 6139 kBps 00m00s
=> rust/crates/ansi_term-0.11.0.tar.gz doesn't seem to exist in
/usr/ports/distfiles/.
=> Attempting to fetch
https://crates.io/api/v1/crates/ansi_term/0.11.0/download?dummy=/rust/crates/ansi_
term-0.11.0.tar.gz
rust/crates/ansi_term-0.11.0.tar.gz      100% of  16 kB  21 MBps 00m00s
=> rust/crates/arrayvec-0.4.7.tar.gz doesn't seem to exist in
/usr/ports/distfiles/.
=> Attempting to fetch
https://crates.io/api/v1/crates/arrayvec/0.4.7/download?dummy=/rust/crates/arrayve
c-0.4.7.tar.gz
rust/crates/arrayvec-0.4.7.tar.gz      100% of  22 kB 3237 kBps 00m00s
=> rust/crates/atty-0.2.9.tar.gz doesn't seem to exist in /usr/ports/distfiles/.
=> Attempting to fetch https://crates.io/api/v1/crates/atty/0.2.9/download?dummy
=/rust/crates/atty-0.2.9.tar.gz
rust/crates/atty-0.2.9.tar.gz          100% of 5898 B  81 MBps 00m00s
=> rust/crates/bitflags-1.0.1.tar.gz doesn't seem to exist in
/usr/ports/distfiles/.
[...]

```

Порт теперь готов к тестовой сборке и дальнейшим настройкам, таким как создание plist, написание описания, добавление информации о лицензии, опций и т.д., как обычно.

Если вы не тестируете свой порт в чистой среде, например, с использованием `poudriere`, не забудьте выполнить `make clean` перед любым тестированием.

#### Пример 58. Включение дополнительных возможностей приложений

Некоторые приложения определяют дополнительные возможности в своем Cargo.toml. Их можно включить при компиляции, установив `CARGO_FEATURES` в порте.

Здесь мы включаем функции `json` и `yaml` в Tokel:

```
CARGO_FEATURES= json yaml
```

#### Пример 59. Кодирование характеристик приложений как параметров порта

Пример раздела `[features]` в Cargo.toml может выглядеть так:

```

[features]
pulseaudio_backend = ["librespot-playback/pulseaudio-backend"]
portaudio_backend = ["librespot-playback/portaudio-backend"]
default = ["pulseaudio_backend"]

```

`pulseaudio_backend` — это функция по умолчанию. Она всегда включена, если мы явно не отключим функции по умолчанию, добавив `--no-default-features` в `CARGO_FEATURES`. Здесь мы превращаем функции `portaudio_backend` и `pulseaudio_backend` в опции порта:

```
CARGO_FEATURES= --no-default-features

OPTIONS_DEFINE= PORTAUDIO PULSEAUDIO

PORTAUDIO_VARS=     CARGO_FEATURES+=portaudio_backend
PULSEAUDIO_VARS=   CARGO_FEATURES+=pulseaudio_backend
```

### Пример 60. Перечисление лицензий крейтов

Крейты имеют собственные лицензии. Важно знать, какие они, при добавлении блока `LICENSE` в порт (см. [Лицензии](#)). Вспомогательная цель `cargo-crates-licenses` попытается перечислить все лицензии всех ящиков, определённых в `CARGO_CRATES`.

```
% make cargo-crates-licenses
aho-corasick-0.6.4  Unlicense/MIT
ansi_term-0.11.0   MIT
arrayvec-0.4.7     MIT/Apache-2.0
atty-0.2.9         MIT
bitflags-1.0.1     MIT/Apache-2.0
byteorder-1.2.2    Unlicense/MIT
[...]
```



Названия лицензий, которые выводит `make cargo-crates-licenses`, являются SPDX 2.1-совместимыми лицензионными выражениями, которые не совпадают с названиями лицензий, определёнными в фреймворке портов. Их необходимо перевести в названия из [Списка предопределённых лицензий](#).

### 6.6.7. Использование `meson`

Для портов, использующих `Meson`, определите `USES=meson`.

Таблица 13. Переменные для портов, использующих `meson`

Переменная	Описание
<code>MESON_ARGS</code>	Порт-специфичные флаги <code>Meson</code> , передаваемые в бинарный файл <code>meson</code> .
<code>MESON_BUILD_DIR</code>	Путь к директории сборки относительно <code>WRKSRC</code> . По умолчанию — <code>_build</code> .

Пример 61. Пример `USES=meson`

Этот фрагмент демонстрирует использование Meson для порта.

```
USES=      meson
MESON_ARGS= -Dfoo=enabled
```

### 6.6.8. Создание приложений на Go

Для портов, использующих Go, определите `USES=go`. Обратитесь к `go` для получения списка переменных, которые можно задать для управления процессом сборки.

Пример 62. Создание порта для приложения на основе модулей Go

В большинстве случаев достаточно установить переменную `GO_MODULE` в значение, указанное директивой `module` в `go.mod`:

```
PORTNAME=      hey
DISTVERSIONPREFIX= v
DISTVERSION=    0.1.4
CATEGORIES=     benchmarks

MAINTAINER=    dmgk@FreeBSD.org
COMMENT=       Tiny program that sends some load to a web application
WWW=           https://github.com/rakyll/hey/

LICENSE=       APACHE20
LICENSE_FILE=  ${WRKSRC}/LICENSE

USES=          go:modules
GO_MODULE=     github.com/rakyll/hey

PLIST_FILES=   bin/hey

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

Если «простой» способ не подходит или требуется больший контроль над зависимостями, полный процесс переноса описан ниже.

Создание порта на основе Go — это процесс из пяти этапов. Сначала необходимо предоставить шаблон портов, который загружает дистрибутивный файл приложения:

```
PORTNAME=      ghq
DISTVERSIONPREFIX= v
DISTVERSION=    0.12.5
CATEGORIES=     devel
```

```
MAINTAINER= tobik@FreeBSD.org
COMMENT=    Remote repository management made easy
WWW=       https://github.com/x-motemen/ghq/

USES=      go:modules
USE_GITHUB= yes
GH_ACCOUNT= motemen

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

Сгенерировать первоначальный distinfo:

```
% make makesum
==> License MIT accepted by the user
=> motemen-ghq-v0.12.5_GH0.tar.gz doesn't seem to exist in /usr/ports/distfiles/.
=> Attempting to fetch
https://codeload.github.com/motemen/ghq/tar.gz/v0.12.5?dummy=/motemen-ghq-
v0.12.5_GH0.tar.gz
fetch: https://codeload.github.com/motemen/ghq/tar.gz/v0.12.5?dummy=/motemen-ghq-
v0.12.5_GH0.tar.gz: size of remote file is not known
motemen-ghq-v0.12.5_GH0.tar.gz                32 kB  177 kBps   00s
```

Теперь файл дистрибутива готов к использованию, и мы можем извлечь необходимые зависимости модуля Go. Этот шаг требует наличия установленного пакета [ports-mgmt/modules2tuple](#):

```
% make gomod-vendor
[...]
GH_TUPLE=  \

Songmu:gitconfig:v0.0.2:songmu_gitconfig/vendor/github.com/Songmu/gitconfig \
daviddengcn:go-
colortext:186a3d44e920:daviddengcn_go_colortext/vendor/github.com/daviddengcn/go-
colortext \
go-yaml:yaml:v2.2.2:go_yaml_yaml/vendor/gopkg.in/yaml.v2 \
golang:net:3ec191127204:golang_net/vendor/golang.org/x/net \
golang:sync:112230192c58:golang_sync/vendor/golang.org/x/sync \
golang:xerrors:3ee3066db522:golang_xerrors/vendor/golang.org/x/xerrors \
motemen:go-
colorine:45d19169413a:motemen_go_colorine/vendor/github.com/motemen/go-colorine \
urfave:cli:v1.20.0:urfave_cli/vendor/github.com/urfave/cli
```

Вывод этой команды необходимо вставить напрямую в Makefile:

```
PORTNAME=  ghq
DISTVERSIONPREFIX=  v
DISTVERSION=  0.12.5
```

```

CATEGORIES= devel

MAINTAINER= tobik@FreeBSD.org
COMMENT= Remote repository management made easy
WWW= https://github.com/x-motemen/ghq/

USES= go:modules
USE_GITHUB= yes
GH_ACCOUNT= motemen
GH_TUPLE=
Songmu:gitconfig:v0.0.2:songmu_gitconfig/vendor/github.com/Songmu/gitconfig \
    dauidengcn:go-
colortext:186a3d44e920:davidengcn_go_colortext/vendor/github.com/davidengcn/go-
colortext \
    go-yaml:yaml:v2.2.2:go_yaml_yaml/vendor/gopkg.in/yaml.v2 \
    golang:net:3ec191127204:golang_net/vendor/golang.org/x/net \
    golang:sync:112230192c58:golang_sync/vendor/golang.org/x/sync \
    golang:xerrors:3ee3066db522:golang_xerrors/vendor/golang.org/x/xerrors \
    motemen:go-
colorine:45d19169413a:motemen_go_colorine/vendor/github.com/motemen/go-colorine \
    urfave:cli:v1.20.0:urfave_cli/vendor/github.com/urfave/cli

.include <bsd.port.mk>

```

distinfo необходимо обновить, чтобы включить все дистрибутивные файлы:

```

% make makesum
=> Songmu-gitconfig-v0.0.2_GH0.tar.gz doesn't seem to exist in
/usr/ports/distfiles/.
=> Attempting to fetch
https://codeload.github.com/Songmu/gitconfig/tar.gz/v0.0.2?dummy=/Songmu-
gitconfig-v0.0.2_GH0.tar.gz
fetch: https://codeload.github.com/Songmu/gitconfig/tar.gz/v0.0.2?dummy=/Songmu-
gitconfig-v0.0.2_GH0.tar.gz: size of remote file is not known
Songmu-gitconfig-v0.0.2_GH0.tar.gz          5662 B  936 kBps   00s
=> dauidengcn-go-colortext-186a3d44e920_GH0.tar.gz doesn't seem to exist in
/usr/ports/distfiles/.
=> Attempting to fetch https://codeload.github.com/davidengcn/go-
colortext/tar.gz/186a3d44e920?dummy=/davidengcn-go-colortext-
186a3d44e920_GH0.tar.gz
fetch: https://codeload.github.com/davidengcn/go-
colortext/tar.gz/186a3d44e920?dummy=/davidengcn-go-colortext-
186a3d44e920_GH0.tar.gz: size of remote file is not known
davidengcn-go-colortext-186a3d44e920_GH0.tar.  4534 B 1098 kBps   00s
[...]

```

Порт теперь готов к тестовой сборке и дальнейшим настройкам, таким как создание plist, написание описания, добавление информации о лицензии, опций и т.д., как обычно.

Если вы не тестируете свой порт в чистой среде, например, с использованием `poudriere`, не забудьте выполнить `make clean` перед любым тестированием.

*Пример 63. Установка имени выходного бинарного файла или пути установки*

Некоторые порты требуют установки результирующего бинарного файла под другим именем или в путь, отличный от стандартного `${PREFIX}/bin`. Это можно сделать с помощью синтаксиса кортежа `GO_TARGET`, например:

```
GO_TARGET= ./cmd/ipfs:ipfs-go
```

установит бинарный файл `ipfs` как `${PREFIX}/bin/ipfs-go` и

```
GO_TARGET= ./dnscrypt-proxy:${PREFIX}/sbin/dnscrypt-proxy
```

установит `dnscrypt-proxy` в `${PREFIX}/sbin`.

### 6.6.9. Построение приложений на Haskell с помощью `cabal`

Для портов, использующих `Cabal`, система сборки определяет `USES=cabal`. Обратитесь к `cabal` для получения списка переменных, которые можно задать для управления процессом сборки.

*Пример 64. Создание порта для приложения Haskell с `Hackage`*

При подготовке порта Haskell `Cabal` требуются программы `devel/hs-cabal-install` и `ports-mgmt/hs-cabal2tuple`, поэтому убедитесь, что они установлены заранее. Сначала необходимо определить общие переменные портов, которые позволяют `cabal-install` загрузить файл дистрибутива пакета:

```
PORTNAME=  ShellCheck
DISTVERSION=  0.6.0
CATEGORIES=  devel

MAINTAINER=  haskell@FreeBSD.org
COMMENT=     Shell script analysis tool
WWW=        https://www.shellcheck.net/

USES=       cabal

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

Этот минимальный `Makefile` загружает файл дистрибутива с помощью вспомогательной цели `cabal-extract`:

```
% make cabal-extract
[...]
Downloading the latest package list from hackage.haskell.org
cabal get ShellCheck-0.6.0
Downloading ShellCheck-0.6.0
Downloaded ShellCheck-0.6.0
Unpacking to ShellCheck-0.6.0/
```

Теперь, когда у нас есть файл описания пакета ShellCheck.cabal в `${WRKSRCS}`, мы можем использовать `cabal-configure` для создания плана сборки:

```
% make cabal-configure
[...]
Resolving dependencies...
Build profile: -w ghc-8.10.7 -01
In order, the following would be built (use -v for more details):
- Diff-0.4.1 (lib) (requires download & build)
- OneTuple-0.3.1 (lib) (requires download & build)
[...]
```

После завершения можно сгенерировать список необходимых зависимостей:

```
% make make-use-cabal
USE_CABAL= QuickCheck-2.12.6.1 \
          hashable-1.3.0.0 \
          integer-logarithms-1.0.3 \
[...]
```

Пакеты Haskell могут содержать ревизии, как и порты FreeBSD. Ревизии могут затрагивать только файлы .cabal. Обратите внимание на дополнительные номера версий после символа `_`. Замените старый список `USE_CABAL` на вновь сгенерированный.

Наконец, файл `distinfo` необходимо регенерировать, чтобы он содержал все файлы дистрибутива:

```
% make makesum
=> ShellCheck-0.6.0.tar.gz doesn't seem to exist in
/usr/local/poudriere/ports/git/distfiles/cabal.
=> Attempting to fetch https://hackage.haskell.org/package/ShellCheck-
0.6.0/ShellCheck-0.6.0.tar.gz
ShellCheck-0.6.0.tar.gz                136 kB  642 kBps   00s
=> QuickCheck-2.12.6.1/QuickCheck-2.12.6.1.tar.gz doesn't seem to exist in
/usr/local/poudriere/ports/git/distfiles/cabal.
=> Attempting to fetch https://hackage.haskell.org/package/QuickCheck-
2.12.6.1/QuickCheck-2.12.6.1.tar.gz
QuickCheck-2.12.6.1/QuickCheck-2.12.6.1.tar.gz  65 kB  361 kBps   00s
```

[...]

Порт теперь готов к тестовой сборке и дальнейшим настройкам, таким как создание plist, написание описания, добавление информации о лицензии, опций и т.д., как обычно.

Если вы не тестируете свой порт в чистой среде, например, с использованием `poudriere`, не забудьте выполнить `make clean` перед любым тестированием.

Некоторые порты Haskell устанавливают различные файлы данных в `share/${PORTNAME}`. В таких случаях требуется особая обработка на стороне порта. Порт должен определить параметр `CABAL_WRAPPER_SCRIPTS`, перечислив каждый исполняемый файл, который будет использовать файлы данных. Более того, в редких случаях портируемая программа использует файлы данных других пакетов Haskell, и тогда на помощь приходит `FOO_DATADIR_VARS`.

*Пример 65. Обработка файлов данных в порте Haskell*

`devel/hs-profiteur` — это приложение на Haskell, которое генерирует одностраничный HTML с некоторым содержимым.

```
PORTNAME=  profiteur

[...]

USES=      cabal

USE_CABAL= OneTuple-0.3.1_2 \
            QuickCheck-2.14.2 \
            [...]

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

Он устанавливает HTML-шаблоны в `share/profiteur`, поэтому необходимо добавить параметр `CABAL_WRAPPER_SCRIPTS`:

```
[...]

USE_CABAL= OneTuple-0.3.1_2 \
            QuickCheck-2.14.2 \
            [...]

CABAL_WRAPPER_SCRIPTS=  ${CABAL_EXECUTABLES}

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

Программа также пытается получить доступ к файлу `jquery.js`, который является частью пакета Haskell `js-jquery-3.3.1`. Чтобы этот файл был найден, необходимо, чтобы скрипт-обёртка искал файлы данных `js-jquery` также в `share/profiteur`. Для этого используется `profiteur_DATADIR_VARS`:

```
[...]  
  
CABAL_WRAPPER_SCRIPTS=      ${CABAL_EXECUTABLES}  
profiteur_DATADIR_VARS=     js-jquery  
  
.include <bsd.port.mk>
```

Теперь порт установит непосредственно бинарный файл в `libexec/cabal/profiteur`, а скрипт — в `bin/profiteur`.

Не существует простого способа определить подходящее значение для параметра `FOO_DATADIR_VARS`, кроме как запустить программу и проверить, что всё работает. К счастью, необходимость использовать `FOO_DATADIR_VARS` возникает очень редко.

Еще один крайний случай при переносе сложных программ на Haskell — наличие зависимостей от систем контроля версий (VCS) в файле `cabal.project`.

*Пример 66. Портинг приложений Haskell с зависимостями от VCS*

`net-p2p/cardano-node` — это чрезвычайно сложное программное обеспечение. В его `cabal.project` содержится множество блоков, подобных этому:

```
[...]  
source-repository-package  
  type: git  
  location: https://github.com/input-output-hk/cardano-crypto  
  tag: f73079303f663e028288f9f4a9e08bccca39a923e  
[...]
```

Зависимости типа `source-repository-package` автоматически подтягиваются `cabal` в процессе сборки. К сожалению, это приводит к использованию сети после этапа `fetch`, что запрещено в рамках системы портов. Эти исходники необходимо указать в Makefile порта. Вспомогательная цель `make-use-cabal` может упростить работу с пакетами, размещёнными на GitHub. Запуск этой цели после стандартных `cabal-extract` и `cabal-configre` позволит получить не только параметр `USE_CABAL`, но и `GH_TUPLE`:

```
% make make-use-cabal  
USE_CABAL= Diff-0.4.1 \  
           Glob-0.10.2_3 \  
           HUnit-1.6.2.0 \  
           [...]
```

```
GH_TUPLE=      input-output-hk:cardano-
base:0f3a867493059e650cda69e20a5cbf1ace289a57:cardano_base/dist-
newstyle/src/cardano-b_-c8db9876882556ed \
      input-output-hk:cardano-
crypto:f73079303f663e028288f9f4a9e08bcca39a923e:cardano_crypto/dist-
newstyle/src/cardano-c_-253fd88117badd8f \
      [...]
```

Может быть полезно отделить элементы `GH_TUPLE`, поступающие из `make-use-cabal`, от остальных, чтобы упростить обновление порта:

```
GH_TUPLE=      input-output-hk:cardano-
base:0f3a867493059e650cda69e20a5cbf1ace289a57:cardano_base/dist-
newstyle/src/cardano-b_-c8db9876882556ed \
      input-output-hk:cardano-
crypto:f73079303f663e028288f9f4a9e08bcca39a923e:cardano_crypto/dist-
newstyle/src/cardano-c_-253fd88117badd8f \
      [...]
```

```
GH_TUPLE+= bitcoin-core:secp256k1:ac83be33d0956faf6b7f61a60ab524ef7d6a473a:secp
```

Версия Haskell-портов с зависимостями от систем контроля версий временно требует следующего обходного решения:

```
BINARY_ALIAS= git=true
```

## 6.7. Использование GNU Autotools

Если порту требуется какое-либо программное обеспечение GNU Autotools, добавьте `USES=autoreconf`. Подробнее см. в `autoreconf`.

## 6.8. Использование GNU `gettext`

### 6.8.1. Простые варианты использования

Если порт требует `gettext`, установите `USES=gettext`, и порт унаследует зависимость от `libintl.so` из пакета `devel/gettext`. Другие значения для использования `gettext` перечислены в `USES=gettext`.

Довольно распространённый случай — порт, использующий `gettext` и `configure`. Обычно GNU `configure` должен автоматически находить `gettext`.

```
USES= gettext
```

```
GNU_CONFIGURE= yes
```

Если он не работает, можно указать расположение `gettext` через `CPPFLAGS` и `LDFLAGS`, используя `localbase` следующим образом:

```
USES= gettext localbase:ldflags
GNU_CONFIGURE= yes
```

## 6.8.2. Оптимальное использование

Некоторые программные продукты позволяют отключать NLS, к примеру, передавая параметр `--disable-nls` сценарию `configure`. В этом случае ваш порт должен использовать `gettext`, в зависимости от значения `NLS`. Для портов небольшой или средней сложности вы можете полагаться на следующую идиому:

```
GNU_CONFIGURE=      yes

OPTIONS_DEFINE=     NLS
OPTIONS_SUB=        yes

NLS_USES=           gettext
NLS_CONFIGURE_ENABLE= nls

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

Или используя старый способ с опциями:

```
GNU_CONFIGURE=      yes

OPTIONS_DEFINE=     NLS

.include <bsd.port.options.mk>

.if ${PORT_OPTIONS:MNLS}
USES+=              gettext
PLIST_SUB+=         NLS=""
.else
CONFIGURE_ARGS+=   --disable-nls
PLIST_SUB+=         NLS="@comment "
.endif

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

Следующий пункт в списке задач — организовать условное включение файлов каталогов сообщений в упаковочный список. Часть, связанная с Makefile, уже предусмотрена идиомой. Это объясняется в разделе [расширенные практики работы с pkg-plist](#). Вкратце,

каждое вхождение `%%NLS%%` в `pkg-plist` будет заменено на `"`@comment "`, если `NLS` отключен, или на пустую строку, если `NLS` включен. Следовательно, строки с префиксом ``%%NLS%%` станут обычными комментариями в итоговом упаковочном списке, если `NLS` выключен; в противном случае префикс будет просто удален. Затем вставьте `%%NLS%%` перед каждым путем к файлу каталога сообщений в `pkg-plist`. Например:

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

В особо сложных случаях вам понадобится использовать более продвинутые техники, чем данный рецепт, такие как [динамические списки упаковки](#).

### 6.8.3. Управление каталогами сообщений

Существует момент, который следует учитывать при установке файлов каталогов сообщений. Целевые каталоги для размещения, расположенные под `LOCALBASE/shared/locale`, редко когда должны создаваться и удаляться портом. Для наиболее популярных языков имеются собственные каталоги, перечисленные в `PORTSDIR/Templates/BSD.local.dist`. Каталоги для множества других языков управляются с помощью порта [devel/gettext](#). Обратите внимание на его `pkg-plist` и посмотрите, куда данный порт собирается установить файлы каталогов сообщений для единственного в своем роде языка.

## 6.9. Использование Perl

Если `MASTER_SITES` установлена в значение `CPAN`, то правильная поддиректория выбирается автоматически. Если подкаталог по умолчанию указан неверно, можно использовать `CPAN/Module` для его изменения. Также можно установить `MASTER_SITES` в старое значение `MASTER_SITE_PERL_CPAN`, тогда предпочтительным значением `MASTER_SITE_SUBDIR` будет имя иерархии выше уровнем. Например, рекомендуемое значение для `p5-Module-Name - Module`. Иерархию верхнего уровня можно посмотреть на [cpan.org](#). Это гарантирует работоспособность порта при смене автора модуля.

Исключением этого правила является отсутствие соответствующего каталога или файла с дистрибутивом в этом каталоге. В качестве `MASTER_SITE_SUBDIR` в этом случае разрешается использовать `id` автора. Можно использовать макрос `CPAN:AUTHOR`, который будет преобразован в хешированный каталог автора. Например, `CPAN:AUTHOR` преобразуется в `authors/id/A/AU/AUTHOR`.

Когда порту требуется поддержка Perl, он должен установить `USES=perl5` с опциональным `USE_PERL5`, как описано в [описание USES для perl5](#).

*Таблица 14. Переменные (только для чтения) для портов, использующих Perl*

Переменные (только для чтения)	Значение
PERL	Полный путь к интерпретатору Perl 5, будь то системный или установленный из порта, но без номера версии. Используйте это, когда программному обеспечению требуется путь к интерпретатору Perl. Для замены строк “`#!” в скриптах используйте `shebangfix.
PERL_VERSION	Полная версия Perl установлена (например, 5.8.9).
PERL_LEVEL	Установленная версия Perl в виде целого числа формата MNNPP (например, 500809).
PERL_ARCH	Где Perl хранит архитектурно-зависимые библиотеки. По умолчанию: \${ARCH}-freebsd.
PERL_PORT	Имя порта Perl, который установлен (например, perl5).
SITE_PERL	Имя каталога, в котором размещаются специфичные для сайта пакеты Perl. Это значение добавляется в PLIST_SUB.



Порты Perl-модулей, у которых нет официального сайта, должны ссылаться на [cpan.org](http://cpan.org) в строке WWW файла Makefile. Предпочтительный формат URL: <https://search.cpan.org/dist/Module-Name/> (включая завершающий слеш).



Не используйте \${SITE\_PERL} в объявлениях зависимостей. Это предполагает, что был включён файл perl5.mk, что не всегда верно. Порты, зависящие от этого порта, будут иметь некорректные зависимости, если файлы этого порта будут перемещены во время обновления. Правильный способ объявления зависимостей модулей Perl показан в примере ниже.

#### Пример 67. Пример зависимости Perl

```
p5-I0-Tee=>0.64:devel/p5-I0-Tee
```

Для портов Perl, которые устанавливают страницы руководства, макросы PERL5\_MAN3 и PERL5\_MAN1 могут использоваться внутри pkg-plist. Например,

```
lib/perl5/5.14/man/man1/event.1.gz
lib/perl5/5.14/man/man3/AnyEvent::I3.3.gz
```

может быть заменено на

```
%%PERL5_MAN1%/event.1.gz
```



Для других разделов (x в 2 и 4–9) макросы `PERL5_MAN_x_` отсутствуют, так как они устанавливаются в обычные каталоги.

*Пример 68. Порт, требующий Perl только для сборки*

Поскольку значение `USE_PERL5` по умолчанию включает `build` и `run`, установите его в:

```
USES=      perl5
USE_PERL5=  build
```

*Пример 69. Порт, который также требует Perl для исправления*

Время от времени использования `sed(1)` для исправлений недостаточно. Когда использование `perl(1)` проще, примените:

```
USES=      perl5
USE_PERL5=  patch build run
```

*Пример 70. Модуль Perl, для сборки которого требуется `ExtUtils::MakeMaker`*

Большинство модулей Perl поставляются с конфигурационным скриптом `Makefile.PL`. В этом случае установите:

```
USES=      perl5
USE_PERL5=  configure
```

*Пример 71. Модуль Perl, для сборки которого требуется `Module::Build`*

Когда модуль Perl поставляется с конфигурационным скриптом `Build.PL`, он может требовать `Module::Build`, и в этом случае установите

```
USES=      perl5
USE_PERL5=  modbuild
```

Если вместо этого требуется `Module::Build::Tiny`, установите

```
USES=      perl5
USE_PERL5=  modbuildtiny
```

## 6.10. Использование X11

### 6.10.1. Компоненты X.Org

Реализация X11, доступная в Коллекции портов, — это X.Org. Если приложение зависит от компонентов X, добавьте `USES= xorg` и укажите необходимые компоненты в `USE_XORG`. Полный список можно найти в `xorg`.

Проект Mesa — это инициатива по предоставлению свободной реализации OpenGL. Чтобы указать зависимость от различных компонентов этого проекта, используйте `USES=gl` и `USE_GL`. См. `gl` для полного списка доступных компонентов. Для обратной совместимости значение `yes` соответствует `glu`.

Пример 72. Пример `USE_XORG`

```
USES=      gl xorg
USE_GL=    glu
USE_XORG=  xrender xft xkbfile xt xaw
```

Таблица 15. Переменные для портов, использующих X

<code>USES= imake</code>	Порт использует <code>imake</code> .
<code>XMKMF</code>	Установить путь к <code>xmkmf</code> , если он отсутствует в <code>PATH</code> . По умолчанию: <code>xmkmf -a</code> .

Пример 73. Использование переменных, связанных с X11

```
# Use some X11 libraries
USES=      xorg
USE_XORG=  x11 xpm
```

### 6.10.2. Порты, требующие Motif

Если порт требует библиотеку Motif, определите `USES= motif` в Makefile. Реализация Motif по умолчанию — это `x11-toolkits/open-motif`. Пользователи могут выбрать `x11-toolkits/lesstif` вместо этого, установив `WANT_LESSTIF` в своём `make.conf`. Аналогично `x11-toolkits/open-motif-devel` можно выбрать, установив `WANT_OPEN_MOTIF_DEVEL` в `make.conf`.

`MOTIFLIB` будет установлен в файле `motif.mk` для ссылки на соответствующую библиотеку Motif. Пожалуйста, исправьте исходный код порта, чтобы использовать `${MOTIFLIB}` везде, где библиотека Motif упоминается в оригинальном Makefile или Imakefile.

Есть два распространённых случая:

- Если порт ссылается на библиотеку Motif как `-lXm` в своём Makefile или Imakefile,

замените это на `${MOTIFLIB}`.

- Если порт использует `XmClientLibs` в своем `Imakefile`, замените это на `${MOTIFLIB} ${XTOOLLIB} ${XLIB}`.

Обратите внимание, что `MOTIFLIB` (обычно) раскрывается в `-L/usr/local/lib -lXm -lXp` или `/usr/local/lib/libXm.a`, поэтому нет необходимости добавлять `-l` или `-l` перед этим.

### 6.10.3. Шрифты X11

Если порт устанавливает шрифты для X Window System, поместите их в `LOCALBASE/lib/X11/fonts/local`.

### 6.10.4. Получение поддельного `DISPLAY` с помощью `Xvfb`

Некоторые приложения требуют рабочего дисплея X11 для успешной компиляции. Это создаёт проблему для машин, работающих без монитора. При использовании этой переменной инфраструктура сборки запустит виртуальный X-сервер с буфером кадров. Рабочий `DISPLAY` затем передаётся в процесс сборки. См. `USES=display` для возможных аргументов.

```
USES= display
```

### 6.10.5. Desktop Entries (пункты рабочего стола)

Desktop entries ([стандарт freedesktop](#)) предоставляют способ автоматической настройки функций рабочего стола при установке новой программы без вмешательства пользователя. Например, вновь установленные программы автоматически появляются в меню приложений совместимых сред рабочего стола. Desktop entries появились в среде GNOME, но теперь стали стандартом и также работают с KDE и Xfce. Эта автоматизация приносит реальную пользу пользователю, поэтому Desktop entries рекомендуются для приложений, которые могут использоваться в среде рабочего стола.

#### 6.10.5.1. Использование predefined файлов `.desktop`

Порты, включающие predefined файлы `*.desktop`, должны добавлять эти файлы в `pkg-plist` и устанавливать их в директорию `$LOCALBASE/share/applications`. Для установки таких файлов полезен макрос `INSTALL_DATA`.

#### 6.10.5.2. Обновление базы данных рабочего стола

Если порт имеет запись `MimeType` в файле `portname.desktop`, базу данных рабочего стола необходимо обновить после установки и удаления. Для этого определите `USES= desktop-file-utils`.

#### 6.10.5.3. Создание Desktop Entries с помощью `DESKTOP_ENTRIES`

Desktop Entries могут быть легко созданы для приложений с использованием `DESKTOP_ENTRIES`. Файл с именем `name.desktop` будет создан, установлен и автоматически добавлен в `pkg-plist`.

Синтаксис следующий:

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

Список возможных категорий доступен на [сайте freedesktop](#). `StartupNotify` указывает, совместимо ли приложение с *уведомлениями о запуске*. Обычно это графический индикатор, например, часы, который появляется у указателя мыши, в меню или на панели, чтобы дать пользователю понять, что программа запускается. Программа, совместимая с уведомлениями о запуске, очищает индикатор после своего старта. Программы, не совместимые с уведомлениями о запуске, никогда не очищают индикатор (что может сбивать с толку и раздражать пользователя), и у них должен быть установлен параметр `StartupNotify` в значение `false`, чтобы индикатор вообще не отображался.

Пример:

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

`DESKTOP_ENTRIES` устанавливаются в директорию, указанную переменной `DESKTOPDIR`. По умолчанию `DESKTOPDIR` имеет значение `${PREFIX}/share/applications`

## 6.11. Использование GNOME

### 6.11.1. Введение

Эта глава объясняет фреймворк GNOME, используемый в портах. Фреймворк можно условно разделить на базовые компоненты, компоненты рабочего стола GNOME и несколько специальных макросов, упрощающих работу сопровождающих портов.

### 6.11.2. Использование `USE_GNOME`

Добавление этой переменной в порт позволяет использовать макросы и компоненты, определённые в `bsd.gnome.mk`. Код в `bsd.gnome.mk` добавляет необходимые зависимости для сборки, выполнения или библиотеки, а также обработку специальных файлов. Приложения GNOME в FreeBSD используют инфраструктуру `USE_GNOME`. Включите все необходимые компоненты в виде списка, разделённого пробелами. Компоненты `USE_GNOME` разделены на следующие виртуальные списки: основные компоненты, компоненты GNOME 3 и устаревшие компоненты. Если порту требуются только библиотеки GTK3, это кратчайший способ определить это:

```
USE_GNOME= gtk30
```

`USE_GNOME` компоненты автоматически добавляют необходимые им зависимости. Подробный

список всех компонентов `USE_GNOME`, а также информацию о том, какие другие компоненты они подразумевают и их зависимости, можно найти в [Компоненты GNOME](#).

Вот пример Makefile для порта GNOME, в котором используются многие методы, описанные в этом документе. Пожалуйста, используйте его в качестве руководства при создании новых портов.

```
PORTNAME=  regexxer
DISTVERSION=  0.10
CATEGORIES=  devel textproc gnome
MASTER_SITES=  GNOME

MAINTAINER=  kwm@FreeBSD.org
COMMENT=     Interactive tool for performing search and replace operations
WWW=        http://regexxer.sourceforge.net/

USES=       gettext gmake localbase:ldflags pathfix pkgconfig tar:xz
GNU_CONFIGURE=  yes
USE_GNOME=  gnomeprefix intlhack gtksourceviewmm3

GLIB_SCHEMAS=  org.regexxer.gschema.xml

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



Макрос `USE_GNOME` без аргументов не добавляет никаких зависимостей к порту. `USE_GNOME` не может быть установлен после `bsd.port.pre.mk`.

### 6.11.3. Переменные

Этот раздел объясняет, какие макросы доступны и как они используются. Как, например, в приведённом выше примере. В [Компоненты GNOME](#) содержится более подробное объяснение. Для использования этих макросов необходимо установить `USE_GNOME`.

#### GLIB\_SCHEMAS

Список всех файлов схем glib, которые устанавливает порт. Макрос добавит файлы в plist порта и обработает регистрацию этих файлов при установке и удалении.

Файлы схем glib написаны в XML и имеют расширение `gschema.xml`. Они устанавливаются в директорию `share/glib-2.0/schemas/`. Эти файлы схем содержат все настройки конфигурации приложений со значениями по умолчанию. Фактическая база данных, используемая приложениями, создаётся с помощью `glib-compile-schema`, которая запускается макросом `GLIB_SCHEMAS`.

```
GLIB_SCHEMAS=foo.gschema.xml
```



Не добавляйте схемы glib в файл `pkg-plist`. Если они указаны в `pkg-plist`, они не будут зарегистрированы, и приложения могут работать

некорректно.

## GCONF\_SCHEMAS

Перечислите все файлы схем gconf. Макрос добавит файлы схем в plist порта и обеспечит их регистрацию при установке и удалении.

GConf — это база данных на основе XML, которую используют практически все приложения GNOME для хранения своих настроек. Эти файлы устанавливаются в директорию etc/gconf/schemas. Эта база данных определяется установленными файлами схем, которые используются для генерации ключевых файлов %gconf.xml. Для каждого файла схемы, устанавливаемого портом, должна быть запись в Makefile:

```
GCONF_SCHEMAS=my_app.schemas my_app2.schemas my_app3.schemas
```



Схемы Gconf перечислены в макросе `GCONF_SCHEMAS`, а не в файле pkg-plist. Если они указаны в pkg-plist, они не будут зарегистрированы, и приложения могут работать некорректно.

## INSTALLS\_OMF

Файлы Open Source Metadata Framework (OMF) часто используются приложениями GNOME 2. Эти файлы содержат информацию о файлах справки приложений и требуют специальной обработки с помощью ScrollKeeper/rarian. Для правильной регистрации файлов OMF при установке приложений GNOME из пакетов убедитесь, что файлы `omf` указаны в `pkg-plist` и что в Makefile порта определено `INSTALLS_OMF`:

```
INSTALLS_OMF=yes
```

При установке `bsd.gnome.mk` автоматически сканирует `pkg-plist` и добавляет соответствующие директивы `@exec` и `@unexec` для каждого файла `.omf`, который необходимо отслеживать в базе данных регистрации OMF.

## 6.12. Компоненты GNOME

Для получения дополнительной помощи с портом GNOME, ознакомьтесь с некоторыми из [существующих портов](#) в качестве примеров. На странице [FreeBSD GNOME](#) указана контактная информация, если требуется дополнительная помощь. Компоненты разделены на используемые в настоящее время компоненты GNOME и устаревшие компоненты. Если компонент поддерживает аргументы, они перечислены в скобках в описании. Первый аргумент является значением по умолчанию. "Both" указывается, если компонент по умолчанию добавляется как в зависимости для сборки, так и для выполнения.

*Таблица 16. Компоненты GNOME*

Компонент	Связанная программа	Описание
atk	accessibility/atk	Инструментарий доступности (АТК)
atkmm	accessibility/atkmm	С++ интерфейс для atk
cairo	graphics/cairo	Векторная графическая библиотека с поддержкой вывода на различные устройства
caiomm	graphics/caiomm	С++ интерфейс для cairo
dconf	devel/dconf	Система базы данных конфигурации (both, build, run)
evolutiondataserver3	databases/evolution-data-server	Бэкенды данных для интегрированного почтового клиента/PIM Evolution
gdkpixbuf2	graphics/gdk-pixbuf2	Графическая библиотека для GTK+
glib20	devel/glib20	Основная библиотека GNOME <b>glib20</b>
glibmm	devel/glibmm	С++ интерфейс для glib20
gnomecontrolcenter3	sysutils/gnome-control-center	Центр управления GNOME 3
gnomedesktop3	x11/gnome-desktop	Библиотека пользовательского интерфейса рабочего стола GNOME 3
gsound	audio/gsound	Библиотека GObject для воспроизведения системных звуков (both, build, run)
gtk-update-icon-cache	graphics/gtk-update-icon-cache	Утилита <b>gtk-update-icon-cache</b> из набора инструментов <b>Gtk+</b>
gtk20	x11-toolkits/gtk20	Набор инструментов Gtk+ 2
gtk30	x11-toolkits/gtk30	Набор инструментов Gtk+ 3
gtkmm20	x11-toolkits/gtkmm20	С++ интерфейс 2.0 для инструментария gtk20
gtkmm24	x11-toolkits/gtkmm24	С++ интерфейс 2.4 для инструментария gtk20
gtkmm30	x11-toolkits/gtkmm30	С++ интерфейс 3.0 для набора инструментов gtk30

Компонент	Связанная программа	Описание
<a href="#">gtksourceview2</a>	x11-toolkits/gtksourceview2	Виджет, добавляющий подсветку синтаксиса в GtkTextView
<a href="#">gtksourceview3</a>	x11-toolkits/gtksourceview3	Текстовая виджет, добавляющая подсветку синтаксиса к виджету GtkTextView
<a href="#">gtksourceviewmm3</a>	x11-toolkits/gtksourceviewmm3	C++ интерфейс для библиотеки gtksourceview3
<a href="#">gvfs</a>	devel/gvfs	Виртуальная файловая система GNOME
<a href="#">intltool</a>	textproc/intltool	Инструмент для интернационализации (см. также intlhack)
<a href="#">introspection</a>	devel/gobject-introspection	Базовые привязки (биндинги) интроспекции и инструменты для генерации привязок интроспекции. В большинстве случаев достаточно <code>:build</code> , <code>:both:run</code> нужны только для приложений, использующих привязки интроспекции. (both, build, run)
<a href="#">libgda5</a>	databases/libgda5	Обеспечивает единообразный доступ к различным типам источников данных
<a href="#">libgda5-ui</a>	databases/libgda5-ui	Библиотека пользовательского интерфейса из библиотеки libgda5
<a href="#">libgdamm5</a>	databases/libgdamm5	привязки C++ для библиотеки libgda5
<a href="#">libgsf</a>	devel/libgsf	Расширяемая абстракция ввода-вывода для работы со структурированными форматами файлов
<a href="#">librsvg2</a>	graphics/librsvg2	Библиотека для разбора и отображения SVG-файлов векторной графики

Компонент	Связанная программа	Описание
libsigc++20	devel/libsigc++20	Фреймворк обратных вызовов для C++
libxml++26	textproc/libxml++26	C++ привязки для библиотеки libxml2
libxml2	textproc/libxml2	Библиотека парсера XML (both, build, run)
libxslt	textproc/libxslt	Библиотека XSLT (сборка, выполнение)
metacity	x11-wm/metacity	Менеджер окон из GNOME
nautilus3	x11-fm/nautilus	GNOME файловый менеджер
pango	x11-toolkits/pango	Открытый фреймворк для разметки и отображения интернационализованного текста
pangomm	x11-toolkits/pangomm	C++ интерфейс для библиотеки pango
py3gobject3	devel/py3-gobject3	Python 3, интерфейс GObject 3.0
pygobject3	devel/py-gobject3	Python 2, интерфейс GObject 3.0
vte3	x11-toolkits/vte3	Виджет терминала с улучшенной поддержкой доступности и интернационализации (I18N)

Таблица 17. Компоненты макросов GNOME

Компонент	Описание
gnomeprefix	Предоставляет <code>configure</code> некоторые стандартные расположения.
intlhack	То же, что и <code>intltool</code> , но с патчами для гарантии использования <code>share/locale/</code> . Используйте только в случае, когда одного <code>intltool</code> недостаточно.
referencehack	Этот макрос предназначен для помощи в разделении API или справочной документации на собственный порт.

Таблица 18. Компоненты GNOME Legacy

Компонент	Связанная программа	Описание
atspi	accessibility/at-spi	Интерфейс поставщика услуг вспомогательных технологий (AT-SPI)
esound	audio/esound	Пакет звука Enlightenment
gal2	x11-toolkits/gal2	Коллекция виджетов, взятых из GNOME 2 gnumeric
gconf2	devel/gconf2	Система базы данных конфигурации для GNOME 2
gconfmm26	devel/gconfmm26	C привязки C для gconf2
gdkpixbuf	graphics/gdk-pixbuf	Графическая библиотека для GTK+
glib12	devel/glib12	Библиотека ядра glib 1.2
gnomedocutils	textproc/gnome-doc-utils	Утилиты документации GNOME
gnomemimedata	misc/gnome-mime-data	База данных MIME и приложений для GNOME 2
gnomesharp20	x11-toolkits/gnome-sharp20	Интерфейсы GNOME 2 для среды выполнения .NET
gnomespeech	accessibility/gnome-speech	GNOME 2 API преобразования текста в речь
gnomevfs2	devel/gnome-vfs	Виртуальная файловая система GNOME 2
gtk12	x11-toolkits/gtk12	Набор инструментов Gtk+ 1.2
gtkhtml3	www/gtkhtml3	Облегченный движок для отображения/печати/редактирования HTML
gtkhtml4	www/gtkhtml4	Облегченный движок для отображения/печати/редактирования HTML
gtksharp20	x11-toolkits/gtk-sharp20	Интерфейсы GTK+ и GNOME 2 для среды выполнения .NET
gtksourceview	x11-toolkits/gtksourceview	Виджет, добавляющий подсветку синтаксиса в GtkTextView
libartgp12	graphics/libart_lgpl	Библиотека для высокопроизводительной 2D графики

<b>Компонент</b>	<b>Связанная программа</b>	<b>Описание</b>
libbonobo	devel/libbonobo	Система компонентов и составных документов для GNOME 2
libbonoboui	x11-toolkits/libbonoboui	Интерфейс для libbonobo в GNOME 2
libgda4	databases/libgda4	Обеспечивает единообразный доступ к различным типам источников данных
libglade2	devel/libglade2	Библиотека glade для GNOME 2
libgnome	x11/libgnome	Библиотеки для GNOME 2, GNU окружения рабочего стола
libgnomecanvas	graphics/libgnomecanvas	Графическая библиотека для GNOME 2
libgnomekbd	x11/libgnomekbd	Динамическая библиотека клавиатуры GNOME 2
libgnomeprint	print/libgnomeprint	Библиотека поддержки печати Gnome 2
libgnomeprintui	x11-toolkits/libgnomeprintui	Библиотека поддержки печати Gnome 2
libgnomeui	x11-toolkits/libgnomeui	Библиотеки для графического интерфейса GNOME 2, среды рабочего стола GNU
libgtkhtml	www/libgtkhtml	Облегченный движок для отображения/печати/редактирования HTML
libgtksourceviewmm	x11-toolkits/libgtksourceviewmm	C++ интерфейс GtkSourceView
libidl	devel/libIDL	Библиотека для создания деревьев файлов CORBA IDL
libsigc++12	devel/libsigc++12	Фреймворк обратных вызовов для C++
libwnck	x11-toolkits/libwnck	Библиотека, используемая для написания пейджеров и списков задач
libwnck3	x11-toolkits/libwnck3	Библиотека, используемая для написания пейджеров и списков задач

Компонент	Связанная программа	Описание
<code>orbit2</code>	<code>devel/ORBit2</code>	Высокопроизводительный CORBA ORB с поддержкой языка C
<code>pygnome2</code>	<code>x11-toolkits/py-gnome2</code>	Интерфейс Python для GNOME 2
<code>pygobject</code>	<code>devel/py-gobject</code>	Python 2, интерфейс GObject 2.0
<code>pygtk2</code>	<code>x11-toolkits/py-gtk2</code>	Набор интерфейсов Python для GTK+
<code>pygtksourceview</code>	<code>x11-toolkits/py-gtksourceview</code>	Интерфейс Python для GtkSourceView 2
<code>vte</code>	<code>x11-toolkits/vte</code>	Виджет терминала с улучшенной поддержкой доступности и интернационализации (I18N)

Таблица 19. Устаревшие компоненты: не использовать

Компонент	Описание
<code>rangox-compat</code>	<code>rangox-compat</code> устарел и был отделён от пакета <code>rango</code> .

## 6.13. Использование Qt



Для портов, которые являются частью самого Qt, см. `qt-dist`.

### 6.13.1. Порты, требующие Qt

Коллекция портов поддерживает Qt 5 и Qt 6 с помощью `USES+=qt:5` и `USES+=qt:6` соответственно. Установите `USE_QT` в список необходимых компонентов Qt (`libraries`, `tools`, `plugins` - библиотеки, инструменты, плагины).

Фреймворк Qt экспортирует ряд переменных, которые могут использоваться портами, некоторые из них перечислены ниже:

Таблица 20. Переменные, предоставляемые портам, использующим Qt

<code>QMAKE</code>	Полный путь к исполняемому файлу <code>qmake</code> .
<code>LRELEASE</code>	Полный путь к утилите <code>lrelease</code> .
<code>MOC</code>	Полный путь к <code>moc</code> .
<code>RCC</code>	Полный путь к <code>rcc</code> .
<code>UIC</code>	Полный путь к <code>uic</code> .

QT_INCDIR	Каталог включаемых файлов Qt.
QT_LIBDIR	Путь к библиотекам Qt.
QT_PLUGINDIR	Путь к плагинам Qt.

### 6.13.2. Выбор компонентов

Отдельные зависимости инструментов и библиотек Qt должны быть указаны в `USE_QT`. Каждый компонент может иметь суффикс `_build` или `_run`, указывающий, требуется ли компонент во время сборки или выполнения. Если суффикс отсутствует, компонент будет требоваться как во время сборки, так и во время выполнения. Обычно компоненты библиотек указываются без суффикса, компоненты инструментов чаще всего указываются с суффиксом `_build`, а компоненты плагинов — с суффиксом `_run`. Наиболее часто используемые компоненты перечислены ниже (все доступные компоненты перечислены в `_USE_QT_ALL`, которая формируется из `_USE_QT_COMMON` и `_USE_QT[56]_ONLY` в `/usr/ports/Mk/Uses/qt.mk`):

Таблица 21. Доступные компоненты библиотеки Qt

Имя	Описание
3d	Модуль Qt3D
5compat	Модуль совместимости Qt 5 для Qt 6
assistant	Браузер документации Qt 5
base	Модуль Qt 6 base
canvas3d	Модуль Qt canvas3d
charts	Модуль Qt 5 charts
concurrent	Модуль многопоточности Qt
connectivity	Модуль Qt для подключения (Bluetooth/NFC)
core	Ядро Qt, неграфический модуль
datavis3d	Модуль визуализации 3D данных Qt 5
dbus	Модуль межпроцессного взаимодействия Qt D-Bus
declarative	Декларативный фреймворк Qt для динамических пользовательских интерфейсов
designer	Интерфейсный конструктор Qt 5 для графического пользовательского интерфейса
diag	Инструмент для сбора диагностической информации о Qt и его окружении
doc	Документация Qt 5
examples	Исходный код примеров Qt 5
gamepad	Модуль Qt 5 Gamepad

<b>Имя</b>	<b>Описание</b>
<code>graphicaleffects</code>	Графические эффекты Qt Quick
<code>gui</code>	Модуль графического интерфейса Qt
<code>help</code>	Модуль интеграции справки Qt в режиме онлайн
<code>l10n</code>	Локализованные сообщения Qt
<code>languageserver</code>	Реализация протокола Language Server Protocol в Qt 6
<code>linguist</code>	Инструмент перевода Qt 5
<code>location</code>	Модуль Qt Location
<code>lottie</code>	Qt 6 QML API для отрисовки графики и анимаций
<code>multimedia</code>	Модуль поддержки аудио, видео, радио и камеры Qt
<code>network</code>	Сетевой модуль Qt
<code>networkauth</code>	Модуль сетевой аутентификации Qt
<code>opengl</code>	Модуль поддержки OpenGL, совместимый с Qt 5
<code>paths</code>	Клиент командной строки для QStandardPaths
<code>phonon4</code>	Мультимедийный фреймворк KDE
<code>pixeltool</code>	Увеличитель экрана Qt 5
<code>plugininfo</code>	Дампер метаданных плагинов Qt 5
<code>positioning</code>	Qt 6 API позиционирования из источников, таких как спутники, Wi-Fi или текстовые файлы.
<code>printsupport</code>	Модуль поддержки печати Qt
<code>qdbus</code>	Интерфейс командной строки Qt для D-Bus
<code>qdbusviewer</code>	Графический интерфейс Qt 5 для D-Bus
<code>qdoc</code>	Генератор документации Qt
<code>qdoc-data</code>	Файлы конфигурации QDoc
<code>qev</code>	Инструмент для интроспекции событий Qt QWidget
<code>qmake</code>	Генератор Makefile Qt
<code>quickcontrols</code>	Набор элементов управления для создания полных интерфейсов в Qt Quick

<b>Имя</b>	<b>Описание</b>
<code>quickcontrols2</code>	Набор элементов управления для создания полных интерфейсов в Qt Quick
<code>remoteobjects</code>	Модуль Qt 5 SXCML
<code>script</code>	Совместимый с Qt 4 модуль для написания сценариев
<code>scripttools</code>	Дополнительные компоненты Qt Script
<code>scxml</code>	Модуль Qt 5 SXCML
<code>sensors</code>	Модуль Qt sensors
<code>serialbus</code>	Функции Qt для доступа к промышленным шинным системам
<code>serialport</code>	Функции Qt для доступа к последовательным портам
<code>shadertools</code>	Инструменты Qt 6 для кроссплатформенного конвейера шейдеров Qt
<code>speech</code>	Доступность в Qt5
<code>sql</code>	Модуль интеграции с базой данных Qt SQL
<code>sql-ibase</code>	Плагин Qt баз данных InterBase/Firebird
<code>sql-mysql</code>	Плагин Qt базы данных MySQL
<code>sql-odbc</code>	Плагин Qt ODBC
<code>sql-pgsql</code>	Плагин Qt базы данных PostgreSQL
<code>sql-sqlite2</code>	Плагин Qt базы данных SQLite 2
<code>sql-sqlite3</code>	Плагин Qt базы данных SQLite 3
<code>sql-tds</code>	Плагин Qt для подключение к базам данных по протоколу TDS
<code>svg</code>	Модуль поддержки SVG в Qt
<code>testlib</code>	Модуль тестирования Qt
<code>tools</code>	Различные инструменты Qt 6
<code>translations</code>	Модуль перевода Qt 6
<code>uiplugin</code>	Интерфейс плагина пользовательского виджета Qt для Qt Designer
<code>uitools</code>	Модуль поддержки форм пользовательского интерфейса Qt Designer
<code>virtualkeyboard</code>	Модуль виртуальной клавиатуры Qt 5
<code>wayland</code>	Оболочка Qt 5 для Wayland
<code>webchannel</code>	Библиотека Qt 5 для интеграции C++/QML с клиентами на HTML/js

Имя	Описание
<code>webengine</code>	Библиотека Qt 5 для отображения веб-содержимого
<code>webkit</code>	QtWebKit с более современной кодовой базой WebKit
<code>websockets</code>	Реализация протокола WebSocket на Qt
<code>websockets-qml</code>	Реализация протокола WebSocket на Qt (привязки QML)
<code>webview</code>	Компонент Qt для отображения веб-содержимого
<code>widgets</code>	Модуль виджетов Qt C++
<code>x11extras</code>	Платформенно-специфичные возможности Qt для систем на основе X11
<code>xml</code>	Реализации SAX и DOM в Qt
<code>xmlpatterns</code>	Поддержка Qt для XPath, XQuery, XSLT и XML Schema

Чтобы определить библиотеки, от которых зависит приложение, выполните `ldd` для основного исполняемого файла после успешной компиляции.

Таблица 22. Доступные компоненты инструментов Qt

Имя	Описание
<code>buildtools</code>	инструменты сборки ( <code>moc</code> , <code>gcc</code> ), необходимые практически для любого приложения Qt.
<code>linguisttools</code>	инструменты локализации: <code>lrelease</code> , <code>lupdate</code>
<code>qmake</code>	Генератор Makefile/утилиты сборки

Таблица 23. Доступные компоненты плагинов Qt

Имя	Описание
<code>imageformats</code>	плагины для графических форматов TGA, TIFF и MNG

Пример 74. Выбор компонентов Qt 5

В этом примере портированное приложение использует библиотеку графического интерфейса Qt 5, основную библиотеку Qt 5, все инструменты генерации кода Qt 5 и генератор Makefile Qt 5. Поскольку библиотека `gui` подразумевает зависимость от основной библиотеки, `core` не нужно указывать. Инструменты генерации кода Qt 5 `moc`, `uic` и `gcc`, а также генератор Makefile `qmake` требуются только во время сборки, поэтому они указаны с суффиксом `_build`:

```
USES= qt:5
```

```
USE_QT= gui buildtools_build qmake_build
```

### 6.13.3. Использование qmake

Если приложение предоставляет файл проекта qmake (\*.pro), определите `USES= qmake` вместе с `USE_QT`. `USES= qmake` уже подразумевает зависимость сборки от qmake, поэтому компонент qmake может быть опущен в `USE_QT`. Подобно `CMake`, qmake поддерживает сборку вне исходного дерева, которую можно включить, указав аргумент `outsources` (см. [пример USES= qmake](#)). Также см. [Возможные аргументы для USES= qmake](#).

Таблица 24. Возможные аргументы для `USES= qmake`

Переменная	Описание
<code>no_configure</code>	Не добавлять цель <code>configure</code> . Это подразумевается при <code>HAS_CONFIGURE=yes</code> и <code>GNU_CONFIGURE=yes</code> . Это требуется, когда сборке нужна только настройка окружения из <code>USES= qmake</code> , но в остальном она запускает qmake самостоятельно.
<code>no_env</code>	Подавить модификацию окружения <code>configure</code> и <code>make</code> . Это требуется только когда qmake используется для настройки программного обеспечения и сборка не понимает окружение, установленное <code>USES= qmake</code> .
<code>norecursive</code>	Не передавать аргумент <code>-recursive</code> в qmake.
<code>outsources</code>	Выполнить сборку вне исходного кода.

Таблица 25. Переменные для портов, использующих qmake

Переменная	Описание
<code>QMAKE_ARGS</code>	Специфичные для порта флаги qmake, передаваемые в бинарный файл qmake.
<code>QMAKE_ENV</code>	Переменные окружения, которые должны быть установлены для бинарного файла qmake. По умолчанию используется <code>\${CONFIGURE_ENV}</code> .
<code>QMAKE_SOURCE_PATH</code>	Путь к файлам проекта qmake (.pro). По умолчанию используется <code>\${WRKSRC}</code> , если запрошена сборка вне исходного кода, в противном случае оставляется пустым.

При использовании `USES= qmake` применяются следующие настройки:

```
CONFIGURE_ARGS+= --with-qt-includes=${QT_INCDIR} \
```

```

--with-qt-libraries=${QT_LIBDIR} \
--with-extra-libs=${LOCALBASE}/lib \
--with-extra-includes=${LOCALBASE}/include

CONFIGURE_ENV+= QTDIR="${QT_PREFIX}" QMAKE="${QMAKE}" \
MOC="${MOC}" RCC="${RCC}" UIC="${UIC}" \
QMAKESPEC="${QMAKESPEC}"

PLIST_SUB+= QT_INCDIR=${QT_INCDIR_REL} \
QT_LIBDIR=${QT_LIBDIR_REL} \
QT_PLUGINDIR=${QT_PLUGINDIR_REL}

```

Некоторые скрипты `configure` не поддерживают указанные выше аргументы. Чтобы отключить изменение `CONFIGURE_ENV` и `CONFIGURE_ARGS`, установите `USES= qmake:no_env`.

*Пример 75. Пример `USES= qmake`*

Этот фрагмент демонстрирует использование `qmake` для порта Qt 5:

```

USES= qmake:outsources qt:5
USE_QT= buildtools_build

```

Приложения Qt часто разрабатываются как кроссплатформенные, и зачастую X11/Unix — не та платформа, на которой они создаются. Это, в свою очередь, приводит к определённым недоработкам, таким как:

- *Отсутствуют дополнительные пути для заголовочных файлов.* Многие приложения поддерживают значки в системном трее, но не учитывают пути для заголовочных файлов и/или библиотек в каталогах X11. Чтобы добавить каталоги в пути поиска заголовочных файлов и библиотек для `qmake` через командную строку, используйте:

```

QMAKE_ARGS+= INCLUDEPATH+=${LOCALBASE}/include \
LIBS+=-L${LOCALBASE}/lib

```

- *Некорректные пути установки.* Иногда данные, такие как иконки или файлы `.desktop`, по умолчанию устанавливаются в каталоги, которые не сканируются приложениями, совместимыми с XDG. Например, [editors/texmaker](#) — посмотрите на файл `patch-texmaker.pro` в директории `files` этого порта, чтобы увидеть шаблон исправления этой проблемы напрямую в проекте файла `qmake`.

## 6.14. Использование KDE

### 6.14.1. Определения переменных KDE

Если приложение зависит от KDE, установите `USES+=kde:5` и `USE_KDE` в список необходимых компонентов. Суффиксы `_build` и `_run` можно использовать для принудительного указания

типа зависимости компонентов (например, `baseapps_run`). Если суффикс не задан, будет использован тип зависимости по умолчанию. Чтобы принудительно задать оба типа, добавьте компонент дважды с обоими суффиксами (например, `ecm_build ecm_run`). Доступные компоненты перечислены ниже (актуальный список компонентов также приведён в `/usr/ports/Mk/Uses/kde.mk`):

Таблица 26. Доступные компоненты KDE

Имя	Описание
<code>activities</code>	Среда выполнения и библиотека KF5 для организации работы в отдельных активностях
<code>activities-stats</code>	KF5 статистика для активностей
<code>activitymanagerd</code>	Системный сервис для управления активностью пользователей, отслеживания шаблонов использования
<code>akonadi</code>	Хранилище данных для KDE-Pim
<code>akonadicalendar</code>	Интеграция Akonadi с Календарем
<code>akonadiconsole</code>	Консоль управления и отладки Akonadi
<code>akonadicontacts</code>	Библиотеки и демоны для реализации управления контактами в Akonadi
<code>akonadiimportwizard</code>	Импорт данных из других почтовых клиентов в KMail
<code>akonadimime</code>	Библиотеки и демоны для реализации базовой обработки электронной почты
<code>akonadinotes</code>	Библиотека KDE для доступа к хранилищам почты в формате MBox
<code>akonadisearch</code>	Библиотеки и демоны для реализации поиска в Akonadi
<code>akregator</code>	Читатель лент от KDE
<code>alarmcalendar</code>	KDE API для будильников KAlarm
<code>apidox</code>	Документация API KF5
<code>archive</code>	Библиотека KF5, предоставляющая классы для работы с форматами архивов
<code>attica</code>	Библиотека API Open Collaboration Services, версия для KDE5
<code>attica5</code>	Библиотека API Open Collaboration Services, версия для KDE5
<code>auth</code>	Абстракция KF5 для системной политики и функций аутентификации

Имя	Описание
<a href="#">baloo</a>	KF5 Фреймворк для поиска и управления пользовательскими метаданными
<a href="#">baloo-widgets</a>	Библиотека BalooWidgets
<a href="#">baloo5</a>	KF5 Фреймворк для поиска и управления пользовательскими метаданными
<a href="#">blog</a>	KDE API для доступа к веб-логам
<a href="#">bookmarks</a>	Библиотека KF5 для закладок и формата XBEL
<a href="#">breeze</a>	Plasma5 artwork, стили и ресурсы для визуального стиля Breeze
<a href="#">breeze-gtk</a>	Plasma5 Breeze визуальный стиль для Gtk
<a href="#">breeze-icons</a>	Тема значков Breeze для KDE
<a href="#">calendarcore</a>	Библиотека доступа к календарю KDE
<a href="#">calendarsupport</a>	Библиотеки поддержки календарей для KDEPim
<a href="#">calendarutils</a>	Утилита KDE и пользовательские функции интерфейса для доступа к календарю
<a href="#">codecs</a>	Библиотека KF5 для работы со строками
<a href="#">completion</a>	KF5 вспомогательные средства и виджеты автодополнения текста
<a href="#">config</a>	Виджеты KF5 для диалогов настройки
<a href="#">configwidgets</a>	Виджеты KF5 для диалогов настройки
<a href="#">contacts</a>	KDE API для управления контактной информацией
<a href="#">coreaddons</a>	KF5 аддоны для QtCore
<a href="#">crash</a>	Библиотека KF5 для обработки анализа сбоев и отчётов об ошибках в приложениях
<a href="#">dbusaddons</a>	KF5 дополнения к QtDBus
<a href="#">decoration</a>	Библиотека Plasma5 для создания оформления окон
<a href="#">designerplugin</a>	Интеграция KF5 виджетов Frameworks в Qt Designer/Creator
<a href="#">discover</a>	Инструменты управления пакетами Plasma5
<a href="#">dnssd</a>	Абстракция KF5 для системных функций DNSSD
<a href="#">doctools</a>	Генерация документации KF5 из docbook
<a href="#">drkonqi</a>	Обработчик сбоев Plasma5

Имя	Описание
esm	Дополнительные модули и скрипты для CMake
emoticons	Библиотека KF5 для преобразования эмотиконов
eventviews	Библиотеки просмотра событий для KDEPim
filemetadata	Библиотека KF5 для извлечения метаданных файлов
frameworkintegration	Плагины рабочего пространства KF5 и интеграции фреймворков
gapi	Библиотека на основе KDE для доступа к сервисам Google
globalaccel	Библиотека KF5 для добавления поддержки глобальных сочетаний клавиш рабочего пространства
grantee-editor	Редактор тем Grantlee
grantee-theme	KDE PIM grantleetheme
gravatar	Библиотека для поддержки gravatar
guiaddons	KF5 аддоны для QtGui
holidays	Библиотека KDE для календарных праздников
hotkeys	Библиотека Plasma5 для горячих клавиш
i18n	KF5 — расширенная инфраструктура интернационализации
iconthemes	Библиотека KF5 для работы с иконками в приложениях
identitymanagement	KDE pim идентификации
idletime	Библиотека KF5 для мониторинга активности пользователей
imap	KDE API для поддержки IMAP
incidenceeditor	Инцидентные редакторские библиотеки для KDEPim
infocenter	Утилита Plasma5, предоставляющая системную информацию
init	Запускатель процессов KF5 для ускорения запуска приложений KDE
itemmodels	Модели KF5 для системы Qt Model/View
itemviews	KF5 виджеты-дополнения для Qt Model/View

<b>Имя</b>	<b>Описание</b>
jobwidgets	Виджеты KF5 для отслеживания экземпляра KJob
js	Библиотека KF5, предоставляющая интерпретатор ECMAScript
jsembed	Библиотека KF5 для привязки объектов JavaScript к QObjects
kaddressbook	Менеджер контактов KDE
kalarm	Планировщик персональных сигналов тревоги
kalarm	Планировщик персональных сигналов тревоги
kate	Базовая структура редактора для системы KDE
kcmutils	KF5 утилиты для работы с KCMModules
kde-cli-tools	Неинтерактивные системные инструменты Plasma5
kde-gtk-config	Plasma5 конфигуратор GTK2 и GTK3
kdeclarative	Библиотека KF5, обеспечивающая интеграцию QML и KDE Frameworks
kded	KF5 расширяемый демон для предоставления системных сервисов
kdelibs4support	Помощник в портировании KF5 из KDELibs4
kdepim-addons	Дополнения KDE PIM
kdepim-apps-libs	Библиотеки KDE PIM, связанные с почтой
kdepim-runtime5	Инструменты и службы KDE PIM
kdeplasma-addons	Дополнения Plasma5 для улучшения работы с Plasma
kdesu	Интеграция KF5 с su для повышенных привилегий
kdewebkit	Библиотека KF5, обеспечивающая интеграцию QtWebKit
kgamma5	Настройки гаммы монитора Plasma5
khtml	Механизм рендеринга KF5 KHTML
kimageformats	Библиотека KF5, обеспечивающая поддержку дополнительных форматов изображений
kio	Абстракция ресурсов и сетевого доступа KF5

Имя	Описание
kirigami2	Набор компонентов на основе QtQuick
kitinerary	Модель данных и система извлечения информации о бронировании путешествий
kmail	Клиент электронной почты KDE
kmail	Клиент электронной почты KDE
kmail-account-wizard	Мастер настройки почтового аккаунта KDE
kmenuedit	Редактор меню Plasma5
knotes	Всплывающие примечания
kontact	KDE Персональный Органайзер
kontact	KDE Персональный Органайзер
kontactinterface	KDE glue для встраивания KParts в Kontact
korganizer	Программа для календаря и планирования
krimdav	Реализация протокола DAV с KJobs
krcpass	Библиотека для работы с файлами паролей Apple Wallet
kross	KF5 мультязыковые прикладные скрипты
kscreen	Библиотека управления экраном Plasma5
kscreenlocker	Архитектура безопасной блокировки экрана Plasma5
ksmtp	Библиотека на основе задач для отправки электронной почты через SMTP-сервер
ksshaskpass	Plasma5 интерфейс для ssh-add
ksysguard	Утилита Plasma5 для отслеживания и управления запущенными процессами
kwallet-pam	Интеграция Plasma5 KWallet с PAM
kwayland-integration	Интеграционные плагины для рабочего стола на основе Wayland
kwin	Менеджер окон Plasma5
kwrited	Демон Plasma5, ожидающий сообщения wall и write
ldap	API доступа к LDAP для KDE
libkcddb	Библиотека KDE CDDB
libkcompactdisc	Библиотека KDE для взаимодействия с аудио-CD
libkdcraw	Интерфейс LibRaw для KDE

<b>Имя</b>	<b>Описание</b>
<code>libkdegames</code>	Библиотеки, используемые играми KDE
<code>libkdepim</code>	Библиотеки KDE PIM
<code>libkeduvcdocument</code>	Библиотека для чтения и записи файлов словарей
<code>libkexiv2</code>	Интерфейс библиотеки Exiv2 для KDE
<code>libkipi</code>	Интерфейс плагинов изображений KDE
<code>libkleo</code>	Менеджер сертификатов для KDE
<code>libksane</code>	Интерфейс библиотеки SANE для KDE
<code>libkscreen</code>	Библиотека управления экраном Plasma5
<code>libksieve</code>	Библиотеки Sieve для KDEPim
<code>libksysguard</code>	Библиотека Plasma5 для отслеживания и управления запущенными процессами
<code>mailcommon</code>	Общие библиотеки для KDEPim
<code>mailimporter</code>	Импорт файлов mbox в KMail
<code>mailtransport</code>	Библиотека KDE для управления транспортом почты
<code>marble</code>	Виртуальный глобус и мировой атлас для KDE
<code>mbox</code>	Библиотека KDE для доступа к хранилищам почты в формате MBox
<code>mbox-importer</code>	Импорт файлов mbox в KMail
<code>mediaplayer</code>	Интерфейс плагина KF5 для функций медиаплеера
<code>messagelib</code>	Библиотека для обработки сообщений
<code>milou</code>	Plasma5 Plasmoid для поиска
<code>mime</code>	Библиотека для обработки данных MIME
<code>newstuff</code>	Библиотека KF5 для загрузки ресурсов приложений из сети
<code>notifications</code>	Абстракция KF5 для системных уведомлений
<code>notifyconfig</code>	Система конфигурации KF5 для KNotify
<code>okular</code>	Универсальная программа для просмотра документов KDE
<code>oxygen</code>	Стиль Plasma5 Oxygen
<code>oxygen-icons5</code>	Тема иконок Oxygen для KDE
<code>package</code>	Библиотека KF5 для загрузки и установки пакетов

Имя	Описание
parts	KF5 система плагинов для работы с документами
people	Библиотека KF5, предоставляющая доступ к контактам
pim-data-exporter	Импорт и экспорт настроек KDE PIM
pimcommon	Общие библиотеки для KDEPim
pimtextedit	Библиотека KDE для утилит редактирования текста, специфичных для PIM
plasma-browser-integration	Компоненты Plasma5 для интеграции браузеров в рабочий стол
plasma-desktop	Plasma5 рабочий стол Plasma
plasma-framework	KF5 - среда выполнения пользовательского интерфейса на основе плагинов, используемая для создания пользовательских интерфейсов
plasma-integration	Плагины интеграции Qt Platform Theme для рабочего окружения Plasma
plasma-pa	Plasma5 Микшер звука Plasma Pulse
plasma-sdk	Приложения Plasma5, полезные для разработки Plasma
plasma-workspace	Plasma5 Рабочее пространство Plasma
plasma-workspace-wallpapers	Обои Plasma5
plotting	KF5 облегченный фреймворк для построения графиков
polkit-kde-agent-1	Демон Plasma5, предоставляющий интерфейс аутентификации polkit
powerdevil	Инструмент Plasma5 для управления настройками энергопотребления
prison	Интерфейс API для создания штрихкодов
pty	Абстракция pty KF5
purpose	Предлагает доступные действия для конкретной цели
qqc2-desktop-style	Стиль Qt QuickControl2 для KDE
runner	KF5 параллелизованная система запросов
service	KF5 расширенные плагины и интроспекция сервисов

Имя	Описание
<code>solid</code>	Интеграция и обнаружение оборудования KF5
<code>sonnet</code>	KF5 библиотека проверки орфографии на основе плагинов
<code>syndication</code>	Библиотека KDE для обработки RSS-лент
<code>syntaxhighlighting</code>	Движок подсветки синтаксиса KF5 для структурированного текста и кода
<code>systemsettings</code>	Настройки системы Plasma5
<code>texteditor</code>	KF5 продвинутый встраиваемый текстовый редактор
<code>textwidgets</code>	KF5 расширенные виджеты для редактирования текста
<code>threadweaver</code>	KF5 дополнения к QtDBus
<code>tnef</code>	KDE API для обработки данных TNEF
<code>unitconversion</code>	Библиотека KF5 для преобразования единиц измерения
<code>user-manager</code>	Пользовательский менеджер Plasma5
<code>wallet</code>	KF5 безопасный и унифицированный контейнер для паролей пользователей
<code>wayland</code>	Обёртка клиентской и серверной библиотек KF5 для библиотек Wayland
<code>widgetsaddons</code>	KF5 аддоны для QtWidgets
<code>windowssystem</code>	Библиотека KF5 для доступа к оконной системе
<code>xmlgui</code>	KF5 настраиваемые пользователем главные окна
<code>xmlrpcclient</code>	Взаимодействие KF5 с XMLRPC-сервисами

#### Пример 76. Пример `USE_KDE`

Это простой пример порта для KDE. `USES= cmake` указывает порту использовать CMake, инструмент конфигурации, широко применяемый в проектах KDE (см. [Использование cmake](#) для подробного описания). `USE_KDE` добавляет зависимость от библиотек KDE. Необходимые компоненты KDE и другие зависимости можно определить через лог конфигурации. `USE_KDE` не подразумевает `USE_QT`. Если порту требуются некоторые компоненты Qt, укажите их в `USE_QT`.

```
USES=      cmake kde:5 qt:5
USE_KDE=   есм
```

```
USE_QT= core buildtools_build qmake_build
```

## 6.15. Использование LXQt

Приложения, зависящие от LXQt, должны устанавливать `USES+= lxqt` и задавать `USE_LXQT` списком необходимых компонентов из таблицы ниже

Таблица 27. Доступные компоненты LXQt

Имя	Описание
<code>buildtools</code>	Помощники для дополнительных модулей CMake
<code>libfmqt</code>	Привязки Libfm к Qt
<code>lxqt</code>	Ядро библиотеки LXQt
<code>qtxdg</code>	Реализация Qt спецификаций freedesktop.org XDG

Пример 77. Пример `USE_LXQT`

Это простой пример, `USE_LXQT` добавляет зависимость от библиотек LXQt. Необходимые компоненты LXQt и другие зависимости можно определить из лога конфигурации.

```
USES= cmake lxqt qt:5 tar:xz
USE_QT= core dbus widgets buildtools_build qmake_build
USE_LXQT= buildtools libfmqt
```

## 6.16. Использование Java

### 6.16.1. Определения переменных

Если порту требуется Java™ Development Kit (JDK™) для сборки, запуска или даже извлечения `distfile`, определите `USE_JAVA`.

В коллекции портов доступно несколько JDK от различных поставщиков и в нескольких версиях. Если порт должен использовать определённую версию, укажите её с помощью переменной `JAVA_VERSION`. Самая актуальная версия — [java/openjdk18](#), также доступны [java/openjdk17](#), [java/openjdk16](#), [java/openjdk15](#), [java/openjdk14](#), [java/openjdk13](#), [java/openjdk12](#), [java/openjdk11](#), [java/openjdk8](#) и [java/openjdk7](#).

Таблица 28. Переменные, которые могут быть установлены портами, использующими Java

Переменная	Значение
<code>USE_JAVA</code>	Определите для остальных переменных, чтобы они имели какой-либо эффект.

Переменная	Значение
JAVA_VERSION	Список подходящих версий Java для порта, разделённых пробелами. Необязательный символ + позволяет указать диапазон версий (допустимые значения: 8[+] 11[+] 17[+] 18[+] 19[+] 20[+] 21[+]).
JAVA_OS	Список разделённых пробелами подходящих операционных систем портов JDK для порта (допустимые значения: <code>native linux</code> ).
JAVA_VENDOR	Список подходящих поставщиков портов JDK для порта через пробел (допустимые значения: <code>openjdk oracle</code> ).
JAVA_BUILD	Когда установлено, добавляет выбранный порт JDK в зависимости сборки.
JAVA_RUN	Когда установлено, добавляет выбранный порт JDK в зависимости времени выполнения.
JAVA_EXTRACT	Когда установлено, добавляет выбранный порт JDK в зависимости для извлечения.

Ниже приведен список всех настроек, которые порт получит после установки `USE_JAVA`:

Таблица 29. Переменные, предоставляемые портам, использующим Java

Переменная	Значение
JAVA_PORT	Имя порта JDK (например, <code>java/openjdk6</code> ).
JAVA_PORT_VERSION	Полная версия порта JDK (например, <code>1.6.0</code> ). Требуются только первые две цифры номера версии, используйте <code>\${JAVA_PORT_VERSION:C/^[0-9])\.([0-9])(.*)\$/\1.\2/}</code> .
JAVA_PORT_OS	Операционная система, используемая портом JDK (например, <code>'native'</code> ).
JAVA_PORT_VENDOR	Поставщик порта JDK (например, <code>'openjdk'</code> ).
JAVA_PORT_OS_DESCRIPTION	Описание операционной системы, используемой портом JDK (например, <code>'Native'</code> ).
JAVA_PORT_VENDOR_DESCRIPTION	Описание поставщика порта JDK (например, <code>'OpenJDK BSD Porting Team'</code> ).
JAVA_HOME	Путь к каталогу установки JDK (например, <code>'/usr/local/openjdk6'</code> ).
JAVAC	Путь к используемому компилятору Java (например, <code>'/usr/local/openjdk6/bin/javac'</code> ).

Переменная	Значение
JAR	Путь к инструменту <code>jar</code> , который следует использовать (например, <code>'/usr/local/openjdk6/bin/jar'</code> или <code>'/usr/local/bin/fastjar'</code> ).
APPLETVIEWER	Путь к утилите <code>appletviewer</code> (например, <code>'/usr/local/openjdk6/bin/appletviewer'</code> ).
JAVA	Путь к исполняемому файлу <code>java</code> . Используется для запуска программ на Java (например, <code>'/usr/local/openjdk6/bin/java'</code> ).
JAVADOC	Путь к программе <code>javadoc</code> .
JAVAH	Путь к программе <code>javah</code> .
JAVAP	Путь к программе <code>javap</code> .
JAVA_KEYTOOL	Путь к утилите <code>keytool</code> .
JAVA_N2A	Путь к инструменту <code>native2ascii</code> .
JAVA_POLICYTOOL	Путь к программе <code>policytool</code> .
JAVA_SERIALVER	Путь к утилите <code>serialver</code> .
RMIC	Путь к генератору RMI-заглушек/скелетов, <code>rmic</code> .
RMIREGISTRY	Путь к программе реестра RMI, <code>rmiregistry</code> .
RMID	Путь к программе демона RMI <code>rmid</code> .
JAVA_CLASSES	Путь к архиву, содержащему файлы классов JDK, <code>\${JAVA_HOME}/jre/lib/rt.jar</code> .

Используйте цель `java-debug` в `make` для получения информации для отладки порта. Она отобразит значения многих из перечисленных ранее переменных.

Кроме того, определены следующие константы, чтобы все порты Java могли быть установлены единообразно:

Таблица 30. Константы, определенные для портов, использующих Java

Константа	Значение
JAVASHAREDIR	Базовый каталог для всего, связанного с Java. По умолчанию: <code>\${PREFIX}/share/java</code> .
JAVAJARDIR	Каталог, в котором установлены JAR-файлы. По умолчанию: <code>\${JAVASHAREDIR}/classes</code> .
JAVALIBDIR	Каталог, в котором расположены JAR-файлы, установленные другими портами. По умолчанию: <code>\${LOCALBASE}/share/java/classes</code> .

Связанные записи определены как в `PLIST_SUB` (документировано в [Изменение pkg-plist на](#)

основе переменных `Make`), так и в `SUB_LIST`.

## 6.16.2. Сборка с Ant

Когда порт должен собираться с использованием Apache Ant, он должен определять `USE_ANT`. Таким образом, Ant считается командой `sub-make`. Если цель `do-build` не определена в порте, будет установлена цель по умолчанию, которая запускает Ant в соответствии с `MAKE_ENV`, `MAKE_ARGS` и `ALL_TARGET`. Это аналогично механизму `USES= gmake`, который документирован в [Building Mechanisms](#).

## 6.16.3. Лучшие практики

При переносе библиотеки Java порт должен устанавливать JAR-файл(ы) в `#{JAVAJARDIR}`, а все остальное — в `#{JAVASHAREDIR}/#{PORTNAME}` (за исключением документации, см. ниже). Чтобы уменьшить размер упаковочного файла, ссылайтесь на JAR-файл(ы) напрямую в Makefile. Используйте следующую инструкцию (где `myport.jar` — имя JAR-файла, устанавливаемого как часть порта):

```
PLIST_FILES+=  #{JAVAJARDIR}/myport.jar
```

При переносе Java-приложения порт обычно устанавливает все компоненты в единый каталог (включая зависимости в виде JAR-файлов). В этом отношении настоятельно рекомендуется использовать `#{JAVASHAREDIR}/#{PORTNAME}`. Портеру предстоит решить, устанавливать ли дополнительные JAR-зависимости в этот каталог или использовать уже установленные (из `#{JAVAJARDIR}`).

При переносе Java™-приложения, которое требует сервера приложений, такого как [www/tomcat7](#), для запуска службы, вендор часто распространяет файл `.war`. `.war` — это веб-архив приложения (Web application ARchive), который извлекается при вызове приложением. Избегайте добавления `.war` в `pkg-plist`. Это не считается лучшей практикой. Сервер приложений развернет архив `war`, но не очистит его должным образом при удалении порта. Более предпочтительный способ работы с этим файлом — извлечь архив, затем установить файлы и, наконец, добавить эти файлы в `pkg-plist`.

```
TOMCATDIR=  #{LOCALBASE}/apache-tomcat-7.0
WEBAPPDIR=  myapplication

post-extract:
  @#{MKDIR}  #{WRKDIR}/#{PORTDIRNAME}
  @#{TAR} xf #{WRKDIR}/myapplication.war -C #{WRKDIR}/#{PORTDIRNAME}

do-install:
  cd #{WRKDIR} && \
  #{INSTALL} -d -o #{WWWOWN} -g #{WWWGRP} #{TOMCATDIR}/webapps/#{PORTDIRNAME}
  cd #{WRKDIR}/#{PORTDIRNAME} && #{COPYTREE_SHARE} \* #{WEBAPPDIR}/#{PORTDIRNAME}
```

Независимо от типа порта (библиотека или приложение), дополнительная документация

устанавливается [в том же месте](#), что и для любого другого порта. Известно, что инструмент Javadoc создает разный набор файлов в зависимости от версии используемого JDK. Для портов, которые не требуют использования конкретной версии JDK, указание списка упаковки (pkg-plist) становится сложной задачей. Это одна из причин, по которой разработчикам портов настоятельно рекомендуется использовать [PORTDOCS](#). Более того, даже если набор файлов, генерируемых [javadoc](#), можно предсказать, размер результирующего pkg-plist говорит в пользу использования [PORTDOCS](#).

Значение по умолчанию для [DATADIR](#) — `${PREFIX}/share/${PORTNAME}`. Рекомендуется переопределить [DATADIR](#) на `${JAVASHAREDIR}/${PORTNAME}` для портов Java. Действительно, [DATADIR](#) автоматически добавляется в [PLIST\\_SUB](#) (документировано в [Изменение pkg-plist на основе переменных Make](#)), поэтому используйте `%DATADIR%` напрямую в pkg-plist.

Что касается выбора между сборкой портов Java из исходного кода или их непосредственной установкой из бинарного дистрибутива, на момент написания документации определённой политики не существует. Однако участники [FreeBSD Java Project](#) рекомендуют сопровождающим портов по возможности собирать их из исходного кода, если это не представляет сложностей.

Все функции, представленные в этом разделе, реализованы в `bsd.java.mk`. Если порту требуется более сложная поддержка Java, сначала ознакомьтесь с [историей изменений bsd.java.mk](#), так как документирование новых функций обычно занимает некоторое время. Затем, если отсутствующая поддержка будет полезна для многих других Java-портов, не стесняйтесь обсудить это на списке рассылки `freebsd-java`.

Хотя существует категория [java](#) для PR, она относится к усилиям по портированию JDK в рамках проекта FreeBSD Java. Поэтому отправляйте порт Java в категорию [ports](#), как и любой другой порт, если только проблема не связана либо с реализацией JDK, либо с `bsd.java.mk`.

Аналогично существует определённая политика в отношении [CATEGORIES](#) для портов Java, которая подробно описана в [Категоризация](#).

## 6.17. Веб-приложения, Apache и PHP

### 6.17.1. Apache

Таблица 31. Переменные для портов, использующих Apache

<a href="#">USE_APACHE</a>	Порт требует Apache. Возможные значения: <a href="#">yes</a> (любая версия), <a href="#">22</a> , <a href="#">24</a> , <a href="#">22-24</a> , <a href="#">22+</a> и т.д. Версия APACHE по умолчанию — <a href="#">22</a> . Подробнее см. в <code>ports/Mk/bsd.apache.mk</code> и на <a href="#">wiki.freebsd.org/Apache/</a> .
<a href="#">APXS</a>	Полный путь к бинарному файлу <a href="#">apxs</a> . Может быть переопределён в порте.
<a href="#">HTTPD</a>	Полный путь к бинарному файлу <a href="#">httpd</a> . Может быть переопределён в порте.

<code>APACHE_VERSION</code>	Версия установленной Apache (переменная только для чтения). Эта переменная доступна только после включения <code>bsd.port.pre.mk</code> . Возможные значения: <code>22</code> , <code>24</code> .
<code>APACHEMODDIR</code>	Каталог для модулей Apache. Эта переменная автоматически раскрывается в <code>pkg-plist</code> .
<code>APACHEINCLUDEDIR</code>	Каталог для заголовков Apache. Эта переменная автоматически раскрывается в <code>pkg-plist</code> .
<code>APACHEETCDIR</code>	Каталог для файлов конфигурации Apache. Эта переменная автоматически раскрывается в <code>pkg-plist</code> .

Таблица 32. Полезные переменные для переноса модулей Apache

<code>MODULENAME</code>	Имя модуля. Значение по умолчанию — <code>PORTNAME</code> . Пример: <code>mod_hello</code>
<code>SHORTMODNAME</code>	Короткое имя модуля. Автоматически определяется из <code>MODULENAME</code> , но может быть переопределено. Пример: <code>hello</code>
<code>AP_FAST_BUILD</code>	Используйте <code>arxs</code> для компиляции и установки модуля.
<code>AP_GENPLIST</code>	Также автоматически создаёт файл <code>pkg-plist</code> .
<code>AP_INC</code>	Добавляет каталог в путь поиска заголовков во время компиляции.
<code>AP_LIB</code>	Добавляет каталог в путь поиска библиотек во время компиляции.
<code>AP_EXTRAS</code>	Дополнительные флаги для передачи в <code>arxs</code> .

## 6.17.2. Веб-приложения

Веб-приложения должны быть установлены в `PREFIX/www/appname`. Этот путь доступен как в `Makefile`, так и в `pkg-plist` как `WWWDIR`, а путь относительно `PREFIX` доступен в `Makefile` как `WWWDIR_REL`.

Пользователь и группа процесса веб-сервера доступны как `WWWOWN` и `WWWGRP`, если необходимо изменить владельца некоторых файлов. Значения по умолчанию для обоих — `www`. Используйте `WWWOWN?= myuser` и `WWWGRP?= mygroup`, если порту требуются другие значения. Это позволяет пользователю легко их переопределить.



Используйте `WWWOWN` и `WWWGRP` с осторожностью. Помните, что каждый файл, доступный для записи веб-серверу, представляет собой потенциальную угрозу безопасности.

Не зависьте от Apache, если веб-приложение явно не требует Apache. Учитывайте, что

пользователи могут захотеть запускать веб-приложение на другом веб-сервере, кроме Apache.

### 6.17.3. PHP

Веб-приложения PHP объявляют свою зависимость от него с помощью `USES=php`. Подробнее см. в [php](#).

### 6.17.4. Модули PEAR

Портирование модулей PEAR — это очень простой процесс.

Добавьте `USES=pear` в Makefile порта. Фреймворк установит соответствующие файлы в нужные места и автоматически сгенерирует plist во время установки.

*Пример 78. Пример Makefile для PEAR Class*

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

MAINTAINER=    someone@example.org
COMMENT=       PEAR Date and Time Zone Classes
WWW=           https://pear.php.net/package/Date/

USES=          pear

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



Модули PEAR будут автоматически преобразованы в порт с флейвором с использованием [флейворов PHP](#).



Если используется нестандартный `PEAR_CHANNEL`, зависимости для сборки и выполнения будут добавлены автоматически.



Модули PEAR не требуют определения `PKGNAME_SUFFIX`, так как он автоматически заполняется с использованием `PEAR_PKGNAMEPREFIX`. Если порту необходимо добавить к `PKGNAMEPREFIX`, он также должен использовать `PEAR_PKGNAMEPREFIX`, чтобы отличать различные флейворы.

#### 6.17.4.1. Модули Horde

Также и перенос модулей Horde является простым процессом.

Добавьте `USES=horde` в Makefile порта. Фреймворк установит соответствующие файлы в нужные места и автоматически сгенерирует plist во время установки.

Переменные `USE_HORDE_BUILD` и `USE_HORDE_RUN` могут использоваться для добавления зависимостей времени сборки и выполнения от других модулей Horde. Полный список доступных модулей можно найти в `Mk/Uses/horde.mk`.

*Пример 79. Пример Makefile для модуля Horde*

```
PORTNAME=  Horde_Core
DISTVERSION=  2.14.0
CATEGORIES=  devel www pear

MAINTAINER=  horde@FreeBSD.org
COMMENT=     Horde Core Framework libraries
WWW=        https://pear.horde.org/

OPTIONS_DEFINE=  KOLAB SOCKETS
KOLAB_DESC=     Enable Kolab server support
SOCKETS_DESC=   Depend on sockets PHP extension

USES=  horde
USE_PHP=  session

USE_HORDE_BUILD=  Horde_Role
USE_HORDE_RUN=   Horde_Role Horde_History Horde_Pack \
                Horde_Text_Filter Horde_View

KOLAB_USE=  HORDE_RUN=Horde_Kolab_Server,Horde_Kolab_Session
SOCKETS_USE=  PHP=sockets

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



Поскольку модули Horde также являются модулями PEAR, они будут автоматически преобразованы с использованием [флейворов PHP](#).

## 6.18. Использование Python

Коллекция портов поддерживает параллельную установку нескольких версий Python. Порты должны использовать правильный интерпретатор `python` в соответствии с настраиваемым пользователем параметром `PYTHON_VERSION`. Важнее всего заменить путь к исполняемому файлу `python` в скриптах на значение `PYTHON_CMD`.

Порты, которые устанавливают файлы в `PYTHON_SITELIBDIR`, должны использовать префикс имени пакета `pyXY-`, чтобы их имя пакета включало версию Python, для которой они предназначены.

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

Таблица 33. Наиболее полезные переменные для портов, использующих Python

<code>USES=python</code>	<p>Порту требуется Python. Минимально необходимая версия может быть указана с такими значениями, как <code>3.10+</code>. Диапазоны версий также можно указать, разделив две версии дефисом: <code>USES=python:3.8-3.9</code>. Обратите внимание, что <code>USES=python</code> не включает Python 2.7, его нужно запрашивать явно с помощью <code>USES=python:2.7+</code>.</p>
<code>USE_PYTHON=distutils</code>	<p>Используйте Python distutils для настройки, компиляции и установки. Это требуется, когда порт поставляется с setup.py. Это переопределяет цели <code>do-build</code> и <code>do-install</code>, а также может переопределить <code>do-configure</code>, если <code>GNU_CONFIGURE</code> не определён. Кроме того, подразумевается <code>USE_PYTHON=flavors</code>.</p>
<code>USE_PYTHON=autolist</code>	<p>Создать список пакетов автоматически. Это также требует установки <code>USE_PYTHON=distutils</code>.</p>
<code>USE_PYTHON=concurrent</code>	<p>Порт будет использовать уникальный префикс, обычно <code>PYTHON_PKGNAMEPREFIX</code>, для определённых каталогов, таких как <code>EXAMPLESDIR</code> и <code>DOCSDIR</code>, а также добавлять суффикс — версию Python из <code>PYTHON_VER</code> — к устанавливаемым бинарным файлам и скриптам. Это позволяет устанавливать порты для разных версий Python одновременно, что в противном случае приводило бы к конфликту файлов.</p>
<code>USE_PYTHON=flavors</code>	<p>Порт не использует distutils, но по-прежнему поддерживает несколько версий Python. <code>FLAVORS</code> будет установлен в поддерживаемые версии Python. Дополнительную информацию см. в <code>USES=python</code> и <code>флейворы</code>.</p>
<code>USE_PYTHON=optsuffix</code>	<p>Если текущая версия Python не является версией по умолчанию, порт получит <code>PKGNAME_SUFFIX=\${PYTHON_PKGNAME_SUFFIX}</code>. Полезно только для флейворов.</p>
<code>PYTHON_PKGNAMEPREFIX</code>	<p>Используется как <code>PKGNAMEPREFIX</code> для различения пакетов разных версий Python. Пример: <code>py27-</code></p>

<code>PYTHON_SITELIBDIR</code>	Расположение дерева site-packages, которое содержит путь установки Python (обычно <code>LOCALBASE</code> ). <code>PYTHON_SITELIBDIR</code> может быть очень полезно при установке модулей Python.
<code>PYTHONPREFIX_SITELIBDIR</code>	Вариант PREFIX-clean для <code>PYTHON_SITELIBDIR</code> . Всегда используйте <code>%%PYTHON_SITELIBDIR%%</code> в pkg-plist, когда это возможно. Значение по умолчанию для <code>%%PYTHON_SITELIBDIR%%</code> — <code>lib/python%%PYTHON_VERSION%%/site-packages</code>
<code>PYTHON_CMD</code>	Интерпретатор командной строки Python, включая номер версии.

Таблица 34. Помощники зависимостей модуля Python

<code>PYNUMERIC</code>	Строка зависимости для числового расширения.
<code>PYNUMPY</code>	Строка зависимости для нового числового расширения, numpy. (PYNUMERIC устарел у вендора).
<code>PYXML</code>	Строка зависимости для расширения XML (не требуется для Python 2.0 и выше, так как оно также входит в базовую поставку).
<code>PY_ENUM34</code>	Условная зависимость от пакета <a href="#">devel/py-enum34</a> в зависимости от версии Python.
<code>PY_ENUM_COMPAT</code>	Условная зависимость от пакета <a href="#">devel/py-enum-compat</a> в зависимости от версии Python.
<code>PY_PATHLIB</code>	Условная зависимость от пакета <a href="#">devel/py-pathlib</a> в зависимости от версии Python.
<code>PY_IPADDRESS</code>	Условная зависимость от пакета <a href="#">net/py-ipaddress</a> в зависимости от версии Python.
<code>PY_FUTURES</code>	Условная зависимость от пакета <a href="#">devel/py-futures</a> в зависимости от версии Python.

Полный список доступных переменных можно найти в `/usr/ports/Mk/Uses/python.mk`.



Все зависимости для портов Python, использующих [флейворы Python](#) (с `USE_PYTHON=distutils` или `USE_PYTHON=flavors`), должны иметь флейвор Python, добавленную к их origin с помощью `@${PY_FLAVOR}`. См. [Makefile для простого модуля Python](#).

```

PORTNAME=  sample
DISTVERSION=  1.2.3
CATEGORIES=  devel

MAINTAINER=  fred.bloggs@example.com
COMMENT=     Python sample module
WWW=        https://example.com/project/sample/

RUN_DEPENDS=  ${PYTHON_PKGNAMEPREFIX}six>0:devel/py-six@${PY_FLAVOR}

USES=        python
USE_PYTHON=  autolist distutils

.include <bsd.port.mk>

```

Некоторые приложения на Python заявляют о поддержке **DESTDIR** (что необходимо для промежуточной сборки), но она не работает (например, Mailman до версии 2.1.16). Это можно обойти, перекомпилировав скрипты. Это можно сделать, например, в цели **post-build**. Предполагая, что Python-скрипты должны находиться в **PYTHONPREFIX\_SITELIBDIR** после установки, можно применить следующее решение:

```

(cd ${STAGEDIR}${PREFIX} \
  && ${PYTHON_CMD} ${PYTHON_LIBDIR}/compileall.py \
  -d ${PREFIX} -f ${PYTHONPREFIX_SITELIBDIR:S;${PREFIX}/;;})

```

Это перекомпилирует исходники с путём, относительным к stage-директории, и добавляет значение **PREFIX** к имени файла, записанному в байт-компилированном выходном файле с помощью **-d**. **-f** требуется для принудительной перекомпиляции, а **:S;\${PREFIX}/;;** удаляет префиксы из значения **PYTHONPREFIX\_SITELIBDIR**, чтобы сделать его относительным к **PREFIX**.

## 6.19. Использование Tcl/Tk

Коллекция Ports поддерживает параллельную установку нескольких версий Tcl/Tk. Порты должны стараться поддерживать как минимум версию Tcl/Tk по умолчанию и выше с помощью **USES=tcl**. Можно указать желаемую версию **tcl**, добавив **:\_xx\_**, например, **USES=tcl:85**.

Таблица 35. Самые полезные переменные только для чтения для портов, использующих Tcl/Tk

<b>TCL_VER</b>	выбранная версия Tcl major.minor
<b>TCLSH</b>	полный путь к интерпретатору Tcl
<b>TCL_LIBDIR</b>	путь к библиотекам Tcl
<b>TCL_INCLUDEDIR</b>	путь к заголовочным файлам Tcl C

<code>TCL_PKG_LIB_PREFIX</code>	Префикс библиотеки, согласно TIP595
<code>TCL_PKG_STUB_POSTFIX</code>	Заглушка библиотеки postfix
<code>TK_VER</code>	выбранная версия Tk major.minor
<code>WISH</code>	полный путь к интерпретатору Tk
<code>TK_LIBDIR</code>	путь к библиотекам Tk
<code>TK_INCLUDEDIR</code>	путь к заголовочным файлам Tk C

См. `USES=tcl` и `USES=tk` в [Использование макросов USES](#) для полного описания этих переменных. Полный список этих переменных доступен в `/usr/ports/Mk/Uses/tcl.mk`.

## 6.20. Использование SDL

`USE_SDL` используется для автоматической настройки зависимостей для портов, которые используют библиотеку на основе SDL, такие как [devel/sdl12](#) и [graphics/sdl\\_image](#).

Эти библиотеки SDL для версии 1.2 распознаются:

- sdl: [devel/sdl12](#)
- console: [devel/sdl\\_console](#)
- gfx: [graphics/sdl\\_gfx](#)
- image: [graphics/sdl\\_image](#)
- mixer: [audio/sdl\\_mixer](#)
- mm: [devel/sdlmm](#)
- net: [net/sdl\\_net](#)
- pango: [x11-toolkits/sdl\\_pango](#)
- sound: [audio/sdl\\_sound](#)
- ttf: [graphics/sdl\\_ttf](#)

Эти библиотеки SDL для версии 2.0 распознаются:

- sdl: [devel/sdl20](#)
- gfx: [graphics/sdl2\\_gfx](#)
- image: [graphics/sdl2\\_image](#)
- mixer: [audio/sdl2\\_mixer](#)
- net: [net/sdl2\\_net](#)
- ttf: [graphics/sdl2\\_ttf](#)

Следовательно, если порт зависит от [net/sdl\\_net](#) и [audio/sdl\\_mixer](#), синтаксис будет следующим:

```
USE_SDL= net mixer
```

Пакет зависимости [devel/sdl12](#), который требуется для [net/sdl\\_net](#) и [audio/sdl\\_mixer](#), также автоматически добавляется.

Использование `USE_SDL` с указанием SDL 1.2 автоматически:

- Добавить зависимость от `sdl12-config` в `BUILD_DEPENDS`
- Добавьте переменную `SDL_CONFIG` в `CONFIGURE_ENV`
- Добавьте зависимости выбранных библиотек в `LIB_DEPENDS`

Используя `USE_SDL` с записями для SDL 2.0, это автоматически:

- Добавить зависимость от `sdl2-config` в `BUILD_DEPENDS`
- Добавьте переменную `SDL2_CONFIG` в `CONFIGURE_ENV`
- Добавьте зависимости выбранных библиотек в `LIB_DEPENDS`

## 6.21. Использование wxWidgets

Этот раздел описывает состояние библиотек wxWidgets в дереве портов и их интеграцию с системой портов.

### 6.21.1. Введение

Существует множество версий библиотек wxWidgets, которые конфликтуют между собой (устанавливают файлы с одинаковыми именами). В дереве портов эта проблема решена путем установки каждой версии под разными именами с использованием суффиксов номеров версий.

Очевидный недостаток этого подхода заключается в том, что каждое приложение необходимо модифицировать для поиска нужной версии. К счастью, большинство приложений вызывают скрипт `wx-config` для определения необходимых флагов компилятора и компоновщика. Имя этого скрипта отличается для каждой доступной версии. Большинство приложений учитывают переменную окружения или принимают аргумент `configure`, чтобы указать, какой скрипт `wx-config` вызывать. В противном случае их необходимо патчить.

### 6.21.2. Выбор версии

Чтобы порт использовал определённую версию wxWidgets, доступны две переменные для определения (если задана только одна, другая будет установлена в значение по умолчанию):

*Таблица 36. Переменные для выбора версий wxWidgets*

Переменная	Описание	Значение по умолчанию
<code>USE_WX</code>	Список версий, которые порт может использовать	Все доступные версии
<code>USE_WX_NOT</code>	Список версий, которые порт не может использовать	Ничего

Доступные версии wxWidgets и соответствующие порты в дереве:

Таблица 37. Доступные версии wxWidgets

Версия	Порт
2.8	<a href="#">x11-toolkits/wxgtk28</a>
3.0	<a href="#">x11-toolkits/wxgtk30</a>

Переменные в [Переменные для выбора версий wxWidgets](#) могут быть установлены в одну или несколько комбинаций, разделенных пробелами:

Таблица 38. Спецификации версий wxWidgets

Описание	Пример
Единственная версия	2.8
Возрастающий диапазон	2.8+
Нисходящий диапазон	3.0-
Полный диапазон (должен быть возрастающим)	2.8-3.0

Существуют также переменные для выбора предпочтительных версий из доступных. Их можно установить в виде списка версий, где первые будут иметь более высокий приоритет.

Таблица 39. Переменные для выбора предпочтительных версий wxWidgets

Имя	Предназначен для
<code>WANT_WX_VER</code>	порт
<code>WITH_WX_VER</code>	пользователь

### 6.21.3. Выбор компонентов

Существуют другие приложения, которые, не являясь библиотеками wxWidgets, связаны с ними. Эти приложения можно указать в `WX_COMPS`. Доступны следующие компоненты:

Таблица 40. Доступные компоненты wxWidgets

Имя	Описание	Ограничение версии
<code>wx</code>	основная библиотека	none
<code>contrib</code>	сторонние библиотеки	none
<code>python</code>	wxPython (интерфейс Python)	2.8-3.0

Тип зависимости может быть выбран для каждого компонента путем добавления суффикса, разделенного точкой с запятой. Если он отсутствует, будет использоваться тип по умолчанию (см. [Типы зависимостей wxWidgets по умолчанию](#)). Доступны следующие типы:

Таблица 41. Доступные типы зависимостей wxWidgets

Имя	Описание
<code>build</code>	Компонент необходим для сборки, эквивалентен <code>BUILD_DEPENDS</code>
<code>run</code>	Компонент необходим для запуска, эквивалентно <code>RUN_DEPENDS</code>
<code>lib</code>	Компонент необходим для сборки и запуска, эквивалентен <code>LIB_DEPENDS</code>

Значения по умолчанию для компонентов подробно описаны в этой таблице:

Таблица 42. Типы зависимостей wxWidgets по умолчанию

Компонент	Тип зависимости
<code>wx</code>	<code>lib</code>
<code>contrib</code>	<code>lib</code>
<code>python</code>	<code>run</code>
<code>mozilla</code>	<code>lib</code>
<code>svg</code>	<code>lib</code>

Пример 81. Выбор компонентов wxWidgets

Этот фрагмент соответствует порту, который использует wxWidgets версии 2.4 и дополнительные библиотеки.

```
USE_WX= 2.8
WX_COMPS= wx contrib
```

#### 6.21.4. Обнаружение установленных версий

Для определения установленной версии определите `WANT_WX`. Если значение не задано для конкретной версии, компоненты будут иметь суффикс версии. `HAVE_WX` будет заполнен после обнаружения.

Пример 82. Обнаружение установленных версий и компонентов wxWidgets

Этот фрагмент может использоваться в порте, который использует wxWidgets, если они установлены или выбран соответствующий параметр.

```
WANT_WX= yes
```

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

.if defined(WITH_WX) || !empty(PORT_OPTIONS:MWX) || !empty(HAVE_WX:Mwx-2.8)
USE_WX=      2.8
CONFIGURE_ARGS+=  --enable-wx
.endif
```

Этот фрагмент может использоваться в порте, который включает поддержку wxPython, если она установлена или выбрана соответствующая опция, в дополнение к wxWidgets, обе версии [2.8](#).

```
USE_WX=      2.8
WX_COMPS=    wx
WANT_WX=     2.8

.include <bsd.port.pre.mk>

.if defined(WITH_WXPYTHON) || !empty(PORT_OPTIONS:MWXPYTHON) ||
!empty(HAVE_WX:Mpython)
WX_COMPS+=   python
CONFIGURE_ARGS+=  --enable-wxpython
.endif
```

### 6.21.5. Переменные, определенные в фреймворке

Эти переменные доступны в порте (после определения одной из [переменных для выбора версий wxWidgets](#)).

Таблица 43. Переменные, определенные для портов, использующих wxWidgets

Имя	Описание
<code>WX_CONFIG</code>	Путь к скрипту <code>wx-config</code> wxWidgets (с другим именем)
<code>WXRC_CMD</code>	Путь к программе <code>wxrc</code> wxWidgets (с другим именем)
<code>WX_VERSION</code>	Версия wxWidgets, которая будет использоваться (например, <a href="#">2.6</a> )

### 6.21.6. Обработка в `bsd.port.pre.mk`

Определите `WX_PREMK`, чтобы иметь возможность использовать переменные сразу после включения `bsd.port.pre.mk`.



При определении `WX_PREMK` версия, зависимости, компоненты и определенные переменные не изменятся при модификации переменных порта wxWidgets *после* включения `bsd.port.pre.mk`.

Этот фрагмент иллюстрирует использование `WX_PREMK` путем запуска скрипта `wx-config` для получения полной строки версии, присвоения её переменной и передачи в программу.

```
USE_WX=    2.8
WX_PREMK=  yes

.include <bsd.port.pre.mk>

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

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



Переменные wxWidgets можно безопасно использовать в командах внутри целей без необходимости в `WX_PREMK`.

### 6.21.7. Дополнительные параметры `configure`

Некоторые скрипты GNU `configure` не могут найти wxWidgets, если задана только переменная окружения `WX_CONFIG`, и требуют дополнительные параметры. `WX_CONF_ARGS` можно использовать для их указания.

Таблица 44. Допустимые значения для `WX_CONF_ARGS`

Возможное значение	Получаемый параметр
<code>absolute</code>	<code>--with-wx-config=\${WX_CONFIG}</code>
<code>relative</code>	<code>--with-wx=\${LOCALBASE} --with-wx-config=\${WX_CONFIG:T}</code>

## 6.22. Использование Lua

Этот раздел описывает состояние библиотек Lua в дереве портов и их интеграцию с системой портов.

### 6.22.1. Введение

Существует множество версий библиотек Lua и соответствующих интерпретаторов, которые конфликтуют между собой (устанавливают файлы с одинаковыми именами). В дереве портов эта проблема решена установкой каждой версии под разными именами с использованием суффиксов номеров версий.

Очевидный недостаток этого заключается в том, что каждое приложение необходимо модифицировать для поиска ожидаемой версии. Однако это можно решить, добавив

дополнительные флаги компилятору и компоновщику.

Приложения, использующие Lua, обычно должны собираться только для одной версии. Однако загружаемые модули для Lua собираются в отдельных флейворах для каждой поддерживаемой версии Lua, и зависимости от таких модулей должны указывать флейвор с использованием суффикса `@${LUA_FLAVOR}` в расположении (origin) порта.

### 6.22.2. Выбор версии

Порт, использующий Lua, должен содержать строку следующего вида:

```
USES= lua
```

Если требуется определённая версия Lua или диапазон версий, его можно указать в виде параметра `XY` (который можно использовать несколько раз), `XY+`, `-XY` или `XY-ZA`. Версия Lua, установленная через `DEFAULT_VERSIONS`, будет использована, если она попадает в запрошенный диапазон, в противном случае будет использована ближайшая к умолчанию запрошенная версия. Например:

```
USES= lua:52-53
```

Обратите внимание, что не предпринимается попытка изменить выбор версии на основе наличия любой уже установленной версии Lua.



Форма указания версии `XY+` не должна использоваться без тщательного обдумывания; Lua API в некоторой степени меняется с каждой версией, и инструменты конфигурации, такие как CMake или Autoconf, скорее всего не будут работать с будущими версиями Lua, пока не будут обновлены для этого.

### 6.22.3. Конфигурация и флаги компилятора

Программное обеспечение, использующее Lua, может быть написано с автоматическим определением версии Lua в использовании. В общем случае порты должны переопределять это предположение и принудительно использовать конкретную выбранную версию Lua, как описано выше. В зависимости от портируемого программного обеспечения, это может потребовать любого или всех из следующих действий:

- Использование `LUA_VER` в качестве части параметра для скрипта конфигурации программного обеспечения через `CONFIGURE_ARGS` или `CONFIGURE_ENV` (или эквивалентные для других систем сборки);
- Добавление `-I${LUA_INCDIR}`, `-L${LUA_LIBDIR}` и `-llua-${LUA_VER}` в `CFLAGS`, `LDFLAGS` и `LIBS` соответственно, где это необходимо;
- Исправьте конфигурационные или файлы сборки программного обеспечения, чтобы выбрать правильную версию.

## 6.22.4. Флейворы версии

Порт, который устанавливает модуль Lua (а не приложение, просто использующее Lua), должен собирать отдельный флейвор для каждой поддерживаемой версии Lua. Это делается путем добавления параметра `module`:

```
USES= lua:module
```

Так же может быть указание номера версии или диапазона версий. Используйте запятую для разделения параметров.

Поскольку каждый флейвор должен иметь уникальное имя пакета, предоставляется переменная `LUA_PKGNAMEPREFIX`, которая будет установлена в соответствующее значение; предполагаемое использование:

```
PKGNAMEPREFIX= ${LUA_PKGNAMEPREFIX}
```

Модульные порты обычно должны устанавливать файлы только в `LUA_MODLIBDIR`, `LUA_MODSHAREDIR`, `LUA_DOCSDIR` и `LUA_EXAMPLESDIR`, все из которых настроены на ссылки в версиянно-зависимые подкаталоги. Установка любых других файлов должна выполняться с осторожностью, чтобы избежать конфликтов между версиями.

Порт (кроме модуля Lua), который хочет собрать отдельный пакет для каждой версии Lua, должен использовать параметр `flavors`:

```
USES= lua:flavors
```

Это работает так же, как параметр `module`, описанный выше, но без предположения, что пакет должен быть задокументирован как модуль Lua (поэтому `LUA_DOCSDIR` и `LUA_EXAMPLESDIR` по умолчанию не определены). Однако порт может определить `LUA_DOCSUBDIR` как подходящее имя подкаталога (обычно `PORTNAME` порта, если это не конфликтует с `PORTNAME` любого модуля), и в этом случае фреймворк определит как `LUA_DOCSDIR`, так и `LUA_EXAMPLESDIR`.

Как и в случае с модульными портами, порт с флейворами должен избегать установки файлов, которые могут конфликтовать между версиями. Обычно это достигается добавлением `LUA_VER_STR` в качестве суффикса к именам программ (например, с использованием `uniquefiles`), а также использованием `LUA_VER` или `LUA_VER_STR` в составе других файлов или поддиректорий, используемых вне `LUA_MODLIBDIR` и `LUA_MODSHAREDIR`.

## 6.22.5. Переменные, определенные в фреймворке

В порте доступны эти переменные.

*Таблица 45. Переменные, определенные для портов, использующих Lua*

Имя	Описание
LUA_VER	Версия Lua, которая будет использоваться (например, 5.4)
LUA_VER_STR	Версия Lua без точек (например, 54)
LUA_FLAVOR	Имя флейвора, соответствующее выбранной версии Lua, используемое для указания зависимостей
LUA_BASE	Префикс, который должен использоваться для поиска Lua (и компонентов), уже установленных
LUA_PREFIX	Префикс, куда этим портом будут установлены Lua (и компоненты)
LUA_INCDIR	Каталог, в котором установлены заголовочные файлы Lua
LUA_LIBDIR	Каталог, в котором установлены библиотеки Lua
LUA_REFMODLIBDIR	Каталог, в котором находятся уже установленные библиотеки модулей Lua (.so)
LUA_REFMODSHAREDIR	Каталог, в котором находятся установленные модули Lua (.lua)
LUA_MODLIBDIR	Каталог, в котором библиотеки модулей Lua (.so) должны быть установлены данным портом
LUA_MODSHAREDIR	Каталог, в котором должны быть установлены модули Lua (.lua) данным портом
LUA_PKGNAMEPREFIX	Префикс имени пакета, используемый модулями Lua
LUA_CMD	Название интерпретатора Lua (например, lua54)
LUAC_CMD	Название компилятора Lua (например, luac54)

Эти дополнительные переменные доступны для портов, которые указали параметр `module`:

Таблица 46. Переменные, определенные для модулей Lua в портах

Имя	Описание
LUA_DOCSDIR	каталог, в который должна быть установлена документация модуля.
LUA_EXAMPLESDIR	каталог, в который должны быть установлены примеры файлов модуля.

## 6.22.6. Примеры

Пример 84. *Makefile* для приложения, использующего Lua

Этот пример показывает, как сослаться на модуль Lua, требуемый во время выполнения. Обратите внимание, что ссылка должна указывать флейвор.

```
PORTNAME= sample
DISTVERSION= 1.2.3
CATEGORIES= whatever

MAINTAINER= fred.bloggs@example.com
COMMENT= Sample
WWW= https://example.com/lua_sample/sample/

RUN_DEPENDS= ${LUA_REFMODLIBDIR}/lpeg.so:devel/lua-lpeg@${LUA_FLAVOR}

USES= lua

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

Пример 85. *Makefile* для простого модуля Lua

```
PORTNAME= sample
DISTVERSION= 1.2.3
CATEGORIES= whatever
PKGNAMEPREFIX= ${LUA_PKGNAMEPREFIX}

MAINTAINER= fred.bloggs@example.com
COMMENT= Sample
WWW= https://example.com/lua_sample/sample/

USES= lua:module

DOCSDIR= ${LUA_DOCSDIR}

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

## 6.23. Использование Guile

Этот раздел описывает состояние Guile в дереве портов и его интеграцию с системой портов.

### 6.23.1. Введение

Существует несколько версий библиотек Guile и соответствующих интерпретаторов, которые конфликтуют между собой (устанавливают файлы с одинаковыми именами). В

дереве портов эта проблема решена путем установки каждой версии под разными именами с использованием суффиксов номеров версий. В большинстве случаев приложения должны определять правильную версию из предоставленных конфигурационных переменных и использовать `pkg-config` для определения имени и связанных путей. Однако некоторые приложения (особенно те, которые используют собственные правила конфигурации для `cmake` или `meson`) всегда будут пытаться использовать последнюю доступную версию. В этом случае либо исправьте порт, либо объявите конфликт сборки (см. опцию `conflicts` ниже), чтобы гарантировать создание правильной зависимости при сборке вне `roudrriere`.

Приложения, использующие Guile, обычно должны собираться только для одной версии, предпочтительно указанной в `DEFAULT_VERSIONS`, или, если это невозможно, для последней поддерживаемой версии. Однако библиотеки Guile или Scheme, а также модули расширения для Guile собираются в отдельных флейворах для каждой поддерживаемой версии Guile. Зависимости от таких портов должны указывать флейвор с использованием суффикса `@${GUILE_FLAVOR}` в расположении (`origin`) порта.

### 6.23.2. Выбор версии

Порт, использующий Guile, должен определять `USES=guile:arg,arg...` с соответствующими параметрами следующим образом:

Таблица 47. Параметры, определенные для портов, использующих Guile

Имя	Описание
<code>X.Y</code>	Объявить совместимость с версией Guile <code>X.Y</code> . Доступные версии: <code>1.8</code> (устарела), <code>2.2</code> и <code>3.0</code> . Можно указать несколько версий.
<code>flavors</code>	Создать флейвор для каждой указанной версии Guile. Версия, указанная в <code>DEFAULT_VERSIONS</code> , станет флейвором по умолчанию. Названия флейворов имеют вид <code>guileXY</code> .
<code>build</code>	Добавить интерпретатор Guile только как зависимость для сборки, а не как зависимость библиотеки. <code>build</code> и <code>run</code> могут быть указаны оба.
<code>run</code>	Добавить интерпретатор Guile только как зависимость во время выполнения, а не как зависимость от библиотеки. <code>build</code> и <code>run</code> могут быть указаны оба.
<code>alias</code>	Добавить значения <code>BINARY_ALIAS</code> для интерпретатора и инструментов.
<code>conflicts</code>	Объявить <code>CONFLICTS_BUILD</code> для версий Guile новее выбранной. Используйте это, когда порт нельзя настроить на использование определённой версии Guile.

Некоторые дополнительные аргументы доступны для обработки нестандартных случаев; подробности см. в `Mk/Uses/guile.mk`.

Если не указано `build` или `run`, то `LIB_DEPENDS` получает зависимость от библиотеки `libguile`, а также любые дополнительные зависимости, требуемые версией `guile`, например, `libgc`. Обычно порту не требуются дополнительные зависимости, связанные с использованием Guile.

### 6.23.3. Флаги конфигурации

Программное обеспечение, использующее Guile, должно использовать механизм `pkg-config` для получения флагов компилятора и компоновщика. Некоторые старые или экзотические порты могут использовать `guile-config` или получать значения напрямую из `guile`, что также должно работать (в некоторых из этих случаев может быть полезен аргумент `alias`).

Фреймворк пытается сообщить порту желаемую версию Guile, используя следующие методы:

- `GUILE_EFFECTIVE_VERSION` добавлен в `CONFIGURE_ENV`;
- Полный путь к исполняемому файлу Guile указан в переменной `GUILE` в `CONFIGURE_ENV` и `MAKE_ENV`;
- Если используется опция `alias`, то желаемые версии бинарных файлов Guile являются теми, которые имеют алиасы;
- Если параметр `alias` не используется, пути к инструментам нужной версии Guile (`guild`, `guile-config` и т.д.) добавляются в `CONFIGURE_ENV` и `MAKE_ENV` в виде переменных `GUILD`, `GUILE_CONFIG` и т.д.

Для некоторых портов может потребоваться указать версию дополнительными способами, например, через `CONFIGURE_ARGS` или `MESON_ARGS`, в зависимости от порта.

Если ни один из этих методов не приводит к тому, что порт выбирает указанную версию Guile при наличии других версий, то предпочтительно исправить его, чтобы это происходило. Если это невозможно, укажите опцию `conflicts`, чтобы предотвратить сборку порта в условиях, когда он обнаруживает неправильную версию.

### 6.23.4. Флейворы версии

Порт, который устанавливает расширение или библиотеку Guile, или библиотеку Scheme, которая предварительно компилируется для Guile, должен собирать отдельный флейвор для каждой поддерживаемой версии Guile. Это делается путем добавления опции `flavors`.

Поскольку каждый флейвор должен иметь уникальное имя пакета, такие порты обычно устанавливают `PKGNAME_SUFFIX`, например:

```
PKGNAME_SUFFIX= -${FLAVOR}
```

Такие порты должны устанавливать файлы Scheme в `GUILE_SITE_DIR`, а не в `GUILE_GLOBAL_SITE_DIR`, даже если файлы не зависят от версии. Это часто требует исправления порта.

Кроме того, если такой порт устанавливает файл `.pc`, он должен быть размещён в `GUILE_PKGCONFIG_PATH`, а не в глобальной директории `pkgconfig`. Это позволяет зависимым портам находить правильную конфигурацию для используемой версии Guile.

Если порт расширения Guile устанавливает файл `.so`, то обычно он должен быть размещён в специфичной для версии Guile директории `extensions`. Обычно не следует использовать

USE\_LDCONFIG.

Любые другие файлы, устанавливаемые портом с флейвором, также должны находиться в версионных каталогах или использовать версионные имена файлов. Для документации и примеров переменные `GUILE_DOCS_DIR` и `GUILE_EXAMPLES_DIR` указывают подходящие расположения, в которых порт должен создать подкаталог (см. ниже).

### 6.23.5. Переменные, определенные в фреймворке

В порте доступны эти переменные.

Таблица 48. Переменные, определенные для портов, использующих Guile

Имя	Пример значения	Описание
<code>GUILE_VERSION</code>	<code>3.0</code>	Используемая версия Guile.
<code>GUILE_SUFFIX</code>	<code>3</code>	Короткий суффикс, используемый в некоторых именах. Используйте с осторожностью; может быть неуникальным или измениться в будущем.
<code>GUILE_FLAVOR</code>	<code>guile30</code>	Название флейвора, соответствующее выбранной версии.
<code>GUILE_PORT</code>	<code>lang/guile3</code>	Расположение порта (origin) для указанной версии Guile.
<code>GUILE_PREFIX</code>	<code>\${PREFIX}</code>	Префикс каталога для использования при установке.
<code>GUILE_CMD</code>	<code>guile-3.0</code>	Имя интерпретатора Guile с суффиксом версии.
<code>GUILE_CMDPATH</code>	<code>\${LOCALBASE}/bin/guile-3.0</code>	Полный путь к интерпретатору Guile.
<code>GUILD_CMD</code>	<code>guild-3.0</code>	Название инструмента Guild, с суффиксом версии.
<code>GUILD_CMDPATH</code>	<code>\${LOCALBASE}/bin/guild-3.0</code>	Полный путь к инструменту Guild.
<code>GUILE_*_CMD</code> <code>GUILE_*_CMDPATH</code>		Как <code>GUILE_CMD</code> и <code>GUILE_CMDPATH</code> , но для других исполняемых файлов утилит.
<code>GUILE_PKGCONFIG_PATH</code>	<code>\${LOCALBASE}/libdata/pkgconfig/guile/3.0</code>	Где пакеты, использующие <code>flavors</code> , должны устанавливать файлы <code>.pc</code> .
<code>GUILE_INFO_PATH</code>	<code>share/info/guile3</code>	Подходящее значение для <code>INFO_PATH</code> для портов, использующих опцию <code>flavors</code> .

Следующие элементы определены как переменные и как записи `PLIST_SUB`. Форма переменной имеет суффикс `_DIR` и представляет собой полный путь (с префиксом `GUILE_PREFIX`).

Таблица 49. Подстановки путей, определенные для портов, использующих Guile

Имя	Пример значения	Описание
GUILE_GLOBAL_SITE	share/guile/site	Каталог сайта, общий для всех версий guile; обычно не должен использоваться.
GUILE_SITE	share/guile/3.0/site	Каталог сайта для выбранной версии Guile.
GUILE_SITE_CCACHE	lib/guile/3.0/site-ccache	Каталог для скомпилированных файлов байт-кода.
GUILE_DOCS	share/doc/guile30	Родительский каталог для документации, специфичной для версий.
GUILE_EXAMPLES	share/examples/guile30	Родительский каталог для примеров, специфичных для версий.

### 6.23.6. Примеры

Пример 86. *Makefile* для приложения, использующего *Guile*

Этот пример демонстрирует, как сослаться на библиотеку Guile, необходимую во время сборки и выполнения. Обратите внимание, что ссылка должна указывать флейвор. В этом примере предполагается, что приложение использует `pkg-config` для поиска зависимостей.

```

PORTNAME=  sample
DISTVERSION=  1.2.3
CATEGORIES=  whatever

MAINTAINER=  fred.bloggs@example.com
COMMENT=     Sample
WWW=         https://example.com/guile_sample/sample/

BUILD_DEPENDS=  guile-lib-`${GUILE_FLAVOR}>=0.2.5:devel/guile-lib@`${GUILE_FLAVOR}
RUN_DEPENDS=    guile-lib-`${GUILE_FLAVOR}>=0.2.5:devel/guile-lib@`${GUILE_FLAVOR}

USES=         guile:2.2,3.0 pkgconfig

.include <bsd.port.mk>

```

## 6.24. Использование `iconv`

В FreeBSD имеется встроенная реализация `iconv` в самой операционной системе.

Для программного обеспечения, требующего `iconv`, определите `USES=iconv`.

Когда порт определяет `USES=iconv`, становятся доступны следующие переменные:

Имя переменной	Назначение	Порт <code>iconv</code> (при использовании расширений <code>WCHAR_T</code> или <code>//TRANSLIT</code> )	Базовый <code>iconv</code>
<code>ICONV_CMD</code>	Каталог, в котором находится бинарный файл <code>iconv</code>	<code>\${LOCALBASE}/bin/iconv</code>	<code>/usr/bin/iconv</code>
<code>ICONV_LIB</code>	аргумент <code>ld</code> для линковки с <code>libiconv</code> (если требуется)	<code>-liconv</code>	(пусто)
<code>ICONV_PREFIX</code>	Каталог, в котором находится реализация <code>iconv</code> (полезно для скриптов <code>configure</code> )	<code>\${LOCALBASE}</code>	<code>/usr</code>
<code>ICONV_CONFIGURE_ARG</code>	Предварительно сконструированный аргумент <code>configure</code> для скриптов <code>configure</code>	<code>--with-libiconv</code> <code>-prefix=\${LOCALBASE}</code>	(пусто)
<code>ICONV_CONFIGURE_BASE</code>	Предварительно сконструированный аргумент <code>configure</code> для скриптов <code>configure</code>	<code>--with</code> <code>-libiconv=\${LOCALBASE}</code>	(пусто)

Эти два примера автоматически заполняют переменные правильным значением для систем, использующих `converters/libiconv` или `iconv`, входящий в состав операционной системы, соответственно:

*Пример 87. Простое использование `iconv`*

```
USES=      iconv
LD_FLAGS+= -L${LOCALBASE}/lib ${ICONV_LIB}
```

*Пример 88. Использование `iconv` с `configure`*

```
USES=      iconv
CONFIGURE_ARGS+=${ICONV_CONFIGURE_ARG}
```

Как показано выше, `ICONV_LIB` пуста, когда присутствует встроенный `iconv`. Это можно

использовать для обнаружения встроенного `iconv` и действовать соответственно.

Иногда в программе аргумент `ld` или путь поиска жестко заданы в Makefile или скрипте `configure`. Для решения этой проблемы можно использовать следующий подход:

*Пример 89. Исправление жестко заданного `-liconv`*

```
USES=      iconv

post-patch:
    @${REINPLACE_CMD} -e 's/-liconv/${ICONV_LIB}/' ${WRKSRC}/Makefile
```

В некоторых случаях необходимо установить альтернативные значения или выполнить операции в зависимости от наличия встроенного `iconv`. `bsd.port.pre.mk` должен быть включен до проверки значения `ICONV_LIB`:

*Пример 90. Проверка доступности встроенной поддержки `iconv`*

```
USES=      iconv

.include <bsd.port.pre.mk>

post-patch:
    .if empty(ICONV_LIB)
        # native iconv detected
        @${REINPLACE_CMD} -e 's|iconv||' ${WRKSRC}/Config.sh
    .endif

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

## 6.25. Использование Xfce

Порты, которым требуются библиотеки или приложения Xfce, устанавливаются `USES=xfce`.

Конкретные зависимости библиотек и приложений Xfce задаются с помощью значений, присвоенных `USE_XFCE`. Они определены в `/usr/ports/Mk/Uses/xfce.mk`. Возможные значения:

Значения `USE_XFCE`

**garcon**

[sysutils/garcon](#)

**libexo**

[x11/libexo](#)

## libgui

[x11-toolkits/libxfce4gui](#)

## libmenu

[x11/libxfce4menu](#)

## libutil

[x11/libxfce4util](#)

## panel

[x11-wm/xfce4-panel](#)

## thunar

[x11-fm/thunar](#)

## xfconf

[x11/xfce4-conf](#)

Пример 91. Пример `USES=xfce`

```
USES=      xfce
USE_XFCE=   libmenu
```

Пример 92. Использование собственных виджетов GTK2 в Xfce

В этом примере портированное приложение использует пакет виджетов, специфичных для GTK2: [x11/libxfce4menu](#) и [x11/xfce4-conf](#).

```
USES=      xfce:gtk2
USE_XFCE=   libmenu xfconf
```



Компоненты Xfce, включённые таким образом, автоматически загрузят все необходимые зависимости. Указывать полный список больше не требуется. Если порту нужен только [x11-wm/xfce4-panel](#), используйте:

```
USES=      xfce
USE_XFCE=   panel
```

Нет необходимости перечислять компоненты [x11-wm/xfce4-panel](#), которые ему самому требуются, вот так:

```
USES=      xfce
USE_XFCE=   libexo libmenu libutil panel
```

Однако компоненты Xfce и зависимости порта, не относящиеся к Xfce, должны быть явно включены. Не рассчитывайте, что компонент Xfce предоставит дополнительную зависимость, кроме себя, для основного порта.

## 6.26. Использование Budgie

Приложения или библиотеки, зависящие от рабочего стола Budgie, должны указывать `USES=budgie` и устанавливать `USE_BUDGIE` в список необходимых компонентов.

Имя	Описание
<code>libbudgie</code>	Ядро рабочего стола (библиотека)
<code>libmagpie</code>	Оконный менеджер X11 и библиотека композитинга Budgie
<code>raven</code>	Универсальный центр в панели для доступа к различным виджетам приложений
<code>screensaver</code>	Рабочий стол: специальная заставка

Все виджеты приложений взаимодействуют через службу `org.budgie_desktop.Raven`.



Зависимость по умолчанию включает время сборки и выполнения, её можно изменить с помощью `:build` или `:run`, например:

```
USES=      budgie
USE_BUDGIE= screensaver:build
```

Пример 93. Пример `USE_BUDGIE`

```
USES=      budgie gettext gnome meson pkgconfig
USE_BUDGIE= libbudgie
```

## 6.27. Использование баз данных

Используйте один из макросов `USES` из [Макросы USES для баз данных](#), чтобы добавить зависимость от базы данных.

Таблица 50. Макросы `USES` для баз данных

База данных	Макрос USES
Berkeley DB	<code>bdb</code>
MariaDB, MySQL, Percona	<code>mysql</code>

База данных	Макрос USES
PostgreSQL	<code>pgsql</code>
SQLite	<code>sqlite</code>

Пример 94. Использование Berkeley DB 6

```
USES=    bdb:6
```

См. `bdb` для получения дополнительной информации.

Пример 95. Использование MySQL

Когда порту требуется клиентская библиотека MySQL, добавьте

```
USES=    mysql
```

См. `mysql` для получения дополнительной информации.

Пример 96. Использование PostgreSQL

Когда порту требуется сервер PostgreSQL версии 9.6 или новее, добавьте

```
USES=    postgresql:9.6+
WANT_PGSQL= server
```

См. `pgsql` для получения дополнительной информации.

Пример 97. Использование SQLite 3

```
USES=    sqlite:3
```

См. `sqlite` для получения дополнительной информации.

## 6.28. Запуск и остановка служб (скрипты `rc`)

`rc.d` скрипты используются для запуска служб при загрузке системы, а также предоставляют администраторам стандартный способ остановки, запуска и перезапуска служб. Порты интегрируются в систему `rc.d`. Подробности использования можно найти в [соответствующей главе Handbook](#). Детальное объяснение доступных команд приведено в `rc(8)` и `rc.subr(8)`. Наконец, существует [статья](#), посвящённая практическим аспектам

написания rc.d скриптов.

С мифическим портом под названием *doorman*, которому необходимо запустить демон *doormand*. Добавьте следующее в Makefile:

```
USE_RC_SUBR= doormand
```

Можно указать несколько скриптов, которые будут установлены. Скрипты должны быть размещены в подкаталоге `files`, и к их имени должен быть добавлен суффикс `.in`. Для этого файла будут выполнены стандартные подстановки `SUB_LIST`. Также настоятельно рекомендуется использовать подстановки `%%PREFIX%%` и `%%LOCALBASE%%`. Подробнее о `SUB_LIST` см. в [соответствующем разделе](#).

Начиная с FreeBSD 6.1-RELEASE, локальные скрипты rc.d (включая те, что установлены через порты) включены в общий `rcorder(8)` базовой системы.

Пример простого скрипта rc.d для запуска демона *doormand*:

```
#!/bin/sh

# PROVIDE: doormand
# REQUIRE: LOGIN
# KEYWORD: shutdown
#
# Add these 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.

. /etc/rc.subr

name=doormand
rcvar=doormand_enable

load_rc_config $name

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

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

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

run_rc_command "$1"
```

Если нет очень веской причины запускать службу раньше или она работает от имени определенного пользователя (не root), все скрипты портов должны использовать:

```
REQUIRE: LOGIN
```

Если скрипт запуска демона требует его остановки, следующий код активирует остановку службы при выключении системы:

```
KEYWORD: shutdown
```

Если скрипт не запускает постоянную службу, это не требуется.

Для необязательных элементов конфигурации предпочтительнее использовать стиль присваивания переменных по умолчанию "=" вместо стиля ":", так как первый устанавливает значение по умолчанию только если переменная не задана, а второй — если переменная не задана *или* равна null. Пользователь может включить что-то вроде:

```
doormand_flags=""
```

в свой `rc.conf.local`, а подстановка переменной с использованием ":" некорректно переопределила бы намерение пользователя. Переменная `_enable` не является опциональной и должна использовать ":" для значения по умолчанию.



Порты *не должны* запускать и останавливать свои службы при установке и удалении. Не злоупотребляйте ключевыми словами `plist`, описанными в разделе `@preexec` `command,@postexec` `command,@preunexec` `command,@postunexec` `command`, выполняя команды, которые изменяют работающую систему, включая запуск или остановку служб.

### 6.28.1. Pre-Commit Checklist

Прежде чем внести порт с `rc.d` скриптом, и что более важно, перед его коммитом, пожалуйста, ознакомьтесь с этим контрольным списком, чтобы убедиться, что он готов.

Порт `devel/rclint` может проверить большинство из них, но он не заменяет тщательного просмотра и проверки.

1. Если это новый файл, имеет ли он расширение `.sh`? Если да, его необходимо изменить на просто `file.in`, поскольку файлы `rc.d` не могут оканчиваться таким расширением.
2. Совпадают ли имя файла (без `.in`), строка `PROVIDE` и ``${name}`? Совпадение имени файла с `'PROVIDE` упрощает отладку, особенно при проблемах с `rcorder(8)`. Совпадение имени файла и ``${name` облегчает понимание того, какие переменные актуальны в `rc.conf[.local]`. Это также является политикой для всех новых скриптов, включая те, что в базовой системе.
3. Установлена ли строка `REQUIRE` в значение `LOGIN`? Это обязательно для скриптов,

выполняемых от имени непривилегированного пользователя. Если скрипт выполняется от имени `root`, есть ли веская причина для его запуска до `LOGIN`? Если нет, он должен запускаться после, чтобы локальные скрипты можно было условно сгруппировать в `rcorder(8)` после запуска большинства компонентов базовой системы.

4. Запускает ли скрипт постоянную службу? Если да, он должен содержать `KEYWORD: shutdown`.
5. Убедитесь, что отсутствует `KEYWORD: FreeBSD`. Это перестало быть необходимым или желательным уже много лет. Это также указывает на то, что новый скрипт был скопирован/вставлен из старого скрипта, поэтому следует проявить дополнительную осторожность при проверке.
6. Если скрипт использует интерпретируемый язык, например `perl`, `python` или `ruby`, убедитесь, что `command_interpreter` установлен корректно. Например, для Perl добавьте `PERL=${PERL}` в `SUB_LIST` и используйте `%%PERL%`. В противном случае,

```
# service name stop
```

вероятно, не будет работать корректно. Дополнительную информацию смотрите в `service(8)`.

7. Проверено, что все вхождения `/usr/local` заменены на `%%PREFIX%%`?
8. Делаются ли присваивания переменным по умолчанию после `load_rc_config`?
9. Используются ли пустые строки при присвоении значений по умолчанию? Такие присвоения должны быть удалены, но перепроверьте, что эти параметры задокументированы в комментариях в начале файла.
10. Действительно ли в сценариях используются значения, присвоенные переменным?
11. Являются ли опции, перечисленные в стандартном `name`_flags``, обязательными? Если да, они должны быть в `command_args`. Флаг `-d` здесь, как красный флаг (простите за каламбур), так как обычно это опция для "демонизации" процесса и, следовательно, фактически обязательна.
12. `_name`_flags` никогда не должны включаться в `command_args` (и наоборот, хотя такая ошибка встречается реже).
13. Выполняет ли скрипт любой код безусловно? Это не приветствуется. Обычно такие вещи должны обрабатываться через `start_precmd`.
14. Все логические проверки должны использовать функцию `checkyesno`. Не допускаются самодельные проверки на `[Yy][Ee][Ss]` и т.п.
15. Если есть цикл (например, ожидание запуска чего-либо), есть ли в нём счётчик для завершения цикла? Мы не хотим, чтобы загрузка зависла навсегда в случае ошибки.
16. Создает ли скрипт файлы или каталоги, требующие определенных разрешений, например, `pid`, который должен принадлежать пользователю, запускающему процесс? Вместо традиционной последовательности `touch(1)/chown(8)/chmod(1)` рассмотрите использование `install(1)` с соответствующими аргументами командной строки, чтобы выполнить всю процедуру за один шаг.

## 6.29. Добавление пользователей и групп

Некоторые порты требуют наличия определённой учётной записи пользователя, обычно для демонов, работающих от имени этого пользователя. Для таких портов выберите *уникальный* UID в диапазоне от 50 до 999 и зарегистрируйте его в ports/UIDs (для пользователей) и ports/GIDs (для групп). Уникальный идентификатор должен быть одинаковым для пользователей и групп.

Пожалуйста, приложите патч для этих двух файлов, если требуется создать нового пользователя или группу для порта.

Затем используйте **USERS** и **GROUPS** в Makefile, и пользователь будет автоматически создан при установке порта.

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

Текущий список зарезервированных UID и GID можно найти в ports/UIDs и ports/GIDs.

## 6.30. Порты, зависящие от исходных кодов ядра

Некоторые порты (например, загружаемые модули ядра) требуют исходные файлы ядра для компиляции порта. Вот правильный способ проверить, установлены ли они у пользователя:

```
USES= kmod
```

Помимо этой проверки, функция **kmod** учитывает большинство аспектов, которые необходимо принимать во внимание данным портам.

## 6.31. Библиотеки Go

Порты не должны упаковывать или устанавливать библиотеки или исходный код Go. Порты Go должны загружать необходимые зависимости в обычное время загрузки и должны устанавливать только программы и то, что нужно пользователям, а не то, что нужно разработчикам на Go.

Порты должны (в порядке предпочтения):

- Использовать зависимости, включенные в исходный код пакета.
- Получить версии зависимостей, указанные вышестоящим проектом (в случае go.mod, vendor.json или аналогичных).
- В крайнем случае (зависимости не включены и версии не указаны точно) получить версии зависимостей, доступные на момент разработки/выпуска вышестоящего проекта.

## 6.32. Библиотеки Haskell

Как и в случае с языком Go, коллекция портов не должна включать или устанавливать библиотеки Haskell. Порты Haskell должны статически линковаться со своими зависимостями и загружать все распространяемые файлы на этапе fetch.

## 6.33. Файлы завершения командной оболочки

Многие современные оболочки (включая bash, fish, tcsh и zsh) поддерживают табуляцию для параметров и/или опций. Эта поддержка обычно обеспечивается файлами завершения, которые содержат определения того, как будет работать завершение по табуляции для определённой команды. Порты иногда поставляются со своими собственными файлами завершения, или разработчики портов могут создавать их самостоятельно.

Если доступны файлы завершения, их всегда следует устанавливать. Нет необходимости создавать для этого опцию. Однако если опция используется, всегда включайте её в `OPTIONS_DEFAULT`.

Таблица 51. Полные имена файлов завершения оболочки

<code>bash</code>	<code>\${PREFIX}/etc/bash_completion.d or \${PREFIX}/share/bash-completion/completions</code>	(любые уникальные имена файлов в одной из этих папок)
<code>fish</code>	<code>\${PREFIX}/share/fish/completions/\${PORTNAME}.fish</code>	
<code>zsh</code>	<code>\${PREFIX}/share/zsh/site-functions/_\${PORTNAME}</code>	

Не регистрируйте зависимости от самих оболочек.

# Глава 7. Флейворы

## 7.1. Введение в флейворы (Flavors)

Флейворы (Flavors) — это способ создания нескольких вариаций порта. Порт собирается несколько раз с различными вариациями.

Например, порт может иметь обычную версию с множеством функций и значительным количеством зависимостей, а также облегчённую "lite"-версию только с базовыми функциями и минимальными зависимостями.

Еще одним примером может быть порт с вариантом GTK и вариантом QT, в зависимости от используемого набора инструментов.

## 7.2. Использование FLAVORS

Чтобы объявить порт с несколькими флейворами, добавьте **FLAVORS** в его Makefile. Первый вариант в **FLAVORS** является вариантом по умолчанию.



Это может помочь упростить логику Makefile, также определив **FLAVOR** как:

```
FLAVOR?= ${FLAVORS:[1]}
```



Чтобы отличать флейворы от опций, которые всегда обозначаются заглавными буквами, названия флейворов могут содержать *только* строчные буквы, цифры и символ подчёркивания `_`.

### Пример 98. Основы использования флейворов

Если порт имеет "облегченный" подчиненный порт (lite slave port), подчиненный порт можно удалить, а порт преобразовать во флейворы с помощью:

```
FLAVORS= default lite
lite_PKGNAME_SUFFIX= -lite
[...]
.if ${FLAVOR:U} != lite
[enable non lite features]
.endif
```

### Пример 99. Еще один пример базового использования флейворов

Если порт имеет подчиненный порт `-nox11`, подчиненный порт можно удалить, а порт преобразовать в флейворы с помощью:

```

FLAVORS=    x11 nox11
FLAVOR?=   ${FLAVORS:[1]}
nox11_PKGNAME_SUFFIX= -nox11
[...]
.if ${FLAVOR} == x11
[enable x11 features]
.endif

```

*Пример 100. Использование флейворов в более сложных примерах*

Вот слегка отредактированный отрывок из того, что присутствует в пакете [devel/libpeas](#), порте, который использует [флейворы Python](#). При стандартных версиях Python 2 и 3, а именно 2.7 и 3.6, он автоматически получит `FLAVORS=py27 py36`

```

USES=      gnome python
USE_PYTHON= flavors

.if ${FLAVOR:U:py27:Mpy2*}
USE_GNOME= pygobject3

CONFIGURE_ARGS+=  --enable-python2 --disable-python3

BUILD_WRKSRС=   ${WRKSRС}/loaders/python
INSTALL_WRKSRС= ${WRKSRС}/loaders/python
.else # py3*
USE_GNOME+=  py3gobject3

CONFIGURE_ARGS+=  --disable-python2 --enable-python3 \
                 ac_cv_path_PYTHON3_CONFIG=${LOCALBASE}/bin/python${PYTHON_VER}-config

BUILD_WRKSRС=   ${WRKSRС}/loaders/python3
INSTALL_WRKSRС= ${WRKSRС}/loaders/python3
.endif

py34_PLIST= ${CURDIR}/pkg-plist-py3
py35_PLIST= ${CURDIR}/pkg-plist-py3
py36_PLIST= ${CURDIR}/pkg-plist-py3

```

Этот порт не использует `USE_PYTHON=distutils`, но всё равно требует флейворы Python. Чтобы избежать ошибки в `make(1)` из-за пустого значения `FLAVOR`, используйте `${FLAVOR:U}` в сравнениях строк вместо `${FLAVOR}`. Привязки Gnome Python `gobject3` имеют два разных названия: `pygobject3` для Python 2 и `py3gobject3` для Python 3. Скрипт `configure` должен выполняться в `${WRKSRС}`, но нас интересует только сборка и установка частей программного обеспечения для Python 2 или Python 3, поэтому установите базовые каталоги сборки и установки соответствующим образом. Подсказка о правильном пути к конфигурационному скрипту Python 3. Список упаковки отличается при сборке с Python 3. Поскольку есть три возможные версии

Python 3, установите **PLIST** для всех трёх с помощью [вспомогательные инструменты флейворов](#).

### 7.2.1. Вспомогательные инструменты для флейворов (Flavors Helpers)

Чтобы упростить написание Makefile, существуют несколько вспомогательных инструментов (помощников) флейворов.

Этот список помощников установит их переменную:

- `flavor_PKGNAMEPREFIX`
- `flavor_PKGNAME_SUFFIX`
- `flavor_PLIST`
- `flavor_DESCR`

Этот список помощников будет добавлен к их переменной:

- `flavor_CONFLICTS`
- `flavor_CONFLICTS_BUILD`
- `flavor_CONFLICTS_INSTALL`
- `flavor_PKG_DEPENDS`
- `flavor_EXTRACT_DEPENDS`
- `flavor_PATCH_DEPENDS`
- `flavor_FETCH_DEPENDS`
- `flavor_BUILD_DEPENDS`
- `flavor_LIB_DEPENDS`
- `flavor_RUN_DEPENDS`
- `flavor_TEST_DEPENDS`

*Пример 101. Специфичный для флейвора **PKGNAME***

Поскольку все пакеты должны иметь уникальные имена, флейворы должны изменять их, используя `flavor_PKGNAMEPREFIX` и `flavor_PKGNAME_SUFFIX`, что упрощает задачу:

```
FLAVORS=    normal lite
lite_PKGNAME_SUFFIX= -lite
```

## 7.3. **USES=php** и флейворы

При использовании `php` с одним из этих аргументов: `phpize`, `ext`, `zend` или `pecl`, порт автоматически получит заполненный параметр `FLAVORS` с версиями PHP, которые он

поддерживает.

Пример 102. Простое расширение `USES=php`

Это создаст пакет для всех поддерживаемых версий:

```
PORTNAME=  some-ext
PORTVERSION=  0.0.1
PKGNAMEPREFIX=  ${PHP_PKGNAMEPREFIX}

USES=      php:ext
```

Это создаст пакет для всех поддерживаемых версий, кроме 7.2:

```
PORTNAME=  some-ext
PORTVERSION=  0.0.1
PKGNAMEPREFIX=  ${PHP_PKGNAMEPREFIX}

USES=      php:ext
IGNORE_WITH_PHP=  72
```

### 7.3.1. Версии PHP с приложениями PHP

Приложения PHP также могут быть созданы с использованием флейворов.

Это позволяет создавать пакеты для всех версий PHP, чтобы пользователи могли использовать их с любой необходимой версией на своих серверах.



Приложения PHP, которые используют флейворы, *обязаны* добавлять `PHP_PKGNAME_SUFFIX` к именам своих пакетов.

Пример 103. Добавление флейворов в PHP-приложения

Добавление поддержки флейворов в PHP-приложение просто:

```
PKGNAME_SUFFIX=  ${PHP_PKGNAME_SUFFIX}

USES=      php:flavors
```



При добавлении зависимости к порту с вариантом PHP используйте `@${PHP_FLAVOR}`. *Никогда* не используйте `FLAVOR` напрямую.

## 7.4. USES=python и флейворы

При использовании `python` и `USE_PYTHON=distutils` порт автоматически получит заполненные `FLAVORS` с версиями Python, которые он поддерживает.

*Пример 104. Простой USES=python*

Предполагая, что поддерживаемые версии Python — 2.7, 3.4, 3.5 и 3.6, а версии Python 2 и 3 по умолчанию — 2.7 и 3.6, порт с параметрами:

```
USES= python
USE_PYTHON= distutils
```

получит следующие флейворы: `py27` и `py36`.

```
USES= python
USE_PYTHON= distutils allflavors
```

получит следующие флейворы: `py27`, `py34`, `py35` и `py36`.

*Пример 105. USES=python с требованиями к версии*

Предполагая, что поддерживаемые версии Python — 2.7, 3.4, 3.5 и 3.6, а версии Python 2 и 3 по умолчанию — 2.7 и 3.6, порт с параметрами:

```
USES= python:-3.5
USE_PYTHON= distutils
```

получит следующие флейвор: `py27`.

```
USES= python:-3.5
USE_PYTHON= distutils allflavors
```

получит следующие флейворы: `py27`, `py34` и `py35`.

```
USES= python:3.4+
USE_PYTHON= distutils
```

получит следующий флейвор: `py36`.

```
USES= python:3.4+
USE_PYTHON= distutils allflavors
```

получит следующие флейворы: `py34`, `py35` и `py36`.

`PY_FLAVOR` доступен для указания правильной версии модулей Python. Все зависимости от вариантов портов Python должны использовать `PY_FLAVOR`, а не `FLAVOR` напрямую.

*Пример 106. Для порта, не использующего `distutils`*

Если версия Python 3 по умолчанию — 3.6, следующая команда установит `PY_FLAVOR` в значение `py36`:

```
RUN_DEPENDS=    ${PYTHON_PKGNAMEPREFIX}mutagen>0:audio/py-mutagen@${PY_FLAVOR}
USES=          python:3.5+
```

## 7.5. `USES=lua` и флейворы

При использовании `lua:module` или `lua:flavors` порт автоматически получит заполненный параметр `FLAVORS` с версиями Lua, которые он поддерживает. Однако предполагается, что обычные приложения (а не модули Lua) не должны использовать эту возможность; большинству приложений, которые встраивают или иным образом используют Lua, следует просто указывать `USES=lua`.

`LUA_FLAVOR` доступен (и должен использоваться) для зависимости от правильной версии зависимостей, независимо от того, использовал ли порт параметры `flavors` или `module`.

См. [Использование Lua](#) для получения дополнительной информации.

## 7.6. `USES=guile` и флейворы

При использовании `guile:flavors` порт автоматически получит заполненное поле `FLAVORS` с версиями Guile, которые он поддерживает. Однако не предполагается, что обычные приложения должны использовать эту возможность; она в первую очередь предназначена для библиотек и расширений, таких как `guile-lib` или `guile-cairo`.

`GUILE_FLAVOR` доступен (и должен использоваться) для зависимости от правильной версии зависимостей с флейворами, независимо от того, использовал ли порт параметр `flavors` или нет.

См. [Использование Guile](#) для получения дополнительной информации.

# Глава 8. Продвинутые практики pkg-plist

## 8.1. Изменение содержимого pkg-plist в зависимости от make-переменных

Некоторые порты, в частности, порты `p5-`, должны менять содержимое своих файлов `pkg-plist` в зависимости от того, с какими параметрами они были отконфигурированы (или в зависимости от версии языка `perl` в случае портов `p5-`). Чтобы облегчить этот процесс, любые вхождения ключевых слов `%%OSREL%%`, `%%PERL_VER%%` и `%%PERL_VERSION%%` в файле `pkg-plist` будут заменяться соответствующими значениями. Значением `%%OSREL%%` является номер версии операционной системы (например, `4.9`). `%%PERL_VERSION%%` и `%%PERL_VER%%` обозначают полный номер версии `perl` (например, `5.8.9`). Некоторые другие `%%VARS%%`, имеющие отношение к файлам документации порта, описаны в [соответствующем разделе](#).

Если вам нужно сделать другие подстановки, вы можете указать в переменной `PLIST_SUB` список пар `VAR=VALUE`, и все вхождения `%%VAR%%` в файле `pkg-plist` будут заменяться на значение `VALUE`.

Например, если порт устанавливает множество файлов в подкаталоге, зависящем от версии, используйте заполнитель для версии, чтобы файл `pkg-plist` не требовал регенерации при каждом обновлении порта. Например, укажите:

```
OCTAVE_VERSION= ${PORTREVISION}
PLIST_SUB= OCTAVE_VERSION=${OCTAVE_VERSION}
```

в файле `Makefile` и использовать `%%OCTAVE_VERSION%%` везде, где нужно указать номер версии в файле `pkg-plist`. Таким образом, при обновлении порта вам не нужно будет менять десятки (а в некоторых случаях и сотни) строк в файле `pkg-plist`.

Если файлы устанавливаются по условию в зависимости от опций, установленных в порте, обычный способ обработки — это добавление префикса `%%OPT%%` для строк в `pkg-plist`, которые нужны при включении опции, или `%%NO_OPT%%`, когда опция отключена, а также добавление `OPTIONS_SUB=yes` в `Makefile`. Подробнее см. `OPTIONS_SUB`.

Например, если есть файлы, которые устанавливаются только при включении опции `X11`, и в `Makefile` указано:

```
OPTIONS_DEFINE= X11
OPTIONS_SUB= yes
```

В `pkg-plist` укажите `%%X11%%` перед строками, которые устанавливаются только при включении опции, например:

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

Эта подстановка будет сделана между выполнением целей `pre-install` и `do-install`, посредством чтения файла `PLIST` и записью в файл `TMPPLIST` (по умолчанию это файл `WRKDIR/.PLIST.mktmp`). Так что если ваш порт строит `PLIST` на лету, делайте это во время или до выполнения цели `pre-install`. Кроме того, если вашему порту требуется отредактировать получающийся файл, делайте это в цели `post-install` изменением файла `TMPPLIST`.

Ещё один способ изменения списка упаковки порта основан на установке переменных `PLIST_FILES` и `PLIST_DIRS`. Значение каждой переменной рассматривается как список путей для записи в `TMPPLIST` вместе с содержимым `PLIST`. Хотя имена, перечисленные в `PLIST_FILES` и `PLIST_DIRS`, подлежат замене `%%VAR%%`, как описано выше, лучше использовать `${VAR}` напрямую. За исключением этого, имена из `PLIST_FILES` появятся в итоговом списке упаковки без изменений, тогда как к именам из `PLIST_DIRS` будет добавлен префикс `@dir`. Чтобы вступить в силу, `PLIST_FILES` и `PLIST_DIRS` должны быть установлены до записи `TMPPLIST`, то есть в `pre-install` или ранее.

Время от времени использования `OPTIONS_SUB` недостаточно. В таких случаях добавление специфичного `TAG` в `PLIST_SUB` внутри `Makefile` со специальным значением `@comment` заставляет инструменты пакетирования игнорировать строку. Например, если некоторые файлы устанавливаются только при включённой опции `X11` и архитектуре `i386`:

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

.if ${PORT_OPTIONS:MX11} && ${ARCH} == "i386"
PLIST_SUB+= X11I386=""
.else
PLIST_SUB+= X11I386="@comment "
.endif
```

## 8.2. Пустые каталоги

### 8.2.1. Очистка пустых каталогов

При удалении порт должен удалить пустые каталоги, которые он создал. Большинство этих каталогов автоматически удаляются с помощью `pkg(8)`, но для каталогов, созданных вне `${PREFIX}`, или пустых каталогов требуется дополнительная работа. Обычно это делается добавлением строк `@dir` для таких каталогов. Подкаталоги должны быть удалены до удаления родительских каталогов.

```
[...]
@dir /var/games/oneko/saved-games
@dir /var/games/oneko
```

### 8.2.2. Создание пустых каталогов

Пустые каталоги, созданные во время установки порта, требуют особого внимания. Они должны присутствовать при создании пакета. Если они не созданы кодом порта, создайте

их в Makefile:

```
post-install:
    ${MKDIR} ${STAGEDIR}${PREFIX}/some/directory
```

Добавьте директорию в pkg-plist так же, как и любую другую. Например:

```
@dir some/directory
```

## 8.3. Файлы конфигурации

Если порт устанавливает файлы конфигурации в PREFIX/etc (или в другое место), *не* указывайте их в pkg-plist. Это приведёт к тому, что **pkg delete** удалит файлы, которые были тщательно отредактированы пользователем, а повторная установка перезапишет их.

Вместо этого устанавливайте образцы файлов с расширением filename.sample. Макрос **@sample** автоматизирует этот процесс; подробности его работы см. в разделе [Расширение списка пакетов с помощью ключевых слов](#). Для каждого образца файла добавьте строку в pkg-plist:

```
@sample etc/orbit.conf.sample
```

Если существует очень веская причина не устанавливать рабочий файл конфигурации по умолчанию, укажите только имя примера файла в pkg-plist, без части **@sample** с последующим пробелом, и добавьте [сообщение](#), указывающее, что пользователь должен скопировать и отредактировать файл перед тем, как программа заработает.



Когда порт устанавливает свою конфигурацию в подкаталоге \${PREFIX}/etc, используйте **ETCDIR**, который по умолчанию равен **\${PREFIX}/etc/\${PORTNAME}**. Это значение может быть переопределено в Makefile порта, если для порта принято использовать другой каталог. Макрос **%%ETCDIR%%** будет использоваться вместо этого в pkg-plist.

Примеры конфигурационных файлов всегда должны иметь суффикс .sample. Если по каким-то историческим причинам использование стандартного суффикса невозможно, или если примеры файлов взяты из другого каталога, используйте эту конструкцию:



```
@sample etc/orbit.conf-dist etc/orbit.conf
```

или

```
@sample %%EXAMPLESDIR%%/orbit.conf etc/orbit.conf
```

## 8.4. Динамический или статический список упаковки

*Статический список упаковки* — это список упаковки, который доступен в Коллекции Портов или как файл pkg-plist (с подстановкой переменных или без неё), или как встроенный в Makefile через `PLIST_FILES` и `PLIST_DIRS`. Даже если содержимое было автоматически сгенерировано инструментом или целью в Makefile до включения в Коллекцию портов коммиттером (например, с использованием `make makeplist`), такой список всё равно считается статическим, поскольку его можно посмотреть без необходимости загрузки или компиляции distfile.

*Динамический список упаковки* — это список упаковки, который генерируется во время компиляции порта на основе установленных файлов и каталогов. Невозможно изучить его до загрузки и компиляции исходного кода портированного приложения или после выполнения команды `make clean`.

Хотя использование динамических списков упаковки не запрещено, сопровождающие должны использовать статические списки упаковки везде, где это возможно, поскольку это позволяет пользователям выполнять `grep(1)` по доступным портам для обнаружения, например, какой порт устанавливает определенный файл. Динамические списки должны быть использованы в основном для сложных портов, для которых изменения в списке упаковки кардинальным образом основаны на возможностях порта, настраиваемых параметрами, (и, таким образом, делая сопровождение статических списков упаковки невозможным), или портов, которые изменяют список упаковки на основе версии используемого им программного обеспечения (например, порты, которые порождают документы при помощи Javadoc).

## 8.5. Автоматическое создание списка упаковки

Сначала убедитесь, что порт почти готов, и отсутствует только файл pkg-plist. Запуск команды `make makeplist` покажет пример для файла pkg-plist. Вывод `makeplist` необходимо дважды перепроверять на корректность, так как он пытается автоматически угадать некоторые вещи и может ошибаться.

Файлы конфигурации пользователя должны устанавливаться как filename.sample, как описано в разделе [Файлы конфигурации](#). info/dir не должен быть указан, а соответствующие строки install-info должны быть добавлены, как указано в разделе [info-файлы](#). Любые библиотеки, устанавливаемые портом, должны быть перечислены, как указано в разделе [общие библиотеки](#).

### 8.5.1. Расширение `PLIST_SUB` с помощью регулярных выражений

Строки, которые нужно заменить, иногда должны быть очень конкретными, чтобы избежать нежелательных замен. Это распространённая проблема с короткими значениями.

Для решения этой проблемы для каждого `PLACEHOLDER=значение` можно задать `PLACEHOLDER_regex=регулярное_выражение`, где часть `regex` более точно соответствует значению.

*Пример 107. Использование `PLIST_SUB` с регулярными выражениями*

Порты Perl могут устанавливать архитектурно-зависимые файлы в специальное дерево. В FreeBSD для упрощения портирования это дерево называется `mach`. Например, порт, который устанавливает файл, чей путь содержит `mach`, может иметь эту часть строки пути заменённой неправильными значениями. Рассмотрим этот Makefile:

```
PORTNAME= Machine-Build
DISTVERSION= 1
CATEGORIES= devel perl5
MASTER_SITES= CPAN
PKGNAMEPREFIX= p5-

MAINTAINER= perl@FreeBSD.org
COMMENT= Building machine
WWW= https://search.cpan.org/dist/Machine-Build

USES= perl5
USE_PERL5= configure

PLIST_SUB= PERL_ARCH=mach
```

Файлы, установленные портом:

```
/usr/local/bin/machine-build
/usr/local/lib/perl5/site_perl/man/man1/machine-build.1.gz
/usr/local/lib/perl5/site_perl/man/man3/Machine::Build.3.gz
/usr/local/lib/perl5/site_perl/Machine/Build.pm
/usr/local/lib/perl5/site_perl/mach/5.20/Machine/Build/Build.so
```

Запуск `make makeplist` ошибочно создает:

```
bin/%%PERL_ARCH%%ine-build
%%PERL5_MAN1%%/%%PERL_ARCH%%ine-build.1.gz
%%PERL5_MAN3%%/Machine::Build.3.gz
%%SITE_PERL%%/Machine/Build.pm
%%SITE_PERL%%/%%PERL_ARCH%%/%%PERL_VER%%/Machine/Build/Build.so
```

Измените строку `PLIST_SUB` в Makefile на:

```
PLIST_SUB= PERL_ARCH=mach \
            PERL_ARCH_regex=\bmach\b
```

Теперь `make makeplist` правильно генерирует:

```
bin/machine-build
%%PERL5_MAN1%/machine-build.1.gz
%%PERL5_MAN3%/Machine::Build.3.gz
%%SITE_PERL%/Machine/Build.pm
%%SITE_PERL%/%%PERL_ARCH%/%%PERL_VER%/Machine/Build/Build.so
```

## 8.6. Расширение списка пакетов используя ключевые слова

Все ключевые слова также могут принимать необязательные аргументы в скобках. Аргументами являются владелец, группа и режим доступа. Этот аргумент применяется к файлу или каталогу, на который ссылаются. Чтобы изменить владельца, группу и режим доступа конфигурационного файла, используйте:

```
@sample(games,games,640) etc/config.sample
```

Аргументы являются необязательными. Если необходимо изменить только группу и режим, используйте:

```
@sample(,games,660) etc/config.sample
```

Если ключевое слово используется в [необязательной записи](#), оно должно быть добавлено после помощника:



```
%%F00%%@sample etc/orbit.conf.sample
```

Это происходит потому, что вспомогательные функции `plist` для опций используются для комментирования строки, поэтому они должны быть указаны первыми. Дополнительную информацию см. в [OPTIONS\\_SUB](#).

### 8.6.1. @desktop-file-utils

Будет выполнять `update-desktop-database -q` после установки и удаления. *Никогда* не используйте напрямую, добавьте `USES=desktop-file-utils` в Makefile.

### 8.6.2. @fc каталог

Добавить запись `@dir` для каталога, переданного в качестве аргумента, и выполнить `fc-cache -fs` для этого каталога после установки и удаления.

### 8.6.3. **@fontsdir** каталог

Добавить запись **@dir** для каталога, переданного в качестве аргумента, и запустить **mkfontscale** и **mkfontdir** в этом каталоге после установки и удаления. Кроме того, при удалении удаляются кэш-файлы **fonts.scale** и **fonts.dir**, если они пусты.

### 8.6.4. **@info** файл

Добавляет файл, переданный в качестве аргумента, в **plist** и обновляет индекс документа **info** при установке и удалении. Кроме того, удаляет индекс, если он пуст, при удалении. Это никогда не следует использовать вручную, а только через **INFO**. Подробнее см. в [Файлы Info](#).

### 8.6.5. **@kld** каталог

Выполняет **kldxref** для каталога при установке и удалении. Дополнительно при удалении каталог будет удалён, если он пуст.

### 8.6.6. **@rmtry** файл

Удаляет файл при удалении и не выдает ошибку, если файл отсутствует.

### 8.6.7. **@sample** файл [файл]

Это используется для обработки установки файлов конфигурации, используя примеры файлов, поставляемых с пакетом. "Реальный" файл (не пример) — это либо второе имя файла, если оно присутствует, либо первое имя файла без расширения **.sample**.

Это делает три вещи. Во-первых, добавляет первый переданный файл в качестве аргумента, образец файла, в **plist**. Затем, при установке, если фактический файл не найден, копирует образец файла в фактический файл. И наконец, при удалении, удаляет фактический файл, если он не был изменён. Дополнительную информацию см. в [Файлы конфигурации](#).

### 8.6.8. **@shared-mime-info** каталог

Выполняет **update-mime-database** для указанного каталога при установке и удалении.

### 8.6.9. **@shell** файл

Добавить файл, переданный в качестве аргумента, в **plist**.

При установке добавить полный путь к **file** в **/etc/shells**, убедившись, что он не добавлен повторно. При удалении удалите его из **/etc/shells**.

### 8.6.10. **@terminfo**

Не использовать самостоятельно. Если порт устанавливает файлы **\*.terminfo**, добавьте **USES=terminfo** в его **Makefile**.

При установке и удалении, если присутствует **tic**, обновить

`${PREFIX}/shared/misc/terminfo.db` из файлов `*.terminfo` в `${PREFIX}/shared/misc`.

## 8.6.11. Основные ключевые слова

Есть несколько ключевых слов, которые жестко закодированы и документированы в [pkg-create\(8\)](#). Для полноты изложения они также документированы здесь.

### 8.6.11.1. @ [файл]

Ключевое слово `empty` является заполнителем, используемым, когда необходимо изменить владельца, группу или права доступа к файлу. Например, чтобы установить группу файла в `games` и добавить бит `setgid`, добавьте:

```
@(,games,2755) sbin/daemon
```

### 8.6.11.2. @preexec команда, @postexec команда, @preunexec команда, @postunexec команда

Выполнить *команду* как часть процесса установки или удаления пакета.

#### @preexec команда

Выполнить *команду* как часть скриптов `pre-install`.

#### @postexec команда

Выполнить *команду* как часть скриптов `post-install`.

#### @preunexec команда

Выполнить *команду* как часть скриптов `pre-deinstall`.

#### @postunexec команда

Выполнить *команду* как часть скриптов `post-deinstall`.

Если в *команде* содержится любая из этих последовательностей, они раскрываются непосредственно. Для следующих примеров предположим, что `@cwd` установлен в `/usr/local`, а последним извлечённым файлом был `bin/emacs`.

#### %F

Раскрывается до последнего извлеченного имени файла (как указано). В примере `bin/emacs`.

#### %D

Раскрыть до текущего префикса директории, установленного с помощью `@cwd`. В этом примере `/usr/local`.

#### %B

Раскрыть до базового имени полного имени файла, то есть префикс текущего каталога плюс последняя спецификация файла, за вычетом имени файла в конце спецификации. В данном примере это будет `/usr/local/bin`.

**%f**

Раскрывается до части имени файла в полном квалифицированном имени, или противоположный случай для **%B**. В примере, `emacs`.



Эти ключевые слова предназначены для помощи в настройке пакета, чтобы он был максимально готов к использованию. Они *не должны* использоваться для запуска служб, остановки служб или выполнения любых других команд, которые изменяют текущую работающую систему.

### 8.6.11.3. **@mode** режим

Установить разрешения по умолчанию для всех последующих извлекаемых файлов в *режим*. Формат такой же, как используется в `chmod(1)`. Использование без аргумента вернёт разрешения по умолчанию (режим файла при упаковке).



Это должен быть числовой режим, например `644`, `4755` или `600`. Нельзя использовать относительный режим, например `u+s`.

### 8.6.11.4. **@owner** пользователь

Установить владельца по умолчанию для всех последующих файлов в *пользователь*. Использование без аргумента вернёт владельца по умолчанию (`root`).

### 8.6.11.5. **@group** группа

Установить группу-владельца по умолчанию для всех последующих файлов в *группу*. Использование без аргумента вернёт группу-владельца по умолчанию (`wheel`).

### 8.6.11.6. **@comment** строка

Эта строка игнорируется при упаковке.

### 8.6.11.7. **@dir** каталог

Объявить имя каталога. По умолчанию каталоги, созданные в `PREFIX` при установке пакета, автоматически удаляются. Используйте эту опцию, если необходимо создать пустой каталог в `PREFIX` или если каталогу требуется нестандартный владелец, группа или права. Каталоги за пределами `PREFIX` необходимо регистрировать. Например, `/var/db/${PORTNAME}` требует записи `@dir`, тогда как `${PREFIX}/shared/${PORTNAME}` — нет, если он содержит файлы или использует стандартные владельца, группу и права.

### 8.6.11.8. **@exec** команда, **@unexec** команда (Устарело)

Выполнить *команду* как часть процесса установки или удаления. Рекомендуется использовать `@rgeexec` *команда* вместо этого.

### 8.6.11.9. **@dirrm** каталог (Устарело)

Объявить имя каталога для удаления при деинсталляции. По умолчанию каталоги,

созданные в **PREFIX** при установке пакета, удаляются при его деинсталляции.

#### 8.6.11.10. **@dirrmtry** каталог (Устарело)

Объявить имя каталога для удаления, аналогично **@dirrm**, но не выводить предупреждение, если каталог не может быть удален.

### 8.6.12. Создание новых ключевых слов

Файлы списка пакетов могут быть расширены ключевыми словами, которые определены в каталоге `${PORTSDIR}/Keywords`. Настройки каждого ключевого слова хранятся в файле UCL с именем `keyword.ucl`. Файл должен содержать как минимум один из следующих разделов:

- **attributes**
- **action**
- **pre-install**
- **post-install**
- **pre-deinstall**
- **post-deinstall**
- **pre-upgrade**
- **post-upgrade**

#### 8.6.12.1. **attributes**

Изменяет владельца, группу или режим доступа, используемые ключевым словом. Содержит ассоциативный массив, в котором возможными ключами являются **owner**, **group** и **mode**. Значениями являются, соответственно, имя пользователя, имя группы и режим доступа к файлу. Например:

```
attributes: { owner: "games", group: "games", mode: 0555 }
```

#### 8.6.12.2. **action**

Определяет, что происходит с параметром ключевого слова. Содержит массив, где возможные значения:

##### **setprefix**

Установите префикс для следующих записей в `plist`.

##### **dir**

Зарегистрировать каталог для создания при установке и удаления при деинсталляции.

##### **dirrm**

Зарегистрировать каталог для удаления при деинсталляции. Устарело.

### **dirmtry**

Зарегистрировать каталог для попытки удаления при деинсталляции. Устарело.

### **file**

Зарегистрировать файл.

### **setmode**

Установить режим для следующих записей plist.

### **setowner**

Установить владельца для следующих записей plist.

### **setgroup**

Установить группу для следующих записей в plist.

### **comment**

Не выполняет никаких действий, эквивалентно отсутствию раздела **action**.

### **ignore\_next**

Игнорировать следующую запись в plist.

#### **8.6.12.3. arguments**

Если установлено значение **true**, добавляется обработка аргументов, разделяя всю строку, **%@**, на нумерованные аргументы, **%1**, **%2**, и так далее. Например, для такой строки:

```
@foo some.content other.content
```

**%1** и **%2** будут содержать:

```
some.content  
other.content
```

Это также влияет на работу записи **action**. Если аргументов больше одного, необходимо указать номер аргумента. Например:

```
actions: [file(1)]
```

#### **8.6.12.4. pre-install, post-install, pre-deinstall, post-deinstall, pre-upgrade, post-upgrade**

Эти ключевые слова содержат **sh(1)** скрипт, который выполняется до или после установки, удаления или обновления пакета. В дополнение к обычным заполнителям **@exec %foo**, описанным в **@preexec команда**, существует новый заполнитель **%@**, который представляет аргумент ключевого слова.

### 8.6.12.5. Примеры пользовательских ключевых слов

Пример 108. Пример ключевого слова `@dirrmtryecho`

Это ключевое слово выполняет две функции: добавляет строку `@dirrmtry directory` в список упаковки и сообщает о том, что директория удаляется при деинсталляции пакета.

```
actions: [dirrmtry]
post-deinstall: <<EOD
    echo "Directory %D/%@ removed."
EOD
```

Пример 109. Реальный пример, как реализован `@sample`

Этот ключевое слово выполняет три действия. Оно добавляет первый *filename*, переданный в качестве аргумента `@sample`, в список упаковки, добавляет в скрипт `post-install` инструкции для копирования образца в фактический файл конфигурации, если он ещё не существует, и добавляет в инструкции `post-deinstall` удаление файла конфигурации, если он не был изменён.

```
actions: [file(1)]
arguments: true
post-install: <<EOD
    case "%1" in
        /*) sample_file="%1" ;;
        *) sample_file="%D/%1" ;;
    esac
    target_file="${sample_file%.sample}"
    set -- %@
    if [ $# -eq 2 ]; then
        target_file=${2}
    fi
    case "${target_file}" in
        /*) target_file="${target_file}" ;;
        *) target_file="%D/${target_file}" ;;
    esac
    if ! [ -f "${target_file}" ]; then
        /bin/cp -p "${sample_file}" "${target_file}" && \
        /bin/chmod u+w "${target_file}"
    fi
EOD
pre-deinstall: <<EOD
    case "%1" in
        /*) sample_file="%1" ;;
        *) sample_file="%D/%1" ;;
    esac
    target_file="${sample_file%.sample}"
```

```
set -- %@
if [ $# -eq 2 ]; then
    set -- %@
    target_file=${2}
fi
case "${target_file}" in
/*) target_file="${target_file}" ;;
*) target_file="%D/${target_file}" ;;
esac
if cmp -s "${target_file}" "${sample_file}"; then
    rm -f "${target_file}"
else
    echo "You may need to manually remove ${target_file} if it is no longer
needed."
fi
EOD
```

# Глава 9. pkg-\*

Есть несколько приёмов работы с файлами pkg-\*, которые мы ещё не описали, но они иногда могут быть очень кстати.

## 9.1. pkg-message

Если вам нужно вывести сообщение для человека, устанавливающего приложение, то вы можете поместить сообщение в файл pkg-message. Эта возможность часто оказывается полезной для вывода дополнительных шагов установки, которые нужно предпринять после выполнения команды `pkg install`, или для вывода информации о лицензировании.



- Файл pkg-message должен содержать только информацию, *критически важную* для настройки и работы в FreeBSD, а также уникальную для данного порта.
- Информация по настройке должна отображаться только при первоначальной установке. Инструкции по обновлению должны показываться только при обновлении с соответствующей версии.
- Не обрамляйте сообщения ни пробелами, ни строками символов (такими как `-----`, или `=====`). Оставьте форматирование `pkg(8)`.
- Коммиттеры имеют полное право ограничивать существующие сообщения диапазонами установки или обновления, используя спецификации формата UCL.
- Пожалуйста, убедитесь, что используете соответствующие инструменты для управления службами.
  - Используйте `service имя start` для запуска службы вместо `/usr/local/etc/rc.d/имя start`
  - Используйте `sysrc name_enable=YES` для изменения параметров в `rc.conf`

pkg-message поддерживает два формата:

### raw

Обычный текстовый файл. Его сообщение отображается только при установке.

### UCL

Если файл начинается с символа “[”, то он считается файлом в формате UCL. Формат UCL описан на странице [libucl's GitHub page](#).



Не добавляйте запись для pkg-message в pkg-plist.

### 9.1.1. UCL в pkg-message

Формат следующий. Это должен быть массив объектов. Сами объекты могут содержать следующие ключевые слова:

## message

Отображаемое сообщение. Этот ключевой параметр является обязательным.

## type

Когда сообщение должно быть отображено.

## maximum\_version

Только если `type` имеет значение `upgrade`. Отображается, если обновление выполняется с версии строго ниже указанной.

## minimum\_version

Только если `type` имеет значение `upgrade`. Отображается, если обновление выполняется с версии, строго большей, чем указанная.

Ключевые слова `maximum_version` и `minimum_version` можно комбинировать.

Ключевое слово `type` может иметь три значения:

## install

Сообщение должно отображаться только при установке пакета.

## remove

Сообщение должно отображаться только при удалении пакета.

## upgrade

сообщение должно отображаться только во время обновления пакета.



Для сохранения совместимости с файлами `pkg-message`, не использующими UCL, первая строка UCL `pkg-message` *ДОЛЖНА* быть одиночным символом “[”, а последняя строка *ДОЛЖНА* быть одиночным символом “]”.

### Пример 110. Короткие строки UCL

Сообщение ограничено двойными кавычками “”, это используется для простых однострочных строк:

```
[
{ type: install
  message: "Simple message"
}
]
```

### Пример 111. Многострочные строки UCL

Многострочные строки используют стандартную нотацию `heredoc`. Разделитель многострочной строки *должен* начинаться сразу после символов `<<` без пробелов и *должен* состоять только из заглавных букв. Чтобы завершить многострочную строку,

добавьте строку-разделитель на отдельной строке без пробелов. Сообщение из раздела [Короткие строки UCL](#) может быть записано как:

```
[
{ type: install
  message: <<EOM
Simple message
EOM
}
]
```

*Пример 112. Показать сообщение при установке/удалении*

Когда сообщение нужно отображать только при установке или удалении, укажите тип:

```
[
{
  type: remove
  message: "package being removed."
}
{ type: install, message: "package being installed."}
]
```

*Пример 113. Показать сообщение при обновлении*

При обновлении порта отображаемое сообщение может быть ещё более адаптировано к потребностям порта.

```
[
{
  type: upgrade
  message: "Package is being upgraded."
}
{
  type: upgrade
  maximum_version: "1.0"
  message: "Upgrading from before 1.0 need to do this."
}
{
  type: upgrade
  minimum_version: "1.0"
  message: "Upgrading from after 1.0 should do that."
}
{
  type: upgrade
  maximum_version: "3.0"
```

```
minimum_version: "1.0"  
message: "Upgrading from > 1.0 and < 3.0 remove that file."  
}  
]
```

При отображении сообщения во время обновления важно ограничить случаи, когда оно показывается пользователю. Чаще всего это делается с помощью `maximum_version`, чтобы ограничить его использование обновлениями до определенной версии, когда требуется выполнить конкретное действие.

## 9.2. pkg-install, pkg-pre-install и pkg-post-install

Если порту необходимо выполнять команды при установке бинарного пакета с помощью `pkg add` или `pkg install`, используйте `pkg-install`. Он запускается дважды через `pkg`: первый раз как `${SH} pkg-install ${PKGNAME} PRE-INSTALL` перед установкой пакета и второй раз как `${SH} pkg-install ${PKGNAME} POST-INSTALL` после его установки. Переменная `$2` может быть проверена, чтобы определить, в каком режиме выполняется скрипт. Переменная окружения `PKG_PREFIX` устанавливается равной имени каталога установки пакета.

Если используется `pkg-pre-install` или `pkg-post-install`, скрипт выполняется только один раз (до или после установки пакета), с единственным аргументом `${PKGNAME}`. Использование `pkg-pre-install.lua` или `pkg-post-install.lua` запускает скрипт на Lua вместо shell-скрипта. Скрипты на Lua, выполняемые `pkg`, предоставляют некоторые расширения и несколько ограничений, которые описаны в [pkg-lua-script\(5\)](#).



Использование `pkg-pre-install` (или `pkg-pre-install.lua`) и `pkg-post-install` (или `pkg-post-install.lua`) предпочтительнее, чем использование `pkg-install`.

Эти скрипты автоматически добавляются в список упаковки.



Эти скрипты предназначены для упрощения настройки пакетов после установки. Они *не должны* использоваться для запуска служб, остановки служб или выполнения любых других команд, которые изменяют текущую работающую систему.

## 9.3. pkg-deinstall, pkg-pre-deinstall и pkg-post-deinstall

Эти скрипты выполняются при удалении пакета.

Скрипт `pkg-deinstall` выполняется дважды командой `pkg delete`. Первый раз как `${SH} pkg-deinstall ${PKGNAME} DEINSTALL` до удаления порта и второй раз как `${SH} pkg-deinstall ${PKGNAME} POST-DEINSTALL` после удаления порта. Переменная `$2` может быть проверена для определения режима, в котором выполняется скрипт. Переменная окружения `PKG_PREFIX`

устанавливается в каталог установки пакета.

Если используется `pkg-pre-deinstall` или `pkg-post-deinstall`, скрипт выполняется только один раз (до или после удаления пакета) с единственным аргументом `${PKGNAME}`. Использование `pkg-pre-deinstall.lua` или `pkg-post-deinstall.lua` запустит скрипт на Lua вместо shell-скрипта. Скрипты на Lua, выполняемые `pkg`, предоставляют некоторые расширения и ограничения, которые описаны в [pkg-lua-script\(5\)](#).



Использование `pkg-pre-deinstall` (или `pkg-pre-deinstall.lua`) и `pkg-post-deinstall` (или `pkg-post-deinstall.lua`) предпочтительнее, чем использование `pkg-deinstall`.

Эти скрипты автоматически добавляются в список упаковки.



Эти скрипты предназначены для упрощения очистки после удаления пакетов. Они *не должны* использоваться для запуска служб, остановки служб или выполнения любых других команд, которые изменяют текущую работающую систему.

## 9.4. Изменение имён файлов `pkg-*`

Все имена файлов `pkg-*` определяются с помощью переменных, так что вы можете изменить их, если это нужно, в вашем файле Makefile. Это особенно полезно, если вы используете одни и те же файлы `pkg-*` совместно между несколькими портами или пишете в один из вышеперечисленных файлов (в главе о [записи в каталоги, отличные от WRKDIR](#) объяснено, почему не рекомендуется осуществлять запись непосредственно в файлы `pkg-*`).

Вот список имён переменных и их значений по умолчанию. (Значение `PKGDIR` по умолчанию равно `${MASTERDIR}`.)

Переменная	Значение по умолчанию
<code>DESCR</code>	<code>\${PKGDIR}/pkg-descr</code>
<code>PLIST</code>	<code>\${PKGDIR}/pkg-plist</code>
<code>PKGINSTALL</code>	<code>\${PKGDIR}/pkg-install</code>
<code>PKGPREINSTALL</code>	<code>\${PKGDIR}/pkg-pre-install</code>
<code>PKGPOSTINSTALL</code>	<code>\${PKGDIR}/pkg-post-install</code>
<code>PKGDEINSTALL</code>	<code>\${PKGDIR}/pkg-deinstall</code>
<code>PKGPREDEINSTALL</code>	<code>\${PKGDIR}/pkg-pre-deinstall</code>
<code>PKGPOSTDEINSTALL</code>	<code>\${PKGDIR}/pkg-post-deinstall</code>
<code>PKGMESSAGE</code>	<code>\${PKGDIR}/pkg-message</code>

## 9.5. Использование `SUB_FILES` и `SUB_LIST`

Переменные `SUB_FILES` и `SUB_LIST` подходят для задания в файлах порта динамических значений, таких как `PREFIX` установки в `pkg-message`.

В переменной `SUB_FILES` указывается перечень файлов для автоматического изменения. Каждый *file* из перечня `SUB_FILES` обязан иметь соответствующий `file.in`, присутствующий в `FILESDIR`. Измененная версия будет создана в `WRKDIR`. Файлы, определенные в качестве значения `USE_RC_SUBR` (или устаревшего `USE_RECORDER`), автоматически добавляются в `SUB_FILES`. Для файлов `pkg-message`, `pkg-install` и `pkg-deinstall` устанавливается соответствующая переменная `Makefile`, указывающая на обработанную версию.

Переменная `SUB_LIST` содержит перечень пар `VAR=VALUE`. В каждом файле из `SUB_FILES` для каждой пары будет произведена замена `%%VAR%%` на `VALUE`. Некоторые общие пары определяются автоматически: `PREFIX`, `LOCALBASE`, `DATADIR`, `DOCSDIR`, `EXAMPLESDIR`, `WWWDIR` и `ETCDIR`. Любая строка, начинающаяся с `@comment`, будет удалена из конечного файла после подстановки переменной.

В следующем примере в `pkg-message` будет сделана замена `%%ARCH%%` на системную архитектуру:

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

Обратите внимание, что в этом примере в `FILESDIR` обязательно существование файла `pkg-message.in`.

Пример хорошего `pkg-message.in`:

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

# Глава 10. Тестирование порта

## 10.1. Запуск `make describe`

Некоторые утилиты FreeBSD для сопровождения портов, например, `portupgrade(1)`, опираются на базу данных с именем `/usr/ports/INDEX`, в которой отслеживаются такие характеристики портов, как их зависимости. Файл `INDEX` создаётся при помощи `ports/Makefile` верхнего уровня по команде `make index`, спускающейся в подкаталог каждого порта и выполняющей в нём `make describe`. Таким образом, если выполнение `make describe` с каким-либо портом завершится неудачно, то никому не удастся создать `INDEX`, при этом много людей вскоре станут несчастны.



Возможность генерировать этот файл очень важна вне зависимости от того, какие параметры присутствуют в `make.conf`, поэтому, пожалуйста, избегайте, таких вещей, как использование декларации `.error`, когда (к примеру) требования к зависимости не было удовлетворено. (Смотрите [Избегайте использования конструкции `.error`](#).)

Если `make describe` выдаёт строчку, а не ошибку, то для вас это пройдёт безболезненно. Обратитесь к файлу `bsd.port.mk`, чтобы выяснить значение выдаваемых строк.

Также обратите внимание, что запуск актуальной версии `portlint` (как указано в следующем разделе) приведёт к автоматическому выполнению `make describe`.

## 10.2. Запуск `make test`

Даже если порт успешно собирается, рекомендуется убедиться, что программа корректно выполняет свои функции. Если исходный проект предоставляет тесты вместе с программным обеспечением, рекомендуется их запустить и проверить, что всё работает, как ожидается.

Порт может автоматически включить тесты, используя переменную `TEST_TARGET`. Когда эта переменная установлена, она содержит имя цели тестирования порта. Обычно это просто `test`, но другие варианты включают `tests`, `check` или, в специфических случаях, такие значения, как `run_tests.py`.

В дополнение к переменной `TEST_TARGET` фреймворк предоставляет следующие переменные для управления выполнением тестов:

- `TEST_WKRSRC` — это каталог для выполнения тестов.
- `TEST_ENV` содержит дополнительные переменные, которые передаются на этап тестирования.
- `TEST_ARGS` содержит любые дополнительные аргументы, переданные на этапе тестирования.

Примеры использования этих переменных можно найти в [cad/xyce](#), [www/libjwt](#) и других.



Убедитесь, что тесты не ломаются при обновлении порта.

## 10.3. Portclippy / Portfmt

Эти инструменты поставляются из пакета:ports-mgmt/portfmt[].

Portclippy — это линтер, проверяющий, расположены ли переменные в файле Makefile в правильном порядке согласно [Порядку переменных в Makefile портов](#).

Portfmt — это инструмент для автоматического форматирования Makefile.

## 10.4. Portlint

Проверьте свою работу командой `portlint` перед тем, как её отослать или перенести в дерево портов. `portlint` предупреждает вас о многих распространённых ошибках, как функциональных, так и стилистических. Для нового (или скопированного внутри хранилища) порта самым подходящим является запуск `portlint -A`; для уже существующего порта достаточно будет запустить `portlint -C`.

Так как для обнаружения ошибок `portlint` использует эвристические методы, то им могут выдаваться и ошибочные предупреждения. Кроме того, время от времени нечто, отмечаемое как некорректность, из-за ограничений механизма создания портов не может быть сделано никак иначе. Если вы сомневаетесь, то лучше всего спросить в [Список рассылки, посвящённый Портам FreeBSD](#).

## 10.5. Инструменты для работы с портами

Программа `ports-mgmt/porttools` входит в состав Коллекции Портов.

`port` является сценарием переднего плана, который может упростить вам задачу тестирования. Если вы хотите проверить новый порт или обновить существующий, то вы можете использовать `port test` для проверки вашего порта, включая проверку `portlint`. Эта команда также находит и отображает любые файлы, которые невключенные в `pkg-plist`. Смотрите следующий пример:

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

## 10.6. PREFIX и DESTDIR

Переменная `PREFIX` определяет, куда будет установлен порт. По умолчанию это `/usr/local`, но может меняться пользователем на собственный путь, такой как `/opt`. В вашем порту значение этой переменной должно учитываться.

Если пользователь установил переменную `DESTDIR`, то она определяет полное альтернативное окружение, обычно, это `jail` или установленная система, смонтированная в месте, отличном от `/`. На самом деле порт устанавливается в `DESTDIR/PREFIX` и

реги­стри­ру­ет­ся в ба­зе дан­ных па­ке­тов в `DESTDIR/var/db/pkg`. По­сколь­ку управ­ле­ние `DESTDIR` про­из­во­дит­ся ав­то­ма­ти­че­ски ин­фра­струк­ту­рой пор­тов с по­мо­щью `chroot(8)`, вам не нуж­ны ни­ка­кие из­ме­не­ния или про­яв­ле­ние осо­бой ос­то­ро­ж­но­сти при на­пи­са­нии пор­тов, со­вмес­ты­мых с `DESTDIR`.

Зна­че­ние пе­ре­мен­ной `PREFIX` бу­дет ус­та­нов­ле­но в `LOCALBASE` (по умол­ча­нию `/usr/local`). Если за­да­на пе­ре­мен­ная `USE_LINUX_PREFIX`, то `PREFIX` при­мет зна­че­ние `LINUXBASE` (по умол­ча­нию `/compat/linux`).

Из­бе­га­ние яв­но про­пи­сы­ва­е­мых пу­тей `/usr/local` в ис­ход­ном ко­де сде­ла­ет порт го­раз­до бо­лее гиб­ким и спо­соб­ным удо­влет­во­рить по­треб­но­сти дру­гих сер­веров. Час­то это­го мож­но до­бить­ся про­стой за­ме­ной строк `/usr/local` в раз­лич­ных фай­лах Makefile вну­три пор­та на `${PREFIX}`. Эта пе­ре­мен­ная ав­то­ма­ти­че­ски пе­ре­да­ёт­ся да­лее на каж­дом эта­пе по­стро­е­ния и ус­та­нов­ки.

Про­верь­те, что ва­ше при­ло­же­ние не ус­та­нав­ли­ва­ет че­го-ли­бо в ка­та­лог `/usr/local` вме­сто `PREFIX`. На­ли­чие яв­но ука­зан­ных пу­тей мож­но бы­стро про­ве­рить сле­ду­ю­щим об­ра­зом:

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

Если что-то бы­ло ус­та­нов­ле­но за пре­де­ла­ми `PREFIX`, то про­цесс соз­да­ния па­ке­та со­об­щи­т об от­сут­ст­вии фай­лов.

Это так­же сто­ит про­ве­рить с ис­поль­зо­ва­ни­ем под­дер­жки ка­та­лога сбор­ки (смот­ри­те [Staging](#)):

```
% make stage && make check-plist && make stage-qa && make package
```

- `check-plist` про­ве­ря­ет от­сут­ст­вую­щие в `plist` фай­лы и фай­лы в `plist`, ко­то­рые не ус­та­нов­ле­ны пор­том.
- `stage-qa` про­ве­ря­ет на­ли­чие рас­про­стран­ён­ных про­блем, та­ких как не­правиль­ный ше­банг (ин­тер­пре­та­тор­ная стро­ка в пер­вой стро­ке скрип­та), сим­воль­ные ссы­лки, ука­зы­ва­ю­щие за пре­де­лы `stage`-ди­рек­то­рии, фай­лы с `setuid` би­том и би­блио­те­ки с от­ла­доч­ной ин­фор­ма­ци­ей...

Эти те­сты не об­на­ру­жат жё­ст­ко за­дан­ные пу­ти в фай­лах пор­та, а так­же не про­ве­ря­ют, что `LOCALBASE` ис­поль­зу­ет­ся кор­рек­тно для ссы­лок на фай­лы из дру­гих пор­тов. Вре­мен­но ус­та­нов­лен­ный порт в `/var/tmp/make -V PORTNAME` дол­жен быть про­те­сти­ро­ван на кор­рек­тную ра­бо­ту, что­бы убе­дить­ся в от­сут­ст­вии про­блем с пу­тя­ми.

`PREFIX` не дол­жен быть яв­но ус­та­нов­лен в Makefile пор­та. Поль­зо­ва­те­ли, ус­та­нав­ли­ва­ю­щие порт, мо­гут за­дать `PREFIX` в дру­гом ме­сте, и порт дол­жен учи­ты­вать эту на­строй­ку.

Об­ра­щай­тесь к про­грам­мам и фай­лам из дру­гих пор­тов с по­мо­щью упо­мя­ну­тых вы­ше пе­ре­мен­ных, а не яв­ных пу­тей. На­при­мер, если порт тре­бу­ет, что­бы мак­рос `PAGER` со­дер­жал пол­ный пу­ть к `less`, не ис­поль­зуй­те яв­ный пу­ть `/usr/local/bin/less`. Вме­сто это­го ис­поль­зуй­те `${LOCALBASE}`:

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

Путь с `LOCALBASE` с большей вероятностью продолжит работать, если системный администратор переместил всё дерево `/usr/local` в другое место.



Все эти тесты выполняются автоматически при запуске `poudriere testport` или `poudriere bulk -t`. Настоятельно рекомендуется каждому участнику разработки портов устанавливать и тестировать свои порты с помощью этого инструмента. Дополнительную информацию можно найти в [poudriere](#).

## 10.7. poudriere

Для контрибьютора портов `poudriere` является одним из самых важных и полезных инструментов для тестирования и сборки. Его основные возможности включают:

- Массовая сборка всего дерева портов, определенных подмножеств дерева портов или отдельного порта с его зависимостями
- Автоматическая упаковка результатов сборки
- Генерация файлов журнала сборки для каждого порта
- Предоставление подписанного репозитория [pkg\(8\)](#)
- Тестирование сборки портов перед отправкой патча в трекер ошибок FreeBSD или внесением изменений в дерево портов
- Тестирование успешных сборок портов с использованием различных параметров

Поскольку `poudriere` выполняет сборку в чистой среде [jail\(8\)](#) и использует возможности [zfs\(8\)](#), он имеет несколько преимуществ по сравнению с традиционным тестированием на основной системе:

- Отсутствие загрязнения основной среды: никаких оставшихся файлов, случайных удалений или изменений существующих конфигурационных файлов.
- Проверяет `pkg-plist` на наличие отсутствующих или лишних записей
- Коммиттеры портов иногда запрашивают журнал `poudriere` вместе с отправкой патча, чтобы оценить, готов ли патч для интеграции в дерево портов

Также его настройка и использование довольно просты, он не имеет зависимостей и будет работать в любой поддерживаемой версии FreeBSD. В этом разделе показано, как установить, настроить и запустить `poudriere` в рамках обычного рабочего процесса разработчика портов.

Примеры в этом разделе показывают стандартную структуру файлов, принятую в FreeBSD. Внесите соответствующие изменения, если у вас используются другие настройки. Дерево портов, обозначаемое как `PORTSDIR`, находится в `/usr/ports`. По умолчанию `LOCALBASE` и `PREFIX` указывают на `/usr/local`.

## 10.7.1. Установка poudriere

poudriere доступен в дереве портов в пакете [ports-mgmt/poudriere](#). Его можно установить с помощью [pkg\(8\)](#) или из портов:

```
# pkg install poudriere
```

или

```
# make -C /usr/ports/ports-mgmt/poudriere install clean
```

Также существует версия poudriere в разработке, которая в конечном итоге станет следующим релизом. Она доступна в пакете:ports-mgmt/poudriere-devel[]. Эта версия используется для официальных сборок пакетов FreeBSD, поэтому она хорошо протестирована. В ней часто появляются новые интересные функции. Коммиттер портов захочет использовать версию в разработке, так как именно она используется в продакшене и содержит все новые функции, которые гарантируют, что всё будет работать идеально. Контрибьютору не обязательно нужны эти функции, так как наиболее важные исправления переносятся в выпущенную версию. Основная причина использования версии в разработке для сборки официальных пакетов заключается в её скорости — она позволяет сократить время полной сборки с 18 до 17 часов при использовании высокопроизводительного сервера с 32 CPU и 128 ГБ оперативной памяти. Эти оптимизации не будут столь значимы при сборке портов на настольном компьютере.

## 10.7.2. Настройка poudriere

Порт устанавливает файл конфигурации по умолчанию, /usr/local/etc/poudriere.conf. Каждый параметр описан в этом файле конфигурации.

Вот минимальный пример конфигурационного файла:

```
ZPOOL=zroot
BASEFS=/usr/local/poudriere
DISTFILES_CACHE=/usr/ports/distfiles
RESOLV_CONF=/etc/resolv.conf
```

### ZPOOL

Имя пула хранения ZFS, который будет использовать poudriere. Должно быть указано в выводе команды `zpool status`.

### BASEFS

Корневая точка монтирования файловых систем poudriere. Эта запись приведет к тому, что poudriere смонтирует `tank/poudriere` в `/poudriere`.

### DISTFILES\_CACHE

Определяет, где хранятся distfiles. В этом примере poudriere и хост используют общий

каталог для хранения distfiles. Это позволяет избежать загрузки тарболов, которые уже присутствуют в системе. Пожалуйста, создайте этот каталог, если он ещё не существует, чтобы poudriere мог его найти.

## RESOLV\_CONF

Используйте файл /etc/resolv.conf хоста внутри клеток для DNS. Это необходимо, чтобы клетки могли разрешать URL-адреса distfiles при загрузке. Это не требуется при использовании прокси. Обратитесь к файлу конфигурации по умолчанию для настройки прокси.

### 10.7.3. Создание клеток poudriere

Создайте базовые клетки, которые poudriere будет использовать для сборки:

```
# poudriere jail -c -j 131Ramd64 -v 13.1-RELEASE -a amd64
```

Загрузите 13.1-RELEASE для amd64 с FTP-сервера, указанного в FREEBSD\_HOST в poudriere.conf, создайте ZFS-файловую систему tank/poudriere/jails/131Ramd64, смонтируйте её в /poudriere/jails/131Ramd64 и распакуйте тарболлы 13.1-RELEASE в эту файловую систему.

```
# poudriere jail -c -j 12i386 -v stable/12 -a i386 -m git+https
```

Создайте tank/poudriere/jails/12i386, смонтируйте его на /poudriere/jails/12i386, затем извлеките верхушку ветки Git FreeBSD-12-STABLE из GIT\_HOST в poudriere.conf или по умолчанию git.freebsd.org в /poudriere/jails/12i386/usr/src, после чего выполните buildworld и установите его в /poudriere/jails/12i386.



Хотя возможно собрать более новую версию FreeBSD на старой версии, в большинстве случаев она не запустится. Например, если требуется клетка на stable/13, то хост также должен работать на stable/13. Запуск 13.1-RELEASE недостаточен.

Для создания клетки poudriere для 14.0-CURRENT:

```
# poudriere jail -c -j 14amd64 -v main -a amd64 -m git+https
```



Для запуска клетки 14.0-CURRENT poudriere хостовая система должна работать под управлением 14.0-CURRENT. В общем случае, более новые ядра могут собирать и запускать более старые клетки. Например, ядро 14.0-CURRENT может собирать и запускать клетку 12.4-STABLE, если параметр ядра COMPAT\_FREEBSD12 был скомпилирован (включен по умолчанию в конфигурации ядра GENERIC 14.0-CURRENT).

Список клеток, известных poudriere, можно вывести с помощью команды poudriere jail -l:

```
# poudriere jail -l
JAILNAME      VERSION      ARCH      METHOD
131Ramd64     13.1-RELEASE amd64     ftp
12i386        12.4-STABLE  i386     git+https
```

### 10.7.4. Обновление клеток poudriere

Управление обновлениями очень простое. Команда:

```
# poudriere jail -u -j JAILNAME
```

обновляет указанную клетку до последней доступной версии. Для релизов FreeBSD обновление до последнего уровня исправлений с помощью [freebsd-update\(8\)](#). Для версий FreeBSD, собранных из исходников, обновление до последней ревизии git в ветке.



Для клеток, использующих метод `git+*`, полезно добавить `-J` **КоличествоПараллельныхСборок** для ускорения сборки за счёт увеличения количества параллельных задач компиляции. Например, если на машине для сборки 6 CPU, используйте:

```
# poudriere jail -u -J 6 -j JAILNAME
```

### 10.7.5. Настройка деревьев портов для использования с poudriere

Существует несколько способов использования деревьев портов в poudriere. Наиболее простой способ — позволить poudriere создать для себя дерево портов по умолчанию, используя [Git](#):

```
# poudriere ports -c -m git+https -B main
```

Эти команды создают `tank/poudriere/ports/default`, монтируют его в `/poudriere/ports/default` и заполняют с помощью Git. После этого он включается в список известных деревьев портов:

```
# poudriere ports -l
PORTSTREE METHOD    TIMESTAMP      PATH
default  git+https  2020-07-20 04:23:56 /poudriere/ports/default
```



Обратите внимание, что дерево портов "default" является особым. Каждая из команд сборки, объяснённых далее, будет неявно использовать это дерево портов, если явно не указано иное. Чтобы использовать другое дерево, добавьте `-p treename` к командам.

Лучший способ работы с локальными изменениями для разработчика портов —

использовать [Git](#). Как и при создании клеток, можно использовать другой метод для создания дерева портов. Чтобы добавить дополнительное дерево портов для тестирования локальных изменений и разработки портов, предпочтительно использовать клонирование дерева через `git` (как описано выше).

### 10.7.6. Использование управляемых вручную деревьев портов с помощью `poudriere`

В зависимости от рабочего процесса может быть крайне полезно использовать деревья портов, которые поддерживаются вручную. Например, если существует локальная копия дерева портов в `/work/ports`, укажите `poudriere` на это расположение:

```
# poudriere ports -c -m null -M /work/ports -p development
```

Это будет указано в таблице известных деревьев:

```
# poudriere ports -l
PORTSTREE  METHOD      TIMESTAMP      PATH
development null        2020-07-20 05:06:33 /work/ports
```



Тире или `null` в колонке `METHOD` означает, что `poudriere` никогда не будет обновлять или изменять это дерево портов. Полностью на пользователе лежит ответственность за поддержку этого дерева, включая все локальные изменения, которые могут использоваться для тестирования новых портов и отправки исправлений.

### 10.7.7. Обновление деревьев портов `poudriere`

Так же просто, как с клетками, описанными ранее:

```
# poudriere ports -u -p PORTSTREE
```

Обновит указанное `PORTSTREE`, дерево, указанное в выводе команды `poudriere -l`, до последней доступной ревизии на официальных серверах.



Деревья портов без метода, см. [Использование вручную управляемых деревьев портов с помощью `poudriere`](#), не могут быть обновлены таким образом и должны обновляться вручную сопровождающим портов.

### 10.7.8. Тестирование портов

После настройки клеток и деревьев портов можно проверить результат изменений, внесенных участником в дерево портов.

Например, локальные изменения в порте [www/firefox](#), расположенном в

/work/ports/www/firefox, можно протестировать в ранее созданной клетке 13.1-RELEASE:

```
# poudriere testport -j 131Ramd64 -p development -o www/firefox
```

Это соберет все зависимости Firefox. Если зависимость уже была собрана ранее и остается актуальной, будет установлен готовый пакет. Если для зависимости нет актуального пакета, он будет собран с параметрами по умолчанию в клетке. Затем будет собран сам Firefox.

Полная сборка каждого порта записывается в /poudriere/data/logs/bulk/131Ri386-development/build-time/logs.

Имя каталога `131Ri386-development` формируется из аргументов `-j` и `-p` соответственно. Для удобства также поддерживается символическая ссылка /poudriere/data/logs/bulk/131Ri386-development/latest. Эта ссылка указывает на последний каталог *времени сборки*. Также в этом каталоге находится файл `index.html`, который позволяет наблюдать за процессом сборки через веб-браузер.

По умолчанию poudriere очищает клетки и оставляет файлы журналов в указанных выше каталогах. Для упрощения анализа клетки можно оставить запущенными после сборки, добавив `-i` к `testport`:

```
# poudriere testport -j 131Ramd64 -p development -i -o www/firefox
```

После завершения сборки, независимо от того, была ли она успешной, в клетке предоставляется оболочка. Эта оболочка используется для дальнейшего исследования. Можно указать poudriere оставить клетку запущенной после завершения сборки с помощью `-I`. poudriere покажет команду для выполнения, когда клетка больше не нужна. Затем можно использовать `jexec(8)` для входа в неё:

```
# poudriere testport -j 131Ramd64 -p development -I -o www/firefox
[...]  
====>> Installing local Pkg repository to /usr/local/etc/pkg/repos  
====>> Leaving jail 131Ramd64-development-n running, mounted at  
/poudriere/data/.m/131Ramd64-development/ref for interactive run testing  
====>> To enter jail: jexec 131Ramd64-development-n env -i TERM=$TERM /usr/bin/login  
-fp root  
====>> To stop jail: poudriere jail -k -j 131Ramd64 -p development  
# jexec 131Ramd64-development-n env -i TERM=$TERM /usr/bin/login -fp root  
# [do some stuff in the jail]  
# exit  
# poudriere jail -k -j 131Ramd64 -p development  
====>> Umounting file systems
```

Неотъемлемой частью инфраструктуры сборки портов FreeBSD является возможность настройки портов под личные предпочтения с помощью опций. Их также можно тестировать с помощью poudriere. Добавление опции `-c`:

```
# poudriere testport -j 131Ramd64 -c -o www/firefox
```

Представляет диалог настройки порта перед его сборкой. Порты, указанные после `-o` в формате `категория/имя_порта`, будут использовать указанные опции, все зависимости будут использовать опции по умолчанию. Тестирование зависимых портов с нестандартными опциями может быть выполнено с использованием наборов, см. [Использование наборов](#).



При тестировании портов, где файл `rkg-plist` изменяется во время сборки в зависимости от выбранных опций, рекомендуется выполнить тестовый запуск со всеми выбранными опциями и один без выбранных опций.

### 10.7.9. Использование наборов

Для всех действий, связанных со сборкой, можно указать так называемый *набор* с помощью `-z имя_набора`. Набор относится к полностью независимой сборке. Это позволяет, например, использовать `testport` с нестандартными параметрами для зависимых портов.

Для использования наборов `poudriere` ожидает, что будет использована структура каталогов, аналогичная `PORT_DBDIR`, по умолчанию `/var/db/ports`, в его конфигурационной директории. Этот каталог затем монтируется с помощью `nullfs(5)` в клетки, где собираются порты и их зависимости. Обычно подходящую начальную точку можно получить, рекурсивно скопировав существующий `PORT_DBDIR` в `/usr/local/etc/poudriere.d/jailname-portname-setname-options`. Это подробно описано в [poudriere\(8\)](#). Например, для тестирования `www/firefox` в определённом наборе с именем `devset`, добавьте параметр `-z devset` к команде `testport`:

```
# poudriere testport -j 131Ramd64 -p development -z devset -o www/firefox
```

Это проверит наличие этих каталогов в следующем порядке:

- `/usr/local/etc/poudriere.d/131Ramd64-development-devset-options`
- `/usr/local/etc/poudriere.d/131Ramd64-devset-options`
- `/usr/local/etc/poudriere.d/131Ramd64-development-options`
- `/usr/local/etc/poudriere.d/devset-options`
- `/usr/local/etc/poudriere.d/development-options`
- `/usr/local/etc/poudriere.d/131Ramd64-options`
- `/usr/local/etc/poudriere.d/options`

Из этого списка `poudriere nullfs(5)` монтирует *первое существующее* дерево каталогов в директорию `/var/db/ports` сборных клеток. Таким образом, все пользовательские настройки используются для всех портов во время этого запуска `testport`.

После предоставления структуры каталогов для набора можно изменить параметры для конкретного порта. Например:

```
# poudriere options -c www/firefox -z devset
```

Отображается диалог настройки [www/firefox](#), где можно редактировать параметры. Выбранные параметры сохраняются в набор [devset](#).



poudriere очень гибок в настройке опций. poudriere можно настроить для конкретных клеток, деревьев портов и для нескольких портов одной командой. Подробности см. в [poudriere\(8\)](#).

### 10.7.10. Предоставление пользовательского файла `make.conf`

Подобно использованию наборов, poudriere также использует пользовательский `make.conf`, если он предоставлен. Для этого не требуется специального аргумента командной строки. Вместо этого poudriere ищет существующие файлы, соответствующие схеме именования, производной от командной строки. Например:

```
# poudriere testport -j 131Ramd64 -p development -z devset -o www/firefox
```

заставляет poudriere проверять наличие этих файлов в следующем порядке:

- `/usr/local/etc/poudriere.d/make.conf`
- `/usr/local/etc/poudriere.d/devset-make.conf`
- `/usr/local/etc/poudriere.d/development-make.conf`
- `/usr/local/etc/poudriere.d/131Ramd64-make.conf`
- `/usr/local/etc/poudriere.d/131Ramd64-development-make.conf`
- `/usr/local/etc/poudriere.d/131Ramd64-devset-make.conf`
- `/usr/local/etc/poudriere.d/131Ramd64-development-devset-make.conf`

В отличие от наборов, все найденные файлы будут добавлены, *в указанном порядке*, в один `make.conf` внутри клеток сборки. Таким образом, можно задать общие переменные `make`, предназначенные для влияния на все сборки, в файле `/usr/local/etc/poudriere.d/make.conf`. Специальные переменные, предназначенные только для определённых клеток или наборов, можно задать в специализированных файлах `make.conf`, например, в `/usr/local/etc/poudriere.d/131Ramd64-development-devset-make.conf`.

*Пример 114. Использование `make.conf` для изменения Perl по умолчанию*

Для сборки набора с нестандартной версией Perl, например, [5.20](#), используя набор с именем `perl5-20`, создайте файл `perl5-20-make.conf` со следующей строкой:

```
DEFAULT_VERSIONS+= perl=5.20
```

Обратите внимание на использование `+=`, чтобы содержимое переменной не было перезаписано, если она уже установлена в стандартном `make.conf`.

### 10.7.11. Удаление ненужных файлов дистрибутива

`poudriere` имеет встроенный механизм для удаления устаревших файлов дистрибутива, которые больше не используются ни одним портом данного дерева. Команда

```
# poudriere distclean -p portstree
```

будет сканировать папку файлов дистрибутива, `DISTFILES_CACHE` в `poudriere.conf`, сравнивая ее с деревом портов, указанным аргументом `-p portstree`, и запрашивать подтверждение на удаление этих файлов дистрибутива. Чтобы пропустить запрос и удалить все неиспользуемые файлы без подтверждения, можно добавить аргумент `-y`:

```
# poudriere distclean -p portstree -y
```

## 10.8. Отладка портов

Иногда что-то идет не так, и порт не работает во время выполнения. Фреймворк предоставляет некоторые средства для отладки портов. Эти вспомогательные инструменты ограничены, поскольку способ отладки порта во многом зависит от используемой технологии. Следующие переменные помогают в отладке портов:

- `WITH_DEBUG`. Если установлено, порты собираются с отладочными символами.
- `WITH_DEBUG_PORTS`. Указывает список портов, которые должны собираться с установленным `WITH_DEBUG`.
- `DEBUG_FLAGS`. Используется для указания дополнительных флагов для `CFLAGS`. По умолчанию `-g`.

Когда `WITH_DEBUG` установлен, глобально или для списка портов, результирующие бинарные файлы не лишаются символов.

Эти переменные могут быть указаны в `make.conf` или в командной строке:

```
# cd category/port && make -DWITH_DEBUG DEBUG_FLAGSS="-g -O0"
```



Если порт собирается с использованием [ports-mgmt/poudriere](#), отладочные переменные должны быть указаны в `make.conf poudriere`, а не в `/etc/make.conf`. Подробности см. в документации [ports-mgmt/poudriere](#).

Пожалуйста, обратитесь к отладочной информации в [Руководстве разработчика](#) для

получения более подробной информации о доступных инструментах отладки.

# Глава 11. Обновление отдельного порта

Когда порт не является самой последней версией, доступной от авторов, обновите локальную рабочую копию `/usr/ports`. Возможно, порт уже был обновлён до новой версии.

При работе с большим количеством портов, вероятно, будет проще использовать Git для поддержания всей коллекции портов в актуальном состоянии, как описано в [Использование коллекции портов](#). Это также позволит отслеживать все зависимости портов.

Следующий шаг — проверить, есть ли уже ожидающее обновление. Для этого есть два варианта. Доступен поиск в [Сообщения о проблемах \(PR\)](#) или [база данных ошибок FreeBSD](#). Выберите **Ports & Packages** в меню множественного выбора **Product** и введите название порта в поле **Summary**.

Если таких отложенных PR не существует, то на следующем этапе следует послать сообщение электронной почты человеку, поддерживающему порт, который выдаётся по команде `make maintainer`. Этот человек может уже работать над обновлением, или иметь причину не обновлять порт прямо сейчас (например, из-за проблем со стабильностью функционирования новой версии); вам нет нужды дублировать их работу. Заметьте, что неподдерживаемые порты перечисляются с адресом сопровождающего `ports@FreeBSD.org`, который является всего лишь адресом общего списка рассылки, так что отправка туда сообщений, скорее всего, в данном случае не поможет.

Если сопровождающий просит вас выполнить обновление, либо сопровождающий отсутствует, то у вас появляется шанс помочь FreeBSD, приготовив обновление самим! Пожалуйста, делайте это с использованием команды `diff(1)` в основной системе.

Чтобы создать подходящий `diff` для одного патча, скопируйте файл, который нужно пропатчить, в `something.orig`, сохраните ваши изменения в `something`, а затем создайте ваше патч:

```
% diff -u something.orig something > something.diff
```

В противном случае используйте метод `git diff` ([Использование Git для создания патчей](#)) или скопируйте содержимое порта в совершенно другой каталог и используйте результат рекурсивного вывода `diff(1)` для новых и старых каталогов портов (например, если изменённый каталог порта называется `superedit`, а исходный находится в нашем дереве как `superedit.bak`, сохраните результат выполнения `diff -ruN superedit.bak superedit`). Подойдёт как унифицированный, так и контекстный `diff`, но коммиттеры портов обычно предпочитают унифицированные `diff`. Обратите внимание на использование опции `-N` — это общепринятый способ заставить `diff` корректно обрабатывать случаи добавления новых файлов или удаления старых. Перед отправкой `diff`, пожалуйста, проверьте вывод, чтобы убедиться, что все изменения имеют смысл. (В частности, не забудьте сначала очистить рабочие каталоги с помощью `make clean`).



Если некоторые файлы были добавлены, скопированы, перемещены или

удалены, добавьте эту информацию в отчёт о проблеме, чтобы коммиттер, принимающий патч, знал, какие команды `git(1)` нужно выполнить.

Для упрощения стандартных операций с файлами исправлений используйте `make makepatch`, как описано в [Применение патчей](#). Существуют и другие инструменты, например `/usr/ports/Tools/scripts/patchtool.py`. Перед его использованием прочтите `/usr/ports/Tools/scripts/README.patchtool`.

Если порт никем не поддерживается, а вы активно его используете, пожалуйста, подумайте над тем, чтобы добровольно стать его сопровождающим. Во FreeBSD имеется более 4000 портов без поддержки, и это как раз та область, где всегда нужны добровольцы. (Детальное описание обязанностей сопровождающего можно найти в разделе [Руководства Разработчика](#).)

Для отправки diff используйте [форму отправки багов](#) (продукт `Ports & Packages`, компонент `Individual Port(s)`). Всегда указывайте категорию с именем порта, за которой следует двоеточие и краткое описание проблемы. Примеры: `категория/имя_порта: добавить опцию F00`; `категория/имя_порта: Обновление до X.Y`. Упоминайте в сообщении все добавленные или удалённые файлы, так как они должны быть явно указаны в `git(1)` при выполнении коммита. Не сжимайте и не кодируйте diff.

Прежде чем отправить сообщение об ошибке, ознакомьтесь с разделом [Написание отчета о проблеме](#) в статье "Отчеты о проблемах". В нем содержится гораздо больше информации о том, как составлять полезные отчеты о проблемах.



Если обновление вызвано соображениями информационной безопасности или наличием серьёзных ошибок в имеющемся порте, пожалуйста, оповестите [Группа Менеджеров Деревя Портов FreeBSD](#) <[portmgr@FreeBSD.org](mailto:portmgr@FreeBSD.org)> о необходимости немедленного перепостроения и повторного распространения пакета данного порта. В противном случае ничего не подозревающие пользователи `pkg` будут продолжать устанавливать старую версию по команде `pkg install` в течение ещё нескольких недель.



Пожалуйста, используйте `diff(1)` или `git diff` для создания обновлений существующих портов. Другие форматы включают весь файл и делают невозможным увидеть только изменённые части. Если различия не включены, всё обновление может быть проигнорировано.

Теперь, когда вы сделали всё это, прочитайте о том, как поддерживать актуальное состояние, в [Актуализация](#).

## 11.1. Использование Git для создания патчей

Когда это возможно, пожалуйста, предоставляйте патч или diff с помощью `git(1)`. Их проще обрабатывать, чем различия между «новым и старым» каталогом. Так легче увидеть, что изменилось, и обновить diff, если что-то было изменено в Коллекции портов с момента начала работы над ней, или если коммиттер просит что-то исправить. Кроме того, патч,

созданный с помощью `git-format-patch(1)` или `git-diff(1)`, можно легко применить с помощью `git-am(1)` или `git-apply(1)`, что сэкономит время коммиттера. Наконец, git-патч, созданный `git-format-patch(1)`, включает информацию об авторе и сообщения коммитов. Они будут записаны в лог репозитория, и это рекомендуемый способ отправки ваших изменений.

```
% git clone https://git.FreeBSD.org/ports.git ~/my_wrkdir ① ②
% cd ~/my_wrkdir
```

- ① Это может быть где угодно; место, в котором производится построение портов, не привязано к `/usr/ports/`.
- ② [git.FreeBSD.org](https://git.FreeBSD.org) — это публичный Git-сервер FreeBSD. Подробнее см. в [таблице URL-репозитория FreeBSD Git](#).

Находясь в каталоге порта, внесите необходимые изменения. Если требуется добавить, переместить или удалить файл, используйте `git` для отслеживания этих изменений:

```
% git add new_file
% git mv old_name new_name
% git rm deleted_file
```

Убедитесь, что проверили порт, используя контрольный список в [Тестирование порта](#) и [Проверка порта с помощью portlint](#).

Также обновите ссылку на контрольную сумму в `distinfo` с помощью `make makesum`.

Прежде чем создавать патч, загрузите последнюю версию репозитория и перебазируйте изменения поверх неё. Внимательно следите за выводом и следуйте ему. Если какие-либо файлы не удалось перебазировать, это означает, что исходные файлы изменились во время вашего редактирования, и конфликты необходимо разрешить вручную.

```
% git fetch origin main
% git rebase origin/main
```

Проверьте изменения, подготовленные для исправления:

```
% git status
% git diff --staged
```

Последний шаг — создать унифицированный diff или патч изменений:

Для создания патча с помощью `git-format-patch(1)`:

```
% git checkout -b my_branch
% git commit
% git format-patch main
```

Это создаст файл исправления с именем вида `0001-foo.patch`. Это предпочтительный способ, так как он включает идентификацию автора, а также удобнее, когда вы делаете серию изменений, которые не должны объединяться вместе.

Или для создания унифицированного diff с помощью `git-diff(1)`:

```
% git diff --staged > ../`make -VPKGNAME`.diff
```

Это создаст файл с различиями с именем вида `foo-1.2.3.diff`. Здесь `foo` заменяется на первую строку сообщения коммита, то есть на тему сообщения коммита.

После создания патча вы можете переключиться на основную ветку для начала других разработок.

```
% git checkout main
```

После принятия и слияния патча вы можете удалить локальную ветку разработки, если хотите:

```
% git branch -D my_branch
```



Если файлы были добавлены, перемещены или удалены, укажите использованные команды `git(1) add`, `mv` и `rm`. Команда `git mv` должна быть выполнена до применения патча. Команды `git add` или `git rm` должны быть выполнены после применения патча.

Отправьте исправление, следуя [рекомендациям по отправке отчетов о проблемах](#).

## 11.2. UPDATING и MOVED

### 11.2.1. /usr/ports/UPDATING

Если обновление порта требует специальных действий, таких как изменение конфигурационных файлов или запуск определённой программы, это должно быть задокументировано в данном файле. Формат записи в этом файле следующий:

```
YYYYMMDD:  
AFFECTS: users of portcategory/portname  
AUTHOR: Your name <Your email address>  
  
Special instructions
```



При включении точных инструкций для `portmaster`, `portupgrade` и/или `pkg`, убедитесь в правильности экранирования в shell. Например, *не* используйте:

```
# pkg delete -g -f docbook-xml* docbook-sk* docbook[2345]??-* docbook-4*
```

Как показано, команда будет работать только с bourne-оболочками. Вместо этого используйте форму, приведённую ниже, которая будет работать как с bourne-оболочкой, так и с c-оболочкой:

```
# pkg delete -g -f docbook-xml\* docbook-sk\* docbook\[2345\]\?-\*\ docbook-4\*
```



Рекомендуется, чтобы строка AFFECTS содержала glob-выражение, соответствующее всем портам, затронутым записью, чтобы автоматизированные инструменты могли максимально легко её обработать. Если обновление касается всех существующих версий BIND 9, содержимое AFFECTS должно быть `users of dns/bind9*`, и оно *не должно* быть `users of BIND 9`

### 11.2.2. /usr/ports/MOVED

Этот файл используется для перечисления перемещённых или удалённых портов. Каждая строка в файле состоит из названия порта, места, куда порт был перемещён, даты и причины. Если порт был удалён, раздел с указанием места перемещения может быть оставлен пустым. Каждый раздел должен быть отделён символом | (вертикальная черта), например:

```
old name|new name (blank for deleted)|date of move|reason
```

Дата должна быть введена в формате ГГГГ-ММ-ДД. Новые записи добавляются в конец списка, чтобы сохранить его в хронологическом порядке, при этом самая старая запись находится в начале списка.

Если порт был удален, но затем восстановлен, удалите строку в этом файле, которая указывает, что он был удален.

Если порт был переименован, а затем переименован обратно в исходное имя, добавьте новую запись с промежуточным именем для старого имени и удалите старую запись, чтобы не создавать цикл.

Любые изменения должны быть проверены с помощью `Tools/scripts/MOVEDlint.awk`.



Если используется каталог портов, отличный от /usr/ports, следует указать:

```
% cd /home/user/ports
```

```
% env PORTSDIR=$PWD Tools/scripts/MOVEDLint.awk
```

# Глава 12. Безопасность

## 12.1. Почему безопасность так важна

Ошибки в программном обеспечении появляются случайно. Возможно, самые опасные из них те, что создают уязвимости безопасности. С технической точки зрения подобные уязвимости должны быть закрыты путем исправления вызывающих их ошибок. Тем не менее, политики обработки несущественных ошибок и уязвимостей очень различаются.

Обычная небольшая ошибка затрагивает только тех пользователей, которые задействуют некоторые комбинации настроек, активирующие эту ошибку. Разработчик в конечном счете выпустит патч, а затем новую версию программного обеспечения, свободного от ошибки, но большинство пользователей не посчитают нужным сразу же произвести обновление, поскольку эта ошибка никогда у них не проявлялась. Критическая ошибка, которая может приводить к потере данных, представляет серьезную проблему. Тем не менее, предусмотрительные пользователи знают, что большинство возможных происшествий, и среди них программные ошибки, скорее всего приводят к потере данных, поэтому они выполняют резервное копирование важных данных; дополнительно, критическая ошибка будет обнаружена очень скоро.

С уязвимостью безопасности всё иначе. Во-первых, она может сохраняться необнаруженной целые годы, потому что чаще всего не вызывает ошибок в работе. Во-вторых, компания злоумышленников может использовать ее для получения неавторизованного доступа к уязвимой системе, уничтожить или подменить важные данные; в худшем случае пользователь даже не заметит нанесенный урон. В-третьих, взлом уязвимой системы часто упрощает задачу проникновения атакующих в другие системы, которые не могут быть скомпрометированы иначе. Таким образом, устранение уязвимости как таковой недостаточно: следует разослать всем заинтересованным уведомления в наиболее понятной и исчерпывающей форме, что позволит оценить риск и предпринять подходящие меры.

## 12.2. Исправление уязвимостей безопасности

Что касается портов и пакетов, уязвимость безопасности изначально может появиться в исходном дистрибутиве или файлах порта. В первом случае, разработчик исходного программного обеспечения скорее всего сразу же выпустит патч или новую версию, и вам лишь понадобится сразу обновить порт в соответствии с исправлением автора. Если исправление по какой-то причине задерживается, вам следует либо [позначить порт как FORBIDDEN](#), либо добавить в порт ваш собственный патч. В случае уязвимости порта просто исправьте этот порт как можно скорее. В любом случае нужно следовать [стандартной процедуре отправки вашего изменения](#), если вы не обладаете правами на коммит изменения непосредственно в дерево портов.



Быть коммиттером портов недостаточно для коммита произвольного порта. Помните, что обычно у портов есть сопровождающие, мнение которых вы должны учитывать.

Пожалуйста, убедитесь, что ревизия порта после закрытия уязвимости увеличена. Вот как пользователи, обновляющие установленные пакеты на постоянной основе, увидят, что им нужно запустить обновление. Кроме того, новый пакет будет собран и распространен через FTP и WWW зеркала, замещая уязвимый. Если в процессе исправления уязвимости не было изменено значение `PORTVERSION`, то должно быть увеличено значение `PORTREVISION`. Вам следует увеличить значение `PORTREVISION` после добавления в порт файла с патчем, но не когда вы обновили порт до последней версии программного обеспечения, попутно затронув при этом `PORTVERSION`. За дальнейшей информацией обращайтесь к [соответствующему разделу](#).

## 12.3. Обеспечение сообщества информацией

### 12.3.1. База данных VuXML

Очень важным и первостепенным шагом при действии как можно раньше после раскрытия уязвимости является уведомление сообщества пользователей порта об опасности. Такие уведомления служат двум целям. Во-первых, в случае действительно серьезной угрозы, будет посоветовано применить мгновенное воздействие. Например, остановить затрагиваемый сетевой сервис или даже удалить порт целиком, пока уязвимость не будет устранена. Во-вторых, масса пользователей имеет тенденцию обновлять установленные пакеты только от случая к случаю. Из уведомления они узнают, что *должны* обновить пакет без промедления сразу же после появления исправленной версии.

Учитывая огромное число портов в дереве, невозможно по каждому случаю выпускать бюллетень безопасности без создания флуда и потери внимания сообщества к моменту появления действительно серьезных причин. Поэтому уязвимости безопасности, обнаруженные в портах, записываются в [базу данных FreeBSD VuXML](#). Члены Команды Офицеров Безопасности также отслеживают её на предмет появления вопросов, требующих их вмешательства.

Если вы обладаете правами коммиттера, вы можете сам обновить базу данных VuXML. Так вы поможете Команде Офицеров Безопасности и своевременно пошлете ценную информацию сообществу. Тем не менее, если вы не являетесь коммиттером или верите, что нашли исключительно серьезную уязвимость, то не задумываясь свяжитесь с Командой Офицеров Безопасности напрямую как это описано на странице [информационной безопасности FreeBSD](#).

База данных VuXML является документом XML. Его исходный файл `vuln.xml` содержится прямо внутри порта [security/vuxml](#). Следовательно, полное имя пути к файлу будет `PORTSDIR/security/vuxml/vuln.xml`. Каждый раз, при обнаружении вами в порте уязвимости безопасности добавьте об этом запись в этот файл. Пока вы не знакомы с VuXML, лучшее, что вы можете сделать, это найти существующую запись, подпадающую под ваш случай, затем скопировать ее и использовать в качестве шаблона.

### 12.3.2. Короткое вступление в VuXML

В совокупности XML является очень сложным форматом, и его описание выходит далеко за

рамки этой книги. Тем не менее, для достижения основного понимания структуры записи VuXML вам понадобится всего лишь понять теги. Имена тегов XML обрамляются в угловые скобки. Каждый открывающий <tag> должен иметь совпадающий закрывающий </tag>. Теги могут быть вложенными. При вложенности внутренние теги должны быть закрыты до закрытия внешних. Существует иерархия тегов, т.е. более сложные правила вкладывания тегов. Это похоже на HTML. Основное отличие в расширяемости XML, т.е. в определении собственных тегов. Из-за своей характерной структуры XML придает форму разрозненным данным. В частности, XML подходит для разметки описаний уязвимостей безопасности.

Теперь рассмотрим настоящую запись VuXML:

```
<vuln vid="f4bc80f4-da62-11d8-90ea-0004ac98a7b9"> ①
  <topic>Several vulnerabilities found in Foo</topic> ②
  <affects>
    <package>
      <name>foo</name> ③
      <name>foo-devel</name>
      <name>ja-foo</name>
      <range><ge>1.6</ge><lt>1.9</lt></range> ④
      <range><ge>2.*</ge><lt>2.4_1</lt></range>
      <range><eq>3.0b1</eq></range>
    </package>
    <package>
      <name>openfoo</name> ⑤
      <range><lt>1.10_7</lt></range> ⑥
      <range><ge>1.2,1</ge><lt>1.3_1,1</lt></range>
    </package>
  </affects>
  <description>
    <body xmlns="http://www.w3.org/1999/xhtml">
      <p>J. Random Hacker reports:</p> ⑦
      <blockquote
        cite="http://j.r.hacker.com/advisories/1">
        <p>Several issues in the Foo software may be exploited
          via carefully crafted QUUX requests. These requests will
          permit the injection of Bar code, mumble theft, and the
          readability of the Foo administrator account.</p>
        </blockquote>
      </body>
    </description>
    <references> ⑧
      <freebsdsa>SA-10:75.foo</freebsdsa> ⑨
      <freebsdpr>ports/987654</freebsdpr> ⑩
      <cvename>CVE-2023-48795</cvename> ⑪
      <certvu>740169</certvu> ⑫
      <uscertta>SA10-99A</uscertta> ⑬
      <mlist
        msgid="201075606@hacker.com">http://marc.theaimsgroup.com/?l=bugtraq&m=20388660782
        5605</mlist> ⑭
      <url>http://j.r.hacker.com/advisories/1</url> ⑮
```

```

</references>
<dates>
  <discovery>2010-05-25</discovery> ⑩
  <entry>2010-07-13</entry> ⑪
  <modified>2010-09-17</modified> ⑫
</dates>
</vuln>

```

Имена тегов должны быть самодокументируемы, чтобы мы сфокусировались только на полях, нужных нам для заполнения:

- ① Это тег верхнего уровня записи VuXML. У него есть обязательный атрибут `vid`, указывающий на универсальный уникальный идентификатор (UUID) для этой записи (в кавычках). Вы должны формировать UUID для каждой новой записи VuXML (и не забудьте заменить ее для шаблона UUID, если вы не пишете запись с нуля). Для получения VuXML UUID вы можете использовать `uuidgen(1)`.
- ② Однострочное описание найденной проблемы.
- ③ Здесь перечислены имена затронутых пакетов. Может быть дано несколько имен, поскольку некоторые пакеты могут быть основаны на одном главном порте или программном продукте. Сюда можно включить стабильную ветвь и ветвь разработки, локализованные версии и подчиненные порты, зависящие от различного выбора важных вариантов конфигурации, указанных на этапе построения.
- ④ Затронутые версии пакета(ов) указаны там как один или несколько диапазонов с использованием комбинации элементов `<lt>`, `<le>`, `<eq>`, `<ge>` и `<gt>`. Убедитесь, что указанные диапазоны версий не перекрываются.  
 В спецификации диапазона `*` (звёздочка) обозначает наименьший номер версии. В частности, `2.*` меньше, чем `2.a`. Таким образом, звёздочка может использоваться в диапазоне для соответствия всем возможным версиям `alpha`, `beta` и `RC`. Например, `<ge>2.</ge><lt>3.</lt>` выборочно соответствует каждой версии `2.x`, тогда как `<ge>2.0</ge><lt>3.0</lt>` — нет, поскольку последний пропускает `2.r3` и включает `3.b`. Приведённый пример указывает, что затронуты версии `1.6` и выше, но не включая `1.9`, версии `2.x` до `2.4_1` и версия `3.0b1`.
- ⑤ Некоторые связанные группы пакетов (в конечном счете, порты) могут быть указаны в разделе `<affected>`. Это можно использовать, если некоторые программные продукты (скажем, `FooBar`, `FreeBar` and `OpenBar`) являются производными от общей кодовой базы и всё еще совместно используют её ошибки и уязвимости. Имейте в виду отличие от перечисления множественных имён в одном разделе `<package>`.
- ⑥ Диапазоны версий должны учитывать `PORTPOCH` и `PORTREVISION`, если это применимо. Пожалуйста, помните, что в соответствии с правилами сравнения строк версия с ненулевым значением `PORTPOCH` выше, чем любая версия без `PORTPOCH`, например, `3.0,1` выше, чем `3.1` или даже `8.9`.
- ⑦ Сводная информация о проблеме. В этом поле используется XHTML. По крайней мере, должны быть обрамляющие `<p>` и `</p>`. Может быть использована более сложная разметка, но только в целях аккуратности и ясности: без эстетства, пожалуйста.
- ⑧ Этот раздел содержит ссылки на имеющие отношение документы. Приветствуется как

можно большее количество ссылок.

- ⑨ Это [бюллетень безопасности FreeBSD](#).
- ⑩ Это [сообщение об ошибке FreeBSD](#).
- ⑪ Идентификатор [MITRE CVE](#).
- ⑫ Это [SecurityFocus Bug ID](#).
- ⑬ Бюллетень безопасности [US-CERT](#).
- ⑭ URL к архивному сообщению в списке рассылки. Атрибут `msgid` является необязательным и может указывать на message ID сообщения.
- ⑮ Это общий URL. Используйте его только если ни одна из других категорий ссылок не подходит.
- ⑯ Это дата, когда проблема была раскрыта (ГГГГ-ММ-ДД).
- ⑰ Это дата добавления записи (ГГГГ-ММ-ДД).
- ⑱ Это дата, когда любая информация в записи была последний раз изменена (ГГГГ-ММ-ДД). Новые записи не должны включать это поле. Добавьте его при редактировании существующей записи.

### 12.3.3. Тестирование ваших изменений в базе данных VuXML

Этот пример описывает новую запись об уязвимости в пакете `dropbear`, которая была исправлена в версии `dropbear-2013.59`.

В качестве предварительного условия установите свежую версию порта [security/vuxml](#).

Сначала проверьте, есть ли уже запись об этой уязвимости. Если бы такая запись существовала, она соответствовала бы предыдущей версии пакета `2013.58`:

```
% pkg audit dropbear-2013.58
```

Если запись не найдена, добавьте новую запись для этой уязвимости.

```
% cd ${PORTSDIR}/security/vuxml  
% make newentry
```

Если уязвимость имеет идентификатор [MITRE CVE](#), можно использовать следующую команду:

```
% cd ${PORTSDIR}/security/vuxml  
% make newentry CVE_ID=CVE-YYYY-XXXX
```

где `CVE-YYYY-XXXX` является действительным идентификатором CVE.

Если уязвимость относится к FreeBSD Security Advisory, вместо этого можно использовать

следующую команду:

```
% cd ${PORTSDIR}/security/vuxml
% make newentry SA_ID=FreeBSD-SA-YY-XXXXXX.asc
```

где `FreeBSD-SA-YY-XXXXXX.asc` является опубликованным [FreeBSD Security Advisory](#).

Проверьте его синтаксис и форматирование:

```
% make validate
```

Предыдущая команда создает файл `vuln-flat.xml`. Его также можно создать с помощью:

```
% make vuln-flat.xml
```



Должен быть установлен хотя бы один из следующих пакетов: [textproc/libxml2](#), [textproc/jade](#).

Проверьте, что раздел `<affected>` записи соответствует правильным пакетам:

```
% pkg audit -f ${PORTSDIR}/security/vuxml/vuln-flat.xml dropbear-2013.58
```

Убедитесь, что запись не создает ложных совпадений в выводе.

Теперь проверьте, соответствуют ли записи правильные версии пакетов:

```
% pkg audit -f ${PORTSDIR}/security/vuxml/vuln-flat.xml dropbear-2013.58 dropbear-
2013.59
dropbear-2012.58 is vulnerable:
dropbear -- exposure of sensitive information, DoS
CVE: CVE-2013-4434
CVE: CVE-2013-4421
WWW: https://portaudit.FreeBSD.org/8c9b48d1-3715-11e3-a624-00262d8b701d.html

1 problem(s) in the installed packages found.
```

Предыдущая версия совпадает, а последняя — нет.

### 12.3.4. Контрольный список для новой записи в VuXML

- Проверьте название порта. Иногда имя проекта в вышестоящем репозитории не полностью совпадает с именем порта.
- Добавить все флейворы. Если у порта есть варианты, все имена пакетов должны быть добавлены в запись как `<package>`. Используйте следующий скрипт для генерации всех

имен пакетов с вариантами:

```
% for flavor in $(make -V FLAVORS); do FLAVOR="${flavor}" make -VPKGNAME;done
```

- Проверить, имеет ли порт `PORTEPOCH`. Приведённый выше фрагмент скрипта помогает в этом. Если порт использует `PORTEPOCH`, обязательно добавить его в тег `<range>`.
- Перепроверьте диапазоны. В случае диапазонов, ограниченных с обеих сторон, убедитесь, что элементы `<ge>` и `<lt>` находятся внутри одного тега `<range>`. В противном случае запись может определить перекрывающийся диапазон.
- Проверка производных версий. Если в вышестоящем проекте обнаружена уязвимость, проверьте, затронуты ли также производные или форки проекта, включённые в дерево портов. Например, если уязвимость обнаружена в [www/firefox](http://www/firefox), оцените, есть ли такая же уязвимость в производных, таких как [www/librewolf](http://www/librewolf), [www/waterfox](http://www/waterfox) или других подобных проектах. Включите все затронутые производные в запись VuXML, чтобы пользователи этих портов были проинформированы. Также проверьте наличие Linux-версий этого порта в дереве. Например, уязвимости в [databases/sqlite3](http://databases/sqlite3) скорее всего затрагивают и пакеты вроде [databases/linux-c7-sqlite3](http://databases/linux-c7-sqlite3).
- Не делайте коммит записи, не запустив сначала `make validate`.

# Глава 13. Что делать нужно, и что делать нельзя

## 13.1. Введение

Вот список часто встречающихся действий, которые нужно и которые нельзя делать во время процесса портирования. Проверьте порт по этому списку, и также проверьте порты в [базе сообщений PR](#), которые присланы другими людьми. Присылайте любые комментарии о портах, которые вы проверили, так, как это описано в статье о [Сообщениях об ошибках и общих замечаниях](#). Проверка портов в базе сообщений PR позволит нам быстрее коммитить их и удостовериться, что вы знаете, что делаете.

## 13.2. WRKDIR

Не пишите ничего в файлы вне каталога **WRKDIR**. Каталог **WRKDIR** является единственным местом, которое гарантированно будет доступно для записи во время построения порта (обратитесь к главе об [установке портов с CDROM](#) за примером построения портов из дерева, доступного только для чтения). Если вам нужно изменить какой-либо из файлов `pkg-*`, сделайте это, [переопределив переменную](#), но не перезаписывая их.

## 13.3. WRKDIRPREFIX

Добейтесь того, чтобы ваш порт принимал во внимание значение переменной **WRKDIRPREFIX**. Большинство портов об этом не заботятся. В частности, если вы обращаетесь к каталогу **WRKDIR** другого порта, заметьте, что его правильным местоположением является `WRKDIRPREFIXPORTSDIR/subdir/name/work`, а не `PORTSDIR/subdir/work` или `.CURDIR/../../subdir/name/work` или что-то подобное. Кроме того, если вы сами задаете **WRKDIR**, то должны поставить перед ним знак `${WRKDIRPREFIX}${.CURDIR}`.

## 13.4. Различение операционных систем и версий ОС

Вы можете встретиться с кодом, который требует модификаций или условной компиляции в зависимости от того, с какой версией FreeBSD Unix он работает. Предпочтительным способом отделения кода для версий FreeBSD является использование макросов `__FreeBSD_version` и `__FreeBSD__`, определённых в [sys/param.h](#). Если этот файл не подключен, добавьте код

```
#include <sys/param.h>
```

в нужном месте файла `.c`.

`__FreeBSD__` определён во всех версиях FreeBSD в качестве старшего номера версии системы.

Например, в FreeBSD 9.x `__FreeBSD__` определён со значением 9.

```
#if __FreeBSD__ >= 9
# if __FreeBSD_version >= 901000
    /* 9.1+ release specific code here */
# endif
#endif
```

Полный список значений `__FreeBSD_version` доступен в [Значения \\_\\_FreeBSD\\_version](#).

## 13.5. Написание чего-либо после `bsd.port.mk`

Не пишите ничего после строки `.include <bsd.port.mk>`. Этой строки можно избежать, включив в где-то в середине вашего файла Makefile файл `bsd.port.pre.mk`, и файл `bsd.port.post.mk` в конец.



Вам нужно включить либо пару файлов `bsd.port.pre.mk/bsd.port.post.mk`, либо только `bsd.port.mk`; не используйте оба этих метода одновременно.

В файле `bsd.port.pre.mk` определяются лишь несколько переменных, которые могут быть использованы в тестах из файла Makefile, в файле `bsd.port.post.mk` заданы остальные.

Вот некоторые важные переменные, определенные в файле `bsd.port.pre.mk` (это не полный список, для выяснения полного списка прочтите, пожалуйста, сам файл `bsd.port.mk`).

Переменная	Описание
<code>ARCH</code>	Архитектура машины в виде, получаемом по команде <code>uname -m</code> (например, <code>i386</code> )
<code>OPSYS</code>	Тип операционной системы, получаемый по команде <code>uname -s</code> (например, <code>FreeBSD</code> )
<code>OSREL</code>	Версия релиза операционной системы (например, <code>2.1.5</code> или <code>2.2.7</code> )
<code>OSVERSION</code>	Версия операционной системы в виде числа, та же, что и <code>__FreeBSD_version</code> .
<code>LOCALBASE</code>	Корень дерева "local" (например, <code>/usr/local</code> )
<code>PREFIX</code>	Куда, собственно, устанавливается порт (обратитесь к <a href="#">подробной информации о PREFIX</a> ).



Если вы задаете переменную `MASTERDIR`, делайте это до подключения `bsd.port.pre.mk`.

Вот несколько примеров того, что вы можете написать после `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
```

Вы не забываете об использовании табуляции вместо пробелов после `BROKEN=`.

## 13.6. Использование выражения `exec` в сценариях обёртках

Если порт устанавливает сценарий на языке shell, который служит для запуска другой программы, и если запуск этой программы является последним действием сценария, убедитесь, что запуск программы производится с использованием выражения `exec`, например:

```
#!/bin/sh
exec %%LOCALBASE%%/bin/java -jar %%DATADIR%%/foo.jar "$@"
```

Выражение `exec` заменяет процесс сценария на указанную программу. Если `exec` опущен, то процесс сценария во время работы программы остается в памяти, бесполезно потребляя системные ресурсы.

## 13.7. Поступайте разумно

Файл Makefile должен выполнять действия просто и небеспричинно. Если вы можете сделать что-то на несколько строк короче или более читабельно, сделайте это. В качестве примеров можно привести использование конструкций `.if` утилиты `make` вместо соответствующей конструкции `if` командного процессора, ненужность переопределения цели `do-extract` при возможности переопределения `EXTRACT*` и использование `GNU_CONFIGURE` вместо `CONFIGURE_ARGS += --prefix=${PREFIX}`.

Если вы обнаружите, что для выполнения чего-то приходится писать много нового кода, то, пожалуйста, просмотрите файл `bsd.port.mk` на предмет того, не содержит ли он решение именно вашей проблемы. Хотя его трудно читать, имеется много проблем, выглядящих сложными, для которых файл `bsd.port.mk` уже содержит быстрое решение.

## 13.8. Относитесь внимательно как к `CC`, так и `CXX`

Порт должен принимать во внимание как переменную `CC`, так и `CXX`. Под этим мы подразумеваем, что порт ни в коем случае не должен устанавливать значения этих переменных, переопределяя имеющиеся значения; вместо этого можно добавлять нужные значения к уже имеющимся. Это связано с тем, что параметры построения, относящиеся ко всем портам, могут быть заданы глобально.

Если порт не учитывает значения этих переменных, добавьте строку `NO_PACKAGE=ignores`

either `CC` or `CXX` в файл Makefile.

Далее следует пример файла Makefile, использующего как переменную `CC`, так и `CXX`. Обратите внимание на использование символов `?=`:

```
CC?= gcc
```

```
CXX?= g++
```

Вот пример, в котором не принимаются во внимание ни `CC`, ни `CXX`:

```
CC= gcc
```

```
CXX= g++
```

В системах FreeBSD обе переменные `CC` и `CXX` могут быть определены в файле `/etc/make.conf`. В первом примере задаётся значение, если оно ранее не было определено в `/etc/make.conf`, что сохраняет любые определения, данные на уровне системы в целом. Второй пример переопределяет всё, что было задано ранее.

## 13.9. Относитесь внимательно к `CFLAGS`

Порт должен учитывать переменную `CFLAGS`. Под этим мы подразумеваем, что порт ни в коем случае не должен устанавливать значения этой переменной, переопределяя имеющиеся значения; вместо этого можно добавлять нужные значения к уже имеющимся. Это связано с тем, что параметры построения, относящиеся ко всем портам, могут быть заданы глобально.

Если порт не учитывает значения этой переменной, добавьте строку `NO_PACKAGE=ignores cflags` в файл Makefile.

Далее следует пример файла Makefile, использующего переменную `CFLAGS`. Обратите внимание на использование символов `+=`:

```
CFLAGS+= -Wall -Werror
```

А вот пример, в котором не учитывается значение переменной `CFLAGS`:

```
CFLAGS= -Wall -Werror
```

В системе FreeBSD переменная `CFLAGS` определена в файле `/etc/make.conf`. В первом примере к переменной `CFLAGS` добавляются дополнительные флаги, при этом сохраняются все

определения, данные ранее на уровне системы. Во втором примере всё, что было задано ранее, игнорируется.

Из сторонних файлов Makefile следует удалить флаги оптимизации. Общесистемные флаги оптимизации находятся в системной переменной `CFLAGS`. Пример из немодифицированного Makefile:

```
CFLAGS= -O3 -funroll-loops -DHAVE_SOUND
```

При использовании системных флагов оптимизации Makefile станет похожим на следующий пример:

```
CFLAGS+= -DHAVE_SOUND
```

## 13.10. Подробные логи сборки

Заставьте систему сборки портов отображать все команды, выполняемые на этапе сборки. Полные логи сборки критически важны для отладки проблем с портами.

Пример неинформативного лога сборки (плохой):

```
CC      source1.o
CC      source2.o
CCLD    someprogram
```

Пример подробного журнала сборки (хороший):

```
cc -O2 -pipe -I/usr/local/include -c -o source1.o source1.c
cc -O2 -pipe -I/usr/local/include -c -o source2.o source2.c
cc -o someprogram source1.o source2.o -L/usr/local/lib -lsomelib
```

Некоторые системы сборки, такие как CMake, ninja и GNU configure, настроены на подробное ведение журнала в рамках инфраструктуры портов. В других случаях портам могут потребоваться индивидуальные изменения.

## 13.11. Обратная связь

Посылайте подходящие изменения/патчи автору/сопровождающему для включения в следующий релиз. Это только сделает вашу работу гораздо легче при выходе следующего релиза.

## 13.12. README.html

README.html не является частью порта и генерируется при помощи `make readme`. Не

включайте этот файл в патчи или коммиты.



Если не удастся выполнить `make readme`, убедитесь, что значение по умолчанию `ECHO_MSG` не изменено внутри порта.

## 13.13. Пометка неустанавливаемого порта как **BROKEN**, **FORBIDDEN** или **IGNORE**

В некоторых случаях пользователи не должны допускаться к установке порта. Для того, чтобы сообщить пользователю, что порт не следует устанавливать, имеется несколько `make`-переменных, которые могут быть использованы в файле Makefile порта. Значения следующих `make`-переменных будут причиной, возвращаемой пользователям, по которой порт отказывает в установке. Пожалуйста, используйте корректные `make`-переменные, так как каждая переменная `make` передает абсолютно различный смысл как для пользователей, так и для автоматизированных систем, которые полагаются на файлы Makefile, таких как [кластер построения портов](#), [FreshPorts](#).

### 13.13.1. Переменные

- **BROKEN** предназначена для портов, которые в настоящее время не компилируются, не устанавливаются или не удаляются правильно. Следует использовать, когда проблема считается временной.

В особых случаях кластер построения будет продолжать попытки собрать их, чтобы показать, решена ли основная проблема. (Однако, как правило, кластер запускается без этой возможности.)

В частности, используйте **BROKEN**, когда порт:

- не компилируется
  - выполняет со сбоями конфигурирование или процесс установки
  - устанавливает файлы ввне `${PREFIX}`
  - не удаляет полностью все свои файлы при деинсталляции (тем не менее, это может быть допустимо, и подходит для портов, оставляющих после себя файлы, измененные пользователем)
  - имеет проблемы во время выполнения на системах, где он должен работать нормально.
- **FORBIDDEN** используется для портов, которые содержат уязвимости в информационной безопасности или являются потенциально вредными в плане обеспечения информационной безопасности системы FreeBSD при установке данного порта (например: заведомо небезопасная программа или программа, которая предоставляет легко взламываемые сервисы). Порты должны помечаться как **FORBIDDEN**, как только в конкретном программном обеспечении обнаружилась уязвимость, но обновление выпущено не было. В идеальном случае порты должны обновляться максимально быстро после обнаружения уязвимости, чтобы уменьшить число уязвимых хостов

FreeBSD (нам нравится иметь репутацию безопасной системы), однако иногда случается значительный временной разрыв между обнаружением уязвимости и выходом обновлённого релиза уязвимого программного обеспечения. Не помечайте порт как **FORBIDDEN**, если причина не вызвана соображениями информационной безопасности.

- **IGNORE** предназначена для портов, которые не должны строиться по какой-либо другой причине. Следует использовать для портов, в случае когда проблема считается структурной. Кластер построения ни при каких условиях не будет строить порты, помеченные как **IGNORE**. В частности, используйте **IGNORE**, когда порт:
  - не работает на установленной версии FreeBSD
  - имеет `distfile`, который не может быть автоматически загружен из-за ограничений лицензирования
  - не работает с некоторыми другими установленными портами (например, порт зависит от [www/drupal7](#), но установлен [www/drupal8](#))



Если порт будет конфликтовать с уже установленным портом (например, если они устанавливают файл в то же место, но с иным функциональным назначением), то **используйте вместо этого CONFLICTS**. **CONFLICTS** сам установит значение **IGNORE**.

### 13.13.2. Заметки о реализации

Не заключайте значения переменных **BROKEN**, **IGNORE** и связанных с ними в кавычки. Из-за способа отображения информации пользователю, формулировка сообщений для каждой переменной отличается:

```
BROKEN= fails to link with base -lcrypto
```

```
IGNORE= unsupported on recent versions
```

и в результате `make describe` выведен информацию в таком виде :

```
==> foobar-0.1 is marked as broken: fails to link with base -lcrypto.
```

```
==> foobar-0.1 is unsupported on recent versions.
```

## 13.14. Архитектурные соображения

### 13.14.1. Общие замечания об архитектурах

FreeBSD работает на гораздо большем количестве архитектур процессоров, чем только хорошо известные x86-совместимые. Некоторые порты имеют ограничения, характерные

для одной или нескольких из этих архитектур.

Для списка поддерживаемых архитектур выполните:

```
cd ${SRCDIR}; make targets
```

Значения отображаются в форме `TARGET/TARGET_ARCH`. Переменная только для чтения `ARCH` в `ports` устанавливается на основе значения `TARGET_ARCH`. Makefile в портах должны проверять значение этой переменной.

### 13.14.2. Пометка порта как архитектурно нейтрального

Порты, которые не имеют зависимых от архитектуры файлов или требований, определяются установкой `NO_ARCH=yes`.

Пакеты, собранные из таких портов, имеют строку архитектуры, оканчивающуюся на `:*` (архитектура с подстановочным символом), в отличие от, например, `freebsd:13:x86:64` (архитектура amd64).



`NO_ARCH` предназначен для указания того, что нет необходимости собирать пакет для каждой из поддерживаемых архитектур. Цель состоит в том, чтобы сократить количество ресурсов, затрачиваемых на сборку и распространение пакетов, таких как сетевой трафик и дисковое пространство на зеркалах и на носителях дистрибутива. Однако в настоящее время наша инфраструктура пакетов (например, менеджеры пакетов, зеркала и сборщики пакетов) не настроена для полного использования преимуществ `NO_ARCH`.

### 13.14.3. Пометка порта как игнорируемого только на определенных архитектурах

- Чтобы пометить порт как `IGNORE` только для определенных архитектур, существуют две другие удобные переменные, которые автоматически устанавливают `IGNORE: ONLY_FOR_ARCHS` и `NOT_FOR_ARCHS`. Примеры:

```
ONLY_FOR_ARCHS= i386 amd64
```

```
NOT_FOR_ARCHS= ia64 sparc64
```

Пользовательское сообщение `IGNORE` можно задать с помощью `ONLY_FOR_ARCHS_REASON` и `NOT_FOR_ARCHS_REASON`. Для отдельных архитектур возможны записи с `ONLY_FOR_ARCHS_REASON_ARCH` и `NOT_FOR_ARCHS_REASON_ARCH`.

- Если порт загружает и устанавливает бинарные файлы i386, установите `IA32_BINARY_PORT`. Если эта переменная задана, `/usr/lib32` должен присутствовать для IA32-версий

библиотек, а ядро должно поддерживать совместимость с IA32. Если одно из этих двух условий не выполняется, **IGNORE** будет установлен автоматически.

#### 13.14.4. Специфические аспекты для кластеров

- Некоторые порты пытаются оптимизировать себя под конкретную машину, на которой они собираются, указывая компилятору **-march=native**. Этого следует избегать: либо добавить этот параметр в опцию, отключенную по умолчанию, либо удалить его полностью.

В противном случае стандартный пакет, созданный кластером сборки, может не запускаться на каждой машине с данной **ARCH**.

### 13.15. Пометка порта на удаление с **DEPRECATED** или **EXPIRATION\_DATE**

Помните, что **BROKEN** и **FORBIDDEN** будут использованы как временное средство, если порт не является работающим. Постоянно неработоспособные порты должны полностью удаляться из дерева.

В подходящих ситуациях пользователи могут быть оповещены о предстоящем удалении через переменные **DEPRECATED** и **EXPIRATION\_DATE**. Первое - это просто строка, сообщающая причину запланированного удаления порта; вторая является строкой в формате ISO 8601 (YYYY-MM-DD). Обе будут показаны пользователю.

Переменную **DEPRECATED** можно установить без использования **EXPIRATION\_DATE** (в частности, при рекомендации новой версии порта), но обратный порядок не имеет никакого смысла.



При пометке порта как **DEPRECATED**, если существуют альтернативные порты, которые можно использовать вместо устаревающего, удобно упомянуть их в сообщении коммита.

Не существует установленного правила о том, насколько заранее нужно уведомлять. Текущая практика предполагает один месяц для проблем, связанных с безопасностью, и два месяца для проблем сборки. Это также дает заинтересованным коммиттерам немного времени на исправление проблем.

### 13.16. Избегайте использования конструкции **.errgo**

Правильным способом подать сигнал для Makefile о том, что порт не может быть установлен из-за какого-то внешнего фактора (например, пользователь указал недопустимую комбинацию опций построения), является установка непустого значения для **IGNORE**. Это значение будет сформатировано и показано пользователю во время **make install**.

Использование для этих целей **.errgo** является распространенной ошибкой. Проблема в том, что в этой ситуации будут повреждены многие инструменты автоматизации, работающие с деревом портов. Наибольшим образом это распространено при попытке построить

/usr/ports/INDEX (смотрите [Запуск make describe](#)). Тем не менее, даже более простые команды, такие как `make maintainer`, в этом случае также вернут ошибку. Это не является приемлемым.

*Пример 115. Как избежать использования .error*

Из следующих двух вариантов строки файла Makefile первый приведёт к неудачному завершению работы `make index`, а второй - нет:

```
.error "option is not supported"
```

```
IGNORE=option is not supported
```

## 13.17. Использование sysctl

Использование `sysctl` не рекомендуется, кроме как при выполнении целей. Это вызвано тем, что вычисление любых `makevar`, таких как во время команды `make index`, с необходимостью запуска этой команды, еще больше замедляет весь процесс.

`sysctl(8)` следует всегда использовать через переменную `SYSCTL`, поскольку она содержит полностью заданный путь, и при необходимости может быть переопределена.

## 13.18. Меняющиеся дистрибутивные файлы

Иногда авторы программного обеспечения изменяют содержимое выпущенных дистрибутивных файлов, не меняя их названия. Убедитесь, что изменения официальные и были выполнены автором. В прошлом случалось, что дистрибутивный файл тихо изменялся на серверах загрузки с целью нанесения вреда или компрометации безопасности конечного пользователя.

Отложите старый `distfile` в сторону, загрузите новый, распакуйте их и сравните содержимое с помощью `diff(1)`. Если ничего подозрительного нет, обновите `distinfo`.



Убедитесь, что вы выделили основные различия в PR и журнале коммитов, чтобы другие люди знали, что ничего плохого не произошло.

Свяжитесь с автором этого программного обеспечения для подтверждения изменений.

## 13.19. Используйте стандарты POSIX

В большинстве случаев порты FreeBSD ожидают соответствия стандарту POSIX. Некоторые программы и системы сборки делают предположения, основанные на конкретной операционной системе или окружении, что может вызывать проблемы при использовании в порте.

Не используйте `/proc`, если есть другие способы получить информацию. Например,

`setprogname(argv[0])` в `main()`, а затем `getprogname(3)` для получения имени исполняемого файла.

Не полагайтесь на поведение, не документированное в POSIX.

Не записывайте метки времени в критическом пути приложения, если оно работает и без них. Получение меток времени может быть медленным в зависимости от точности меток времени в ОС. Если метки времени действительно необходимы, определите, насколько точными они должны быть, и используйте API, которое, согласно документации, предоставляет только необходимую точность.

Ряд простых системных вызовов (например, `gettimeofday(2)`, `getpid(2)`) работают намного быстрее в Linux® по сравнению с любой другой операционной системой из-за кэширования и используемой оптимизации `vsyscall`. Не полагайтесь на их дешевизну в критичных к производительности приложениях. В целом, старайтесь избегать системных вызовов там, где это возможно.

Не полагайтесь на специфичное для Linux® поведение сокета. В частности, отличаются размеры буфера сокета по умолчанию (выполните вызов `setsockopt(2)` с `SO_SNDBUF` и `SO_RCVBUF`, и в то время как в Linux® при заполнении буфера сокета `send(2)` блокируется, FreeBSD возвращает ошибку и устанавливает `ENOBUFS` в качестве значения `errno`.

Если требуется рассчитывать на нестандартное поведение, инкапсулируйте это должным образом в общий для всех API с проверкой поведения на этапе конфигурации, и если требуемое поведение не найдено, прекращайте выполнение.

Используйте [страницы справочника](#) для проверки, относится ли функция к интерфейсу POSIX (ищите раздел "STANDARDS" на странице справочника).

Не рассчитывайте на то, что в качестве `/bin/sh` используется `bash`. Убедитесь, что командная строка, переданная в `system(3)`, будет работать в POSIX-совместимой оболочке.

Список основных `bash`-измов расположен [здесь](#).

Проверьте, что используемые заголовочные файлы включены в POSIX или список, рекомендуемый страницей справочника, т.к. например, забыть подключить `sys/types.h` - не такая уж проблема в Linux®, однако это не так во FreeBSD.

## 13.20. Разное

Файлы `pkg-descr` и `pkg-plist` должны проверяться дважды. Если вы пересматриваете порт и думаете, что его можно описать иначе, сделайте это.

Будьте внимательны с юридическими вопросами! Не делайте из нас нелегальных распространителей ПО!

# Глава 14. Примерный Makefile

Образец Makefile, который можно использовать для создания нового порта.

Вам рекомендуется следовать этому формату (соблюдая [порядок](#) следования переменных, пустые строки между разделами, и так далее). Этот формат разработан для того, чтобы важная информация была легко найдена. Обратитесь [главе о тестировании](#), чтобы узнать больше о lint, утилитах для форматирования и проверки файла Makefile.

```
PORTNAME=  xdvi ①
DISTVERSION=  18.2
CATEGORIES=  print
MASTER_SITES=  ${MASTER_SITE_XCONTRIB} ②
MASTER_SITE_SUBDIR=  applications
PKGNAMEPREFIX=  ja-
DISTNAME=  xdvi-pl18
EXTRACT_SUFX=  .tar.Z ③

PATCH_SITES=  ftp://ftp.sra.co.jp/pub/X11/japanese/ ④
PATCHFILES=  xdvi-18.patch1.gz xdvi-18.patch2.gz
PATCH_DIST_STRIP=  -p1 ⑤

MAINTAINER=  asami@FreeBSD.org ⑥
COMMENT=  DVI Previewer for the X Window System
WWW=  http://xdvi.sourceforge.net/

LICENSE=  BSD2CLAUSE ⑦
LICENSE_FILE=  ${WRKSRC}/LICENSE

RUN_DEPENDS=  gs:print/ghostscript ⑧

USES=  gmake ⑨

⑩
IS_INTERACTIVE=  yes ⑪
WRKSRC=  ${WRKDIR}/xdvi-new ⑫
GNU_CONFIGURE=  yes ⑬

⑭
OPTIONS_DEFINE=  DOCS EXAMPLES FOO
OPTIONS_DEFAULT=FOO
OPTIONS_SUB=  yes ⑮

FOO_DESC=  Enable foo support
FOO_CONFIGURE_ENABLE=  foo

⑯
MY_FAVORITE_RESPONSE=  "yeah, right"
```

⑰

```

pre-fetch:
    i go fetch something, yeah

post-patch:
    мне кое-что сделать после применения патча, великолепно

pre-install:
    и потом еще кое-что перед установкой, ого

.include <bsd.port.mk> ⑱

```

- ① Секция для описания самого порта и его главного сайта: первыми идут переменные `PORTNAME` и `PORTVERSION` или `DISTVERSION*`, на ними `CATEGORIES`, затем `MASTER_SITES`, после которой идет `MASTER_SITE_SUBDIR`. Если нужно, то после нее идут `PKGNAMEPREFIX` и `PKGNAME_SUFFIX`. Затем следуют `DISTNAME`, `EXTRACT_SUFFIX` и/или `DISTFILES`, и уже потом, если нужно, `EXTRACT_ONLY`.
- ② Не забывайте про завершающую косую черту (`/`), если вы не используете макросы `MASTER_SITE_*`.
- ③ Задайте это, если исходный код поставляется не в виде стандартного файла `".tar.gz"`.
- ④ Секция патчей — может быть пустой.
- ⑤ Если распространяемые патчи не были созданы относительно `${WRKSRC}`, возможно, это потребует исправления вручную.
- ⑥ Сопровождающий; **обязательное поле!** Это человек, который добровольно занимается обновлениями порта и неисправностями при построении, и которому пользователь может направлять вопросы и сообщения об ошибках. Для сохранения как можно более высокого качества Коллекции Портов мы больше не принимаем новые порты, назначенные на [ports@FreeBSD.org](mailto:ports@FreeBSD.org).
- ⑦ Лицензия — не следует оставлять пустым.
- ⑧ Зависимости — могут быть пустыми.
- ⑨ Если порт требует GNU `make` вместо стандартного FreeBSD `make` (`make(1)`) для сборки. Например, некоторым приложениям X требуется выполнение `xmkmf -a`, в этом случае порту понадобится `USES=imake`.
- ⑩ Этот раздел посвящён другим стандартным переменным `bsd.port.mk`, которые не относятся ни к одной из вышеперечисленных категорий.
- ⑪ Если порты задают интерактивные вопросы во время настройки, сборки, установки.
- ⑫ Если извлечение происходит в каталог, отличный от `DISTNAME`.
- ⑬ Если требуется запустить скрипт `configure`, сгенерированный GNU `autoconf`.
- ⑭ Этот раздел предназначен для настройки параметров портов.
- ⑮ Установите `OPTIONS_SUB`, если параметры изменят список файлов в `plist`.
- ⑯ В правилах ниже используются нестандартные переменные.
- ⑰ Специальные правила, в порядке их вызова фреймворком портов.

⑱ Наконец, эпилог.

# Глава 15. Порядок переменных в Makefile портов

Первые разделы Makefile всегда должны идти в одном и том же порядке. Это стандартное правило позволяет любому легко читать любой порт, не тратя время на поиск переменных в произвольном порядке.



Описанные здесь разделы и переменные являются обязательными в обычном порте. В подчиненном порте многие разделы и переменные могут быть пропущены.



Каждый следующий блок должен быть отделен от предыдущего одним пустым пробелом.

В следующих блоках устанавливайте только переменные, которые требуются для порта. Определяйте эти переменные в порядке, указанном здесь.

## 15.1. Блок **PORTNAME**

Этот блок является наиболее важным. Он определяет имя порта, версию, расположение файла дистрибутива и категорию. Переменные должны быть в следующем порядке:

- **PORTNAME \* PORTVERSION[1]**
- **DISTVERSIONPREFIX \* DISTVERSION[1]**
- **DISTVERSIONSUFFIX**
- **PORTREVISION**
- **PORTEPOCH**
- **CATEGORIES**
- **MASTER\_SITES**
- **MASTER\_SITE\_SUBDIR** (устарело)
- **PKGNAMEPREFIX**
- **PKGNAMEPREFIX**
- **DISTNAME**
- **EXTRACT\_SUFX**
- **DISTFILES**
- **DIST\_SUBDIR**
- **EXTRACT\_ONLY**



Может быть использован только один из параметров — **PORTVERSION** или

## 15.2. Блок **PATCHFILES**

Этот блок является необязательным. Переменные:

- **PATCH\_SITES**
- **PATCHFILES**
- **PATCH\_DIST\_STRIP**

## 15.3. Блок **MAINTAINER**

Этот блок является обязательным. Переменные следующие:

- **MAINTAINER**
- **COMMENT**
- **WWW**

## 15.4. Блок **LICENSE**

Этот блок является необязательным, хотя настоятельно рекомендуется. Переменные:

- **LICENSE**
- **LICENSE\_COMB**
- **LICENSE\_GROUPS** или **LICENSE\_GROUPS\_NAME**
- **LICENSE\_NAME** или **LICENSE\_NAME\_NAME**
- **LICENSE\_TEXT** или **LICENSE\_TEXT\_NAME**
- **LICENSE\_FILE** или **LICENSE\_FILE\_NAME**
- **LICENSE\_PERMS** или **LICENSE\_PERMS\_NAME\_**
- **LICENSE\_DISTFILES** или **LICENSE\_DISTFILES\_NAME**

Если имеется несколько лицензий, отсортируйте различные переменные **LICENSE\_VAR\_NAME** по названию лицензии.

## 15.5. Общие сообщения **BROKEN/IGNORE/DEPRECATED**

Этот блок необязателен. Переменные:

- **DEPRECATED**
- **EXPIRATION\_DATE**
- **FORBIDDEN**
- **BROKEN**

- `BROKEN_*`
- `IGNORE`
- `IGNORE_*`
- `ONLY_FOR_ARCHS`
- `ONLY_FOR_ARCHS_REASON*`
- `NOT_FOR_ARCHS`
- `NOT_FOR_ARCHS_REASON*`



`BROKEN_*` и `IGNORE_*` могут быть любыми общими переменными, например, `IGNORE_amd64`, `BROKEN_FreeBSD_10` и т.д. За исключением переменных, которые зависят от `USES`, их следует размещать в `USES` и `USE_x`. Например, `IGNORE_WITH_PHP` работает только если установлен `php`, а `BROKEN_SSL` — только если установлен `ssl`.

Если порт помечен как `BROKEN` при выполнении определённых условий, и эти условия можно проверить только после включения `bsd.port.options.mk` или `bsd.port.pre.mk`, то такие переменные должны быть установлены позже, в [Остальные Переменные](#).

## 15.6. Блок зависимостей

Этот блок необязателен. Переменные:

- `FETCH_DEPENDS`
- `EXTRACT_DEPENDS`
- `PATCH_DEPENDS`
- `BUILD_DEPENDS`
- `LIB_DEPENDS`
- `RUN_DEPENDS`
- `TEST_DEPENDS`

## 15.7. Флейворы

Этот блок необязателен.

Начните этот раздел с определения `FLAVORS`. Затем рассмотрите возможные вспомогательные инструменты флейворов. Дополнительную информацию см. в разделе [Использование флейворов \(FLAVORS\)](#).

Конструкции, устанавливающие переменные, недоступные в виде помощников, с использованием `.if ${FLAVOR:U} == foo`, должны быть размещены в соответствующих разделах ниже.

## 15.8. USES и USE\_x

Начните этот раздел с определения `USES`, а затем возможных `USE_x`.

Держите связанные переменные рядом. Например, если используется `USE_GITHUB`, всегда размещайте переменные `GH_*` сразу после неё.

## 15.9. Стандартные переменные `bsd.port.mk`

Этот блок раздела предназначен для переменных, которые могут быть определены в `bsd.port.mk` и не принадлежат ни к одному из предыдущих блоков разделов.

Порядок не важен, однако старайтесь держать схожие переменные вместе. Например, переменные `uid` и `gid` `USERS` и `GROUPS`. Конфигурационные переменные `CONFIGURE_*` и `*_CONFIGURE`. Списки файлов и директорий `PORTDOCS` и `PORTEXAMPLES`.

## 15.10. Параметры и помощники

Если порт использует [фреймворк опций](#), сначала определите `OPTIONS_DEFINE` и `OPTIONS_DEFAULT`, затем остальные переменные `OPTIONS_*`, далее описания `*_DESC`, а затем вспомогательные опции. Старайтесь сортировать их все в алфавитном порядке.

*Пример 116. Пример порядка переменных-опций*

Опции `FOO` и `BAR` не имеют стандартного описания, поэтому его необходимо написать. Остальные опции уже имеют описание в `Mk/bsd.options.desc.mk`, поэтому его создание не требуется. Переменные `DOCS` и `EXAMPLES` используют вспомогательные цели для установки своих файлов, они приведены здесь для полноты, хотя относятся к разделу [Цели](#), поэтому перед ними могут быть вставлены другие переменные и цели.

```
OPTIONS_DEFINE= DOCS EXAMPLES FOO BAR
OPTIONS_DEFAULT= FOO
OPTIONS_RADIO= SSL
OPTIONS_RADIO_SSL= OPENSSSL GNUTLS
OPTIONS_SUB= yes

BAR_DESC= Enable bar support
FOO_DESC= Enable foo support

BAR_CONFIGURE_WITH= bar=${LOCALBASE}
FOO_CONFIGURE_ENABLE= foo
GNUTLS_CONFIGURE_ON= --with-ssl=gnutls
OPENSSSL_CONFIGURE_ON= --with-ssl=openssl

post-install-DOCS-on:
    ${MKDIR} ${STAGEDIR}${DOCSDIR}
    cd ${WRKSRC}/doc && ${COPYTREE_SHARE} . ${STAGEDIR}${DOCSDIR}
```

```
post-install-EXAMPLES-on:
  ${MKDIR} ${STAGEDIR}${EXAMPLESDIR}
  cd ${WRKSRCS}/ex && ${COPYTREE_SHARE} . ${STAGEDIR}${EXAMPLESDIR}
```

## 15.11. Остальные переменные

И затем, остальные переменные, которые не упоминались в предыдущих блоках.

## 15.12. Цели

После определения всех переменных можно определить дополнительные цели `make(1)`. Следует располагать `pre-` перед `post-` и в том же порядке, в котором выполняются различные этапы:

- `fetch`
- `extract`
- `patch`
- `configure`
- `build`
- `install`
- `test`

При использовании опций `helpers` для цели `target` сохраняйте их в алфавитном порядке, но оставляйте `-on` перед `-off`. Если также используется основная цель, размещайте её перед дополнительными:



```
post-install:
  # install generic bits

post-install-DOCS-on:
  # Install documentation

post-install-X11-on:
  # Install X11 related bits

post-install-X11-off:
  # Install bits that should be there if X11 is disabled
```

# Глава 16. Актуализация

Коллекция Портов FreeBSD постоянно изменяется. Здесь находится некоторая информация о том, как поддерживать её в актуальном состоянии.

## 16.1. FreshPorts

Самым простым способом отслеживать уже произошедшие обновления является подписка на [FreshPorts](#). Для мониторинга вы можете выбрать несколько портов. Сопровождающим порта настоятельно рекомендуется подписаться здесь, потому что они будут получать уведомления не только о собственных изменениях, но и об изменениях, сделанных любым другим коммиттером FreeBSD. (Это часто необходимо для синхронизации с изменениями на более низком технологическом уровне, хотя более корректным было бы получение предупреждений от тех, кто вносит подобные изменения, иногда этот этап пропускается или он просто непрактичен. Кроме того, в некоторых случаях изменения по своей природе весьма незначительны. Мы полагаем, что любой разработчик в таких ситуациях будет руководствоваться здравым смыслом).

Если вы хотите использовать FreshPorts, то вам нужна только учётная запись. Если регистрационный адрес вашей электронной почты будет иметь вид [@FreeBSD.org](#), то справа на Web-страницах вы увидите дополнительную ссылку. Для тех из вас, кто уже получил учётную запись FreshPorts, но не использовал собственный адрес электронной почты [@FreeBSD.org](#), достаточно сменить адрес на [@FreeBSD.org](#), подписаться, а затем сменить его обратно.

Во FreshPorts имеется также функция проверки правильности, которая автоматически проверяет каждое изменение, внесённое в дерево портов FreeBSD. Если вы подпишетесь на эту услугу, то будете оповещаться обо всех ошибках, обнаруженных FreshPorts при проверке внесённых вами изменений.

## 16.2. Web-интерфейс к хранилищу исходных текстов

Файлы в хранилище исходных текстов можно просматривать при помощи Web-интерфейса. Изменения, которые касаются в целом всей системы портов, теперь документируются в файле [CHANGES](#). Изменения, касающиеся отдельных портов, отражаются теперь в файле [UPDATING](#). Однако однозначный ответ на любой вопрос можно найти, только прочитав исходных код [bsd.port.mk](#) и связанных с ним файлов.

## 16.3. Список рассылки FreeBSD, посвящённый портам

Если вы поддерживаете порты, то должны следить за [Список рассылки, посвящённый Портам FreeBSD](#). О важных изменениях, отражающихся на работе портов, будет сообщаться здесь, а затем они переносятся в [CHANGES](#).

Если данный список рассылки слишком загружен сообщениями, вы можете отслеживать [Список рассылки анонсов FreeBSD Ports](#), который модерирован и не является местом для дискуссий.

## 16.4. Кластер построения портов FreeBSD

Одной из наименее известных сильных сторон FreeBSD является тот факт, что для непрерывного построения Коллекции Портов для каждого из основных релизов ОС для каждой архитектуры уровня поддержки Tier-1 выделен целый кластер машин.

Отдельные порты собираются, если они специально не помечены как **IGNORE**. Для портов, помеченных как **BROKEN**, попытки будут продолжены для того, чтобы увидеть, если основная проблема была решена. (Это сделано через использование переменной **TRYBROKEN** для Makefile порта.)

## 16.5. Portscout: сканер дистрибутивных файлов портов FreeBSD

Кластер построения выделен для выполнения самого последнего релиза каждого из портов, дистрибутивные файлы которых уже были сгружены. Однако из-за постоянных изменений в Internet дистрибутивные файлы могут быстро исчезать. **Portscout**, средство сканирования дистрибутивных файлов FreeBSD пытается опросить каждый из сайтов, доступных для сгрузки каждого из портов, для определения того, доступны ли ещё дистрибутивные файлы. Portscout может готовить отчёты в HTML и рассылать электронные письма об имеющихся обновлениях для портов тем, кто это запрашивает. Сопровождающие периодически запрашивают наличие изменений либо вручную, либо используя ленту RSS.

Главная страница Portscout отображает email сопровождающего порта, количество портов, за которые ответственен сопровождающий, количество портов с новыми дистрибутивными файлами и процент устаревших портов. Функция поиска выполняет поиск сопровождающего по адресу электронной почты и позволяет выбирать между всеми портами или только устаревшими.

При щелчке по адресу электронной почты сопровождающего отображается список всех его портов, разделённых по категориям, вместе с текущим номером версии, информацией о наличии новой версии, временем последнего обновления порта и временем его последней проверки. Функция поиска на этой странице позволяет пользователю выполнять поиск конкретного порта.

По щелчку на название порта в списке отображается информация о порте [FreshPorts](#).

Другим полезным ресурсом является [репозиторий Portscout](#).

# Глава 17. Использование макроса **USES**

## 17.1. Введение в **USES**

Макросы **USES** упрощают объявление требований и настроек для порта. Они могут добавлять зависимости, изменять поведение при сборке, добавлять метаданные в пакеты и так далее, просто выбирая предустановленные значения.

Каждый раздел в этой главе описывает возможное значение для **USES**, а также его возможные аргументы. Аргументы добавляются к значению после двоеточия (:). Несколько аргументов разделяются запятыми (,).

*Пример 117. Использование нескольких значений*

```
USES=  bison perl
```

*Пример 118. Добавление аргумента*

```
USES=  tar:xz
```

*Пример 119. Добавление нескольких аргументов*

```
USES=  drupal:7,theme
```

*Пример 120. Смешивая Все Вместе*

```
USES=  postgresql:9.3+ сре python:2.7,build
```

## 17.2. **7z**

Возможные аргументы: (нет), **p7zip**, **partial**

Извлечение с использованием **7z(1)** вместо **bsdtar(1)** и устанавливает **EXTRACT\_SUFX=.7z**. Опция **p7zip** добавляет зависимость от **7z** из [archivers/p7zip](#), если версия из базовой системы не может извлечь файлы. **EXTRACT\_SUFX** не изменяется, если используется опция **partial**, это может быть полезно, если основной дистрибутивный файл не имеет расширения **.7z**.

## 17.3. `ada`

Возможные аргументы: (нет), `6`, `12`, (запуск)

Зависит от компилятора с поддержкой Ada и устанавливает `CC` соответствующим образом. По умолчанию используется `gcc6-aux` из портов.

## 17.4. `angr`

Возможные аргументы: `binaries`, `nose`

Обеспечить поддержку портов, требующих [платформу бинарного анализа angr](#).

Если присутствует аргумент `binaries`, для тестирования порта требуются специальные бинарные файлы `angr`.

Если присутствует аргумент `nose`, порт использует `nosetests` для цели тестирования. Этот аргумент подразумевает `USES=python:test`.

Фреймворк предоставляет следующие переменные, которые могут быть установлены портом:

### `ANGR_VERSION`

Версия программ проекта `angr`.

### `ANGR_BINARIES_TAGNAME`

Имя тега для бинарных файлов `angr`.

### `ANGR_NOSETESTS`

Путь к программе `nosetests`.

## 17.5. `ansible`

Возможные аргументы: `env`, `module`, `plugin`

Обеспечивает поддержку портов, зависящих от пакета `sysutils/ansible`.

Если присутствует аргумент `env`, порт не зависит от `sysutils/ansible`, но требует установки некоторых переменных Ansible.

Если присутствует аргумент `module`, то порт является модулем Ansible.

Если присутствует аргумент `plugin`, то порт является плагином Ansible.

Фреймворк предоставляет следующие переменные порту:

### `ANSIBLE_CMD`

Путь к программе `ansible`.

### ANSIBLE\_DOC\_CMD

Путь к программе ansible-doc.

### ANSIBLE\_RUN\_DEPENDS

RUN\_DEPENDS с портом Ansible.

### ANSIBLE\_DATADIR

Путь к корню структуры каталогов, где хранятся все модули и плагины Ansible.

### ANSIBLE\_ETCDIR

Путь к каталогу etc Ansible.

### ANSIBLE\_PLUGINS\_PREFIX

Путь к директории "plugins" в `${ANSIBLE_DATADIR}`.

### ANSIBLE\_MODULESDIR

Путь к каталогу для локальных модулей Ansible.

### ANSIBLE\_PLUGINS\_DIR

Путь к каталогу для локальных плагинов Ansible.

### ANSIBLE\_PLUGIN\_TYPE

Тип плагина Ansible (например, "connection", "inventory" или "vars").

## 17.6. apache

Возможные аргументы: (нет), `2.4`, `build`, `run`, `server`

Обеспечивает поддержку портов, зависящих от веб-сервера Apache.

Аргумент `version` можно использовать для указания конкретной версии Apache httpd. Можно задать определённую версию (`USES=apache:2.4`), минимальную версию (`USES=apache:2.4+`) или максимальную версию (`USES=apache:-2.4`).

Если указан аргумент `build`, к порту добавляется зависимость для сборки.

Если указан аргумент `run`, к порту добавляется зависимость времени выполнения.

Если указан аргумент `server`, это означает, что порт является серверным.

Фреймворк предоставляет следующие переменные, которые могут быть установлены портом:

### AP\_FAST\_BUILD

Автоматическая сборка модуля

### AP\_GENPLIST

Автоматическое создание `PLIST` плюс добавление модуля в отключенном состоянии в `httpd.conf` (только если нет файла `pkg-plist`)

## MODULENAME

Имя модуля Apache. По умолчанию: `${PORTNAME}`

## SHORTMODNAME

Краткое название модуля Apache. По умолчанию: `${MODULENAME:S/mod_//}`

## SRC\_FILE

Исходный файл модуля APACHE. По умолчанию: `${MODULENAME}.c`

Следующие переменные могут быть доступны для порта:

## APACHE\_VERSION

Основная-вспомогательная версия выбранного сервера Apache, например 2.4

## APACHEETCDIR

Расположение каталога конфигурации Apache. По умолчанию: `${LOCALBASE}/etc/apache2`

## APACHEINCLUDEDIR

Расположение include-файлов Apache. По умолчанию: `${LOCALBASE}/include/apache24`

## APACHEMODDIR

Расположение модулей Apache. По умолчанию: `${LOCALBASE}/libexec/apache24`

`APACHE_DEFAULT::`Версия Apache по умолчанию

## 17.7. autoreconf

Возможные аргументы: (нет), `build`

Выполняет `autoreconf`. Эта команда объединяет функциональность `aclocal`, `autoconf`, `autoheader`, `automake`, `autopoint` и `libtoolize`. Каждая из этих команд применяется к `${AUTORECONF_WKSRCS}/configure.ac` или его старому названию `${AUTORECONF_WKSRCS}/configure.in`. Если `configure.ac` определяет подкаталоги с их собственными `configure.ac` с использованием `AC_CONFIG_SUBDIRS`, `autoreconf` также рекурсивно обновит их. Аргумент `:build` только добавляет зависимости времени сборки на эти инструменты, но не запускает `autoreconf`. Порт может установить `AUTORECONF_WKSRCS`, если `WKSRCS` не содержит путь к `configure.ac`.

## 17.8. azurepy

Возможные аргументы: (отсутствуют)

Обеспечить поддержку портов `py-azure*`. Удаляет пространства имен `azure` и очищает общие файлы.

## 17.9. blaslapack

Возможные аргументы: (нет), `atlas`, `netlib` (по умолчанию), `gotoblas`, `openblas`

Добавляет зависимости от библиотек Blas / Lapack.

## 17.10. bdb

Возможные аргументы: (отсутствуют), `5` (по умолчанию), `18`

Добавить зависимость от библиотеки Berkeley DB. По умолчанию используется `databases/db5`. Также может зависеть от `databases/db18` при использовании аргумента `:18`. Можно объявить диапазон допустимых значений: `:5+` находит самую высокую установленную версию и возвращается к `5`, если ничего другого не установлено. `INVALID_BDB_VER` можно использовать для указания версий, которые не работают с этим портом. Фреймворк предоставляет порту следующие переменные:

### `BDB_LIB_NAME`

Имя библиотеки Berkeley DB. Например, при использовании `databases/db5` она содержит `db-5.3`.

### `BDB_LIB_CXX_NAME`

Название библиотеки Berkeley DB C++. Например, при использовании `databases/db5` она содержит `db_cxx-5.3`.

### `BDB_INCLUDE_DIR`

Расположение каталога с заголовочными файлами Berkeley DB. Например, при использовании пакета `databases/db5`, он будет содержать `${LOCALBASE}/include/db5`.

### `BDB_LIB_DIR`

Расположение каталога библиотеки Berkeley DB. Например, при использовании `databases/db5`, он содержит `${LOCALBASE}/lib`.

### `BDB_VER`

Обнаруженная версия Berkeley DB. Например, при использовании `USES=bdb:5+` и установленной Berkeley DB 18, будет содержать `18`.



`databases/db48` устарел и не поддерживается. Он не должен использоваться ни одним портом.

## 17.11. bison

Возможные аргументы: (нет), `build`, `run`, `both`

Использует пакет `devel/bison` По умолчанию, без аргументов или с аргументом `build`, подразумевается, что `bison` является зависимостью на этапе сборки, `run` — зависимостью на этапе выполнения, а `both` — зависимостью как на этапе сборки, так и на этапе выполнения.

## 17.12. budgie

Возможные аргументы: (отсутствуют)

Предоставить поддержку окружения рабочего стола Budgie. Используйте `USE_BUDGIE` для выбора необходимых компонентов порта. Дополнительную информацию см. в разделе [Использование Budgie](#).

## 17.13. cabal



Порты не следует создавать для библиотек Haskell, подробнее см. в [Библиотеки Haskell](#).

Возможные аргументы: (отсутствуют), `hpack`, `nodefault`

Устанавливает значения и цели по умолчанию, используемые для сборки программного обеспечения на Haskell с помощью Cabal. Добавляется зависимость для сборки на порт компилятора Haskell (`lang/ghc`). Если в переменной `BUILD_DEPENDS` уже указана другая версия GHC (например, `lang/ghc810`), она будет использована вместо версии по умолчанию. Если указан аргумент `hpack`, добавляется зависимость для сборки на `devel/hs-hpack`, и `hpack` вызывается на этапе конфигурации для генерации файла `.cabal`. Если указан аргумент `nodefault`, фреймворк не будет пытаться загрузить основной дистрибутивный файл из Hackage. Этот аргумент добавляется неявно, если присутствует `USE_GITHUB` или `USE_GITLAB`.

Фреймворк предоставляет следующие переменные:

### `CABAL_REVISION`

Пакеты Haskell, размещённые на Hackage, могут иметь ревизии. Установите этот параметр в целочисленное значение, чтобы использовать исправленное описание пакета.

### `USE_CABAL`

Если программное обеспечение использует зависимости на Haskell, перечислите их в этой переменной. Каждый элемент должен присутствовать на Hackage и быть указан в формате `имяпакета-0.1.2`. Зависимости также могут иметь ревизии, которые указываются после символа `_`. Поддерживается автоматическое формирование списка зависимостей, см. [Сборка приложений на Haskell с помощью cabal](#).

### `CABAL_FLAGS`

Список флагов, передаваемых `cabal-install` на этапах настройки и сборки. Флаги передаются в исходном виде. Эта переменная обычно используется для включения или отключения флагов, объявленных в файле `.cabal`. Передайте `foo`, чтобы включить флаг `foo`, и `-foo`, чтобы отключить его.

### `CABAL_EXECUTABLES`

Список исполняемых файлов, устанавливаемых портом. Значение по умолчанию: `${PORTNAME}`. Для получения списка возможных значений этой переменной обратитесь к файлу `.cabal` портируемого проекта. Каждое значение соответствует разделу `executable` в

файле `.cabal`. Элементы из этого списка автоматически добавляются в `pkg-plist`.

### `SKIP_CABAL_PLIST`

Если определено, не добавлять элементы из `#{CABAL_EXECUTABLES}` в `pkg-plist`.

### `opt_USE_CABAL`

Добавляет элементы в `#{USE_CABAL}` в зависимости от опции `opt`.

### `opt_CABAL_EXECUTABLES`

Добавляет элементы в `#{CABAL_EXECUTABLES}` в зависимости от опции `opt`.

### `opt_CABAL_FLAGS`

Если `opt` включён, добавить значение к `#{CABAL_FLAGS}`. В противном случае добавить `-value`, чтобы отключить флаг. Обратите внимание, что это поведение немного отличается от простого `CABAL_FLAGS`, так как оно не принимает значения, начинающиеся с `-`.

### `CABAL_WRAPPER_SCRIPTS`

Подмножество `#{CABAL_EXECUTABLES}`, содержащее программы на Haskell, которые будут обёрнуты в shell-скрипт, устанавливающий переменные окружения `*_datadir` перед запуском программы. Это также приводит к тому, что фактический бинарный файл Haskell устанавливается в директорию `libexec/cabal/`. Данная настройка необходима для программ на Haskell, которые устанавливают свои файлы данных в директорию `share/`.

### `FOO_DATADIR_VARS`

Список дополнительных пакетов Haskell, чьи файлы данных должны быть доступны исполняемому файлу с именем `FOO`. Исполняемый файл должен быть частью `#{CABAL_WRAPPER_SCRIPTS}`. Указанные пакеты Haskell не должны иметь суффикса версии.

### `CABAL_PROJECT`

Некоторые проекты на Haskell могут уже иметь файл `cabal.project`, который также создаётся фреймворком портов. Если это так, используйте эту переменную, чтобы указать, что делать с оригинальным файлом `cabal.project`. Установка этой переменной в значение `remove` приведёт к удалению оригинального файла. Установка этой переменной в значение `append` приведёт к следующему:

1. Исходный файл переместится в `cabal.project.#{PORTNAME}` на этапе `extract`.
2. Исходный файл `cabal.project.#{PORTNAME}` и сгенерированный `cabal.project` объединятся в один файл после этапа `patch`. Использование `append` позволяет выполнить патчинг исходного файла перед его объединением.

## 17.14. cargo

Возможные аргументы: (отсутствуют)

Использует Cargo для настройки, сборки и тестирования. Может применяться для портирования приложений на Rust, использующих систему сборки Cargo. Дополнительную информацию смотрите в [Сборка приложений на Rust с помощью cargo](#).

## 17.15. `charsetfix`

Возможные аргументы: (отсутствуют)

Предотвращает установку файла `charset.alias` портом. Этот файл должен устанавливаться только пакетом `converters/libiconv`. Переменная `CHARSETFIX_MAKEFILEIN` может быть установлена в путь относительно `WRKSRC`, если `charset.alias` не устанавливается через `${WRKSRC}/Makefile.in`.

## 17.16. `cl`

Возможные аргументы: (отсутствуют)

Предоставляет поддержку портов Common Lisp.

Фреймворк предоставляет следующие переменные, которые могут быть установлены портами:

### `ASDF_MODULES`

Список модулей `ASDF` для сборки, когда установлен `FASL_TARGET` (по умолчанию `PORTNAME`)

### `FASL_TARGET`

Собрать `fasl` вариант порта (один из `ccl`, `clisp` или `sbcl`)

### `USE_ASDF`

Зависит от пакета `devel/cl-asdf`

### `USE_ASDF_FASL`

Зависит от `devel/cl-asdf-<FASL_TARGET>`

### `USE_CCL`

Зависит от пакета `lang/ccl`; подразумевается при `FASL_TARGET=ccl`

### `USE_CLISP`

Зависит от пакета `lang/clisp`; подразумевается при `FASL_TARGET=clisp`

### `USE_SBCL`

Зависит от пакета `lang/sbcl`; подразумевается, если `FASL_TARGET=SBCL`

Фреймворк предоставляет следующие переменные, которые могут быть прочитаны портами:

### `ASDF_PATHNAME`

Путь к исходному коду CL

### `ASDF_REGISTRY`

Путь к реестру CL, содержащему файлы `asd`

## CCL

Путь к компилятору Clozure Common Lisp

## CLISP

Путь к компилятору GNU Common Lisp

## CL\_LIBDIR\_REL

Каталог библиотек CL относительно `LOCALBASE` или `PREFIX`

## FASL\_DIR\_REL

Относительный путь к скомпилированным fasl-файлам; зависит от `FASL_TARGET`

## FASL\_PATHNAME

Путь к CL fasl

## LISP\_EXTRA\_ARG

Дополнительные аргументы, используемые при сборке fasl

## SBCL

Путь к компилятору Steel Bank Common Lisp

## 17.17. cmake

Возможные аргументы: (отсутствуют), `insource`, `noninja`, `run`, `testing`

Используйте CMake для настройки порта и генерации системы сборки.

По умолчанию выполняется сборка в дереве вне исходного кода, оставляя исходные файлы в `WRKSRC` свободными от артефактов сборки. С аргументом `insource` вместо этого будет выполнена сборка в исходном коде. Этот аргумент должен быть исключением и использоваться только в случае, когда обычная сборка вне исходного кода не работает.

По умолчанию для сборки используется Ninja (`devel/ninja`). В некоторых случаях это может работать некорректно. С аргументом `noninja` сборка будет использовать обычный `make`. Этот аргумент следует применять только если сборка на основе Ninja не работает.

С аргументом `run` регистрируется зависимость во время выполнения в дополнение к зависимости при сборке.

С аргументом `testing`, добавляется цель тестирования, использующая CTest. При запуске тестов порт будет переконфигурирован для тестирования и пересобран.

Для получения дополнительной информации см. [Использование cmake](#).

## 17.18. compiler

Возможные аргументы: (нет), `env` (по умолчанию, подразумевается), `C++17-lang`, `C++14-lang`, `C++11-lang`, `gcc-C++11-lib`, `C++11-lib`, `C++0x`, `c11`, `nestedfct`, `features`

Определяет, какой компилятор использовать, исходя из заданных предпочтений. Используйте `C++17-lang`, если порту требуется компилятор с поддержкой C++17, `C++14-lang`, если порту требуется компилятор с поддержкой C++14, `C++11-lang`, если порту требуется компилятор с поддержкой C++11, `gcc-C++11-lib`, если порту требуется компилятор `g++` с библиотекой C++11, или `C++11-lib`, если порту требуется стандартная библиотека с поддержкой C++11. Если порту требуется компилятор, понимающий C++0X, C11 или вложенные функции, следует использовать соответствующие параметры.

Используйте `features` для запроса списка возможностей, поддерживаемых компилятором по умолчанию. После включения `bsd.port.pre.mk` порт может проверить результаты с помощью следующих переменных:

- `COMPILER_TYPE`: компилятор по умолчанию в системе, `gcc` или `clang`
- `ALT_COMPILER_TYPE`: альтернативный компилятор в системе, `gcc` или `clang`. Устанавливается только при наличии двух компиляторов в базовой системе.
- `COMPILER_VERSION`: первые две цифры версии компилятора по умолчанию.
- `ALT_COMPILER_VERSION`: первые две цифры версии альтернативного компилятора, если он присутствует.
- `CHOSEN_COMPILER_TYPE`: выбранный компилятор, `gcc` или `clang`
- `COMPILER_FEATURES`: возможности, поддерживаемые компилятором по умолчанию. В настоящее время указана библиотека C++.

## 17.19. `cpe`

Возможные аргументы: (отсутствуют)

Включает информацию о Common Platform Enumeration (CPE) в манифест пакета в виде строки формата CPE 2.3. Подробности см. в [спецификации CPE](#). Чтобы добавить информацию CPE в порт, выполните следующие шаги:

1. Поищите официальную запись CPE для программного продукта, используя либо [поисковую систему CPE](#) от NVD, либо [официальный словарь CPE](#) (предупреждение: очень большой XML-файл). *Никогда не создавайте данные CPE самостоятельно.*
2. Добавьте `cpe` в `USES` и сравните результат выполнения `make -V CPE_STR` с записью в словаре CPE. Продолжайте шаг за шагом, пока результат `make -V CPE_STR` не станет корректным.
3. Если название продукта (второе поле, по умолчанию `PORTNAME`) указано неверно, определите `CPE_PRODUCT`.
4. Если название производителя (первое поле, по умолчанию `CPE_PRODUCT`) указано неверно, определите `CPE_VENDOR`.
5. Если поле версии (третье поле, по умолчанию `PORTVERSION`) указано неверно, определите `CPE_VERSION`.
6. Если поле обновления (четвертое поле, по умолчанию пустое) указано неверно, определите `CPE_UPDATE`.
7. Если это всё ещё неверно, проверьте файл `Mk/Uses/cpe.mk` для получения

дополнительной информации или свяжитесь с Ports Security Team <[ports-secteam@FreeBSD.org](mailto:ports-secteam@FreeBSD.org)>.

8. Извлекайте как можно больше информации для имени CPE из существующих переменных, таких как `PORTNAME` и `PORTVERSION`. Используйте модификаторы переменных для извлечения соответствующих частей из этих переменных, вместо того чтобы жестко прописывать имя.
9. *Всегда* выполняйте `make -V CPE_STR` и проверяйте вывод перед коммитом любых изменений, затрагивающих `PORTNAME`, `PORTVERSION` или любые другие переменные, используемые для формирования `CPE_STR`.

## 17.20. `cran`

Возможные аргументы: (нет), `auto-plist`, `compiles`

Использует Comprehensive R Archive Network. Укажите `auto-plist` для автоматического создания `pkg-plist`. Укажите `compiles`, если порт содержит код, который необходимо компилировать.

## 17.21. `desktop-file-utils`

Возможные аргументы: (отсутствуют)

Использует `update-desktop-database` из пакета `devel/desktop-file-utils`. Дополнительный шаг `post-install` будет выполнен без вмешательства в уже существующие шаги `post-install` в `Makefile` порта. Строка с `@desktop-file-utils` будет добавлена в `plist`. Используйте этот макрос только если порт предоставляет файл `.desktop`, содержащий запись `MimeType`.

## 17.22. `desthack`

Возможные аргументы: (отсутствуют)

Изменяет поведение GNU `configure` для корректной поддержки `DESTDIR` в случае, если исходное программное обеспечение этого не делает.

## 17.23. `display`

Возможные аргументы: (отсутствуют), `ARGS`

Настраивает виртуальное окружение для отображения. Если переменная окружения `DISPLAY` не установлена, то `Xvfb` добавляется как зависимость при сборке, а `CONFIGURE_ENV` расширяется с указанием номера порта текущего запущенного экземпляра `Xvfb`. Параметр `ARGS` по умолчанию имеет значение `install` и управляет фазой, вокруг которой запускается и останавливается виртуальный дисплей.

## 17.24. dos2unix

Возможные аргументы: (отсутствуют)

Порт содержит файлы с символами конца строки в формате DOS, которые необходимо преобразовать. Несколько переменных могут быть установлены для контроля, какие файлы будут преобразованы. По умолчанию преобразуются все файлы, включая бинарные. См. [Простые автоматические замены](#) для примеров.

- `DOS2UNIX_REGEX`: сопоставлять имена файлов на основе регулярного выражения.
- `DOS2UNIX_FILES`: соответствуют точным именам файлов.
- `DOS2UNIX_GLOB`: сопоставлять имена файлов на основе шаблона файлов оболочки.
- `DOS2UNIX_WRKSRC`: каталог, с которого начинать преобразования. По умолчанию `${WRKSRC}`.

## 17.25. drupal

Возможные аргументы: `7`, `module`, `theme`

Автоматизирует установку порта, который является темой или модулем Drupal. Использовать с версией Drupal, которую ожидает порт. Например, `USES=drupal:7,module` означает, что этот порт создает модуль Drupal 7. Тему Drupal 7 можно указать с помощью `USES=drupal:7,theme`.

## 17.26. ebur128

Возможные аргументы: (нет), `build`, `lib`, `run`, `test`

Добавляет зависимость от пакета `audio/ebur128`. Позволяет прозрачно зависеть от вариантов `rust` или `legacy`, используя `DEFAULT_VERSIONS` в `make.conf`. Например, для использования устаревшей версии укажите `DEFAULT_VERSIONS+=ebur128=legacy`

Без аргументов поведение аналогично случаю с предоставлением аргумента `lib`. Остальные аргументы указывают соответствующую категорию зависимости.

## 17.27. eigen

Возможные аргументы: `2`, `3`, `build` (по умолчанию), `run`

Добавить зависимость от пакета `math/eigen`.

## 17.28. electronfix

Возможные аргументы: `31`, `32`, `33`

Предоставить поддержку для простого портирования Electron-приложений, распространяемых в бинарной форме. Добавляет зависимость на этапах сборки и выполнения от `devel/electron31`, `devel/electron32` или `devel/electron33` в зависимости от

используемого аргумента.

Фреймворк предоставляет следующие переменные, которые могут быть установлены портами:

#### **ELECTRONFIX\_SYMLINK\_FILES**

Список файлов для создания символьных ссылок из дистрибутива Electron.

#### **ELECTRONFIX\_MAIN\_EXECUTABLE**

Имя файла основного исполняемого файла, который будет заменен оригинальным бинарным файлом Electron.

## 17.29. **elfctl**

Возможные аргументы: (отсутствуют), **build** (по умолчанию), **stage**

Установите управляющие заметки функций ELF-бинарных файлов, задав **ELF\_FEATURES**.

Когда не указан аргумент или указан аргумент **build**, операции выполняются над бинарными файлами в **BUILD\_WRKSRC**, а файлы, перечисленные в **ELF\_FEATURES**, указываются относительно **BUILD\_WRKSRC**. Когда указан аргумент **stage**, операции выполняются над бинарными файлами в **STAGEDIR**, а файлы, перечисленные в **ELF\_FEATURES**, указываются относительно **STAGEDIR**.

*Пример 121. Uses=elfctl*

```
ELF_FEATURES= featurelist:path/to/file1 \  
              featurelist:path/to/file2
```

Формат **featurelist** описан в [elfctl\(1\)](#).

## 17.30. **elixir**

Возможные аргументы: (отсутствуют)

Предоставить поддержку для портов, использующих [lang/elixir](#). Добавляет зависимость во время сборки и выполнения на [lang/elixir](#).

Предоставляемые фреймворком переменные:

#### **ELIXIR\_APP\_NAME**

Название приложения Elixir, как оно установлено в каталоге lib Elixir

#### **ELIXIR\_LIB\_ROOT**

Путь к библиотекам Elixir по умолчанию

## ELIXIR\_APP\_ROOT

Корневая директория для этого приложения Elixir

## ELIXIR\_HIDDEN

Приложения, которые необходимо скрыть из пути выполнения кода; обычно `${PORTNAME}`

## ELIXIR\_LOCALE

Локаль UTF-8, которая будет использоваться Elixir во время сборки (подойдет любая локаль UTF-8)

## MIX\_CMD

Команда `mix`

## MIX\_COMPILE

Команда `mix`, используемая для компиляции приложения на Elixir

## MIX\_REWRITE

Автоматически заменять зависимости Mix на пути к коду

## MIX\_BUILD\_DEPS

Список `BUILD_DEPENDS` в формате категория/имя\_порта (часто упоминаемый как "deps" в Erlang и Elixir)

## MIX\_RUN\_DEPS

Список `RUN_DEPENDS` в формате категория/имя порта

## MIX\_DOC\_DIRS

Дополнительные каталоги документации для установки в `DOCSDIR`

## MIX\_DOC\_FILES

Дополнительные файлы документации для установки в `DOCSDIR` (обычно README.md)

## MIX\_ENV

Окружение для сборки Mix (в том же формате, что и `MAKE_ENV`)

## MIX\_ENV\_NAME

Имя среды сборки Mix, обычно "prod"

## MIX\_BUILD\_NAME

Имя выходного файла сборки в `_build/`, обычно `${MIX_ENV_NAME}`

## MIX\_TARGET

Имя цели Mix, обычно "compile"

## MIX\_EXTRA\_APPS

Список подприложений для сборки, если имеются

## MIX\_EXTRA\_DIRS

Список дополнительных каталогов для установки в `ELIXIR_APP_ROOT`

## MIX\_EXTRA\_FILES

Список дополнительных файлов для установки в `ELIXIR_APP_ROOT`

# 17.31. emacs

Возможные аргументы: (нет) (по умолчанию), `build`, `run`, `noflavors`

Предоставляет поддержку для портов, требующих Emacs. Аргумент `build` создает зависимость сборки от Emacs. Аргумент `run` создает зависимость выполнения от Emacs. Если оба аргумента `build` и `run` отсутствуют, создаются зависимости сборки и выполнения от Emacs. Аргумент `noflavors` запрещает флейворы и подразумевается, если нет зависимости выполнения от Emacs.

Стандартный вариант Emacs для портов с `USES=emacs` можно определить в `make.conf`. Например, для варианта `nox` используйте `DEFAULT_VERSIONS+= emacs=nox`. Допустимые флейворы: `full`, `canna`, `nox`, `wayland`, `devel_full`, `devel_nox`.

Переменные, которые могут быть установлены портами:

## EMACS\_FLAVORS\_EXCLUDE

НЕ собирать эти флейворы Emacs. Если `EMACS_FLAVORS_EXCLUDE` не определена и:

- существует зависимость во время выполнения от Emacs
- аргумент `noflavors` не указан

то предполагаются все допустимые флейворы Emacs.

## EMACS\_NO\_DEPENDS

НЕ добавлять зависимости сборки или выполнения от Emacs. Это предотвратит создание вариантов, и никакие файлы байт-кода не будут сгенерированы как часть пакета.

Переменные, которые могут быть прочитаны портами:

## EMACS\_CMD

Команда Emacs с полным путём (например, `/usr/local/bin/emacs-30.1`)

## EMACS\_FLAVOR

Используется для зависимостей (например, `BUILD_DEPENDS=dash.e1${EMACS_PKGNAME_SUFFIX}>0:devel/dash@${EMACS_FLAVOR}`)

## EMACS\_LIBDIR

Каталог библиотек Emacs без `${PREFIX}` (например, `share/emacs`)

## EMACS\_LIBDIR\_WITH\_VER

Каталог библиотеки без `${PREFIX}`, включая версию (например, `share/emacs/30.1`)

## EMACS\_MAJOR\_VER

Основная версия Emacs (например, 30)

## EMACS\_PKGNAME\_SUFFIX

PKGNAME\_SUFFIX для различия вариантов Emacs

## EMACS\_SITE\_LISPDIR

Каталог site-lisp Emacs без `#{PREFIX}` (например, share/emacs/site-lisp)

## EMACS\_VER

Версия Emacs (например, 30.1)

## EMACS\_VERSION\_SITE\_LISPDIR

Каталог site-lisp Emacs, включая номер версии (например, share/emacs/30.1/site-lisp)

## 17.32. erlang

Возможные аргументы: (нет), `enc`, `rebar`, `rebar3`

Добавляет зависимость на время сборки и выполнения от `lang/erlang`. В зависимости от аргумента, добавляет дополнительные зависимости для сборки. `enc` добавляет зависимость от `devel/erlang-native-compiler`, `rebar` добавляет зависимость от `devel/rebar`, а `rebar3` добавляет зависимость от `devel/rebar3`.

В дополнение, следующие переменные доступны для порта:

- `ERL_APP_NAME`: Имя приложения Erlang, как оно установлено в каталоге lib Erlang (без указания версии)
- `ERL_APP_ROOT`: Корневой каталог для этого приложения Erlang
- `REBAR_CMD`: Путь к команде "rebar"
- `REBAR3_CMD`: Путь к команде "rebar3"
- `REBAR_PROFILE`: Профиль Rebar
- `REBAR_TARGETS`: Список целей Rebar (обычно compile, возможно escriptize)
- `ERL_BUILD_NAME`: Имя сборки для rebar3
- `ERL_BUILD_DEPS`: Список BUILD\_DEPENDS в формате категория/имя\_порта
- `ERL_RUN_DEPS`: Список RUN\_DEPENDS в формате категория/имя\_порта
- `ERL_DOCS`: Список файлов и каталогов документации

## 17.33. fakeroot

Возможные аргументы: (отсутствуют)

Изменяет некоторые стандартные поведения систем сборки для разрешения установки от имени пользователя. Дополнительную информацию о `fakeroot` можно найти на <https://wiki.debian.org/FakeRoot>.

## 17.34. fam

Возможные аргументы: (нет), `fam`, `gamin`

Использует монитор изменений файлов (FAM — File Alteration Monitor) как зависимость от библиотеки, либо `devel/fam`, либо `devel/gamin`. Конечные пользователи могут задать `WITH_FAM_SYSTEM`, чтобы указать свои предпочтения.

## 17.35. firebird

Возможные аргументы: (отсутствуют), `25`

Добавить зависимость от клиентской библиотеке базы данных Firebird.

## 17.36. fonts

Возможные аргументы: (отсутствуют), `fc`, `fontsdir` (по умолчанию), `none`

Добавляет зависимость во время выполнения на инструменты, необходимые для регистрации шрифтов. В зависимости от аргумента добавляет строку `@fc ${FONTSDIR}`, строку `@fontsdir ${FONTSDIR}` или не добавляет строку, если аргумент `none`, в `plist`. `FONTSDIR` по умолчанию имеет значение `${PREFIX}/share/fonts/${FONTNAME}`, а `FONTNAME` — `${PORTNAME}`. Добавляет `FONTSDIR` в `PLIST_SUB` и `SUB_LIST`

## 17.37. fortran

Возможные аргументы: `gcc` (по умолчанию)

Использует компилятор GNU Fortran.

## 17.38. fpc

Возможные аргументы: (нет), `run`

Обеспечить поддержку портов на основе Free Pascal. Установит компилятор Free Pascal и модули.

Добавляет зависимость сборки от `lang/fpc`.

Если указан аргумент `run`, также добавляется зависимость запуска.

## 17.39. fuse

Возможные аргументы: `2` (по умолчанию), `3`

Порт будет зависеть от библиотеки FUSE и обрабатывать зависимость от модуля ядра в зависимости от версии FreeBSD.

## 17.40. `gem`

Возможные аргументы: (отсутствуют), `noautoplist`

Обработка сборки с RubyGems. Если используется `noautoplist`, список упаковки не генерируется автоматически.

Это подразумевает `USES=ruby`.

## 17.41. `gettext`

Возможные аргументы: (отсутствуют)

Устарело. Будет включать как `gettext-runtime`, так и `gettext-tools`.

## 17.42. `gettext-runtime`

Возможные аргументы: (отсутствуют), `lib` (по умолчанию), `build`, `run`

Использует пакет `devel/gettext-runtime`. По умолчанию, без аргументов или с аргументом `lib`, подразумевает зависимость от библиотеки `libintl.so`. Аргументы `build` и `run` подразумевают, соответственно, зависимость во время сборки и во время выполнения от `gettext`.

## 17.43. `gettext-tools`

Возможные аргументы: (отсутствуют), `build` (по умолчанию), `run`

Использует пакет `devel/gettext-tools`. По умолчанию, без аргумента или с аргументом `build`, регистрируется зависимость во время сборки от `msgfmt`. С аргументом `run` регистрируется зависимость во время выполнения.

## 17.44. `ghostscript`

Возможные аргументы: `X`, `build`, `run`, `nox11`

Можно указать конкретную версию `X`. Доступные версии: `7`, `8`, `9` и `agpl` (по умолчанию). `nox11` указывает, что требуется версия порта `-nox11`. `build` и `run` добавляют зависимости на Ghostscript во время сборки и выполнения соответственно. По умолчанию добавляются зависимости как на сборку, так и на выполнение.

## 17.45. `gl`

Возможные аргументы: (отсутствуют)

Предоставляет простой способ зависеть от компонентов GL. Компоненты должны быть перечислены в `USE_GL`. Доступные компоненты:

## egl

Добавить зависимость от библиотеки libEGL.so из пакета [graphics/libglvnd](#)

## gbm

Добавить зависимость от библиотеки libgbm.so из пакета [graphics/mesa-libs](#)

## gl

Добавить зависимость от библиотеки libGL.so из пакета [graphics/libglvnd](#)

## glesv2

Добавить зависимость от библиотеки libGLESv2.so из пакета [graphics/libglvnd](#)

## glew

Добавить зависимость от библиотеки libGLEW.so из пакета [graphics/glew](#)

## glu

Добавить зависимость от библиотеки libGLU.so из [graphics/libGLU](#)

## glut

Добавить зависимость от библиотеки libglut.so из [graphics/freelut](#)

## opengl

Добавить зависимость от библиотеки libOpenGL.so из [graphics/libglvnd](#)

## 17.46. gmake

Возможные аргументы: (отсутствуют)

Использует пакет [devel/gmake](#) как зависимость во время сборки и настраивает окружение для использования [gmake](#) в качестве стандартного [make](#) при сборке.

## 17.47. gnome

Возможные аргументы: (отсутствуют)

Предоставляет простой способ зависеть от компонентов GNOME. Компоненты должны быть перечислены в [USE\\_GNOME](#). Доступные компоненты:

- [atk](#)
- [atkmm](#)
- [cairo](#)
- [caiomm](#)
- [dconf](#)
- [esound](#)
- [evolutiondataserver3](#)

- gconf2
- gconfmm26
- gdkpixbuf
- gdkpixbuf2
- glib12
- glib20
- glibmm
- gnomecontrolcenter3
- gnomedesktop3
- gnomedesktop4
- gnomedocutils
- gnomemenu3
- gnomemimedata
- gnomeprefix
- gnomesharp20
- gnomevfs2
- gsound
- gtk-update-icon-cache
- gtk12
- gtk20
- gtk30
- gtkhtml3
- gtkhtml4
- gtkmm20
- gtkmm24
- gtkmm30
- gtksharp20
- gtksourceview
- gtksourceview2
- gtksourceview3
- gtksourceviewmm3
- gvfs
- intlhack
- intltool
- introspection

- `libartlgpl2`
- `libbonobo`
- `libbonoboui`
- `libgda5`
- `libgda5-ui`
- `libgdamm5`
- `libglade2`
- `libgnome`
- `libgnomecanvas`
- `libgnomekbd`
- `libgnomeprint`
- `libgnomeprintui`
- `libgnomeui`
- `libgsf`
- `libgtkhtml`
- `libgtksourceviewmm`
- `libidl`
- `librsvg2`
- `libsigc++12`
- `libsigc++20`
- `libwnck`
- `libwnck3`
- `libxml++26`
- `libxml2`
- `libxslt`
- `metacity`
- `nautilus3`
- `orbit2`
- `pango`
- `pangomm`
- `pangox-compat`
- `py3gobject3`
- `pygnome2`
- `pygobject`
- `pygobject3`

- `pygtk2`
- `pygtksourceview`
- `referencehack`
- `vte`
- `vte3`

Зависимость по умолчанию — на время сборки и выполнения, её можно изменить с помощью `:build` или `:run`. Например:

```
USES=      gnome
USE_GNOME= gnomemenu3:build intlhack
```

См. [Использование GNOME](#) для получения дополнительной информации.

## 17.48. go



Порты не следует создавать для библиотек Go, дополнительную информацию см. в [Библиотеки Go](#).

Возможные аргументы: (нет), `N.NN`, `N.NN-devel`, `modules`, `no_targets`, `run`

Устанавливает значения и цели по умолчанию, используемые для сборки ПО на Go. Добавляется зависимость сборки от порта компилятора Go, сопровождающие порта могут установить требуемую версию. По умолчанию сборка выполняется в режиме GOPATH. Если ПО на Go использует модули, режим с поддержкой модулей можно включить с помощью аргумента `modules`. `no_targets` настроит окружение сборки, как `GO_ENV`, `GO_BUILDFLAGS`, но пропустит создание целей извлечения (`extract`) и сборки (`build`). `run` также добавит зависимость выполнения от порта компилятора Go.

Процесс сборки контролируется несколькими переменными:

### GO\_MODULE

Имя модуля приложения, указанное директивой `module` в `go.mod`. В большинстве случаев это единственная необходимая переменная для портов, использующих модули Go.

### GO\_PKGNAME

Имя пакета Go при сборке в режиме GOPATH. Это каталог, который будет создан в `${GOPATH}/src`. Если не задано явно и присутствует `GH_SUBDIR` или `GL_SUBDIR`, то `GO_PKGNAME` будет выведено из них. Не требуется при сборке в режиме с поддержкой модулей.

### GO\_TARGET

Пакеты для сборки. Значение по умолчанию — `${GO_PKGNAME}`. `GO_TARGET` также может быть кортежем в формате `package:path`, где `path` может быть либо простым именем файла, либо полным путём, начинающимся с `${PREFIX}`.

## GO\_TESTTARGET

Пакеты для тестирования. Значение по умолчанию — `./...` (текущий пакет и все подпакеты).

## CGO\_CFLAGS

Дополнительные значения `CFLAGS`, передаваемые компилятору C с помощью `go`.

## CGO\_LDFLAGS

Дополнительные значения `LDFLAGS`, передаваемые компилятору C через `go`.

## GO\_BUILDFLAGS

Дополнительные аргументы сборки, передаваемые в `go build`.

## GO\_TESTFLAGS

Дополнительные аргументы сборки, передаваемые в `go test`.

См. [Сборка приложений на Go](#) для примеров использования.

## 17.49. gperf

Возможные аргументы: (отсутствуют)

Добавить зависимость во время сборки на `devel/gperf`, если `gperf` отсутствует в базовой системе.

## 17.50. grantlee

Возможные аргументы: `5`, `selfbuild`

Обработать зависимость от Grantlee. Указать `5` для зависимости от версии на основе Qt5, `devel/grantlee5`. `selfbuild` используется внутри `devel/grantlee5` для получения номеров их версий.

## 17.51. groff

Возможные аргументы: `build`, `run`, `both`

Регистрирует зависимость от `textproc/groff`, если пакет отсутствует в базовой системе.

## 17.52. gssapi

Возможные аргументы: (отсутствуют), `base` (по умолчанию), `heimdal`, `mit`, `flags`, `bootstrap`

Обрабатывает зависимости, необходимые для использования GSS-API. Доступны только библиотеки, предоставляющие механизм Kerberos. По умолчанию (или при значении `base`) используется библиотека GSS-API из базовой системы. Также можно установить значение `heimdal` для использования `security/heimdal` или `mit` для использования `security/krb5`.

Если локальная установка Kerberos не находится в `LOCALBASE`, установите `HEIMDAL_HOME` (для `heimdal`) или `KRB5_HOME` (для `krb5`) на каталог установки Kerberos.

Эти переменные экспортируются для использования портами:

- `GSSAPIBASEDIR`
- `GSSAPICPPFLAGS`
- `GSSAPIINCDIR`
- `GSSAPILDFLAGS`
- `GSSAPILIBDIR`
- `GSSAPILIBS`
- `GSSAPI_CONFIGURE_ARGS`

Опция `flags` может быть указана вместе с `base`, `heimdal` или `mit` для автоматического добавления `GSSAPICPPFLAGS`, `GSSAPILDFLAGS` и `GSSAPILIBS` в `CFLAGS`, `LDFLAGS` и `LDADD` соответственно. Например, используйте `base,flags`.

Опция `bootstrap` — это специальный префикс, предназначенный только для использования в `security/krb5` и `security/heimdal`. Например, используйте `bootstrap,mit`.

*Пример 122. Типичное использование*

```
OPTIONS_SINGLE= GSSAPI
OPTIONS_SINGLE_GSSAPI= GSSAPI_BASE GSSAPI_HEIMDAL GSSAPI_MIT GSSAPI_NONE

GSSAPI_BASE_USES= gssapi
GSSAPI_BASE_CONFIGURE_ON= --with-gssapi=${GSSAPIBASEDIR}
${GSSAPI_CONFIGURE_ARGS}
GSSAPI_HEIMDAL_USES= gssapi:heimdal
GSSAPI_HEIMDAL_CONFIGURE_ON= --with-gssapi=${GSSAPIBASEDIR}
${GSSAPI_CONFIGURE_ARGS}
GSSAPI_MIT_USES= gssapi:mit
GSSAPI_MIT_CONFIGURE_ON= --with-gssapi=${GSSAPIBASEDIR}
${GSSAPI_CONFIGURE_ARGS}
GSSAPI_NONE_CONFIGURE_ON= --without-gssapi
```

## 17.53. `gststreamer`

Возможные аргументы: (отсутствуют)

Предоставляет простой способ зависимости от компонентов GStreamer. Компоненты должны быть перечислены в `USE_GSTREAMER`. Доступные компоненты:

- `a52dec`
- `aalib`

- amrnb
- amrwbdec
- aom
- assrender
- bad
- bs2b
- cairo
- cdio
- cdparanoia
- chromaprint
- curl
- dash
- dtls
- dts
- dv
- dvd
- dvdread
- editing-services
- faac
- faad
- flac
- flite
- gdkpixbuf
- gl
- gme
- gnonlin
- good
- gsm
- gtk4
- gtk
- hal
- hls
- jack
- jpeg
- kate

- kms
- ladspa
- lame
- libav
- libcaca
- libde265
- libmms
- libvisual
- lv2
- mm
- modplug
- mpeg2dec
- mpeg2enc
- mpg123
- mplex
- musepack
- neon
- ogg
- opencv
- openexr
- openh264
- openjpeg
- openmpt
- opus
- pango
- png
- pulse
- qt
- resindvd
- rsvg
- rtmp
- shout2
- sidplay
- smoothstreaming
- sndfile

- `sndio`
- `soundtouch`
- `soup`
- `spandsp`
- `speex`
- `srtplib`
- `taglib`
- `theora`
- `ttml`
- `twolame`
- `ugly`
- `v4l2`
- `vorbis`
- `vpx`
- `vulkan`
- `wavpack`
- `webp`
- `webrtcdsp`
- `x264`
- `x265`
- `x`
- `ximagesrc`
- `zbar`

## 17.54. `guile`

Возможные аргументы: `(нет)`, `X.Y`, `flavors`, `build`, `run`, `alias`, `conflicts`

Добавляет зависимость от Guile. По умолчанию это зависимость от соответствующей библиотеки `libguile*.so`, если не переопределено опциями `build` и/или `run`. Опция `alias` настраивает `BINARY_ALIAS` соответствующим образом (см. [Использование BINARY\\_ALIAS](#)).

Версия по умолчанию устанавливается с помощью обычного механизма `DEFAULT_VERSIONS`; если версия по умолчанию не входит в список указанных версий, то используется последняя доступная версия из списка.

Приложения, использующие Guile, обычно собираются только для одной версии Guile. Однако модули расширений или библиотек должны использовать опцию `flavors` для сборки с несколькими флейворами.

Для получения дополнительной информации см. [Использование Guile](#).

## 17.55. horde

Возможные аргументы: (отсутствуют)

Добавить зависимости времени сборки и выполнения для [devel/pear-channel-horde](#). Другие зависимости Horde можно добавить с помощью `USE_HORDE_BUILD` и `USE_HORDE_RUN`. Дополнительную информацию см. в разделе [Модули Horde](#).

## 17.56. iconv

Возможные аргументы: (нет), `lib`, `build`, `patch`, `translit`, `wchar_t`

Использует функции `iconv`, либо из порта [converters/libiconv](#) как зависимость на этапе сборки и выполнения, либо из базовой системы. По умолчанию, без аргументов или с аргументом `lib`, подразумевает `iconv` с зависимостями на этапе сборки и выполнения. `build` подразумевает зависимость на этапе сборки, а `patch` — на этапе патчинга. Если порт использует расширения `WCHAR_T` или `//TRANSLIT` для `iconv`, добавьте соответствующие аргументы, чтобы использовалась правильная версия `iconv`. Для получения дополнительной информации см. [Использование iconv](#).

## 17.57. imake

Возможные аргументы: (нет), `env`, `notall`, `noman`

Добавить [devel/imake](#) как зависимость на этапе сборки и выполнить `xmkmf -a` на этапе `configure`. Если указан аргумент `env`, цель `configure` не устанавливается. Если флаг `-a` вызывает проблемы для порта, добавьте аргумент `notall`. Если `xmkmf` не генерирует цель `install.man`, добавьте аргумент `noman`.

## 17.58. java

Возможные аргументы: (нет), `ant`, `build`, `extract`, `run`

По умолчанию используется `USES=java:build,run`, если аргументы не предоставлены и `NO_BUILD` не определен. Если `NO_BUILD` определен, используется `USES=java:run`. Если указан аргумент `ant`, порт использует Apache Ant. Если указан аргумент `build`, порт JDK добавляется в зависимости сборки. Если указан аргумент `extract`, порт JDK добавляется в зависимости извлечения. Если указан аргумент `run`, порт JDK добавляется в зависимости выполнения.

Фреймворк предоставляет следующие переменные, которые могут быть установлены портом:

### JAVA\_VERSION

Список подходящих версий Java для порта, разделенных пробелами. Необязательный символ `+` позволяет указать диапазон версий. (допустимые значения `8[+]`, `11[+]`, `17[+]`, `18[+]`, `19[+]`, `20[+]`, `21[+]`, `22[+]`, `22[+]`)

## JAVA\_OS

Список поддерживаемых операционных систем для порта JDK, разделённых пробелами. (допустимые значения: `native`, `linux`)

## JAVA\_VENDOR

Список подходящих поставщиков портов JDK для порта, разделённых пробелами. (допустимые значения: `openjdk`, `oracle`)

Фреймворк предоставляет следующие переменные для чтения портом:

## JAVA\_PORT

Имя порта JDK. (например, 'java/openjdk8')

## JAVA\_PORT\_VERSION

Версия порта JDK. (например, '8')

## JAVA\_PORT\_OS

Используемая операционная система для порта JDK. (например, 'linux')

## JAVA\_PORT\_VENDOR

Поставщик порта JDK. (например, 'openjdk')

## JAVA\_PORT\_OS\_DESCRIPTION

Описание операционной системы, используемой портом JDK. (например, 'Linux')

## JAVA\_PORT\_VENDOR\_DESCRIPTION

Описание поставщика порта JDK. (например, 'OpenJDK BSD Porting Team')

## JAVA\_HOME

Путь к каталогу установки JDK. (например, /usr/local/openjdk8)

## JAVAC

Путь к используемому компилятору Java. (например, /usr/local/openjdk8/bin/javac или /usr/local/bin/javac)

## JAR

Путь к используемому инструменту JAR. (например, /usr/local/openjdk8/bin/jar или /usr/local/bin/fastjar)

## APPLETVIEWER

Путь к утилите appletviewer. (например, /usr/local/linux-jdk1.8.0/bin/appletviewer)

## JAVA

Путь к исполняемому файлу `java`. Используется для запуска программ на Java. (например, /usr/local/openjdk8/bin/java)

## JAVADOC

Путь к программе `javadoc`.

## JAVAH

Путь к программе `javah`.

## JAVAP

Путь к программе `javap`.

## JAVA\_KEYTOOL

Путь к утилите `keytool`.

## JAVA\_N2A

Путь к инструменту `native2ascii`.

## JAVA\_POLICYTOOL

Путь к программе `policytool`.

## JAVA\_SERIALVER

Путь к утилите `serialver`.

## RMIC

Путь к генератору RMI-заглушек/скелетов, `rmic`.

## RMIREGISTRY

Путь к программе реестра RMI, `rmiregistry`.

## RMID

Путь к программе демона RMI.

## JAVA\_CLASSES

Путь к архиву, содержащему файлы классов JDK. В большинстве JDK это `${JAVA_HOME}/jre/lib/rt.jar`.

## JAVASHAREDIR

Базовый каталог для всех общих ресурсов Java.

## JAVAJARDIR

Каталог, в котором порт должен устанавливать JAR-файлы.

## JAVAILBDIR

Каталог, в котором находятся JAR-файлы, установленные другими портами.

## 17.59. jpeg

Возможные аргументы: `lib` (по умолчанию, подразумевается), `build`, `run`

Помощь в обработке зависимостей от `jpeg`.

Если указан аргумент `lib` или аргументы не предоставлены, то в порт добавляется зависимость от библиотеки.

Если указан аргумент `build`, то в порт добавляется зависимость сборки.

Если указан аргумент `run`, то к порту добавляется зависимость времени выполнения.

Если указан аргумент `both`, то к порту добавляется зависимость для сборки и зависимость для выполнения.

Фреймворк предоставляет следующую переменную, которая может быть установлена портами:

### JPEG\_PORT

Указывает реализацию JPEG для использования. Возможные значения:

- `graphics/jpeg-turbo` (по умолчанию)
- `graphics/mozjpeg`

## 17.60. kde

Возможные аргументы: 5

Добавить зависимость от компонентов KDE. Подробнее см. в [Использование KDE](#).

## 17.61. kmod

Возможные аргументы: (отсутствуют), `debug`

Заполняет шаблон для портов модулей ядра, в настоящее время:

- Добавьте `kld` в `CATEGORIES`.
- Установите `SSP_UNSAFE`.
- Установите `IGNORE`, если исходные коды ядра не найдены в `SRC_BASE`.
- Определить `KMODDIR` по умолчанию как `/boot/modules`, добавить его в `PLIST_SUB` и `MAKE_ENV`, а также создать его при установке. Если `KMODDIR` установлен в `/boot/kernel`, он будет перезаписан в `/boot/modules`. Это предотвращает повреждение пакетов при обновлении ядра из-за переименования `/boot/kernel` в `/boot/kernel.old` в процессе.
- Обрабатывать перекрестные ссылки на модули ядра при установке и удалении, используя `@kld`.
- Если указан аргумент `debug`, порт может установить отладочную версию модуля в `KERN_DEBUGDIR/KMODDIR`. По умолчанию `KERN_DEBUGDIR` копируется из `DEBUGDIR` и устанавливается в `/usr/lib/debug`. Фреймворк позаботится о создании и удалении необходимых каталогов.

## 17.62. kodi

Возможные аргументы: (отсутствуют), `noautoplist`

Обеспечить поддержку дополнений для [multimedia/kodi](#). Если указан аргумент `noautoplist`, автоматическое создание `plist` не выполняется.

## 17.63. `lazarus`

Возможные аргументы: (отсутствуют), `gtk2` (по умолчанию), `qt5`, `qt6`, `flavors`

Обеспечить поддержку портов на основе [editors/lazarus](#).

Если аргументы не предоставлены или указан `gtk2`, приложение `lazarus-app` будет собрано с интерфейсом `gtk2`, и порт [editors/lazarus](#) будет собран с интерфейсом `gtk2`.

Если указан аргумент `qt5`, приложение `lazarus-app` собирается с интерфейсом `qt5`.

Если указан аргумент `qt6`, приложение `lazarus-app` собирается с интерфейсом `qt6`.

Если указан аргумент `flavors`, приложение `lazarus-app` собирается с поддержкой функций флейворов.

Если порт не требует автоматической компиляции файлов проекта `lazarus`, можно определить следующую переменную:

```
NO_LAZBUILD= yes
```

Доступны следующие переменные для портов:

### `LAZARUS_PROJECT_FILES`

Список `lri`-файлов. Он не должен быть пустым. По умолчанию: пусто

### `LAZARUS_DIR`

Путь к директории установки `lazarus`. По умолчанию: `${LOCALBASE}/share/lazarus-${LAZARUS_VER}`

### `LAZBUILD_ARGS`

Дополнительные аргументы `lazbuild`. В большинстве случаев это может быть `-d`. Подробнее см. [lazbuild\(1\)](#). По умолчанию: пусто

### `LAZARUS_NO_FLAVORS`

Не собирать эти флейворы `lazarus`. Если `LAZARUS_NO_FLAVORS` не определена, то предполагаются все допустимые флейворы `lazarus`.

### `WANT_LAZARUS_DEVEL`

Если установлено значение `yes`, то используйте [lazarus/devel](#) как зависимость сборки.

## 17.64. `ldap`

Возможные аргументы: (нет), `<версия>`, клиент, сервер

Регистрирует зависимость от пакета [net/openldap](#). Использует конкретную `<версию>` (без

точечной нотации), если она указана. В противном случае пытается найти установленную версию. При необходимости возвращается к версии по умолчанию, указанной в `bsd.default-versions.mk`. `client` указывает на зависимость во время выполнения от клиентской библиотеки. Это также значение по умолчанию. `server` указывает на зависимость во время выполнения от сервера.

Следующие переменные могут быть доступны для порта:

#### **IGNORE\_WITH\_OPENLDAP**

Эта переменная может быть определена, если порты не поддерживают одну или несколько версий OpenLDAP.

#### **WITH\_OPENLDAP\_VER**

Пользовательская переменная для установки версии OpenLDAP.

#### **OPENLDAP\_VER**

Обнаруженная версия OpenLDAP.

## **17.65. lha**

Возможные аргументы: (отсутствуют)

Установите `EXTRACT_SUFFIX` в `.lzh`

## **17.66. libarchive**

Возможные аргументы: (отсутствуют)

Регистрирует зависимость от `archivers/libarchive`. Любые порты, зависящие от `libarchive`, должны включать `USES=libarchive`.

## **17.67. libedit**

Возможные аргументы: (отсутствуют)

Регистрирует зависимость от `devel/libedit`. Все порты, зависящие от `libedit`, должны включать `USES=libedit`.

## **17.68. libtool**

Возможные аргументы: (нет), `keep1a`, `build`

Исправляет скрипты `libtool`. Это должно быть добавлено во все порты, использующие `libtool`. Аргумент `keep1a` может быть использован для сохранения файлов `.la`. Некоторые порты не поставляются с собственной копией `libtool` и требуют зависимость во время сборки от `devel/libtool`, используйте аргумент `:build` для добавления такой зависимости.

## 17.69. linux

Возможные аргументы: `c6`, `c7`

Порт фреймворка совместимости с Linux. Укажите `c6` для зависимостей от пакетов CentOS 6. Укажите `c7` для зависимостей от пакетов CentOS 7. Доступные пакеты:

- `allegro`
- `alsa-plugins-oss`
- `alsa-plugins-pulseaudio`
- `alsalib`
- `atk`
- `avahi-libs`
- `base`
- `cairo`
- `cups-libs`
- `curl`
- `cyrus-sasl2`
- `dbusglib`
- `dbuslibs`
- `devtools`
- `dri`
- `expat`
- `flac`
- `fontconfig`
- `gdkpixbuf2`
- `gnutls`
- `graphite2`
- `gtk2`
- `harfbuzz`
- `jasper`
- `jbigkit`
- `jpeg`
- `libasyncns`
- `libaudiofile`
- `libelf`
- `libgcrypt`

- [libgfortran](#)
- [libgpg-error](#)
- [libmng](#)
- [libogg](#)
- [libpciaccess](#)
- [libsndfile](#)
- [libsoup](#)
- [libssh2](#)
- [libtasn1](#)
- [libthai](#)
- [libtheora](#)
- [libv4l](#)
- [libvorbis](#)
- [libxml2](#)
- [mikmod](#)
- [naslibs](#)
- [ncurses-base](#)
- [nspr](#)
- [nss](#)
- [openal](#)
- [openal-soft](#)
- [openldap](#)
- [openmotif](#)
- [openssl](#)
- [pango](#)
- [pixman](#)
- [png](#)
- [pulseaudio-libs](#)
- [qt](#)
- [qt-x11](#)
- [qtwebkit](#)
- [scimlibs](#)
- [sdl12](#)
- [sdlimage](#)
- [sdlmixer](#)

- `sqlite3`
- `tc185`
- `tcp_wrappers-libs`
- `tiff`
- `tk85`
- `ucl`
- `xorglibs`

## 17.70. `llvm`

Возможные аргументы: (нет), `XY`, `min=XY`, `max=XY`, `build`, `run`, `lib`

Добавляет зависимость от LLVM. По умолчанию это зависимость для сборки, если не переопределено опциями `run` или `lib`. Версия по умолчанию задаётся в `LLVM_DEFAULT`. Также можно указать конкретную версию. Минимальную и максимальную версии можно указать с помощью параметров `min` и `max` соответственно. Фреймворк портов экспортирует следующие переменные в порт:

### `LLVM_VERSION`

Версия, выбранная из аргументов к `llvm.mk`

### `LLVM_PORT`

Выбранный порт `llvm`

### `LLVM_CONFIG`

`llvm-config` выбранного порта

### `LLVM_LIBLLVM`

`libLLVM.so` выбранного порта

### `LLVM_PREFIX`

Префикс инсталляции выбранного порта

## 17.71. `localbase`

Возможные аргументы: (отсутствуют), `ldflags`

Гарантирует использование библиотек из зависимостей в `LOCALBASE` вместо библиотек из базовой системы. Указывает `ldflags` для добавления `-L${LOCALBASE}/lib` в `LD_FLAGS` вместо `LIBS`. Порты, зависящие от библиотек, которые также присутствуют в базовой системе, должны использовать эту опцию. Она также используется внутри несколькими другими `USES`.

## 17.72. `lua`

Возможные аргументы: (нет), `XY`, `XY+`, `-XY`, `XY-ZA`, `module`, `flavors`, `build`, `run`, `env`

Добавляет зависимость от Lua. По умолчанию это зависимость от библиотеки, если не переопределено опциями `build` и/или `run`. Опция `env` предотвращает добавление любой зависимости, при этом все обычные переменные остаются определенными.

Версия по умолчанию устанавливается с помощью обычного механизма `DEFAULT_VERSIONS`, если только версия или диапазон версий не указаны в качестве аргумента, например, `51` или `51-54`.

Приложения, использующие Lua, обычно собираются только для одной версии Lua. Однако модули библиотек, предназначенные для загрузки кодом Lua, должны использовать опцию `module` для сборки с несколькими вариантами.

Для получения дополнительной информации см. [Использование Lua](#).

## 17.73. `luajit`

Возможные аргументы: (нет), `X`

Добавляет зависимость от среды выполнения `luajit`. Можно указать конкретную версию `X`.  
Доступные версии: `luajit`, `luajit-devel`, `luajit-openresty`

После включения `bsd.port.options.mk` или `bsd.port.pre.mk` порт может проверять эти переменные:

### `LUAJIT_VER`

Выбранная версия `luajit`

### `LUAJIT_INCDIR`

Путь к заголовочным файлам `luajit`

### `LUAJIT_LUAVER`

Какой версии спецификации `luajit` выбрана (2.0 для `luajit`, иначе 2.1)

Для получения дополнительной информации см. [Использование Lua](#).

## 17.74. `lxqt`

Возможные аргументы: (отсутствуют)

Обработка зависимостей для рабочей среды LXQt. Используйте `USE_LXQT` для выбора необходимых компонентов для порта. Дополнительную информацию см. в разделе [Использование LXQt](#).

## 17.75. `magick`

Возможные аргументы: (нет), `X`, `build`, `nox11`, `run`, `test`

Добавить зависимость библиотеки от `ImageMagick`. Можно указать конкретную версию `X`.  
Доступные версии: `6` и `7` (по умолчанию). `nox11` означает, что требуется версия порта `-nox11`.

`build`, `run` и `test` добавляют зависимости на сборку, выполнение и тестирование для ImageMagick.

## 17.76. `makeinfo`

Возможные аргументы: (отсутствуют)

Добавить зависимость во время сборки на `makeinfo`, если его нет в базовой системе.

## 17.77. `makeself`

Возможные аргументы: (отсутствуют)

Указывает, что файлы дистрибутива являются архивами `makeself` и устанавливает соответствующие зависимости.

## 17.78. `mate`

Возможные аргументы: (отсутствуют)

Предоставляет простой способ зависимостей от компонентов MATE. Компоненты должны быть перечислены в `USE_MATE`. Доступные компоненты:

- `autogen`
- `caja`
- `common`
- `controlcenter`
- `desktop`
- `dialogs`
- `docutils`
- `icontheme`
- `intlhack`
- `intltool`
- `libmatekbd`
- `libmateweather`
- `marco`
- `menus`
- `notificationdaemon`
- `panel`
- `pluma`
- `polkit`

- `session`
- `settingsdaemon`

Зависимость по умолчанию — на время сборки и выполнения, её можно изменить с помощью `:build` или `:run`. Например:

```
USES=      mate
USE_MATE=  menus:build intlhack
```

## 17.79. `meson`

Возможные аргументы: (отсутствуют)

Предоставить поддержку для проектов на основе Meson. Дополнительную информацию смотрите в [Использование meson](#).

## 17.80. `metaport`

Возможные аргументы: (отсутствуют)

Устанавливает следующие переменные для упрощения создания метапорта: `MASTER_SITES`, `DISTFILES`, `EXTRACT_ONLY`, `NO_BUILD`, `NO_INSTALL`, `NO_MTREE`, `NO_ARCH`.

## 17.81. `minizip`

Возможные аргументы: (отсутствуют), `ng`

Добавляет зависимость библиотеки от [archivers/minizip](#) или [archivers/minizip-ng](#) соответственно.

## 17.82. `mlt`

Возможные аргументы: `7`, `nodepend`

Обеспечить поддержку портов, зависящих от [multimedia/mlt7](#).

Если указан аргумент `nodepend`, зависимости от библиотек не создаются. Этот аргумент имеет смысл только для портов `multimedia/mlt7*`.

## 17.83. `mysql`

Возможные аргументы: (отсутствуют), `версия`, `client` (по умолчанию), `server`, `embedded`

Предоставить поддержку MySQL. Если версия не указана, попытаться определить установленную версию. В случае неудачи использовать версию по умолчанию, MySQL-5.6. Возможные версии: `55`, `55m`, `55p`, `56`, `56p`, `56w`, `57`, `57p`, `80`, `100m`, `101m` и `102m`. Суффиксы `m` и `p`

обозначают флейворс MariaDB и Percona для MySQL. Параметры `server` и `embedded` добавляют зависимости во время сборки и выполнения на сервер MySQL. При использовании `server` или `embedded` добавьте `client`, чтобы также включить зависимость от `libmysqlclient.so`. Порт может установить `IGNORE_WITH_MYSQL`, если некоторые версии не поддерживаются.

Фреймворк устанавливает `MYSQL_VER` в обнаруженную версию MySQL.

## 17.84. `mono`

Возможные аргументы: (отсутствуют), `nuget`

Добавляет зависимость от фреймворка Mono (в настоящее время только C#), устанавливая соответствующие зависимости.

Укажите `nuget`, если порт использует пакеты nuget. `NUGET_DEPENDS` должен содержать имена и версии пакетов nuget в формате `имя=версия`. Можно добавить необязательное расположение пакета (origin), используя `имя=версия:_ расположение _`.

Вспомогательная цель `buildnuget` выведет содержимое `NUGET_DEPENDS` на основе предоставленного файла `packages.config`.

## 17.85. `motif`

Возможные аргументы: (отсутствуют)

Использует `x11-toolkits/open-motif` как зависимость библиотеки. Конечные пользователи могут установить `WANT_LESSTIF` в `make.conf`, чтобы использовать `x11-toolkits/lesstif` как зависимость вместо `x11-toolkits/open-motif`. Аналогично, установка `WANT_OPEN_MOTIF_DEVEL` в `make.conf` добавит зависимость от `x11-toolkits/open-motif-devel`

## 17.86. `mpi`

Возможные аргументы: `mpich` (по умолчанию), `openmpi`

Обеспечить поддержку портов, зависящих от `MPI`.

Если указан аргумент `mpich`, в порт добавляется зависимость от `net/mpich`.

Если указан аргумент `openmpi`, в порт добавляется зависимость от `net/openmpi`.

Фреймворк портов предоставляет следующие переменные, которые могут быть прочитаны портом:

### `MPI_LIBS`

Библиотеки, необходимые для связывания программ с использованием `MPI`.

### `MPI_CFLAGS`

Флаги компилятора, необходимые для сборки программ с использованием `MPI`.

## **MPICC**

Расположение исполняемого файла `mpicc`. По умолчанию: `${MPI_HOME}/bin/mpicc`.

## **MPICXX**

Расположение исполняемого файла `mpicxx`. По умолчанию: `${MPI_HOME}/bin/mpicxx`.

## **MPIF90**

Расположение исполняемого файла `mpif90`. По умолчанию: `${MPI_HOME}/bin/mpif90`.

## **MPIFC**

То же, что и выше.

## **MPI\_HOME**

Каталог установки **MPI**. По умолчанию используется `${LOCALBASE}` для **MPICH**.

## **MPIEXEC**

Расположение исполняемого файла `mpiexec`. По умолчанию: `${MPI_HOME}/bin/mpiexec`.

## **MPIRUN**

Расположение исполняемого файла `mpirun`. По умолчанию: `${MPI_HOME}/bin/mpirun`.

## **17.87. ncurses**

Возможные аргументы: (нет), `base`, `port`

Использует `ncurses` и устанавливает некоторые полезные переменные.

## **17.88. nextcloud**

Возможные аргументы: (отсутствуют)

Добавляет поддержку приложений Nextcloud, добавляя зависимость во время выполнения на [www/nextcloud](http://www.nextcloud).

## **17.89. ninja**

Возможные аргументы: (нет), `build`, `make` (по умолчанию), `run`

Если указаны аргументы `build` или `run`, это соответственно добавляет зависимость во время сборки или выполнения от пакета `devel/ninja`. Если указан `make` или аргументы не предоставлены, используется `ninja` для сборки порта вместо `make`. `make` подразумевает `build`. Если переменная `NINJA_DEFAULT` установлена в `samurai`, тогда зависимости устанавливаются для пакета `devel/samurai` вместо этого.

## **17.90. nodejs**

Возможные аргументы: (нет), `build`, `run`, `current`, `lts`, `10`, `14`, `16`, `17`.

Использует `nodejs`. Добавляет зависимость от пакета `www/node*`. Если указана поддерживаемая версия, то также необходимо указать `run` и/или `build`.

## 17.91. `objc`

Возможные аргументы: (отсутствуют)

Добавить зависимости Objective C (компилятор, библиотека времени выполнения), если базовая система их не поддерживает.

## 17.92. `ocaml`

Возможные аргументы: (нет), `build`, `camlp4`, `dune`, `findlib`, `findplist`, `ldconfig`, `run`, `tk`, `tkbuild`, `tkrun`, `wash`

Обеспечить поддержку OCaml.

Если аргументы не указаны, по умолчанию используются `build`, `run`.

Если указан аргумент `build`, то `lang/ocamlc` добавляется в `BUILD_DEPENDS`, `EXTRACT` и `PATCH_DEPENDS`.

Если указан аргумент `camlp4`, то для сборки используется `devel/ocamlp4`.

Если указан аргумент `dune`, то `devel/ocaml-dune` используется как система сборки.

Если указан аргумент `findlib`, то для установки пакетов будет использоваться `ocamlfind`. Директории пакетов будут автоматически удалены.

Если указан аргумент `findplist`, то содержимое целевых каталогов `findlib` будет добавлено автоматически.

Если указан аргумент `ldconfig`, то файл `ld.conf` OCaml будет обработан автоматически. При использовании `dune` Dune может устанавливать `stubs` в директориях пакетов `site-lib` или в отдельной директории ниже `DUNE_LIBDIR site-lib`. Установите, если ваш порт устанавливает общие библиотеки в `ocaml`

Если указан аргумент `run`, добавить `ocamlc` в `RUN_DEPENDS`.

Если указан аргумент `tk`, то в порт добавляется зависимость на сборку и выполнение от пакета `x11-toolkits/ocaml-labltk`. Подразумевает `tkbuild` и `tkrun`.

Если указан аргумент `tkbuild`, то пакет `x11-toolkits/ocaml-labltk` добавляется в `BUILD_DEPENDS`, `EXTRACT` и `PATCH_DEPENDS`.

Если указан аргумент `tkrun`, то `x11-toolkits/ocaml-labltk` добавляется в `RUN_DEPENDS`.

Если указан аргумент `wash`, общие каталоги OCaml будут очищены при удалении. Полезно при установке в нестандартный `PREFIX`.

Портом могут быть установлены следующие переменные:

## OCAML\_PKGDIRS

Каталоги в site-lib для обработки, если указан аргумент `findlib`. По умолчанию: `${PORTNAME}`

## OCAML\_LDLIBS

Каталоги в `PREFIX`, которые будут автоматически добавлены/удалены из `ld.conf`. По умолчанию: `${OCAML_SITELIBDIR}/${PORTNAME}`

## OCAML\_PACKAGES

Список пакетов для сборки и установки. По умолчанию `${PORTNAME}`

## 17.93. octave

Возможные аргументы: (нет), `env`

Использует `math/octave`. `env` загружает только одну переменную окружения `OCTAVE_VERSION`.

## 17.94. openal

Возможные аргументы: `al`, `soft` (по умолчанию), `si`, `alut`

Использует OpenAL. Бэкенд может быть указан, с программной реализацией по умолчанию. Пользователь может указать предпочтительный бэкенд с помощью `WANT_OPENAL`. Допустимые значения для этой настройки: `soft` (по умолчанию) и `si`.

## 17.95. pathfix

Возможные аргументы: (отсутствуют)

Ищите `Makefile.in` и `configure` в `PATHFIX_WKSRSC` (по умолчанию `WKSRSC`) и исправляйте стандартные пути, чтобы они соответствовали иерархии FreeBSD. Например, исправляется каталог установки для файлов `.pc pkgconfig` на `${PREFIX}/libdata/pkgconfig`. Если порт использует `USES=autoreconf`, `Makefile.am` будет автоматически добавлен в `PATHFIX_MAKEFILEIN`.

Если порт `USES=cmake`, он будет искать файл `CMakeLists.txt` в `PATHFIX_WKSRSC`. При необходимости это имя файла по умолчанию можно изменить с помощью `PATHFIX_CMAKELISTSTXT`.

## 17.96. pear

Возможные аргументы: `env`

Добавляет зависимость от пакета `devel/pear`. Настраивает поведение по умолчанию для программного обеспечения, использующего PHP Extension and Application Repository. Использование аргументов `env` только устанавливает переменные окружения `PEAR`. Дополнительную информацию см. в [Модули PEAR](#).

## 17.97. perl5

Возможные аргументы: (отсутствуют)

Зависит от Perl. Настройка выполняется с помощью `USE_PERL5`.

`USE_PERL5` может содержать фазы, в которых используется Perl: `extract`, `patch`, `build`, `run` или `test`.

`USE_PERL5` также может содержать `configure`, `modbuild` или `modbuilddtiny`, если требуется `Makefile.PL`, `Build.PL` или вариант `Build.PL` для `Module::Build::Tiny`.

`USE_PERL5` по умолчанию имеет значение `build run`. При использовании `configure`, `modbuild` или `modbuilddtiny`, `build` и `run` подразумеваются автоматически.

См. [Использование Perl](#) для получения дополнительной информации.

## 17.98. pgsql

Возможные аргументы: (нет), `X.Y`, `X.Y+`, `X.Y-`, `X.Y-Z.A`

Предоставить поддержку PostgreSQL. Ответственный за порт может указать требуемую версию. Можно указать минимальную и максимальную версии или диапазон; например, `9.0-`, `8.4+`, `8.4-9.2`

По умолчанию добавляемая зависимость будет клиентской, но если порту требуются дополнительные компоненты, это можно указать с помощью `WANT_PGSQL=компонент[:цель]`; например, `WANT_PGSQL=server:configure pltcl plperl`. Доступные компоненты:

- `client`
- `contrib`
- `docs`
- `pgtcl`
- `plperl`
- `plpython`
- `pltcl`
- `server`

## 17.99. php

Возможные аргументы: (нет), `phpize`, `ext`, `zend`, `build`, `cli`, `cgi`, `mod`, `web`, `embed`, `pecl`, `flavors`, `noflavors`

Обеспечить поддержку PHP. Добавить зависимость во время выполнения на версию PHP по умолчанию, [lang/php81](#).

## phpize

Используется для создания расширения PHP. Поддерживает флейворы.

## ext

Используется для сборки, установки и регистрации расширения PHP. Поддерживает флейворы.

## zend

Используется для сборки, установки и регистрации Zend-расширения. Поддерживает флейворы.

## build

Установить PHP также как зависимость во время сборки.

## cli

Требуется версия PHP для командной строки.

## cgi

Требуется CGI-версия PHP.

## mod

Требуется модуль Apache для PHP.

## web

Требуется модуль Apache или CGI-версия PHP.

## embed

Требуется встроенная версия библиотеки PHP.

## pecl

Установить значения по умолчанию для загрузки расширений PHP из репозитория PECL. Включает флейворы.

## flavors

Включить автоматическую генерацию [флейворов PHP](#). Флейворы будут созданы для всех версий PHP, за исключением указанных в `IGNORE_WITH_PHP`.

## noflavors

Отключить автоматическое создание флейворов PHP. *Должно* использоваться только с расширениями, предоставляемыми самим PHP.

Переменные используются для указания необходимых модулей PHP, а также версий PHP, которые поддерживаются.

## USE\_PHP

Список необходимых расширений PHP во время выполнения. Добавьте `:build` к названию расширения, чтобы указать зависимость во время сборки. Пример: `pecl xml:build gettext`

## IGNORE\_WITH\_PHP

Порт не работает с PHP указанной версии. Возможные значения можно посмотреть в содержимом `_ALL_PHP_VERSIONS` в `Mk/Uses/php.mk`.

При сборке расширения PHP или Zend с помощью `:ext` или `:zend`, можно задать следующие переменные:

## PHP\_MODNAME

Имя расширения PHP или Zend. Значение по умолчанию: `${PORTNAME}`.

## PHP\_HEADER\_DIRS

Список подкаталогов, из которых следует устанавливать заголовочные файлы. Фреймворк всегда будет устанавливать заголовочные файлы, находящиеся в том же каталоге, что и расширение.

## PHP\_MOD\_PRIO

Приоритет загрузки расширения. Это число от `00` до `99`.

Для расширений, которые не зависят от других расширений, приоритет автоматически устанавливается в `20`, а для расширений, зависящих от другого расширения, приоритет автоматически устанавливается в `30`. Некоторые расширения могут требовать загрузки перед всеми остальными, например, [www/php56-opcache](#). Некоторые могут требовать загрузки после расширения с приоритетом `30`. В таком случае добавьте `PHP_MOD_PRIO=XX` в Makefile порта. Например:

```
USES=      php:ext
USE_PHP=   wddx
PHP_MOD_PRIO= 40
```

Эти переменные доступны для использования в `PKGNAMEPREFIX` или `PKGNAME_SUFFIX`:

## PHP\_PKGNAMEPREFIX

Содержит `php_XY_`, где `XY` — версия PHP текущей редакции. Используется с расширениями и модулями PHP.

## PHP\_PKGNAME\_SUFFIX

Содержит `-php_XY_`, где `XY` — версия PHP текущего варианта. Используется с PHP-приложениями.

## PECL\_PKGNAMEPREFIX

Содержит `php_XY_-pecl-`, где `XY` — версия PHP текущей редакции. Используется с модулями PECL.



С вариантами сборки все расширения PHP, расширения PECL, модули PECL должны иметь разные имена пакетов, поэтому они должны использовать одну из трёх переменных в `PKGNAMEPREFIX` или `PKGNAME_SUFFIX`.

## 17.100. pkgconfig

Возможные аргументы: (отсутствуют), `build` (по умолчанию), `run`, `both`

Использует `devel/pkgconf`. Без аргументов или с аргументом `build` подразумевает зависимость от `pkg-config` во время сборки. `run` подразумевает зависимость во время выполнения, а `both` — зависимости как во время выполнения, так и во время сборки.

## 17.101. pure

Возможные аргументы: (нет), `ffi`

Использует `lang/pure`. В основном применяется для сборки портов, зависящих от `pure`. С аргументом `ffi` подразумевает `devel/pure-ffi` как зависимость во время выполнения.

## 17.102. pyqt

Возможные аргументы: (нет), `4`, `5`

Использует PyQt. Если порт является частью самого PyQt, установите `PYQT_DIST`. Используйте `USE_PYQT` для выбора необходимых порту компонентов. Доступные компоненты:

- `core`
- `dbus`
- `dbussupport`
- `demo`
- `designer`
- `designerplugin`
- `doc`
- `gui`
- `multimedia`
- `network`
- `opengl`
- `qscintilla2`
- `sip`
- `sql`
- `svg`
- `test`
- `webkit`
- `xml`
- `xmlpatterns`

Эти компоненты доступны только с PyQt4:

- `assistant`
- `declarative`
- `help`
- `phonon`
- `script`
- `scripttools`

Эти компоненты доступны только с PyQt5:

- `multimediawidgets`
- `printsupport`
- `qml`
- `serialport`
- `webkitwidgets`
- `widgets`

Зависимость по умолчанию для каждого компонента — это время сборки и выполнения. Чтобы выбрать только сборку или выполнение, добавьте `_build` или `_run` к имени компонента. Например:

```
USES=      pyqt
USE_PYQT=  core doc_build designer_run
```

## 17.103. `pytest`

Возможные аргументы: (нет), 4

Вводит новую зависимость от `devel/pytest`. Он определяет цель `do-test`, которая будет правильно запускать тесты. Используйте аргумент, чтобы зависеть от определённой версии `devel/pytest`. Для портов, использующих `devel/pytest`, рекомендуется использовать это вместо конкретной цели `do-test`. Фреймворк предоставляет порту следующие переменные:

### `PYTEST_ARGS`

Дополнительные аргументы для `pytest` (по умолчанию пусто).

### `PYTEST_IGNORED_TESTS`

списки шаблонов `pytest -k` для игнорирования тестов (по умолчанию пустые). Для тестов, которые не должны проходить, например, требующих доступа к базе данных.

### `PYTEST_BROKEN_TESTS`

списки шаблонов `pytest -k` тестов для игнорирования (по умолчанию пустые). Для сломанных тестов, которые требуют исправления.

В дополнение следующие переменные могут быть заданы пользователем:

#### **PYTEST\_ENABLE\_IGNORED\_TESTS**

Включить тесты, которые в противном случае игнорируются **PYTEST\_IGNORED\_TESTS**.

#### **PYTEST\_ENABLE\_BROKEN\_TESTS**

Включить тесты, которые в противном случае игнорируются **PYTEST\_BROKEN\_TESTS**.

#### **PYTEST\_ENABLE\_ALL\_TESTS**

Включить тесты, которые в противном случае игнорируются **PYTEST\_IGNORED\_TESTS** и **PYTEST\_BROKEN\_TESTS**.

## **17.104. python**

Возможные аргументы: (нет), **X.Y**, **X.Y+**, **-X.Y**, **X.Y-Z.A**, **patch**, **build**, **run**, **test**

Использует Python. Можно указать поддерживаемую версию или диапазон версий. Если Python требуется только во время сборки, выполнения или тестирования, его можно установить как зависимость для сборки, выполнения или тестирования с помощью **build**, **run** или **test**. Если Python также требуется на этапе исправлений, используйте **patch**. Дополнительную информацию см. в разделе [Использование Python](#).

**USES=python:env** можно использовать, когда необходимы переменные, экспортируемые фреймворком, но зависимость от Python не требуется. Это может быть полезно при использовании с **USES=shebangfix**, если цель состоит только в исправлении shebang без добавления зависимости от Python.

## **17.105. qmail**

Возможные аргументы: (нет), **build**, **run**, **both**, **vars**

Использует [mail/qmail](#). С аргументом **build** подразумевается зависимость от **qmail** во время сборки. Аргумент **run** подразумевает зависимость во время выполнения. Использование без аргументов или с аргументом **both** подразумевает зависимости как во время выполнения, так и во время сборки. Аргумент **vars** только устанавливает переменные QMAIL для использования в порте.

## **17.106. qmake**

Возможные аргументы: (отсутствуют), **norecursive**, **outsource**, **no\_env**, **no\_configure**

Использует QMake для настройки. Для получения дополнительной информации см. [Использование qmake](#).

## **17.107. qt**

Возможные аргументы: **5**, **6**, **no\_env**

Добавить зависимость от компонентов Qt. `no_env` передаётся напрямую в `USES= qmake`. Подробнее см. в [Использование Qt](#).

## 17.108. qt-dist

Possible arguments: (none) or 5 and (none) or 6 and (none) or one of 3d, 5compat, base, charts, connectivity, datavis3d, declarative, doc languageserver, gamepad, graphicaleffects, imageformats, locat ion, lottie, multimedia, networkauth, positioning, quick3d, quickcontrols2, quickcontrols, quicktimeline, remoteobjects, script, scxml ` , `sensors, serialbus, serialport, shadertools, speech, svg, tools, translations, virtualkeyboard, wayland, webchannel, webengine, webglplugin, websockets, webview, x11extras, xmlpatterns.

Предоставляет поддержку сборки компонентов Qt 5 и Qt 6. Обеспечивает настройку соответствующей конфигурации окружения для сборки порта.

*Пример 123. Сборка компонентов Qt 5*

Порт представляет собой компонент `networkauth` из Qt 5, который входит в файл дистрибутива `networkauth`.

```
PORTNAME= networkauth
DISTVERSION= ${QT5_VERSION}

USES= qt-dist:5
```

*Пример 124. Сборка компонентов Qt 6*

Порт представляет собой компонент `websockets` из Qt 6, который входит в файл дистрибутива `websockets`.

```
PORTNAME= websockets
PORTVERSION= ${QT6_VERSION}

USES= qt-dist:6
```

Если `PORTNAME` не совпадает с именем компонента, его можно передать как аргумент в `qt-dist`.

*Пример 125. Сборка компонентов Qt 5 с разными именами*

Порт представляет собой компонент `gui` из Qt 5, который входит в файл дистрибутива `base`.

```
PORTNAME= gui
DISTVERSION= ${QT5_VERSION}
```

```
USES= qt-dist:5,base
```

## 17.109. `readline`

Возможные аргументы: (нет), `port`

Использует `readline` в качестве зависимости библиотеки и устанавливает `CPPFLAGS` и `LDFLAGS` по необходимости. Если используется аргумент `port` или если `readline` отсутствует в базовой системе, добавляет зависимость от `devel/readline`

## 17.110. `ruby`

Возможные аргументы: (нет), `build`, `extconf`, `run`, `setup`

Предоставить поддержку для портов, связанных с Ruby. `(none)` без аргументов добавляет зависимость во время выполнения на `lang/ruby`. `build` добавляет зависимость на `lang/ruby` во время сборки. `extconf` указывает, что порт использует `extconf.rb` для настройки. `run` добавляет зависимость на `lang/ruby` во время выполнения. Это также значение по умолчанию. `setup` указывает, что порт использует `setup.rb` для настройки и сборки.

Пользователь может определить следующие переменные:

### `RUBY_VER`

Альтернативная короткая версия `ruby` в виде `x.y`.

### `RUBY_DEFAULT_VER`

Установите (например) `2.7`, чтобы использовать `ruby27` в качестве версии по умолчанию.

### `RUBY_ARCH`

Установите имя архитектуры (например, `i386-freebsd7`).

Следующие переменные экспортируются для использования портом:

### `RUBY`

Установлена в полный путь к `ruby`. Если задано, значения следующих переменных автоматически получаются из исполняемого файла `ruby`: `RUBY_ARCH`, `RUBY_ARCHLIBDIR`, `RUBY_LIBDIR`, `RUBY_SITELIBDIR`, `RUBY_VER` и `RUBY_VERSION`

### `RUBY_VER`

Установлена в альтернативную короткую версию `ruby` в формате `x.y`.

### `RUBY_EXTCONF`

Установлена в альтернативное имя для `extconf.rb` (по умолчанию: `extconf.rb`).

### `RUBY_EXTCONF_SUBDIRS`

Установлена в список подкаталогов, если включено несколько модулей.

## RUBY\_SETUP

Установлена в альтернативное имя для setup.rb (по умолчанию: setup.rb).

## 17.111. samba

Возможные аргументы: `build`, `env`, `lib`, `run`

Обработать зависимость от Samba. `env` не добавит никаких зависимостей, а только установит переменные. `build` и `run` добавляют зависимости во время сборки и выполнения на `smbd`. `lib` добавит зависимость на `libsmbclient.so`. Экспортируемые переменные:

### SAMBA\_PORT

Расположение порта Samba по умолчанию.

### SAMBA\_INCLUDEDIR

Расположение заголовочных файлов Samba.

### SAMBA\_LIBS

Каталог, в котором доступны общие библиотеки Samba.

### SAMBA\_LDB\_PORT

Расположение порта `ldb`, используемого выбранной версией Samba (например, [databases/ldb28](#)). Он должен использоваться, если порту требуется зависимость от той же версии `ldb`, что и у выбранной версии Samba.

### SAMBA\_TALLOC\_PORT

Расположение порта `talloc`, используемого выбранной версией Samba. Следует использовать, если порту требуется зависеть от той же версии `talloc`, что и выбранная версия Samba.

### SAMBA\_TDB\_PORT

Расположение порта `TDB`, используемого выбранной версией Samba. Его следует использовать, если порту требуется зависеть от той же версии `TDB`, что и выбранная версия Samba.

### SAMBA\_TEVENT\_PORT

Расположение порта `tevent`, используемого выбранной версией Samba. Это следует использовать, если порту необходимо зависеть от той же версии `tevent`, что и выбранная версия Samba.

## 17.112. sconS

Возможные аргументы: (отсутствуют)

Предоставить поддержку для использования [devel/sconS](#). Дополнительную информацию смотрите в [Использование sconS](#).

## 17.113. sdl

Возможные аргументы: `sdl`

Обеспечить поддержку использования пакетов `SDL`. Переменная `USE_SDL` является обязательной и указывает, какие компоненты добавить в зависимости.

Поддерживаемые в настоящее время модули `SDL1.2`:

- `sdl`
- `console`
- `gfx`
- `image`
- `mixer`
- `mm`
- `net`
- `pango`
- `sound`
- `ttf`

Текущие поддерживаемые модули `SDL2`:

- `sdl2`
- `gfx2`
- `image2`
- `mixer2`
- `net2`
- `sound2`
- `ttf2`

Текущие поддерживаемые модули `SDL3`:

- `sdl3`
- `image3`
- `ttf3`

## 17.114. shared-mime-info

Возможные аргументы: (отсутствуют)

Использует `update-mime-database` из пакета `misc/shared-mime-info`. Это автоматически добавит шаг `post-install` таким образом, что сам порт всё ещё может указать собственный

шаг post-install при необходимости. Также добавляет запись `@shared-mime-info` в `plist`.

## 17.115. shebangfix

Возможные аргументы: (отсутствуют)

Множество программ используют некорректные расположения для интерпретаторов скриптов, особенно `/usr/bin/perl` и `/bin/bash`. Макрос `shebangfix` исправляет строки `shebang` в скриптах, перечисленных в `SHEBANG_REGEX`, `SHEBANG_GLOB` или `SHEBANG_FILES`.

### SHEBANG\_REGEX

Содержит *одно* расширенное регулярное выражение и используется с аргументом `-iregex` в `find(1)`. См. [USEShebangfix](#) с `SHEBANG_REGEX`.

### SHEBANG\_GLOB

Содержит список шаблонов, используемых с аргументом `-name` в `find(1)`. См. [USEShebangfix](#) с `SHEBANG_GLOB`.

### SHEBANG\_FILES

Содержит список файлов или шаблонов `sh(1)`. Макрос `shebangfix` выполняется из `${WRKSR}`, поэтому `SHEBANG_FILES` может содержать пути, относительные к `${WRKSR}`. Также он может работать с абсолютными путями, если требуется исправление файлов вне `${WRKSR}`. См. [USEShebangfix](#) с `SHEBANG_FILES`.

В настоящее время Bash, Java, Ksh, Lua, Perl, PHP, Python, Ruby, Tcl и Tk поддерживаются по умолчанию.

Существует три переменных конфигурации:

### SHEBANG\_LANG

Список поддерживаемых интерпретаторов.

### \_interp\_\_CMD

Путь к интерпретатору команд в FreeBSD. Значение по умолчанию — `${LOCALBASE}/bin/interp`.

### \_interp\_\_OLD\_CMD

Список неправильных вызовов интерпретаторов. Обычно это устаревшие пути или пути, используемые в других операционных системах, которые неверны в FreeBSD. Они будут заменены на правильные пути в `_interp__CMD`.



Эти пути *всегда* будут частью `interp__OLD_CMD`: `"/usr/bin/env _interp" /bin/interp /usr/bin/interp /usr/local/bin/interp`.



`_interp__OLD_CMD` содержит несколько значений. Любая запись с пробелами должна быть заключена в кавычки. См. [Указание всех путей при добавлении интерпретатора в USEShebangfix](#).



Исправление shebang-строк выполняется на этапе `patch`. Если скрипты создаются с некорректными shebang-строками на этапе `build`, процесс сборки (например, скрипт `configure` или `Makefiles`) должен быть исправлен или ему должен быть указан правильный путь (например, с помощью `CONFIGURE_ENV`, `CONFIGURE_ARGS`, `MAKE_ENV` или `MAKE_ARGS`) для генерации корректных shebang-строк.

Правильные пути для поддерживаемых интерпретаторов доступны в `_interp_CMD`.



При использовании с `USES=python`, если цель состоит только в исправлении shebang, но зависимость от самого Python не требуется, используйте `USES=python:env`.

*Пример 126. Добавление другого интерпретатора в `USES=shebangfix`*

Чтобы добавить другой интерпретатор, установите `SHEBANG_LANG`. Например:

```
SHEBANG_LANG= lua
```

*Пример 127. Указание всех путей при добавлении интерпретатора в `USES=shebangfix`*

Если они еще не были определены и не было значений по умолчанию для `_interpOLD_CMD` и `_interpCMD`, запись Ksh можно определить как:

```
SHEBANG_LANG= ksh
ksh_OLD_CMD=  "/usr/bin/env ksh" /bin/ksh /usr/bin/ksh
ksh_CMD=     "${LOCALBASE}/bin/ksh
```

*Пример 128. Добавление нестандартного расположения интерпретатора*

Некоторое программное обеспечение использует нестандартные пути для интерпретатора. Например, приложение может ожидать, что Python будет расположен в `/opt/bin/python2.7`. Нестандартный путь, который нужно заменить, можно указать в Makefile порта:

```
python_OLD_CMD= /opt/bin/python2.7
```

*Пример 129. `USES=shebangfix` с `SHEBANG_REGEX`*

Для исправления всех файлов в `${WRKSRCDIR}/scripts`, оканчивающихся на `.pl`, `.sh` или `.cgi`, выполните:

```
USES= shebangfix
SHEBANG_REGEX= ./scripts/.*\.(sh|pl|cgi)
```



`SHEBANG_REGEX` используется при выполнении `find -E`, который применяет современные регулярные выражения, также известные как расширенные регулярные выражения. Подробнее см. в [re\\_format\(7\)](#).

*Пример 130.* `USES=shebangfix` с `SHEBANG_GLOB`

Для исправления всех файлов в `${WRKSRC}` с окончанием `.pl` или `.sh` выполните:

```
USES= shebangfix
SHEBANG_GLOB= *.sh *.pl
```

*Пример 131.* `USES=shebangfix` с `SHEBANG_FILES`

Для исправления файлов `script/foobar.pl` и `script/*.sh` в `${WRKSRC}` выполните:

```
USES= shebangfix
SHEBANG_FILES= scripts/foobar.pl scripts/*.sh
```

## 17.116. `sqlite`

Возможные аргументы: (нет), `2`, `3`

Добавить зависимость от SQLite. Используемая по умолчанию версия — `3`, но также возможна версия `2` с использованием модификатора `:2`.

## 17.117. `sbrk`

Возможные аргументы: (отсутствуют)

Помечает порт как `BROKEN` для `aarch64` и `riscv64`.

## 17.118. `ssl`

Возможные аргументы: (нет), `build`, `run`

Обеспечить поддержку OpenSSL. Зависимость только для сборки или выполнения может быть указана с использованием `build` или `run`. Эти переменные доступны для использования портом, а также добавлены в `MAKE_ENV`:

## OPENSSLBASE

Путь к базовой установке OpenSSL.

## OPENSSLDIR

Путь к файлам конфигурации OpenSSL.

## OPENSSLIB

Путь к библиотекам OpenSSL.

## OPENSSLINC

Путь к заголовочным файлам OpenSSL.

## OPENSSLRPATH

Если определено, путь, который требуется компоновщику для поиска библиотек OpenSSL.



Если порт не собирается с вариантом OpenSSL, установите переменную `BROKEN_SSL`, а также, возможно, `BROKEN_SSL_REASON__flavor_`:

```
BROKEN_SSL= libressl
BROKEN_SSL_REASON_libressl= needs features only available in OpenSSL
```

## 17.119. tar

Возможные аргументы: (нет), `Z`, `bz2`, `bzip2`, `lzma`, `tbz`, `tbz2`, `tgz`, `txz`, `xz`, `zst`, `zstd`

Установите `EXTRACT_SUFX` в `.tar`, `.tar.Z`, `.tar.bz2`, `.tar.tbz2`, `.tar.lzma`, `.tar.tbz`, `.tar.tbz2`, `.tar.tgz`, `.tar.txz`, `.tar.xz`, `.tar.zst` или `.tar.zstd` соответственно.

## 17.120. tcl

Возможные аргументы: `version`, `wrapper`, `build`, `run`, `tea`

Добавьте зависимость от Tcl. Конкретная версия может быть запрошена с помощью `version`. Версия может быть пустой, одной или несколькими точными номерами версий (в настоящее время `84`, `85` или `86`), либо минимальным номером версии (в настоящее время `84+`, `85+` или `86+`). Чтобы запросить только неспецифичную для версии обёртку, используйте `wrapper`. Зависимость только на время сборки или выполнения может быть указана с помощью `build` или `run`. Для сборки порта с использованием Tcl Extension Architecture используйте `tea`. После включения `bsd.port.pre.mk` порт может проверить результаты с помощью этих переменных:

- `TCL_VER`: выбранная версия Tcl в формате `major.minor`
- `TCLSH`: полный путь к интерпретатору Tcl
- `TCL_LIBDIR`: путь к библиотекам Tcl

- `TCL_INCLUDEDIR`: путь к заголовочным файлам Tcl на языке C
- `TCL_PKG_LIB_PREFIX`: Префикс библиотеки, согласно TIR595
- `TCL_PKG_STUB_POSTFIX`: Постфикс библиотеки заглушки
- `TK_VER`: выбранная версия Tk в формате major.minor
- `WISH`: полный путь к интерпретатору Tk
- `TK_LIBDIR`: путь к библиотекам Tk
- `TK_INCLUDEDIR`: путь к заголовочным файлам Tk на языке C

## 17.121. `terminfo`

Возможные аргументы: (отсутствуют)

Добавляет `@terminfo` в файл `plist`. Используется, когда порт устанавливает файлы `*.terminfo` в каталог `${PREFIX}/share/misc`.

## 17.122. `tex`

Возможные аргументы: (отсутствуют)

Обеспечить поддержку `tex`. Загружает все стандартные переменные для портов, связанных с `TEX`, и не добавляет зависимостей от других портов.

Переменные используются для указания того, какие модули `TEX` требуются.

### `USE_TEX`

Список необходимых расширений `TEX` во время выполнения. Добавьте `:build` к названию расширения, чтобы добавить зависимость на время сборки, `:run` — для зависимости во время выполнения, `:test` — для зависимости во время тестирования, `:extract` — для зависимости во время извлечения. Пример: `base texmf:build source:run`

Текущие возможные аргументы следующие:

- `base`
- `texmf`
- `source`
- `docs`
- `web2c`
- `kpathsea`
- `ptexenc`
- `basic`
- `tlmgr`
- `texlua`

- `texluajit`
- `synctex`
- `xpdfopen`
- `dvipsk`
- `dvipdfmx`
- `xdvik`
- `gbklatex`
- `formats`
- `tex`
- `latex`
- `pdftex`
- `jadetex`
- `luatex`
- `ptex`
- `xetex`
- `xmltex`
- `texhash`
- `updmap`
- `fmtutil`

## 17.123. `tk`

Так же, как аргументы для `tcl`

Небольшая обертка при использовании `Tcl` и `Tk`. Возвращаются те же переменные, что и при использовании `Tcl`.

## 17.124. `trigger`

Возможные аргументы: (отсутствуют)

Предоставить поддержку для портов, требующих выполнения триггеров с помощью `pkg(8)`. Триггеры выполняются в конце транзакции, если условия выполнены.

Следующая переменная может быть установлена портами:

### TRIGGERS

Список триггеров для пакета. По умолчанию используется `${PORTNAME}`.

Триггеры указываются в формате UCL и обычно размещаются в каталоге `files/` порта.

## 17.125. **uidfix**

Возможные аргументы: (отсутствуют)

Изменяет некоторые стандартные настройки (в основном переменные) системы сборки, чтобы позволить установку этого порта обычным пользователем. Попробуйте это в порте перед использованием `USES=fakeroot` или исправлением.

## 17.126. **uniquefiles**

Возможные аргументы: (нет), `dirs`

Сделать файлы или каталоги 'уникальными', добавляя префикс или суффикс. Если используется аргумент `dirs`, порту требуется префикс (и только префикс) на основе `UNIQUE_PREFIX` для стандартных каталогов `DOCSDIR`, `EXAMPLESDIR`, `DATADIR`, `WWWDIR`, `ETCDIR`. Эти переменные доступны для портов:

- `UNIQUE_PREFIX`: Префикс, используемый для каталогов и файлов. По умолчанию: `${PKGNAMEPREFIX}`.
- `UNIQUE_PREFIX_FILES`: Список файлов, которые необходимо предварить префиксом. По умолчанию: пусто.
- `UNIQUE_SUFFIX`: Суффикс, используемый для файлов. По умолчанию: `${PKGNAME_SUFFIX}`.
- `UNIQUE_SUFFIX_FILES`: Список файлов, к которым необходимо добавить суффикс. По умолчанию: пусто.

## 17.127. **vala**

Возможные аргументы: `build`, `lib`, `no_depend`

Добавляет зависимости сборки или библиотеки на `lang/vala`. Аргумент `no_depend` зарезервирован для самого `lang/vala`.

## 17.128. **varnish**

Возможные аргументы: 4 (по умолчанию), 6, 7

Обрабатывает зависимости для Varnish Cache. Добавляет зависимость от пакета `www/varnish*`.

## 17.129. **waf**

Возможные аргументы: (отсутствуют)

Обеспечить поддержку портов, использующих систему сборки `waf`.

Это подразумевает `USES=python:build`.

Следующие переменные экспортируются для использования портом:

## WAF\_CMD

Расположение скрипта `waf`. Установите этот параметр, если скрипт `waf` не находится в `WRKSRC/waf`.

## CONFIGURE\_TARGET

Цель для `configure`. По умолчанию – `configure`.

## ALL\_TARGET

Цель для `all`. По умолчанию `build`.

## INSTALL\_TARGET

Цель для `install`. По умолчанию `install`.

## 17.130. webplugin

Возможные аргументы: (нет), `ARGS`

Автоматически создавать и удалять символические ссылки для каждого приложения, поддерживающего фреймворк `webplugin`. `ARGS` может быть одним из:

- `gecko`: поддержка плагинов на основе Gecko
- `native`: поддержка плагинов для Gecko, Opera и WebKit-GTK
- `linux`: поддержка Linux плагинов
- `all` (по умолчанию, неявно): поддержка всех типов плагинов
- (отдельные записи): поддерживаются только перечисленные браузеры

Эти переменные можно настроить:

- `WEBPLUGIN_FILES`: Значение по умолчанию отсутствует, должно быть установлено вручную. Файлы плагинов для установки.
- `WEBPLUGIN_DIR`: Каталог для установки файлов плагина, по умолчанию `PREFIX/lib/browser_plugins/WEBPLUGIN_NAME`. Установите это значение, если порт устанавливает файлы плагина вне стандартного каталога, чтобы избежать битых символических ссылок.
- `WEBPLUGIN_NAME`: Конечный каталог для установки файлов плагина, по умолчанию `PKGBASE`.

## 17.131. xfce

Возможные аргументы: (нет), `gtk2`

Предоставить поддержку для портов, связанных с Xfce. Подробности см. в [Использование Xfce](#).

Аргумент `gtk2` указывает, что порт требует поддержки GTK2. Он добавляет дополнительные возможности, предоставляемые некоторыми основными компонентами, например, `x11/libxfce4menu` и `x11-wm/xfce4-panel`.

## 17.132. **хогг**

Возможные аргументы: (отсутствуют)

Предоставляет простой способ зависеть от компонентов X.org. Компоненты должны быть перечислены в `USE_XORG`. Доступные компоненты:

Таблица 52. Доступные компоненты X.Org

Имя	Описание
<code>dmx</code>	Библиотека расширений DMX
<code>fontenc</code>	Библиотека fontenc
<code>fontutil</code>	Создать индекс файлов шрифтов X в каталоге
<code>ice</code>	Библиотека Inter Client Exchange для X11
<code>libfs</code>	Библиотека FS
<code>pciaccess</code>	Универсальная библиотека доступа к PCI
<code>pixman</code>	Библиотека для низкоуровневого управления пикселями
<code>sm</code>	Библиотека управления сеансами для X11
<code>x11</code>	Библиотека X11
<code>xau</code>	Библиотека протокола аутентификации для X11
<code>xaw</code>	Библиотека X Athena Widgets
<code>xaw6</code>	Библиотека X Athena Widgets
<code>xaw7</code>	Библиотека X Athena Widgets
<code>xbitmaps</code>	Данные растровых изображений X.Org
<code>xcb</code>	Библиотека с интерфейсом языка C для X протокола (XCB)
<code>xcomposite</code>	Библиотека расширения X Composite
<code>xcursor</code>	X библиотека загрузки курсоров на стороне клиента
<code>xdamage</code>	Библиотека расширения X Damage
<code>xdmcp</code>	Библиотека протокола управления дисплейным менеджером X (XDMCP)
<code>xext</code>	Библиотека расширений X11
<code>xfixes</code>	Библиотека расширений X Fixes
<code>xfont</code>	Библиотека шрифтов X
<code>xfont2</code>	Библиотека шрифтов X
<code>xft</code>	Клиентский API шрифтов для приложений X

Имя	Описание
<code>xi</code>	Библиотека расширения X Input
<code>xinerama</code>	Библиотека X11 Xinerama
<code>xkbfile</code>	Библиотека файлов ХКВ
<code>xmu</code>	Библиотека X Miscellaneous Utilities
<code>xmuu</code>	Библиотека X Miscellaneous Utilities
<code>xorg-macros</code>	X.Org макросы разработки асlocal
<code>xorg-server</code>	Сервер X.Org X и относящиеся к нему программы
<code>xorgproto</code>	Заголовочные файлы протокола xorg
<code>xpm</code>	Библиотека X Pixmap
<code>xpresent</code>	Библиотека расширений X Present
<code>xrandr</code>	Библиотека расширений X Resize and Rotate
<code>xrender</code>	Библиотека расширения X Render
<code>xres</code>	Библиотека мониторинга ресурсов X Resource usage
<code>xscrsaver</code>	Библиотека XScrnSaver
<code>xshmfence</code>	Примитив синхронизации "SyncFence" в разделяемой памяти
<code>xt</code>	Библиотека X Toolkit
<code>xtrans</code>	Абстрактный сетевой код для X
<code>xtst</code>	Расширение X Test
<code>xv</code>	Библиотека расширения X Video
<code>xvnc</code>	Библиотека X Video Extension Motion Compensation
<code>xxf86dga</code>	Расширение X DGA
<code>xxf86vm</code>	Расширение X Vidmode

## 17.133. `xorg-cat`

Возможные аргументы: `app`, `data`, `doc`, `driver`, `font`, `lib`, `proto`, `util`, `xserver` и (без аргументов) или один из `autotools` (по умолчанию), `meson`

Обеспечивает поддержку сборки компонентов Xorg. Управляет настройкой общих зависимостей и необходимой конфигурационной среды. Предназначено только для компонентов Xorg.

Категория должна соответствовать категориям вышестоящего репозитория.

Второй аргумент — используемая система сборки. По умолчанию используется `autotools`, но также поддерживается `meson`.

## 17.134. `zip`

Возможные аргументы: (нет), `infozip`

Указывает, что файлы дистрибутива используют алгоритм сжатия ZIP. Для файлов, использующих алгоритм InfoZip, необходимо передать аргумент `infozip`, чтобы установить соответствующие зависимости.

# Глава 18. Значения `__FreeBSD_version`

Здесь удобный список значений `__FreeBSD_version` как определено в [sys/param.h](#):

## 18.1. Версии FreeBSD 15

Таблица 53. Значения `__FreeBSD_version` в FreeBSD 15

Значение	Версия	Дата	Релиз
1500000	<a href="#">29a16ce065db</a>	24 августа 2023	15.0-CURRENT.
1500001	<a href="#">a6662c37b6ff</a>	17 сентября 2023	15.0-CURRENT после реализации <code>fpu_kern_enter</code> и <code>fpu_kern_leave</code> для <code>powerpc</code> .
1500002	<a href="#">17f5e2b904af</a>	18 октября 2023	15.0-CURRENT после изменения внутреннего KAPI между модулями <code>nfsccommon</code> и <code>nfsc1</code> .
1500003	<a href="#">ef85fd507e6e</a>	1 ноября 2023	15.0-CURRENT после удаления кода обратной совместимости для преобразования <code>inode64</code> .
1500004	<a href="#">7fabea328fed</a>	23 ноября 2023	15.0-CURRENT после добавления новой функции VFS под названием <code>vfs_exjail_clone()</code> , которая будет использоваться модулем ZFS.

Значение	Версия	Дата	Релиз
1500005	<a href="#">21fce617d1de</a>	27 ноября 2023	15.0-CURRENT после серии механических изменений в дереве: идентификаторы SCCS удалены, закомментированные строки с авторскими правами удалены с помощью <code>if 0</code> , механические исправления стиля после этих изменений и удаление некоторых макросов из <code>[.filename]#sys/cdefs.h</code> .
1500006	<a href="#">c711af772782</a>	8 декабря 2023	15.0-CURRENT после обновления llvm, clang, compiler-rt, libc++, libunwind, lld, lldb и openmp до llvmorg-17.0.6-0-g6009708b4367, также известного как релиз 17.0.6.
1500007	<a href="#">8ccd0b876e67</a>	11 декабря 2023	15.0-CURRENT после предоставления доступа к ехесвре для совместимости с Linux в libc.
1500008	<a href="#">9bf957fc9b37</a>	24 декабря 2023	15.0-CURRENT после изменений в LinuxKPI.
1500009	<a href="#">b068bb09a1a8</a>	11 января 2024	15.0-CURRENT после добавления <code>vnode_pager_clean_async(9)</code> и <code>vnode_pager_clean_sync(9)</code> .
1500010	<a href="#">a2da1bdb61bc</a>	12 января 2024	15.0-CURRENT после изменения внутреннего KPI между модулями nfsccommon и nfsc.

Значение	Версия	Дата	Релиз
1500011	<a href="#">a2da1bdb61bc</a>	17 января 2024	15.0-CURRENT после добавления поддержки <code>zfs.dataset</code> в <code>jail(8)</code> .
1500012	<a href="#">120ceeabab5d4</a>	24 января 2024	15.0-CURRENT после добавления <code>kern_openatfp(9)</code> и <code>ksmp(2)</code> .
1500013	<a href="#">d04abb05375d</a>	7 февраля 2024	15.0-CURRENT после добавления <code>libsys</code> .
1500014	<a href="#">ed27ae8df4b1</a>	11 февраля 2024	15.0-CURRENT после переключения <code>clang</code> и других исполняемых файлов LLVM на сборку как PIE.
1500015	<a href="#">a7b9f4d96e8b</a>	13 марта 2024	15.0-CURRENT после удаления избыточных аргументов <code>type</code> и <code>rid</code> из нескольких функций в API ресурсов <code>new-bus</code> .
1500016	<a href="#">60bc9617e79e</a>	18 марта 2024	15.0-CURRENT после введения <code>livedump_start_vnode(9)</code> .
1500017	<a href="#">bcd401b5a39c</a>	20 марта 2024	15.0-CURRENT после исправления утверждения или сбоя <code>clang</code> при сборке последних библиотек <code>boost</code> .
1500018	<a href="#">0192eda105b3</a>	6 апреля 2024	15.0-CURRENT после обновления <code>llvm</code> , <code>clang</code> , <code>compiler-rt</code> , <code>libc++</code> , <code>libunwind</code> , <code>lld</code> , <code>lldb</code> и <code>openmp</code> до версии <code>lvmorg-18.1.3-0-gc13b7485b879</code> , также известной как релиз 18.1.3.

Значение	Версия	Дата	Релиз
1500019	<a href="#">e03e8b077433</a>	31 мая 2024	15.0-CURRENT после переопределения <code>CLOCK_BOOTTIME</code> как псевдонима для <code>CLOCK_MONOTONIC</code> вместо <code>CLOCK_UPTIME</code> .
1500020	<a href="#">7818c2d37c2c</a>	12 июля 2024	15.0-CURRENT после удаления поддержки сборки armv6.
1500021	<a href="#">24388fccd551</a>	21 июля 2024	15.0-CURRENT после изменений в LinuxKPI.
1500022	<a href="#">a1740cb93639</a>	29 июля 2024	15.0-CURRENT после удаления поддержки подкачки стека ядра.
1500023	<a href="#">1206cf04a717</a>	30 июля 2024	15.0-CURRENT после добавления новых флагов в <code>malloc(9)</code> .
1500024	<a href="#">e3953c036f9d</a>	2 октября 2024	15.0-CURRENT после увеличения версии <code>libmd.so.6</code> до <code>libmd.so.7</code> .
1500025	<a href="#">9d52823bf1df</a>	6 октября 2024	15.0-CURRENT после расширения поля <code>flags</code> в <code>vm_object</code> .
1500026	<a href="#">f3dbef108212</a>	23 октября 2024	15.0-CURRENT после обновления <code>llvm</code> , <code>clang</code> , <code>compiler-rt</code> , <code>libc++</code> , <code>libunwind</code> , <code>lld</code> , <code>lldb</code> и <code>openmp</code> до версии <code>llvmorg-19.1.2-0-g7ba7d8e2f7b6</code> , также известной как релиз 19.1.2.
1500027	<a href="#">893d044346d5</a>	14 ноября 2024	15.0-CURRENT после скрытия <code>struct ifnet</code> и изменения KPI регистрации устройств <code>sound(4)</code> .

Значение	Версия	Дата	Релиз
1500028	<a href="#">cab31f5633c1</a>	25 ноября 2024	15.0-CURRENT после добавления флага <code>TDA_PSELECT</code> для раннего восстановления масок сигналов.
1500029	<a href="#">46297859a745</a>	6 декабря 2024	15.0-CURRENT после добавления <code>bus_attach_children</code> , <code>bus_detach_children</code> и <code>bus_identify_children</code> .
1500030	<a href="#">b196276c20b5</a>	2 января 2025	15.0-CURRENT после изменения <code>bus_generic_detach</code> для удаления дочерних устройств после их отключения.
1500037	<a href="#">7acd5af48cf1</a>	12 апреля 2025	15.0-CURRENT после внесения изменений в alloc в LinuxKPI.
1500038	<code>gitref:[repository="src", length=12]</code>	19 апреля 2025	15.0-CURRENT после удаления <code>vm_page_next()</code> и <code>_prev</code> .
1500039	<code>gitref:[repository="src", length=12]</code>	4 мая 2025	15.0-CURRENT после введения правильно типизированных <code>jiffies</code> .
1500040	<code>gitref:[repository="src", length=12]</code>	4 мая 2025	15.0-CURRENT после изменения внутреннего API между модулями <code>nfsccommon</code> и <code>nfsc</code> .
1500045	<a href="#">a02180cf60a6</a>	3 июня 2025	15.0-CURRENT после внесения изменений <code>dma-mapping.h</code> из <code>drm-kmod</code> в LinuxKPI.
1500062	<a href="#">567e6250c003</a>	17 августа 2025	15.0-CURRENT после добавления <code>VTYPE_ISDEV()</code> , <code>VN_ISDEV()</code> и <code>VATTR_ISDEV()</code> .

## 18.2. Версии FreeBSD 14

Таблица 54. Значения `__FreeBSD_version` в FreeBSD 14

Значение	Версия	Дата	Релиз
1400000	<a href="#">a53ce3fc4938</a>	22 января 2021	14.0-CURRENT.
1400001	<a href="#">739ecbcf1c4f</a>	23 января 2021	14.0-CURRENT после добавления поддержки символьных ссылок к бесблокировочному поиску.
1400002	<a href="#">2cf84258922f</a>	26 января 2021	14.0-CURRENT после исправления утверждения clang при сборке порта <a href="#">devel/onetbb</a> .
1400003	<a href="#">d386f3a3c32f</a>	28 января 2021	14.0-CURRENT после добавления различных компонентов LinuxKPI, конфликтующих с <code>drm-kmod</code> .
1400004	<a href="#">68f6800ce05c</a>	8 февраля 2021	14.0-CURRENT после изменения интерфейсов ядра для диспетчеризации криптографических операций.
1400005	<a href="#">45eabf5754ac</a>	17 февраля 2021	14.0-CURRENT после изменения API <a href="#">ptrace(2)</a> <code>PT_GETDBREGS</code> / <code>PT_SETDBREGS</code> на arm64.
1400006	<a href="#">c96151d33509</a>	17 марта 2021	14.0-CURRENT после добавления перечисляющих ioctl в <a href="#">sndstat(4)</a> .
1400007	<a href="#">d36d68161517</a>	6 апреля 2021	14.0-CURRENT после исправления некорректного <a href="#">dmpi_tls_data</a> .

Значение	Версия	Дата	Релиз
1400008	<a href="#">e152bbeeb221</a>	11 апреля 2021	14.0-CURRENT после изменения внутреннего КАПИ между модулями <code>krpc</code> и NFS.
1400009	<a href="#">9ca874cf740e</a>	20 апреля 2021	14.0-CURRENT после добавления поддержки TCP LRO для VLAN и VxLAN.
1400010	<a href="#">a3a02acde100</a>	21 апреля 2021	14.0-CURRENT после изменения схемы и определений <code>nvlst</code> для ioctl <code>sndstat(4)</code> .
1400015	<a href="#">d72cd275187c</a>	25 мая 2021	14.0-CURRENT после добавления дополнительных изменений LinuxKPI, требующих корректировки <code>drm-kmod</code> .
1400016	<a href="#">21e3c1fbe246</a>	25 мая 2021	14.0-CURRENT после удаления поддержки программных бэкендов KTLS.
1400017	<a href="#">beb817edfe22</a>	25 мая 2021	14.0-CURRENT после добавления <code>crypto_cursor_segment()</code> .
1400018	<a href="#">a4b07a2701f5</a>	30 мая 2021	14.0-CURRENT после разрешения реализации <code>VFS_QUOTACTL(9)</code> указывать изменения состояния занятости.
1400019	<a href="#">37d64dcdfa51</a>	7 июня 2021	14.0-CURRENT после включения <code>pr_err_once()</code> в LinuxKPI <code>printk.h</code> .

Значение	Версия	Дата	Релиз
1400020	<a href="#">8a1a42b2a7a4</a>	9 июня 2021	14.0-CURRENT после добавления макросов для <code>might_lock_nested()</code> и <code>lockdep_(re/un/)pin_lock()</code> в LinuxKPI.
1400021	<a href="#">b47f461c8e67</a>	10 июня 2021	14.0-CURRENT после добавления макроса <code>list_for_each_entry_lockless()</code> в LinuxKPI.
1400022	<a href="#">40cc9a3a6b81</a>	11 июня 2021	14.0-CURRENT после коммита <a href="#">e1a907a25cfa</a> изменил внутренний KAPI между модулями <code>krcp</code> и <code>nfserver</code> .
1400023	<a href="#">d409305fa383</a>	13 июня 2021	14.0-CURRENT после обновления <code>llvm</code> , <code>clang</code> , <code>compiler-rt</code> , <code>libc++</code> , <code>libunwind</code> , <code>lld</code> , <code>lldb</code> и <code>openmp</code> до версии <code>llvmtorg-12.0.0-0-gd28af7c654d8</code> , также известной как релиз 12.0.0.
1400024	<a href="#">41dfd8bd6466</a>	18 июня 2021	14.0-CURRENT после различных добавлений в LinuxKPI.
1400025	<a href="#">5fa1eb1cd927</a>	5 июля 2021	14.0-CURRENT после различных добавлений в LinuxKPI.
1400026	<a href="#">fad3f322efb5</a>	16 июля 2021	14.0-CURRENT после изменения внутреннего KAPI между модулями <code>nfsccommon</code> и <code>nfscd</code> .

Значение	Версия	Дата	Релиз
1400027	<a href="#">cc55ee8009a5</a>	28 июля 2021	14.0-CURRENT после добавления вспомогательных функций LSE атомарных операций вне строки в <code>libcompiler_rt.a</code> для архитектуры <code>aarch64</code> .
1400028	<a href="#">792b602a337d</a>	31 июля 2021	14.0-CURRENT после обеспечения потокобезопасности разделов FPU в <code>LinuxKPI</code> .
1400029	<a href="#">245ec7651e42</a>	5 августа 2021	14.0-CURRENT после добавления <code>fspacectl(2)</code> , <code>vn_deallocate(9)</code> и <code>VOP_DEALLOCATE(9)</code> .
1400030	<a href="#">95941b963606</a>	12 августа 2021	14.0-CURRENT после изменений параметров <code>VOP_DEALLOCATE(9)</code> и добавления поддержки <code>fspacectl(2)</code> для POSIX разделяемой памяти.
1400031	<a href="#">1a4c5061fc5b</a>	24 августа 2021	14.0-CURRENT после изменения <code>fspacectl(2)</code> , <code>vn_deallocate(9)</code> и <code>VOP_DEALLOCATE(9)</code> для обновления <code>rmsr.r_offset</code> до значимого значения.
1400032	<a href="#">76321d2d432e</a>	25 августа 2021	14.0-CURRENT после изменения <code>fspacectl(2)</code> , <code>vn_deallocate(9)</code> и <code>VOP_DEALLOCATE(9)</code> для упрощения подсчёта количества обнулённых байт.

Значение	Версия	Дата	Релиз
1400033	<a href="#">c751d067c166</a>	7 сентября 2021	14.0-CURRENT после перемещения блокировок буфера сокета в содержащий сокет и переименования sb(un)lock в SOCK_IO_RECV_LOCK, SOCK_IO_RECV_UNLOCK, SOCK_IO_SEND_LOCK и SOCK_IO_SEND_UNLOCK.
1400034	<a href="#">c751d067c166</a>	29 сентября 2021	14.0-CURRENT после изменений в LinuxKPI.
1400035	<a href="#">16f1ee11e657</a>	4 октября 2021	14.0-CURRENT после разделения libtinfow и libncurses.
1400036	<a href="#">ac847dbf7368</a>	6 октября 2021	14.0-CURRENT после расширения шифров AES-CCM и ChaCha20-Poly1305 в OCF для поддержки нескольких длин одноразовых номеров.
1400037	<a href="#">2b68eb8e1dbb</a>	11 октября 2021	14.0-CURRENT после удаления аргумента thread из <a href="#">VOP_STAT(9)</a> и <a href="#">fo_stat</a> .
1400038	<a href="#">0d6516b45346</a>	17 октября 2021	14.0-CURRENT после того, как LinuxKPI получил поддержку отложенного выделения VAR.
1400039	<a href="#">bd49c454ca62</a>	19 октября 2021	14.0-CURRENT после изменений в аллокаторе страниц.
1400040	<a href="#">f38bef2ce417</a>	30 октября 2021	14.0-CURRENT после увеличения номера версии разделяемой библиотеки libdialog.

Значение	Версия	Дата	Релиз
1400041	<a href="#">0c276dee030b</a>	6 ноября 2021	14.0-CURRENT после изменения аргументов для <a href="#">VOP_ALLOCATE(9)</a> .
1400042	<a href="#">20aa359773be</a>	13 ноября 2021	14.0-CURRENT после обновления llvm, clang, compiler-rt, libc++, libunwind, lld, lldb и openmp до llvmorg-13.0.0-0-gd7b669b3a303, также известного как релиз 13.0.0.
1400043	<a href="#">7e1d3eefd410</a>	25 ноября 2021	14.0-CURRENT после удаления неиспользуемого аргумента потока из <a href="#">NDINIT(9)*</a> .
1400044	<a href="#">ec434c85b46d</a>	9 декабря 2021	14.0-CURRENT после изменения встроенных программных криптографических преобразований для поддержки AEAD-шифров и изменения аутентификационных преобразований Blake-2S/B для поддержки Init перед Setkey, как в других аутентификационных преобразованиях.
1400045	<a href="#">b214fcsceacad</a>	15 декабря 2021	14.0-CURRENT после изменения аргумента cookies в <a href="#">VOP_READDIR(9)</a> на <b>**uint64_t</b> .
1400046	<a href="#">e2650af157bc</a>	30 декабря 2021	14.0-CURRENT после приведения макросов CPU_SET в соответствие с glibc.

Значение	Версия	Дата	Релиз
1400047	<a href="#">ed6417cd8d0b</a>	January 17, 2022	14.0-CURRENT после множества изменений LinuxKPI, необходимых для <code>drm-kmod</code> .
1400048	<a href="#">dd2f7a4b45eb</a>	18 января 2022	14.0-CURRENT после добавления <code>&lt;crypto/chacha20_poly1305.h&gt;</code> .
1400049	<a href="#">2c4b65cc3d22</a>	January 24, 2022	14.0-CURRENT after adding <code>&lt;crypto/curve25519.h&gt;</code> .
1400050	<a href="#">213e91399b79</a>	25 января 2022	14.0-CURRENT после <code>iflib</code> добавляет возможность, при которой драйвер может установить свою собственную функцию выбора TX-очереди как <code>ift_txq_select</code> в структуре <code>if_txrx</code> .
1400051	<a href="#">59d465e200bb</a>	25 января 2022	14.0-CURRENT после добавления поддержки <code>i2c</code> для LinuxKPI.
1400052	<a href="#">05f0b24bfb34</a>	February 14, 2022	14.0-CURRENT после добавления поддержки <code>GUID_INIT</code> и <code>rm_qos.h</code> для LinuxKPI.
1400053	<a href="#">ba87e9bf7420</a>	February 17, 2022	14.0-CURRENT после добавления <code>mmap_lock.h</code> в LinuxKPI.
1400054	<a href="#">50bb3a33d879</a>	28 марта 2022	14.0-CURRENT после изменения <code>irq_work_queue</code> для возврата типа <code>bool</code> в LinuxKPI в соответствии с API 5.10.

Значение	Версия	Дата	Релиз
1400055	<a href="#">d69af4758be9</a>	29 марта 2022	14.0-CURRENT после добавления <code>for_each_sgtable_dma_sg</code> и <code>for_each_sgtable_dma_page</code> в LinuxKPI
1400056	<a href="#">ab8ac4c28574</a>	31 марта 2022	14.0-CURRENT после обновления zlib до версии 1.2.12
1400057	<a href="#">e68b35e40881</a>	22 апреля 2022	14.0-CURRENT после изменения прототипа <code>udp_tun_func_t()</code> .
1400058	<a href="#">2e32d4e41d20</a>	7 мая 2022	14.0-CURRENT после изменений в <code>newbus</code> для удаления аргументов <code>devclass</code> .
1400059	<a href="#">3a9a9c0ca44e</a>	14 мая 2022	14.0-CURRENT после обновления <code>llvm</code> , <code>clang</code> , <code>compiler-rt</code> , <code>libc++</code> , <code>libunwind</code> , <code>lld</code> , <code>lldb</code> и <code>openmp</code> до версии <code>lvmorg-14.0.3-0-g1f9140064dfb</code> , также известной как релиз 14.0.3.
1400060	<a href="#">85d7875d4291</a>	6 июня 2022	14.0-CURRENT после исправлений <code>LinuxKPI dmi_matches()</code> .
1400061	<a href="#">c4c5981c14d5</a>	8 июня 2022	14.0-CURRENT после изменений структуры <code>mbuf(9)</code> .
1400062	<a href="#">8c309d48aabf</a>	18 июня 2022	14.0-CURRENT после изменений структуры <code>kinfo_file</code> .
1400063	<a href="#">8cff8e6e13a6</a>	29 июня 2022	14.0-CURRENT после множества изменений LinuxKPI, необходимых для <code>drm-kmod</code> .
1400064	<a href="#">ddd9004e7a5d</a>	18 июля 2022	14.0-CURRENT после удаления <code>OBJT_DEFAULT</code> .

Значение	Версия	Дата	Релиз
1400065	<a href="#">b273f93657cf</a>	8 августа 2022	14.0-CURRENT после множества изменений LinuxKPI, необходимых для <code>drm-kmod</code> .
1400066	<a href="#">ff7812ee7d44</a>	18 августа 2022	14.0-CURRENT после множества изменений LinuxKPI, необходимых для <code>drm-kmod</code> .
1400069	<a href="#">f95c0bc89ea4</a>	22 сентября 2022	14.0-CURRENT после нескольких изменений в LinuxKPI.
1400070	<a href="#">6bddde307e21</a>	22 сентября 2022	14.0-CURRENT после изменений KPI в <code>rmap_unmapdev()</code> и <code>kmem_*</code> .
1400071	<a href="#">d3f96f661050</a>	26 сентября 2022	14.0-CURRENT после изменений KPI, когда списки OID <code>sysctl</code> были преобразованы в RB-деревья.
1400072	<a href="#">8a96874eeee</a>	22 сентября 2022	14.0-CURRENT после изменения прототипа <code>qsort_r</code> для соответствия POSIX.
1400073	<a href="#">9c9501390512</a>	17 октября 2022	14.0-CURRENT after introduction of v2 of TX Queue Select Functionality.
1400074	<a href="#">e28932c643e8</a>	9 декабря 2022	14.0-CURRENT после добавления запасных слотов <code>fops</code> в <code>fileops</code> .
1400078	<a href="#">4b56afaf7bf4</a>	13 января 2023	14.0-CURRENT после изменения LinuxKPI <code>pci.h</code> .

Значение	Версия	Дата	Релиз
1400079	<a href="#">3264f6b88fce</a>	8 февраля 2023	14.0-CURRENT после обновления llvm, clang, compiler-rt, libc++, libunwind, lld, lldb и openmp до llvmmorg-15.0.7-0-g8dfdcc7b7bf6, также известного как релиз 15.0.7.
1400084	<a href="#">ea3061526e9c</a>	23 марта 2023	14.0-CURRENT после изменения структур reg, greg, trapframe и pcb для архитектуры arm64.
1400085	<a href="#">1ceb9298cf2</a>	28 марта 2023	14.0-CURRENT после нескольких изменений в LinuxKPI.
1400086	<a href="#">c17eb99a66e7</a>	8 апреля 2023	14.0-CURRENT после изменений аргументов vn_lock_pair().
1400087	<a href="#">af22da75a035</a>	22 апреля 2023	14.0-CURRENT после обновлений LinuxKPI.
1400088	<a href="#">97583aa25675</a>	24 апреля 2023	14.0-CURRENT после миграции LinuxKPI на IfAPI.
1400089	<a href="#">9fb6718d1b18</a>	25 апреля 2023	14.0-CURRENT после динамического выделения массива stoprcbs в smp.
1400090	<a href="#">653738e895ba</a>	7 июня 2023	14.0-CURRENT после того, как ptrace начал очищать TDB_BORN во время PT_DETACH.

Значение	Версия	Дата	Релиз
1400091	<a href="#">a681cba16d89</a>	22 июня 2023	14.0-CURRENT после обновления llvm, clang, compiler-rt, libc++, libunwind, lld, lldb и openmp до llvmmorg-16.0.6-0-g7cbf1a259152, также известного как релиз 16.0.6.
1400092	<a href="#">9ead001d5b42</a>	24 июня 2023	14.0-CURRENT после импорта OpenSSL 3.0.9 в базовую систему.
1400093	<a href="#">ba8cc6d7271a</a>	5 июля 2023	14.0-CURRENT после использования <code>__enum_uint8</code> для <code>vtype</code> и <code>vstate</code> в VFS
1400097	<a href="#">29a16ce065db</a>	24 августа 2023	14.0-STABLE после ветвления stable/14
1400500	<a href="#">29a16ce065db</a>	8 сентября 2023	14.0-STABLE после ветвления releng/14.0
1400501	<a href="#">91e53779b4fc</a>	19 ноября 2023	14.0-STABLE после реализации <code>fpu_kern_enter</code> и <code>fpu_kern_leave</code> для powerpc.
1400502	<a href="#">092abb839d1d</a>	24 декабря 2023	14.0-STABLE после изменения внутреннего API между модулями <code>kgssapi</code> и <code>krpc</code> .
1400503	<a href="#">ba99d960884d</a>	29 декабря 2023	14.0-STABLE после изменения внутреннего KAPI между модулями <code>nfsccommon</code> и <code>nfscsl</code> .

Значение	Версия	Дата	Релиз
1400504	<a href="#">68584c97ecfb</a>	7 января 2024	14.0-STABLE после обновления llvm, clang, compiler-rt, libc++, libunwind, lld, lldb и openmp до Lvmorg-17.0.6-0-g6009708b4367, также известного как релиз 17.0.6.
1400505	<a href="#">64e869e9b93c</a>	7 января 2024	14.0-STABLE после добавления <a href="#">vnode_pager_clean_async(9)</a> и <a href="#">vnode_pager_clean_sync(9)</a> .
1400506	<a href="#">d90417109582</a>	19 января 2024	14.0-STABLE после изменения внутреннего KAPI между модулями nfscommon и nfscd.
1400507	<a href="#">b566e44b2b88</a>	31 января 2024	14.0-STABLE после добавления <a href="#">kern_openatfp(9)</a> и <a href="#">kcmp(2)</a> .
1400508	<a href="#">2d120981e26d</a>	18 февраля 2024	14.0-STABLE после обновлений LinuxKPI.
1400509	<a href="#">b392b36d3776</a>	18 февраля 2024	14.0-STABLE после изменения внутренней структуры <a href="#">struct ieee80211var</a> в net80211.
1400510	<a href="#">69da6e087983</a>	23 марта 2024	14.0-STABLE после исправления утверждения или падения clang при сборке последних библиотек boost.

Значение	Версия	Дата	Релиз
1400511	<a href="#">7c41358a2b0a</a>	20 апреля 2024	14.0-STABLE после обновления llvm, clang, compiler-rt, libc++, libunwind, lld, lldb и openmp до версии llvmtorg-18.1.3-0-gc13b7485b879, также известной как релиз 18.1.3.
1401500	<a href="#">7b082bdf72e6</a>	2 мая 2024	14.1-STABLE после переименования из 14.1-PRERELEASE.
1401501	<a href="#">f285eabc89ce</a>	6 июня 2024	14.1-STABLE после добавления модуля linuxkpi_video.
1401502	<a href="#">b37a6d41a046</a>	2 августа 2024	14.1-STABLE после изменений в LinuxKPI.
1401503	<a href="#">8a5a9dbf389e</a>	15 октября 2024	14.1-STABLE после расширения поля <code>flags</code> в <code>vm_object</code> .
1402500	<a href="#">4e8444d5750a</a>	31 октября 2024	14.2-STABLE после переименования из 14.2-PRERELEASE.
1402501	<a href="#">35d2f335e855</a>	1 декабря 2024	14.2-STABLE после обновления llvm, clang, compiler-rt, libc++, libunwind, lld, lldb и openmp до llvmtorg-19.1.4-0-gaadaa00de76e, также известного как релиз 19.1.4.
1402502	<a href="#">d172f42e4ba7</a>	27 февраля 2025	14.2-STABLE после удаления избыточных аргументов <code>type</code> и <code>rid</code> из нескольких функций в API ресурсов new-bus.

Значение	Версия	Дата	Релиз
1402503	<a href="#">4aed8b3b613c</a>	27 февраля 2025	14.2-STABLE после добавления <code>bus_attach_children</code> , <code>bus_detach_children</code> и <code>bus_identify_children</code> .
1402505	<a href="#">a3b2d8e360c3</a>	18 апреля 2025	14.2-STABLE после изменения аллокации в <code>LinuxKPI</code> и удаления микропрограммы <code>iwlfwifi</code> .
1403503	<a href="#">6cdcf08c9c5e</a>	13 июля 2025	14.3-STABLE после изменений в <code>LinuxKPI</code> <code>dma-mapping.h</code> and <code>acpi</code> .

## 18.3. FreeBSD 13 Версии

Таблица 55. FreeBSD 13 Значения `__FreeBSD_version`

Значение	Версия	Дата	Релиз
1300000	<a href="#">339436</a>	19 октября 2018	13.0-CURRENT.
1300001	<a href="#">339730</a>	25 октября 2018	13.0-CURRENT после увеличения номеров версий разделяемых библиотек <code>OpenSSL</code> .
1300002	<a href="#">339765</a>	25 октября 2018	13.0-CURRENT после восстановления <code>sys/joystick.h</code> .
1300003	<a href="#">340055</a>	2 ноября 2018	13.0-CURRENT после изменения API <code>vor_symlink</code> (параметр <code>a_target</code> теперь объявлен как <code>const</code> .)
1300004	<a href="#">340841</a>	23 ноября 2018	13.0-CURRENT после включения кода <code>crtbegin</code> и <code>crtend</code> .
1300005	<a href="#">341836</a>	11 декабря 2018	13.0-CURRENT после включения контрольных сумм <code>inode</code> в <code>UFS</code> .

Значение	Версия	Дата	Релиз
1300006	<a href="#">342398</a>	24 декабря 2018	13.0-CURRENT после исправления включения <code>sys/random.h</code> для использования из C++.
1300007	<a href="#">342629</a>	30 декабря 2018	13.0-CURRENT после изменения размера <code>struct linux_cdev</code> на 32-битных платформах.
1300008	<a href="#">342772</a>	4 января 2019	13.0-CURRENT после добавления системных переменных <code>kern.smp.threads_per_core</code> и <code>kern.smp.cores</code> .
1300009	<a href="#">343213</a>	20 января 2019	13.0-CURRENT после изменения структуры <code>struct ieee80211var</code> для устранения состояния гонки между <code>ioctl</code> и <code>detach</code> в структуре <code>ieee80211com</code> .
1300010	<a href="#">343485</a>	27 января 2019	13.0-CURRENT после увеличения <code>SPECNAMELEN</code> с 63 до <code>MAXNAMELEN</code> (255).
1300011	<a href="#">344041</a>	12 февраля 2019	13.0-CURRENT после исправления <code>renameat(2)</code> для работы с ядрами, собранными с опцией <code>CAPABILITIES</code> .

Значение	Версия	Дата	Релиз
1300012	<a href="#">344062</a>	12 февраля 2019	13.0-CURRENT после того, как <code>taskqgroup_attach()</code> и <code>taskqgroup_attach_cpu()</code> принимают аргументы <code>device_t</code> и указатель на структуру <code>resource</code> для обозначения прерываний устройства.
1300013	<a href="#">344300</a>	19 февраля 2019	13.0-CURRENT после удаления <code>drm</code> и <code>drm2</code> .
1300014	<a href="#">344779</a>	4 марта 2019	13.0-CURRENT после обновления <code>clang</code> , <code>llvm</code> , <code>lld</code> , <code>lldb</code> , <code>compiler-rt</code> и <code>libc++</code> до версии 8.0.0 rc3.
1300015	<a href="#">345196</a>	15 марта 2019	13.0-CURRENT после деанонимизации перечислений состояний потоков и процессов, что позволяет приложениям пользовательского пространства использовать их без переопределения имен значений.
1300016	<a href="#">345236</a>	16 марта 2019	13.0-CURRENT после включения LLVM OpenMP 8.0.0 rc5 на amd64 по умолчанию.
1300017	<a href="#">345305</a>	19 марта 2019	13.0-CURRENT после раскрытия размера буфера <code>Rx mbuf</code> для драйверов в <code>iflib</code> .
1300018	<a href="#">346012</a>	16 марта 2019	13.0-CURRENT после введения системного вызова <code>funlinkat</code> в <a href="#">345982</a> .

Значение	Версия	Дата	Релиз
1300019	<a href="#">346282</a>	16 апреля 2019	13.0-CURRENT после добавления <a href="#">is_random_seeded(9)</a> в <a href="#">random(4)</a> .
1300020	<a href="#">346358</a>	18 апреля 2019	13.0-CURRENT после восстановления доступности <a href="#">random(4)</a> с учетом компромиссов до <a href="#">346250</a> и добавления новых настроек и диагностических <code>sysctl</code> для программного обнаружения проблем с ранней инициализацией семени после загрузки.
1300021	<a href="#">346645</a>	24 апреля 2019	13.0-CURRENT после того, как LinuxKPI использует <a href="#">bus_dma(9)</a> для совместимости с IOMMU.
1300022	<a href="#">347089</a>	4 мая 2019	13.0-CURRENT после исправления регрессии, возникшей после <a href="#">346645</a> в LinuxKPI.
1300023	<a href="#">347192</a>	6 мая 2019	13.0-CURRENT после преобразования конфигурации устройства дампа ядра в список.
1300024	<a href="#">347325</a>	8 мая 2019	13.0-CURRENT после увеличения номеров версий драйверов Mellanox ( <a href="#">mlx4en(4)</a> ; <a href="#">mlx5en(4)</a> ).

Значение	Версия	Дата	Релиз
1300025	<a href="#">347532</a>	13 мая 2019	13.0-CURRENT после переименования <code>vm.max_wired</code> в <code>vm.max_user_wired</code> и изменения его типа.
1300026	<a href="#">347596</a>	14 мая 2019	13.0-CURRENT после добавления члена контекста к <code>ww_mutex</code> в LinuxKPI.
1300027	<a href="#">347601</a>	14 мая 2019	13.0-CURRENT после добавления <code>prereq</code> в <code>pm_ops</code> в LinuxKPI.
1300028	<a href="#">347925</a>	17 мая 2019	13.0-CURRENT после удаления драйверов <code>bm</code> , <code>cs</code> , <code>de</code> , <code>ed</code> , <code>ep</code> , <code>ex</code> , <code>fe</code> , <code>pcn</code> , <code>sf</code> , <code>sn</code> , <code>tl</code> , <code>tx</code> , <code>txp</code> , <code>vx</code> , <code>wb</code> и <code>xe</code> .
1300029	<a href="#">347984</a>	20 мая 2019	13.0-CURRENT после удаления некоторых загрязнений заголовков из-за <code>sys/eventhandler.h</code> . Затронутые файлы теперь могут требовать явного включения одного или нескольких заголовков: <code>sys/eventhandler.h</code> , <code>sys/ktr.h</code> , <code>sys/lock.h</code> или <code>sys/mutex.h</code> , тогда как ранее они могли включаться неявно до версии 1300029.

Значение	Версия	Дата	Релиз
1300030	<a href="#">348350</a>	29 мая 2019	13.0-CURRENT после добавления поддержки перемещения в libdwarf на powerpc64 для исправления обработки DWARF-информации в несвязанных объектах. Оригинальный коммит в ссылке: <a href="https://svnweb.freebsd.org/changeset/base/348347">https://svnweb.freebsd.org/changeset/base/348347</a> [348347].
1300031	<a href="#">348808</a>	8 июня 2019	13.0-CURRENT после добавления исправлений разделов drcpu и vnet в модули ядра i386 для предотвращения паники в определённых условиях. Модули ядра i386 необходимо перекомпилировать с включёнными изменениями в скрипт компоновщика, иначе они откажутся загружаться.
1300032	<a href="#">349151</a>	17 июня 2019	13.0-CURRENT после выделения реализации <code>gsc32()</code> ядра в отдельный заголовочный файл ( <code>gsb_crc32.h</code> ) и переименования исходного файла в <code>gsb_crc32.c</code> .
1300033	<a href="#">349277</a>	June 21, 2019	13.0-CURRENT после добавлений в список <code>rcu</code> LinuxKPI.

Значение	Версия	Дата	Релиз
1300034	<a href="#">349352</a>	24 июня 2019	13.0-CURRENT после удаления NAND и NANDFS.
1300035	<a href="#">349846</a>	8 июля 2019	13.0-CURRENT после объединения механизмов удержания и фиксации <code>vm_page</code> .
1300036	<a href="#">349972</a>	13 июля 2019	13.0-CURRENT после добавления <code>arm_drain_writebuf()</code> и <code>arm_sync_icache()</code> для совместимости с NetBSD и OpenBSD.
1300037	<a href="#">350307</a>	24 июля 2019	13.0-CURRENT после удаления <code>libcap_random(3)</code> .
1300038	<a href="#">350437</a>	30 июля 2019	13.0-CURRENT после удаления поддержки gzip'ed a.out.
1300039	<a href="#">350665</a>	7 августа 2019	13.0-CURRENT после объединения fusefs из projects/fuse2.
1300040	<a href="#">351140</a>	16 августа 2019	13.0-CURRENT после удаления sys/dir.h, который был устаревшим с 1997 года.
(не изменено)	<a href="#">351423</a>	23 августа 2019	13.0-CURRENT после изменения большинства аргументов в <code>ping6(8)</code> .
1300041	<a href="#">351480</a>	25 августа 2019	13.0-CURRENT после удаления zlib 1.0.4 по завершении унификации zlib в ядре.
1300042	<a href="#">351522</a>	27 августа 2019	13.0-CURRENT после добавления поддержки TLS внутри ядра на уровне ядра.

Значение	Версия	Дата	Релиз
1300043	<a href="#">351698</a>	2 сентября 2019	13.0-CURRENT после удаления <a href="#">gets(3)</a> .
1300044	<a href="#">351701</a>	2 сентября 2019	13.0-CURRENT после добавления функций создания/удаления <code>sysfs</code> , обрабатывающих несколько файлов за один вызов, в LinuxKPI.
1300045	<a href="#">351729</a>	3 сентября 2019	13.0-CURRENT после добавления системного вызова <a href="#">sysctlbyname(3)</a> .
1300046	<a href="#">351937</a>	6 сентября 2019	13.0-CURRENT после улучшений LinuxKPI <code>sysfs</code> .
1300047	<a href="#">352110</a>	9 сентября 2019	13.0-CURRENT после изменения правил синхронизации для подсчета ссылок <a href="#">vm_page</a> .
1300048	<a href="#">352700</a>	25 сентября 2019	13.0-CURRENT после добавления системного вызова <code>shm_open2</code> для поддержки готовящегося системного вызова <a href="#">memfd_create(2)</a> .
1300049	<a href="#">353274</a>	7 октября 2019	13.0-CURRENT после вынесения проверки отключения VNET в отдельное поле структуры <code>vnet</code> .
1300050	<a href="#">353358</a>	9 октября 2019	13.0-CURRENT после обновления <code>llvm</code> , <code>clang</code> , <code>compiler-rt</code> , <code>libc++</code> , <code>libunwind</code> , <code>lld</code> , <code>lldb</code> и <code>openmp</code> до финального релиза 9.0.0 r372316.

Значение	Версия	Дата	Релиз
1300051	<a href="#">353685</a>	17 октября 2019	13.0-CURRENT после выделения более универсального <a href="#">debugnet(4)</a> из <a href="#">netdump(4)</a> .
1300052	<a href="#">353698</a>	17 октября 2019	13.0-CURRENT после преобразования поля <code>busy page</code> в полноценную блокировку, которая больше не требует блокировки объекта для обеспечения согласованности.
1300053	<a href="#">353700</a>	17 октября 2019	13.0-CURRENT после реализации NetGDB.
1300054	<a href="#">353868</a>	21 октября 2019	13.0-CURRENT после удаления устаревших KPIs, которые использовались для доступа к спискам адресов интерфейсов.
1300055	<a href="#">354335</a>	4 ноября 2019	13.0-CURRENT после включения атрибутов группы классов устройств в <code>LinuxKPI</code> .
1300056	<a href="#">354460</a>	7 ноября 2019	13.0-CURRENT после исправления потенциальной проблемы безопасности с чтением за границами в <code>libc++</code> .
1300057	<a href="#">354694</a>	13 ноября 2019	13.0-CURRENT после добавления поддержки <code>AT_EXCPATH</code> в <a href="#">elf_aux_info(3)</a> .
1300058	<a href="#">354820</a>	18 ноября 2019	13.0-CURRENT после расширения поля <code>aflags</code> в <code>vm_page</code> до 16 бит.

Значение	Версия	Дата	Релиз
1300059	<a href="#">354835</a>	18 ноября 2019	13.0-CURRENT после преобразования встроенных целей <code>sysent</code> для использования нового <code>makesyscalls.lua</code> .
1300060	<a href="#">354922</a>	20 ноября 2019	13.0-CURRENT после добавления символической ссылки <code>/etc/os-release</code> на <code>/var/run/os-release</code> .
1300061	<a href="#">354977</a>	21 ноября 2019	13.0-CURRENT после добавления функций в <code>bitstring(3)</code> для поиска непрерывных последовательностей установленных или сброшенных битов.
1300062	<a href="#">355309</a>	2 декабря 2019	13.0-CURRENT после добавления поддержки <code>TCP_STATS</code> .
1300063	<a href="#">355537</a>	8 декабря 2019	13.0-CURRENT после удаления <code>VI_DOOMED</code> (используйте <code>VN_IS_DOOMED</code> вместо этого).
1300064	<a href="#">355658</a>	9 декабря 2019	13.0-CURRENT после исправления проверки версии C++ для объявления <code>timespec_get(3)</code> .
1300065	<a href="#">355643</a>	12 декабря 2019	13.0-CURRENT после добавления расширений <code>sigsetop</code> , которые обычно встречаются в <code>musl libc</code> и <code>glibc</code> .

Значение	Версия	Дата	Релиз
1300066	<a href="#">355679</a>	12 декабря 2019	13.0-CURRENT после изменения внутреннего интерфейса между модулями NFS в рамках внедрения NFS 4.2.
1300067	<a href="#">355732</a>	13 декабря 2019	13.0-CURRENT после удаления устаревших функций <code>callout_handle_init</code> , <code>timeout</code> и <code>untimeout</code> .
1300068	<a href="#">355828</a>	16 декабря 2019	13.0-CURRENT после удвоения значения <code>ARG_MAX</code> для 64-битных платформ.
1300069	<a href="#">356051</a>	24 декабря 2019	13.0-CURRENT после добавления шаблонов <code>busdma</code> .
1300070	<a href="#">356113</a>	27 декабря 2019	13.0-CURRENT после устранения последнего различия MI в определениях <code>AT_*</code> (для <code>powerpc</code> ).
1300071	<a href="#">356135</a>	27 декабря 2019	13.0-CURRENT после изменения статистики USB для каждого устройства вместо каждой шины.
1300072	<a href="#">356185</a>	29 декабря 2019	13.0-CURRENT после удаления класса <code>GEOM_SCHED</code> и утилиты <code>gsched</code> .
1300073	<a href="#">356263</a>	2 января 2020	13.0-CURRENT после удаления <code>arm/arm</code> как допустимой цели.
1300074	<a href="#">356337</a>	3 января 2020	13.0-CURRENT после удаления аргумента <code>flags</code> из <code>VOP_UNLOCK</code> .

Значение	Версия	Дата	Релиз
1300075	<a href="#">356409</a>	6 января 2020	13.0-CURRENT после добавления собственного счетчика отмененных USB-передач.
1300076	<a href="#">356511</a>	8 января 2020	13.0-CURRENT после внедрения реализации <code>vpor</code> в слой <code>fileop</code> в <code>posix_fallocate(2)</code> .
(не изменено)	<a href="#">357396</a>	2 февраля 2020	13.0-CURRENT после удаления кода архитектуры <code>armv5</code> из дерева <code>src</code> .
1300077	<a href="#">357455</a>	3 февраля 2020	13.0-CURRENT после удаления кода архитектуры <code>sparc64</code> из дерева <code>src</code> .
1300078	<a href="#">358020</a>	17 февраля 2020	13.0-CURRENT после изменения <code>struct vnet</code> и волшебного cookie <code>VNET</code> .
1300079	<a href="#">358164</a>	20 февраля 2020	13.0-CURRENT после обновления <code>ncurses</code> до версии 6.2.x
1300080	<a href="#">358172</a>	20 февраля 2020	13.0-CURRENT после добавления системного вызова <code>realpathat</code> в <code>VFS</code> .
1300081	<a href="#">358218</a>	21 февраля 2020	13.0-CURRENT после недавних изменений в <code>linuxkri</code> .
1300082	<a href="#">358497</a>	1 марта 2020	13.0-CURRENT после удаления <code>bktr(4)</code> .
1300083	<a href="#">358834</a>	10 марта 2020	13.0-CURRENT после удаления <code>amd(8)</code> , <code>r358821</code> .

Значение	Версия	Дата	Релиз
1300084	<a href="#">358851</a>	10 марта 2020	13.0-CURRENT после обновления llvm, clang, compiler-rt, libc++, libunwind, lld, lldb и openmp до версии 10.0.0-rc3 c290cb61fdc.
1300085	<a href="#">359261</a>	23 марта 2020	13.0-CURRENT после импорта тестового фреймворка куа.
1300086	<a href="#">359347</a>	26 марта 2020	13.0-CURRENT после переключения powerpc и powerpcspe на компоновщик lld.
1300087	<a href="#">359374</a>	27 марта 2020	13.0-CURRENT после рефакторинга интерфейсов драйвера и потребителя для внутриядерного шифрования.
1300088	<a href="#">359530</a>	1 апреля 2020	13.0-CURRENT после удаления поддержки отладки процессов через procfs.
1300089	<a href="#">359727</a>	8 апреля 2020	13.0-CURRENT после разделения интерфейса RCU на части с возможностью ожидания и без неё в LinuxKPI.
1300090	<a href="#">359747</a>	9 апреля 2020	13.0-CURRENT после удаления старого драйвера устройства блокировки NFS, использующего Giant.
1300091	<a href="#">359839</a>	12 апреля 2020	13.0-CURRENT после реализации системного вызова <a href="#">close_range(2)</a> .

Значение	Версия	Дата	Релиз
1300092	<a href="#">359920</a>	14 апреля 2020	13.0-CURRENT после переработки немэпированных mbuf в KTLS для хранения <code>ext_pgs</code> в самом mbuf.
1300093	<a href="#">360418</a>	27 апреля 2020	13.0-CURRENT после добавления поддержки выгрузки приема TLS в ядре.
1300094	<a href="#">360796</a>	7 мая 2020	13.0-CURRENT после изменений в linuxkpi.
1300095	<a href="#">361275</a>	20 мая 2020	13.0-CURRENT после добавления поддержки сокетов HyperV для гостевых систем FreeBSD.
1300096	<a href="#">361410</a>	23 мая 2020	13.0-CURRENT после обновления llvm, clang, compiler-rt, libc++, libunwind, lld, lldb и openmp до версии 10.0.1 rc1 f79cd71e145.
1300097	<a href="#">361724</a>	2 июня 2020	13.0-CURRENT после реализации макроса функции <code>__is_constexpr()</code> в LinuxKPI.
1300098	<a href="#">362159</a>	14 июня 2020	13.0-CURRENT после изменения поля <code>export_args ex_flags</code> , чтобы оно было 64-битным.
1300099	<a href="#">362453</a>	20 июня 2020	13.0-CURRENT после перевода liblzma на использование реализации SHA256 из libmd.

Значение	Версия	Дата	Релиз
1300100	<a href="#">362640</a>	June 26, 2020	13.0-CURRENT после изменения внутреннего API между модулями ядра NFS.
1300101	<a href="#">363077</a>	10 июля 2020	13.0-CURRENT после реализации функции <code>array_size()</code> в LinuxKPI.
1300102	<a href="#">363562</a>	26 июля 2020	13.0-CURRENT после реализации бесблокировочного поиска в слое VFS.
1300103	<a href="#">363757</a>	1 августа 2020	13.0-CURRENT после того, как права для NDINIT_ALL стали обязательными.
1300104	<a href="#">363783</a>	2 августа 2020	13.0-CURRENT после изменений в структуре vnode.
1300105	<a href="#">363894</a>	5 августа 2020	13.0-CURRENT после изменения <code>vaccess()</code> .
1300106	<a href="#">364092</a>	11 августа 2020	13.0-CURRENT после добавления аргумента в <code>newnfs_connect()</code> , указывающего на использование TLS для соединения.
1300107	<a href="#">364109</a>	11 августа 2020	13.0-CURRENT после изменения для клонирования полей структуры задачи, связанных с RCU.
1300108	<a href="#">364233</a>	14 августа 2020	13.0-CURRENT после добавления нескольких функций <code>wait_bit</code> в linuxkpi, которые необходимы для DRM из Linux v5.4.

Значение	Версия	Дата	Релиз
1300109	<a href="#">364274</a>	16 августа 2020	13.0-CURRENT после удаления аргумента <code>vget()</code> и перенумерации флагов <code>namei</code> .
(не изменено)	<a href="#">364284</a>	16 августа 2020	13.0-CURRENT после обновления <code>llvm</code> , <code>clang</code> , <code>compiler-rt</code> , <code>libc++</code> , <code>libunwind</code> , <code>lld</code> , <code>lldb</code> и <code>openmp</code> до версии <code>release/11.x</code> <code>llvmorg-11.0.0-rc1-47-gff47911ddfc</code> .
1300110	<a href="#">364331</a>	18 августа 2020	13.0-CURRENT после удаления неиспользуемого аргумента <code>use_ext</code> в <code>nfsc_l_reqstart()</code> .
1300111	<a href="#">364476</a>	22 августа 2020	13.0-CURRENT после добавления поддержки TLS в RPC ядра.
1300112	<a href="#">364747</a>	August 25, 2020	13.0-CURRENT после объединения поддержки OpenZFS.
1300113	<a href="#">364753</a>	August 25, 2020	13.0-CURRENT после добавления атомарных функций и функций <code>bswap</code> в <code>libcompiler_rt</code> .
1300114	<a href="#">365459</a>	8 сентября 2020	13.0-CURRENT после изменения определений <code>AT_HWCAP</code> для <code>arm64</code> в <code>elf_aux_info(3)</code> .
1300115	<a href="#">365705</a>	14 сентября 2020	13.0-CURRENT после исправления сборки приложения <code>crunchgen(1)</code> с <code>WARNS=6</code> .
1300116	<a href="#">366062</a>	22 сентября 2020	13.0-CURRENT после введения архитектуры <code>powerpc64le</code> .

Значение	Версия	Дата	Релиз
1300117	<a href="#">366070</a>	23 сентября 2020	13.0-CURRENT после перереализации <code>purgevfs</code> для итерации по <code>vnodes</code> вместо всего хэша.
1300118	<a href="#">366374</a>	2 октября 2020	13.0-CURRENT после добавления поддержки подсветки и функций <code>dmi_*</code> в <code>linuxkpi</code> .
1300119	<a href="#">366432</a>	6 октября 2020	13.0-CURRENT после заполнения поля контекста получения <code>ww_mutex</code> в <code>LinuxKPI</code> .
1300120	<a href="#">366666</a>	13 октября 2020	13.0-CURRENT после исправления отображений только для записи на <code>arm64</code> .
1300121	<a href="#">366719</a>	15 октября 2020	13.0-CURRENT после добавления <code>VOP_EAGAIN</code> .
1300122	<a href="#">366782</a>	17 октября 2020	13.0-CURRENT после добавления <code>ptsname_r</code> .
1300123	<a href="#">366871</a>	20 октября 2020	13.0-CURRENT после изменений <code>VOP</code> , <code>VPTOCNP</code> и <code>INACTIVE</code> .
1300124	<a href="#">367162</a>	30 октября 2020	13.0-CURRENT после добавления <code>cache_vop_mkdir</code> и переименования <code>cache_rename</code> в <code>cache_vop_rename</code> .
1300125	<a href="#">367347</a>	4 ноября 2020	13.0-CURRENT после использования блокировки <code>gms</code> для обработки демонтажа в <code>zfs</code> .
1300126	<a href="#">367384</a>	5 ноября 2020	13.0-CURRENT после оптимизации зон на каждый процессор.

Значение	Версия	Дата	Релиз
1300127	<a href="#">367432</a>	6 ноября 2020	13.0-CURRENT после перемещения <code>malloc_type_internal</code> в <code>malloc_type</code> .
1300128	<a href="#">367522</a>	9 ноября 2020	13.0-CURRENT после добавлений LinuxKPI для реализации частей ACPI, необходимых <code>drm-kmod</code> в базовой системе.
1300129	<a href="#">367627</a>	12 ноября 2020	13.0-CURRENT после удаления <code>malloc_last_fail</code> .
1300130	<a href="#">367777</a>	17 ноября 2020	13.0-CURRENT после разделения <code>p_pd / pwddesc</code> и <code>p_fd / filedesc</code> .
1300131	<a href="#">368417</a>	7 декабря 2020	13.0-CURRENT после удаления криптографических файловых дескрипторов.
1300132	<a href="#">368659</a>	15 декабря 2020	13.0-CURRENT после улучшения обработки альтернативных настроек в стеке USB.
1300133	<a href="#">2ed0c8d801f5</a>	23 декабря 2020	13.0-CURRENT после изменения внутреннего API между модулями NFS и RPC ядра.
1300134	<a href="#">a84b0e94cdbf</a>	7 января 2021	13.0-CURRENT после выделения аппаратно-независимой части поддержки USB HID в новый модуль.
1300135	<a href="#">35a39dc5b349</a>	12 января 2021	13.0-CURRENT после добавления <code>kernel_fpu_begin/kernel_fpu_end</code> в LinuxKPI.

Значение	Версия	Дата	Релиз
1300136	<a href="#">72c551930be1</a>	17 января 2021	13.0-CURRENT после переработки очереди <code>irq_work</code> в LinuxKPI на основе быстрой <code>taskqueue</code> .
1300137	<a href="#">010196adcfaf</a>	30 января 2021	13.0-CURRENT после исправления утверждения clang при сборке порта <a href="#">devel/onetbb</a> .
1300138	<a href="#">dcee9964238b</a>	1 февраля 2021	13.0-ALPHA3 после добавления блокировки при поиске символьных ссылок в кэше VFS.
1300139	<a href="#">91a07ed50ffc</a>	2 февраля 2021	13.0-ALPHA3 после добавления различных компонентов LinuxKPI, конфликтующих с <code>drm-kmod</code> .
1300500	<a href="#">3c6a89748a01</a>	5 февраля 2021	13.0-STABLE после ветвления releng/13.0.
1300501	<a href="#">c3f97dd75a1c</a>	23 апреля 2021	13.0-STABLE после исправления <code>dl_iterate_phdr()</code> в <code>rtld</code> .
1300501	<a href="#">c3f97dd75a1c</a>	23 апреля 2021	13.0-STABLE после исправления <code>dl_iterate_phdr()</code> в <code>rtld</code> .
1300502	<a href="#">da6a8ccfa293</a>	23 апреля 2021	13.0-STABLE после реализации <code>atomic_dec_and_lock_irqsave()</code> в LinuxKPI.
1300503	<a href="#">d60c6dc8f69b</a>	23 апреля 2021	13.0-STABLE после изменения внутреннего KAPI между <code>krpc</code> и NFS.

Значение	Версия	Дата	Релиз
1300504	<a href="#">fb34817c686c</a>	30 апреля 2021	13.0-STABLE после обновления LinuxKPI для поддержки обновления drmkmod 5.5.
1300505	<a href="#">8f81f190a640</a>	10 мая 2021	13.0-STABLE после изменения внутреннего KAPI между модулями nscl.ko и nfscommon.ko.
1300506	<a href="#">e31579b8558d</a>	2 июня 2021	13.0-STABLE после добавления поддержки TCP LRO для VLAN и VxLAN.
1300507	<a href="#">c64d1bd7145b</a>	2 июня 2021	13.0-STABLE после добавления нового элемента в структуру отслеживания <a href="#">EPOCH(9)</a> .
1300508	<a href="#">658f5eed38c3</a>	11 июня 2021	13.0-STABLE после добавления макросов для <code>might_lock_nested()</code> и <code>lockdep_(re/un/)pin_lock()</code> в LinuxKPI.
1300509	<a href="#">210349325af9</a>	14 июня 2021	13.0-STABLE после добавления макроса <code>list_for_each_entry_lockless()</code> в LinuxKPI.
1300510	<a href="#">eb3397588e1b</a>	26 июня 2021	13.0-STABLE после изменения внутреннего KAPI между модулями kgrc и nfsd.

Значение	Версия	Дата	Релиз
1300511	<a href="#">2622570aeb3d</a>	7 июля 2021	13.0-STABLE после изменения <code>softdep_prelink()</code> для выполнения синхронизации только в случае, если другой поток изменил метаданные vnode с момента предыдущего <code>prelink</code> .
1300512	<a href="#">f72db34d2295</a>	18 июля 2021	13.0-STABLE после различных слияний LinuxKPI, OFED, net80211 и драйверов.
1300513	<a href="#">af732203b8f7</a>	31 июля 2021	13.0-STABLE после обновления llvm, clang, compiler-rt, libc++, libunwind, lld, lldb и openmp до версии llv Morg-12.0.1-0-gfed41342a82f, также известной как релиз 12.0.1.
1300514	<a href="#">53d162819c20</a>	3 августа 2021	Несовместимые изменения в KBI внутренних интерфейсов между NFS требуют пересборки модулей.
1300515	<a href="#">0437d10e359e</a>	22 сентября 2021	13.0-STABLE возвращается к KBI 13.0 для linuxkpi.
1300518	<a href="#">a017868e2818</a>	21 октября 2021	13.0-STABLE после добавления <code>crypto_cursor_segment()</code> .
1300519	<a href="#">fe2827f1678b</a>	21 октября 2021	13.0-STABLE после расширения шифров AES-CCM и ChaCha20-Poly1305 в OCF для поддержки нескольких длин одноразовых номеров.

Значение	Версия	Дата	Релиз
1300521	<a href="#">29745cf91cfc</a>	19 ноября 2021	13.0-STABLE после различных слияний с LinuxKPI и net80211.
1300522	<a href="#">0c8684ae2001</a>	24 ноября 2021	13.0-STABLE после изменения внутреннего KAPI между модулями NFS.
(не изменено)	<a href="#">7224d4125ab5</a>	6 декабря 2021	13.0-STABLE после обновления llvm, clang, compiler-rt, libc++, libunwind, lld, lldb и openmp до версии llvmtorg-13.0.0-0-gd7b669b3a303, также известной как релиз 13.0.0.
1300523	<a href="#">690bcf605d84</a>	18 декабря 2021	13.0-STABLE после добавления двух аргументов в <a href="#">VOP_ALLOCATE(9)</a> .
1300524	<a href="#">dc4114875ef1</a>	14 января 2022	13.0-STABLE после обеспечения совместимости макросов CPU_SET с glibc.
1300525	<a href="#">dee0854a009c</a>	22 января 2022	13.0-STABLE после множества изменений LinuxKPI, необходимых для drm-kmod.
1300526	<a href="#">c39ff2415cb9</a>	20 февраля 2022	13.0-STABLE после нескольких изменений LinuxKPI, пересекающихся, но не конфликтующих с drm-kmod.
1301000	<a href="#">ad329796bdb2</a>	10 марта 2022	Ветка releng/13.1 создана.
1301500	<a href="#">08523c8c63bb</a>	10 марта 2022	13.1-STABLE после ветвления releng/13.1.

Значение	Версия	Дата	Релиз
1301501	<a href="#">6663718bb496</a>	27 марта 2022	13.1-STABLE после различных слияний с LinuxKPI и net80211.
1301502	<a href="#">2278cf4e48e7</a>	27 апреля 2022	13.1-STABLE после различных слияний с LinuxKPI.
1301503	<a href="#">b2aa64d05bd8</a>	19 мая 2022	13.1-STABLE после добавления альтернативных макросов DRIVER_MODULE без аргумента devclass.
1301504	<a href="#">a13b6fc61908</a>	4 июня 2022	13.1-STABLE после обновления llvm, clang, compiler-rt, libc++, libunwind, lld, lldb и openmp до версии llvmorg-14.0.3-0-g1f9140064dfb, также известной как релиз 14.0.3.
1301505	<a href="#">6f93a76ffeab</a>	21 июня 2022	13.1-STABLE после различных слияний с LinuxKPI.
1301506	<a href="#">8e6cfc632cf6</a>	13 июля 2022	13.1-STABLE после добавления <crypto/chacha20_poly1305.h> и <crypto/curve25519.h>.
1301507	<a href="#">9cbba5950123</a>	21 июня 2022	13.1-STABLE после различных слияний с LinuxKPI.
1301508	<a href="#">83ac15a799e3</a>	17 октября 2022	13.1-STABLE после различных слияний в LinuxKPI и для удаления макросов из pause().
1301509	<a href="#">baa97013121a</a>	19 октября 2022	13.1-STABLE после введения версии 2 функциональности выбора очереди ТХ.

Значение	Версия	Дата	Релиз
1301510	<a href="#">6820a0512fa6</a>	8 декабря 2022	13.1-STABLE после исправлений LinuxKPI dmi_matches().
1301511	<a href="#">17333d92643d</a>	17 декабря 2022	13.1-STABLE после добавления нового гс: <code>machine_id</code> для генерации <code>/etc/machine-id</code> .
1302500	<a href="#">c243de11cf7c</a>	9 февраля 2023	13.2-STABLE после ветвления releng/13.2.
1302501	<a href="#">e3068d2655e2</a>	16 февраля 2023	13.2-STABLE после добавления <code>totalram_pages()</code> в LinuxKPI.
1302502	<a href="#">5ca371f4f536</a>	17 февраля 2023	13.2-STABLE после различных слияний с LinuxKPI.
1302503	<a href="#">aaca677fee21</a>	21 февраля 2023	13.2-STABLE после различных слияний с LinuxKPI.
1302504	<a href="#">d6852eed98ed</a>	12 марта 2023	13.2-STABLE после объединения machine-id в <code>hostid_save</code> .
1302505	<a href="#">85e32e957fcc</a>	9 апреля 2023	13.2-STABLE после обновления llvm, clang, compiler-rt, libc++, libunwind, lld, lldb и openmp до llvmorg-15.0.7-0-g8dfdcc7b7bf6, также известного как релиз 15.0.7.
1302506	<a href="#">e982b1cf1fe1</a>	26 июня 2023	13.2-STABLE после различных слияний с LinuxKPI.

Значение	Версия	Дата	Релиз
1302507	<a href="#">b2acc21dfbd6</a>	23 июля 2023	13.2-STABLE после обновления llvm, clang, compiler-rt, libc++, libunwind, lld, lldb и openmp до lvmorg-16.0.6-0-g7cbf1a259152, также известного как релиз 16.0.6.
1302508	<a href="#">21ccba43f511</a>	6 сентября 2023	13.2-STABLE после того, как ptrace начал очищать TDB_BORN во время PT_DETACH.
1302509	<a href="#">faedeaf7377b</a>	2 декабря 2023	13.2-STABLE после добавления новой функции VFS под названием <code>vfs_exjail_clone()</code> , которая будет использоваться модулем ZFS.
1302510	<a href="#">45758665781d</a>	7 января 2024	13.2-STABLE после обновления llvm, clang, compiler-rt, libc++, libunwind, lld, lldb и openmp до lvmorg-17.0.6-0-g6009708b4367, также известного как релиз 17.0.6.
1303001	<a href="#">a75a3d7afcc8</a>	19 февраля 2024	13.3-БЕТА3 после изменения внутренней структуры <code>struct ieee80211var</code> в net80211.
1303501	<a href="#">a7e1fc7f620d</a>	19 февраля 2024	13.3-STABLE после изменения внутренней структуры <code>struct ieee80211var</code> в net80211.

Значение	Версия	Дата	Релиз
1303502	<a href="#">07839ae99c06</a>	23 марта 2024	13.3-STABLE после исправления утверждения или падения clang при сборке последних библиотек boost.
1303503	<a href="#">055e875e6077</a>	20 апреля 2024	13.3-STABLE после обновления llvm, clang, compiler-rt, libc++, libunwind, lld, lldb и openmp до llvmmorg-18.1.3-0-gc13b7485b879, также известного как релиз 18.1.3.
1304500	<a href="#">77064cddb948</a>	1 августа 2024	13.4-STABLE после переименования из 13.4-PRERELEASE.
1304501	<a href="#">b802ab153dd2</a>	1 декабря 2024	13.4-STABLE после обновления llvm, clang, compiler-rt, libc++, libunwind, lld, lldb и openmp до llvmmorg-19.1.4-0-gaadaa00de76e, также известного как релиз 19.1.4.

## 18.4. Версии FreeBSD 12

Таблица 56. FreeBSD 12 Значения `__FreeBSD_version`

Значение	Версия	Дата	Релиз
1200000	<a href="#">302409</a>	7 июля 2016	12.0-CURRENT.
1200001	<a href="#">302628</a>	12 июля 2016	12.0-CURRENT после удаления правил сортировки из диапазонов типа [a-z].

Значение	Версия	Дата	Релиз
1200002	<a href="#">304395</a>	18 августа 2016	12.0-CURRENT после удаления неиспользуемого и устаревшего системного вызова <code>openbsd_poll</code> .
1200003	<a href="#">304608</a>	22 августа 2016	12.0-CURRENT после добавления поддержки <code>thread_local</code> из C++11 в ревизии <a href="#">303795</a> .
1200004	<a href="#">304752</a>	24 августа 2016	12.0-CURRENT после исправления LC*MASK для <code>newlocale(3)</code> и <code>querylocale(3)</code> (rev <a href="#">304703</a> ).
1200005	<a href="#">304789</a>	25 августа 2016	12.0-CURRENT после изменения некоторых интерфейсов <code>ioctl</code> в ревизии <a href="#">304787</a> между пользовательскими программами iSCSI и ядром.
1200006	<a href="#">305256</a>	1 сентября 2016	12.0-CURRENT после исправления <code>META_MODE</code> в <code>crunchgen(1)</code> в <a href="#">305254</a> .
1200007	<a href="#">305421</a>	5 сентября 2016	12.0-CURRENT после разрешения взаимоблокировки между <code>device_detach()</code> и <code>usbd_do_request_flags(9)</code> .
1200008	<a href="#">305833</a>	15 сентября 2016	12.0-CURRENT после удаления совместимого с 4.3BSD макроса <code>m_copy()</code> в <a href="#">305824</a> .

Значение	Версия	Дата	Релиз
1200009	<a href="#">306077</a>	21 сентября 2016	12.0-CURRENT после удаления <code>bio_taskqueue()</code> в <a href="#">305988</a> .
1200010	<a href="#">306276</a>	23 сентября 2016	12.0-CURRENT после монтирования <code>msdosfs(5)</code> с поддержкой <code>longnames</code> по умолчанию.
1200011	<a href="#">306556</a>	1 октября 2016	12.0-CURRENT после добавления поля <code>fb_memattr</code> в <code>fb_info</code> в <a href="#">306555</a> .
1200012	<a href="#">306592</a>	2 октября 2016	12.0-CURRENT после изменений в <code>net80211(4)</code> (rev <a href="#">306590</a> , <a href="#">306591</a> ).
1200013	<a href="#">307140</a>	12 октября 2016	12.0-CURRENT после установки заголовочных файлов, необходимых для разработки с <code>libzfs_core</code> .
1200014	<a href="#">307529</a>	17 октября 2016	12.0-CURRENT после объединения общего кода в <code>rtwn(4)</code> и <code>urtn(4)</code> , а также добавления поддержки устройств 802.11ac.
1200015	<a href="#">308874</a>	20 ноября 2016	12.0-CURRENT после некоторого изменения ABI для исправления powerpc.
1200016	<a href="#">309017</a>	22 ноября 2016	12.0-CURRENT после удаления полей, связанных с <code>PG_CACHED</code> , из <code>vmmeter</code> .

Значение	Версия	Дата	Релиз
1200017	<a href="#">309124</a>	25 ноября 2016	12.0-CURRENT после обновления копий clang, llvm, lldb, compiler-rt и libc++ до версии 3.9.0 и добавления lld 3.9.0.
1200018	<a href="#">309676</a>	7 декабря 2016	12.0-CURRENT после добавления члена <code>ki_moretdname</code> в структуры <code>struct kinfo_proc</code> и <code>struct kinfo_proc32</code> для экспорта полного имени потока в пользовательские утилиты.
1200019	<a href="#">310149</a>	16 декабря 2016	12.0-CURRENT после начала закладки основы для поддержки 11ас.
1200020	<a href="#">312087</a>	13 января 2017	12.0-CURRENT после удаления <code>fgetsock</code> и <code>fputsock</code> .
1200021	<a href="#">313858</a>	16 февраля 2017	12.0-CURRENT после удаления поддержки MCA и EISA.
1200022	<a href="#">314040</a>	21 февраля 2017	12.0-CURRENT после обеспечения сохранности структуры задач LinuxKPI между системными вызовами.
(не изменено)	<a href="#">314373</a>	2 марта 2017	12.0-CURRENT после удаления поддержки бинарной совместимости с System V Release 4.
1200023	<a href="#">314564</a>	2 марта 2017	12.0-CURRENT после обновления копий clang, llvm, lld, lldb, compiler-rt и libc++ до версии 4.0.0.

Значение	Версия	Дата	Релиз
1200024	<a href="#">314865</a>	7 марта 2017	12.0-CURRENT после удаления <code>rsarp-int.h</code>
1200025	<a href="#">315430</a>	16 марта 2017	12.0-CURRENT после добавления заголовочного файла <code>&lt;dev/mmc/mmc_ioctl.h&gt;</code> .
1200026	<a href="#">315662</a>	16 марта 2017	12.0-CURRENT после скрытия <code>struct inpcb</code> и <code>struct tcpcb</code> от пользовательского пространства.
1200027	<a href="#">315673</a>	21 марта 2017	12.0-CURRENT после того, как блокировка CAM SIM стала опциональной.
1200028	<a href="#">316683</a>	10 апреля 2017	12.0-CURRENT после переименования <code>smp_no_rendevous_barrier()</code> в <code>smp_no_rendezvous_barrier()</code> в <a href="#">316648</a> .
1200029	<a href="#">317176</a>	April 19, 2017	12.0-CURRENT после удаления <code>struct vmmeter</code> из <code>struct rcpu</code> в <a href="#">317061</a> .
1200030	<a href="#">317383</a>	24 апреля 2017	12.0-CURRENT после удаления поддержки NATM, включая <code>en(4)</code> , <code>fatm(4)</code> , <code>hatm(4)</code> и <code>patm(4)</code> .
1200031	<a href="#">318736</a>	23 мая 2017	12.0-CURRENT после расширения типов <code>ino_t</code> , <code>dev_t</code> , <code>nlink_t</code> до 64 бит и изменения структуры <code>struct dirent</code> (также известное как <code>ino64</code> ).
1200032	<a href="#">319664</a>	8 июня 2017	12.0-CURRENT после удаления <code>groff</code> .

Значение	Версия	Дата	Релиз
1200033	<a href="#">320043</a>	17 июня 2017	12.0-CURRENT после того, как тип члена <code>data</code> структуры <code>struct event</code> был увеличен до 64 бит, и добавлены члены расширенной структуры.
1200034	<a href="#">320085</a>	19 июня 2017	12.0-CURRENT после изменения клиента и сервера NFS для фактического использования 64-битного <code>ino_t</code> .
1200035	<a href="#">320317</a>	June 24, 2017	12.0-CURRENT после добавления флага <code>MAP_GUARD</code> в <code>mmap(2)</code> .
1200036	<a href="#">320347</a>	26 июня 2017	12.0-CURRENT после изменения <code>time_t</code> на 64 бита на powerpc (32-битная версия).
1200037	<a href="#">320545</a>	1 июля 2017	12.0-CURRENT после очистки и встраивания функций <code>bus_dma_map*</code> ( <a href="#">320528</a> ).
1200038	<a href="#">320879</a>	10 июля 2017	12.0-CURRENT после коммита MMC CAM ( <a href="#">320844</a> ).
1200039	<a href="#">321369</a>	22 июля 2017	12.0-CURRENT после обновления копий clang, llvm, lld, lldb, compiler-rt и libc++ до версии 5.0.0 (trunk r308421).
1200040	<a href="#">321688</a>	29 июля 2017	12.0-CURRENT после добавления поддержки принудительного демонтажа клиента NFS <code>umount -N</code> .

Значение	Версия	Дата	Релиз
1200041	<a href="#">322762</a>	21 августа 2017	12.0-CURRENT после того, как инструкция WRFSBASE стала работоспособной на amd64.
1200042	<a href="#">322900</a>	25 августа 2017	12.0-CURRENT после изменения счетчиков PLPMTUD для использования <a href="#">counter(9)</a> .
1200043	<a href="#">322989</a>	28 августа 2017	12.0-CURRENT после уменьшения CACHE_LINE_SIZE для x86 до 64 байт.
1200044	<a href="#">323349</a>	8 сентября 2017	12.0-CURRENT после реализации <a href="#">poll_wait()</a> в LinuxKPI.
1200045	<a href="#">323706</a>	18 сентября 2017	12.0-CURRENT после добавления поддержки разделяемой памяти в LinuxKPI. ( <a href="#">323703</a> ).
1200046	<a href="#">323910</a>	22 сентября 2017	12.0-CURRENT после добавления поддержки 32-битных совместимых IOCTL в LinuxKPI.
1200047	<a href="#">324053</a>	26 сентября 2017	12.0-CURRENT после удаления M_HASHTYPE_RSS_UDP_IPV4_EX. ( <a href="#">324052</a> ).
1200048	<a href="#">324227</a>	2 октября 2017	12.0-CURRENT после скрытия <a href="#">struct socket</a> и <a href="#">struct unpcb</a> из пользовательского пространства.
1200049	<a href="#">324281</a>	4 октября 2017	12.0-CURRENT после добавления поля <a href="#">value.u16</a> в структуру <a href="#">struct diocgattr_arg</a> .

Значение	Версия	Дата	Релиз
1200050	<a href="#">324342</a>	5 октября 2017	12.0-CURRENT после добавления <code>armv7 MACHINE_ARCH</code> . ( <a href="#">324340</a> ).
1200051	<a href="#">324455</a>	9 октября 2017	12.0-CURRENT после удаления <code>libstand.a</code> как публичного интерфейса. ( <a href="#">324454</a> ).
1200052	<a href="#">325028</a>	26 октября 2017	12.0-CURRENT после исправления <code>ptrace()</code> , чтобы всегда очищать правильное событие потока при возобновлении.
1200053	<a href="#">325506</a>	7 ноября 2017	12.0-CURRENT после изменения структуры <code>struct mbuf</code> для добавления опциональных аппаратных меток времени для принимаемых пакетов.
1200054	<a href="#">325852</a>	15 ноября 2017	12.0-CURRENT после изменения структуры <code>struct vmtotal</code> для поддержки отчёта больших счётчиков памяти.
1200055	<a href="#">327740</a>	9 января 2018	12.0-CURRENT после добавления поддержки <code>cpucontrol</code> -е.
1200056	<a href="#">327952</a>	14 января 2018	12.0-CURRENT после обновления clang, llvm, lld, lldb, compiler-rt и libc++ до версии 6.0.0 (ветки/ <code>release_60r321788</code> ).
1200057	<a href="#">329033</a>	8 февраля 2018	12.0-CURRENT после применения исправления в clang 6.0.0 для корректной сборки портов wine.

Значение	Версия	Дата	Релиз
1200058	<a href="#">329166</a>	12 февраля 2018	12.0-CURRENT после включения загрузчика Lua.
1200059	<a href="#">330299</a>	2 марта 2018	12.0-CURRENT после удаления объявления <code>union semun</code> , если не определено <code>_WANT_SEMUN</code> . Также удаление <code>struct mmsg</code> и переименование членов <code>struct semid_ds</code> и <code>struct msgid_ds</code> , предназначенных только для ядра.
1200060	<a href="#">330384</a>	4 марта 2018	12.0-CURRENT после обновления clang, llvm, lld, lldb, compiler-rt и libc++ до версии 6.0.0.
1200061	<a href="#">332100</a>	6 апреля 2018	12.0-CURRENT после изменения <code>syslog(3)</code> для генерации сообщений в формате RFC 5424.
1200062	<a href="#">332423</a>	12 апреля 2018	12.0-CURRENT после изменения API Netmap.
1200063	<a href="#">333446</a>	10 мая 2018	12.0-CURRENT после переработки параметров интерфейса и внутренней части CTL для использования <code>nv(3)</code> , разрешено создание нескольких портов ioctl интерфейса.
1200064	<a href="#">334074</a>	22 мая 2018	12.0-CURRENT после изменения <code>ifnet address</code> и <code>multicast address TAILQ</code> на <code>CK_STAILQ</code> .

Значение	Версия	Дата	Релиз
1200065	<a href="#">334290</a>	28 мая 2018	12.0-CURRENT после изменения <a href="#">dwatch(1)</a> для разрешения использования '-E code' для переопределения EVENT_DETAILS в профиле.
1200066	<a href="#">334466</a>	1 июня 2018	12.0-CURRENT после удаления внутриядерных таблиц rmc для Intel.
1200067	<a href="#">334892</a>	9 июня 2018	12.0-CURRENT после добавления констант DW_LANG в libdwarf.
1200068	<a href="#">334930</a>	12 июня 2018	12.0-CURRENT после изменения интерфейса между модулями NFS.
1200069	<a href="#">335237</a>	15 июня 2018	12.0-CURRENT после изменения <a href="#">struct kerneldumpheader</a> до версии 4 (аналогично версии 2 в 11-STABLE и более ранних).
1200070	<a href="#">335873</a>	2 июля 2018	12.0-CURRENT после встраивания <a href="#">atomic(9)</a> в модули на amd64 и i386, что потребовало пересборки всех модулей потребителей для этих архитектур.
1200071	<a href="#">335930</a>	4 июля 2018	12.0-CURRENT после изменения ABI и API <a href="#">epoch(9)</a> ( <a href="#">335924</a> ), что потребовало пересборки модулей потребителей.
1200072	<a href="#">335979</a>	5 июля 2018	12.0-CURRENT после изменения ABI и API <a href="#">struct xinpcb</a> и связанных структур.

Значение	Версия	Дата	Релиз
1200073	<a href="#">336313</a>	15 июля 2018	12.0-CURRENT после изменения ABI и API структур <code>struct if_shared_ctx</code> и <code>struct if_softc_ctx</code> , что потребовало пересборки модулей потребителей <code>iflib(9)</code> .
1200074	<a href="#">336360</a>	16 июля 2018	12.0-CURRENT после обновления конфигурации <code>libstdc++</code> для использования функций C99.
1200075	<a href="#">336538</a>	19 июля 2018	12.0-CURRENT после объединения <code>zfsloader</code> в <code>loader</code> , после добавления <code>ntpd:ntpd</code> как <code>uid:gid 123:123</code> и после удаления поддержки <code>big-endian</code> для архитектуры <code>arm</code> ( <code>MACHINE_ARCH=arm eb</code> ).
1200076	<a href="#">336914</a>	30 июля 2018	12.0-CURRENT после изменений KPI в <code>timespecadd</code> .
1200077	<a href="#">337576</a>	10 августа 2018	12.0-CURRENT после добавления в систему <code>timespec_get(3)</code> .
1200078	<a href="#">337863</a>	15 августа 2018	12.0-CURRENT после выполнения хука <code>exec.created</code> для клеток.
1200079	<a href="#">338061</a>	19 августа 2018	12.0-CURRENT после перевода <code>arc4random</code> на использование алгоритма ChaCha20 и устаревания <code>arc4random_stir</code> и <code>arc4random_addrandom</code> .

Значение	Версия	Дата	Релиз
1200080	<a href="#">338172</a>	22 августа 2018	12.0-CURRENT после удаления драйверов <code>drm</code> .
1200081	<a href="#">338182</a>	21 августа 2018	12.0-CURRENT после изменений KPI для NVMe.
1200082	<a href="#">338285</a>	24 августа 2018	12.0-CURRENT после отмены удаления драйверов <code>drm</code> .
1200083	<a href="#">338331</a>	26 августа 2018	12.0-CURRENT после удаления <code>arc4random_stir</code> и <code>arc4random_addrandom</code> .
1200084	<a href="#">338478</a>	5 сентября 2018	12.0-CURRENT после обновления <code>objcopy(1)</code> для корректной обработки little-endian MIPS64 объектных файлов.
1200085	<a href="#">339270</a>	19 октября 2018	12.0-STABLE после обновления OpenSSL до версии 1.1.1.
1200086	<a href="#">339732</a>	25 октября 2018	12.0-STABLE после обновления номеров версий разделяемых библиотек OpenSSL.
1200500	<a href="#">340471</a>	16 ноября 2018	12-STABLE после ветвления <code>releng/12.0</code> .
1200501	<a href="#">342801</a>	6 января 2019	12-STABLE после слияния исправления поведения <code>linux_destroy_dev()</code> , когда остаются открытые файлы из уничтожаемого <code>cdev</code> .
1200502	<a href="#">343126</a>	January 17, 2019	12-STABLE после включения <code>#include &lt;sys/random.h&gt;</code> в C++.
1200503	<a href="#">344152</a>	15 февраля 2019	12-STABLE после слияния исправления <code>renameat(2)</code> для ядер с CAPABILITIES.

Значение	Версия	Дата	Релиз
1200504	<a href="#">345169</a>	15 марта 2019	12-STABLE после слияния ССМ для работы с портом ZoF.
1200505	<a href="#">345327</a>	20 марта 2019	12-STABLE после объединения поддержки выборочного отключения ZFS без отключения загрузчика.
1200506	<a href="#">346168</a>	12 апреля 2019	12-STABLE после слияния llvm, clang, compiler-rt, libc++, libunwind, lld, lldb и openmp 8.0.0 финальный релиз r356365.
1200507	<a href="#">346337</a>	17 апреля 2019	12-STABLE после слияния изменений iflib из ревизий <a href="#">345303</a> , <a href="#">345658</a> , и частично из <a href="#">345305</a> .
1200508	<a href="#">346784</a>	27 апреля 2019	12-STABLE после появления <code>ether_gen_addr</code> .
1200509	<a href="#">347790</a>	16 мая 2019	12-STABLE после обновления номеров версий драйверов Mellanox ( <a href="#">mlx4en(4)</a> ; <a href="#">mlx5en(4)</a> ).
1200510	<a href="#">348036</a>	21 мая 2019	12-STABLE после изменения структуры в linuxkpi из <a href="#">348035</a> .
1200511	<a href="#">348243</a>	24 мая 2019	12-STABLE после MFC изменения <a href="#">347843</a> : добавление элемента <code>group_leader</code> в структуру <code>task_struct</code> для LinuxKPI.
1200512	<a href="#">348245</a>	24 мая 2019	12-STABLE после добавления элемента контекста к <code>ww_mutex</code> в LinuxKPI.

Значение	Версия	Дата	Релиз
1200513	<a href="#">349763</a>	5 июля 2019	12-STABLE после MFC <a href="#">epoch(9)</a> изменил: <a href="#">349763</a> , <a href="#">340404</a> , <a href="#">340415</a> , <a href="#">340417</a> , <a href="#">340419</a> , <a href="#">340420</a> .
1200514	<a href="#">350083</a>	17 июля 2019	12-STABLE после добавлений в список <code>gcu</code> LinuxKPI.
1200515	<a href="#">350877</a>	11 августа 2019	12-STABLE после слияния изменений <a href="#">349891</a> (реорганизация списков SRCS в виде одного файла на строку с последующей сортировкой по алфавиту) и <a href="#">349972</a> (добавление обёрток системных вызовов <code>arm_sync_icache()</code> и <code>arm_drain_writebuf()</code> для <code>sysarch</code> ).
1200516	<a href="#">351276</a>	20 августа 2019	12-STABLE после слияния различных изменений в <code>iflib</code> (MFC) <a href="#">351276</a> .
1200517	<a href="#">352076</a>	9 сентября 2019	12-STABLE после добавления функций создания/удаления <code>sysfs</code> , обрабатывающих несколько файлов за один вызов в LinuxKPI.
1200518	<a href="#">352114</a>	10 сентября 2019	12-STABLE после дополнительных обновлений LinuxKPI в <code>sysfs</code> .
1200519	<a href="#">352351</a>	15 сентября 2019	12-STABLE после переноса (MFC) нового драйвера <code>fusefs</code> .

Значение	Версия	Дата	Релиз
1201000	<a href="#">352546</a>	20 сентября 2019	releng/12.1 ответвился от stable/12@r352480.
1201500	<a href="#">352547</a>	20 сентября 2019	12-STABLE после ветвления releng/12.1.
1201501	<a href="#">354598</a>	10 ноября 2019	12-STABLE после исправления потенциальной проблемы безопасности ООВ-чтения в libc++.
1201502	<a href="#">354613</a>	11 ноября 2019	12-STABLE после включения атрибутов группы классов устройств в LinuxKPI.
1201503	<a href="#">354928</a>	21 ноября 2019	12-STABLE после добавления поддержки <code>AT_EXCPATH</code> в <code>elf_aux_info(3)</code> .
1201504	<a href="#">355658</a>	10 ноября 2019	12-STABLE после исправления проверки версии C++ для объявления <code>timespec_get(3)</code> .
1201505	<a href="#">355899</a>	19 декабря 2019	12-STABLE после добавления расширений <code>sigsetop</code> , которые часто встречаются в musl libc и glibc.
1201506	<a href="#">355968</a>	21 декабря 2019	12-STABLE после удвоения значения <code>ARG_MAX</code> для 64-битных платформ.
1201507	<a href="#">356306</a>	2 января 2020	12-STABLE после добавления функций в <code>bitstring(3)</code> для поиска непрерывных последовательностей установленных или сброшенных битов.

Значение	Версия	Дата	Релиз
1201508	<a href="#">356394</a>	6 января 2020	12-STABLE после изменения статистики USB для каждого устройства вместо каждой шины.
1201509	<a href="#">356460</a>	7 января 2020	12-STABLE после обновления llvm, clang, compiler-rt, libc++, libunwind, lld, lldb и openmp до финального релиза 9.0.0 r372316.
1201510	<a href="#">356679</a>	13 января 2020	12-STABLE после добавления собственного счётчика для отменённых USB-передач.
1201511	<a href="#">357333</a>	31 января 2020	12-STABLE после добавления символической ссылки /etc/os-release на /var/run/os-release.
1201512	<a href="#">357612</a>	6 февраля 2020	12-STABLE после недавних изменений в LinuxKPI.
1201513	<a href="#">359957</a>	15 апреля 2020	12-STABLE после разделения интерфейса RCU на допускающий и не допускающий сон части в LinuxKPI.
1201514	<a href="#">360525</a>	1 мая 2020	12-STABLE после реализации полной поддержки <a href="#">bus_dma(9)</a> в LinuxKPI и включения всех зависимостей.

Значение	Версия	Дата	Релиз
1201515	<a href="#">360545</a>	1 мая 2020	12-STABLE после обновления llvm, clang, compiler-rt, libc++, libunwind, lld, lldb и openmp до версии 10.0.0.
1201516	<a href="#">360620</a>	4 мая 2020	12-STABLE после перемещения <code>id_mapped</code> в конец структуры <code>bus_dma_impl</code> для сохранения KPI.
1201517	<a href="#">361350</a>	21 мая 2020	12-STABLE после переименования <code>vm.max_wired</code> в <code>vm.max_user_wired</code> и изменения его типа.
1201518	<a href="#">362319</a>	18 июня 2020	12-STABLE после реализации макроса функции <code>__is_constexpr()</code> в LinuxKPI.
1201519	<a href="#">362916</a>	4 июля 2020	12-STABLE после перевода liblzma на использование реализации SHA256 из libmd.
1201520	<a href="#">363494</a>	24 июля 2020	12-STABLE после обновления llvm, clang, compiler-rt, libc++, libunwind, lld, lldb и openmp до версии 10.0.1.
1201521	<a href="#">363790</a>	3 августа 2020	12-STABLE после реализации функции <code>array_size()</code> в LinuxKPI.
1201522	<a href="#">363832</a>	4 августа 2020	12-STABLE после добавления системного вызова <code>sysctlbyname</code> .

Значение	Версия	Дата	Релиз
1201523	<a href="#">364390</a>	19 августа 2020	12-STABLE после изменения для клонирования полей структуры задачи, связанных с RCU.
1201524	<a href="#">365356</a>	5 сентября 2020	12-STABLE после выделения XDR в отдельный модуль ядра для минимизации зависимостей ZFS.
1201525	<a href="#">365471</a>	8 сентября 2020	12-STABLE после добавления атомарных функций и функций <code>bswap</code> в <code>libcompiler_rt</code> .
1201526	<a href="#">365608</a>	10 сентября 2020	12-STABLE после обновления net80211 и изменений API проверки привилегий ядра.
1202000	<a href="#">365618</a>	11 сентября 2020	releng/12.2 ответвился от stable/12@r365618.
1202500	<a href="#">365619</a>	11 сентября 2020	12-STABLE после ветвления releng/12.2.
1202501	<a href="#">365661</a>	12 сентября 2020	12-STABLE после последующих коммитов в <code>libcompiler_rt</code> .
1202502	<a href="#">365816</a>	16 сентября 2020	12-STABLE после исправления сборки приложения <code>crunchgen(1)</code> с <code>WARNS=6</code> .
1202503	<a href="#">366878</a>	20 октября 2020	12-STABLE после заполнения поля контекста захвата <code>ww_mutex</code> в LinuxKPI.
1202504	<a href="#">367511</a>	9 ноября 2020	12-STABLE после добавления <code>ptsname_r(3)</code> .

Значение	Версия	Дата	Релиз
1202505	<a href="#">f3d75bed5475</a>	28 декабря 2020	12-STABLE после улучшения обработки альтернативных настроек в стеке USB.
1202506	<a href="#">d36cc12ddfe3</a>	30 апреля 2021	12-STABLE после изменения внутреннего KAPI между krcs и NFS.
1202507	<a href="#">1e279fe9deae</a>	10 мая 2021	12-STABLE после изменения внутреннего KAPI между модулями nscl.ko и nfscommon.ko.
1202508	<a href="#">489236b04748</a>	26 июня 2021	12-STABLE после изменения внутреннего KAPI между модулями krcs и nfsd.
1203500	<a href="#">f2900e784cb0</a>	20 октября 2021	12-STABLE после ветвления releng/12.3.
1203501	<a href="#">b148c7b87148</a>	22 декабря 2021	12-STABLE после добавления атомарных функций и функций <code>bswap</code> в <code>libcompiler_rt</code> .
1203502	<a href="#">4772e4135cb3</a>	22 декабря 2021	12-STABLE после обновления llvm, clang, compiler-rt, libc++, libunwind, lld, lldb и openmp до версии 11.0.1.
1203503	<a href="#">e405b2dc913c</a>	25 декабря 2021	12-STABLE после обновления llvm, clang, compiler-rt, libc++, libunwind, lld, lldb и openmp до версии 12.0.0.

Значение	Версия	Дата	Релиз
1203504	<a href="#">1a398266112e</a>	25 декабря 2021	12-STABLE после добавления вспомогательных функций LSE атомарных операций вне строки в libcompiler_rt.a для архитектуры aarch64.
1203505	<a href="#">0b7be89b329e</a>	25 декабря 2021	12-STABLE после обновления llvm, clang, compiler-rt, libc++, libunwind, lld, lldb и openmp до версии 13.0.0.
1203506	<a href="#">f591279d9c93</a>	12 февраля 2022	12-STABLE после восстановления компромисса доступности random(4).
1203507	<a href="#">180d95e04e93</a>	9 апреля 2022	12-STABLE после объединения zlib.
1203508	<a href="#">6c717a28505d</a>	19 октября 2022	12-STABLE после iflib: Разрешить драйверам определять, в какую очередь передавать данные.
1204000	<a href="#">fce871fe3520</a>	20 октября 2022	releng/12.4 — ветка от stable/12.
1204500	<a href="#">6a9031c5e2ba</a>	20 октября 2022	12-STABLE после ветвления releng/12.4.

## 18.5. FreeBSD 11 Версии

Таблица 57. FreeBSD 11 Значения `__FreeBSD_version`

Значение	Версия	Дата	Релиз
1100000	<a href="#">256284</a>	10 октября 2013	11.0-CURRENT.

Значение	Версия	Дата	Релиз
1100001	<a href="#">256776</a>	19 октября 2013	11.0-CURRENT после добавления поддержки сценариев <code>гс.d</code> для "первой загрузки", что позволяет портам использовать эту возможность.
1100002	<a href="#">257696</a>	5 ноября 2013	11.0-CURRENT после прекращения поддержки устаревших <code>ioctl</code> .
1100003	<a href="#">258284</a>	17 ноября 2013	11.0-CURRENT после изменений в <code>iconv</code> .
1100004	<a href="#">259424</a>	15 декабря 2013	11.0-CURRENT после изменения поведения <code>gss_pseudo_random</code> , внесённого в <a href="#">259286</a> .
1100005	<a href="#">260010</a>	28 декабря 2013	11.0-CURRENT после <a href="#">259951</a> - Не объединять записи в <code>vm_map_stack(9)</code> .
1100006	<a href="#">261246</a>	28 января 2014	11.0-CURRENT после обновления <code>libelf</code> и <code>libdwarf</code> .
1100007	<a href="#">261283</a>	30 января 2014	11.0-CURRENT после обновления <code>libc++</code> до версии 3.4.
1100008	<a href="#">261881</a>	February 14, 2014	11.0-CURRENT после исправления совместимости ABI в <code>libc++</code> 3.4.
1100009	<a href="#">261991</a>	16 февраля 2014	11.0-CURRENT после обновления <code>llvm/clang</code> до версии 3.4.
1100010	<a href="#">262630</a>	28 февраля 2014	11.0-CURRENT после обновления <code>ncurses</code> до версии 5.9 (ревизия <a href="#">262629</a> ).
1100011	<a href="#">263102</a>	13 марта 2014	11.0-CURRENT после изменения ABI в структуре <code>if_data</code> .

Значение	Версия	Дата	Релиз
1100012	<a href="#">263140</a>	14 марта 2014	11.0-CURRENT после удаления поддержки протокола Novell IPX.
1100013	<a href="#">263152</a>	14 марта 2014	11.0-CURRENT после удаления поддержки протокола AppleTalk.
1100014	<a href="#">263235</a>	March 16, 2014	11.0-CURRENT после переименования <code>&lt;sys/capability.h&gt;</code> в <code>&lt;sys/capsicum.h&gt;</code> во избежание конфликта с одноименными заголовочными файлами в других операционных системах. Совместимый заголовочный файл оставлен для уменьшения количества проблем при сборке, но в будущем будет устаревшим.
1100015	<a href="#">263620</a>	22 марта 2014	11.0-CURRENT после переименования <code>cnt</code> в <code>vm_cnt</code> .
1100016	<a href="#">263660</a>	23 марта 2014	11.0-CURRENT после добавления <code>armv6hf</code> <code>TARGET_ARCH</code> .
1100017	<a href="#">264121</a>	4 апреля 2014	11.0-CURRENT после удаления поддержки GCC для определения <code>__block</code> .
1100018	<a href="#">264212</a>	6 апреля 2014	11.0-CURRENT после добавления поддержки протокола UDP-Lite (RFC 3828).
1100019	<a href="#">264289</a>	8 апреля 2014	11.0-CURRENT после FreeBSD-SA-14:06.openssl (ревизия <a href="#">264265</a> ).

Значение	Версия	Дата	Релиз
1100020	<a href="#">265215</a>	1 мая 2014	11.0-CURRENT после удаления <code>lindv</code> в пользу наличия <code>/dev/full</code> по умолчанию (rev <a href="#">265212</a> ).
1100021	<a href="#">266151</a>	6 мая 2014	11.0-CURRENT после изменений в <code>src.opts.mk</code> , отделяющих <code>make.conf(5)</code> от <code>buildworld</code> (rev <a href="#">265419</a> ).
1100022	<a href="#">266904</a>	30 мая 2014	11.0-CURRENT после изменений в <code>strcasecmp(3)</code> , перемещении <code>strcasecmp_l(3)</code> и <code>strncasecmp_l(3)</code> из <code>&lt;string.h&gt;</code> в <code>&lt;strings.h&gt;</code> для соответствия POSIX 2008 (rev <a href="#">266865</a> ).
1100023	<a href="#">267440</a>	13 июня 2014	11.0-CURRENT после подключения библиотеки CUSE и модуля ядра к сборке по умолчанию.
1100024	<a href="#">267992</a>	27 июня 2014	11.0-CURRENT после изменения API <code>sysctl(3)</code> .
1100025	<a href="#">268066</a>	30 июня 2014	11.0-CURRENT после обновления библиотеки <code>regex(3)</code> для добавления разделителей <code>&gt;</code> и <code>&lt;</code> .
1100026	<a href="#">268118</a>	1 июля 2014	11.0-CURRENT после изменения внутреннего интерфейса между модулями NFS, включая <code>krpc</code> , в (rev <a href="#">268115</a> ).

Значение	Версия	Дата	Релиз
1100027	<a href="#">268441</a>	8 июля 2014	11.0-CURRENT после FreeBSD-SA-14:17.kmem (ссылка на ревизию: <a href="https://svnweb.freebsd.org/changeset/base/268431">https://svnweb.freebsd.org/changeset/base/268431</a> [268431]).
1100028	<a href="#">268945</a>	21 июля 2014	11.0-CURRENT после исправления соответствия <a href="#">hdestroy(3)</a> изменился ABI.
1100029	<a href="#">270173</a>	3 августа 2014	11.0-CURRENT после исправления ошибки <a href="#">SOCK_DGRAM</a> (rev <a href="#">269489</a> ).
1100030	<a href="#">270929</a>	1 сентября 2014	11.0-CURRENT после того, как сокеты <a href="#">SOCK_RAW</a> были изменены так, чтобы вообще не модифицировать пакеты.
1100031	<a href="#">271341</a>	9 сентября 2014	11.0-CURRENT после FreeBSD-SA-14:18.openssl (ссылка на ревизию: <a href="https://svnweb.freebsd.org/changeset/base/269686">https://svnweb.freebsd.org/changeset/base/269686</a> [269686]).
1100032	<a href="#">271438</a>	11 сентября 2014	11.0-CURRENT после изменений API в <a href="#">ifa_ifwithbroadaddr</a> , <a href="#">ifa_ifwithdstaddr</a> , <a href="#">ifa_ifwithnet</a> и <a href="#">ifa_ifwithroute</a> .
1100033	<a href="#">271657</a>	9 сентября 2014	11.0-CURRENT после изменения <a href="#">access</a> , <a href="#">eaccess</a> и <a href="#">faccessat</a> для проверки аргумента mode.
1100034	<a href="#">271686</a>	16 сентября 2014	11.0-CURRENT после FreeBSD-SA-14:19.tcp (rev <a href="#">271666</a> ).

Значение	Версия	Дата	Релиз
1100035	<a href="#">271705</a>	17 сентября 2014	11.0-CURRENT после добавления поддержки аппаратного контекста i915.
1100036	<a href="#">271724</a>	17 сентября 2014	Увеличение версии для различия в ABI-примечании бинарных файлов, готовых к строгой проверке флагов <a href="#">mmap(2)</a> (изменение <a href="#">271724</a> ).
1100037	<a href="#">272674</a>	6 октября 2014	11.0-CURRENT после добавления <a href="#">explicit_bzero(3)</a> (изменение: <a href="https://svnweb.freebsd.org/changeset/base/272673">https://svnweb.freebsd.org/changeset/base/272673</a> [ <a href="#">272673</a> ]).
1100038	<a href="#">272951</a>	October 11, 2014	11.0-CURRENT после очистки заголовков TCP wrapper.
1100039	<a href="#">273250</a>	18 октября 2014	11.0-CURRENT после удаления <a href="#">MAP_RENAME</a> и <a href="#">MAP_NORESERVE</a> .
1100040	<a href="#">273432</a>	21 октября 2014	11.0-CURRENT после FreeBSD-SA-14:23 (ссылка на ревизию: <a href="https://svnweb.freebsd.org/changeset/base/273146">https://svnweb.freebsd.org/changeset/base/273146</a> [ <a href="#">273146</a> ]).
1100041	<a href="#">273875</a>	30 октября 2014	11.0-CURRENT после изменений API в <a href="#">syscall_register</a> , <a href="#">syscall32_register</a> , <a href="#">syscall_register_helper</a> и <a href="#">syscall32_register_helper</a> (rev <a href="#">273707</a> ).
1100042	<a href="#">274046</a>	3 ноября 2014	11.0-CURRENT после изменения <a href="#">struct tcb</a> .

Значение	Версия	Дата	Релиз
1100043	<a href="#">274085</a>	4 ноября 2014	11.0-CURRENT после включения <a href="#">vt(4)</a> по умолчанию.
1100044	<a href="#">274116</a>	4 ноября 2014	11.0-CURRENT после добавления новых библиотек/утилит ( <a href="#">dpr</a> и <a href="#">figpar</a> ) для визуализации пропускной способности данных.
1100045	<a href="#">274162</a>	4 ноября 2014	11.0-CURRENT после FreeBSD-SA-14:23, FreeBSD-SA-14:24 и FreeBSD-SA-14:25.
1100046	<a href="#">274470</a>	13 ноября 2014	11.0-CURRENT после изменения сигнатуры <a href="#">kern_poll</a> (rev <a href="#">274462</a> ).
1100047	<a href="#">274476</a>	13 ноября 2014	11.0-CURRENT после удаления no-at версий вспомогательных системных вызовов VFS, таких как <a href="#">kern_open</a> .
1100048	<a href="#">275358</a>	1 декабря 2014	11.0-CURRENT после начала процесса удаления использования устаревшего флага "M_FLOWID" из сетевого кода.
1100049	<a href="#">275633</a>	9 декабря 2014	11.0-CURRENT после импорта важного исправления в векторизатор LLVM, которое в некоторых случаях могло приводить к переполнению буфера.
1100050	<a href="#">275732</a>	12 декабря 2014	11.0-CURRENT после добавления AES-ICM и AES-GCM в OpenCrypto.

Значение	Версия	Дата	Релиз
1100051	<a href="#">276096</a>	December 23, 2014	11.0-CURRENT после удаления старого кода клиента и сервера NFS из ядра.
1100052	<a href="#">276479</a>	31 декабря 2014	11.0-CURRENT после обновления clang, lvm и lldb до версии 3.5.0.
1100053	<a href="#">276781</a>	7 января 2015	11.0-CURRENT после того, как <a href="#">MCLGET(9)</a> получил возвращаемое значение (rev <a href="#">276750</a> ).
1100054	<a href="#">277213</a>	15 января 2015	11.0-CURRENT после переработки подсистемы вызовов.
1100055	<a href="#">277528</a>	22 января 2015	11.0-CURRENT после отмены изменений callout в <a href="#">277213</a> .
1100056	<a href="#">277610</a>	23 января 2015	11.0-CURRENT после добавления системных вызовов <a href="#">futimens</a> и <a href="#">utimensat</a> .
1100057	<a href="#">277897</a>	29 января 2015	11.0-CURRENT после удаления <a href="#">d_thread_t</a> .
1100058	<a href="#">278228</a>	5 февраля 2015	11.0-CURRENT после добавления поддержки запроса страницы расширенного запроса SCSI VPD (0x86).
1100059	<a href="#">278442</a>	9 февраля 2015	11.0-CURRENT после импорта xz 5.2.0, который добавил многопоточное сжатие, и lzma получила зависимость от libthr (rev <a href="#">278433</a> ).

Значение	Версия	Дата	Релиз
1100060	<a href="#">278846</a>	16 февраля 2015	11.0-CURRENT после пересылки <code>FBIO_BLANK</code> клиентам фреймбуфера.
1100061	<a href="#">278964</a>	18 февраля 2015	11.0-CURRENT после добавления <code>CDAI_FLAG_NONE</code> .
1100062	<a href="#">279221</a>	23 февраля 2015	11.0-CURRENT после добавлений API <code>mtio(4)</code> и <code>sa(4)</code> , а также <code>ioctl(2)</code> .
1100063	<a href="#">279728</a>	7 марта 2015	11.0-CURRENT после добавления поддержки мьютексов в API <code>pps_ioctl()</code> в ядре.
1100064	<a href="#">279729</a>	7 марта 2015	11.0-CURRENT после добавления поддержки PPS в драйверы USB-последовательных портов.
1100065	<a href="#">280031</a>	15 марта 2015	11.0-CURRENT после обновления clang, llvm и lldb до версии 3.6.0.
1100066	<a href="#">280306</a>	20 марта 2015	11.0-CURRENT после удаления поддержки SSLv2 из OpenSSL.
1100067	<a href="#">280630</a>	25 марта 2015	11.0-CURRENT после удаления поддержки SSLv2 из <code>fetch(1)</code> и <code>fetch(3)</code> .
1100068	<a href="#">281172</a>	6 апреля 2015	11.0-CURRENT после изменения системной настройки <code>net.inet6.ip6.mif6table</code> .
1100069	<a href="#">281550</a>	15 апреля 2015	11.0-CURRENT после удаления квалификатора <code>const</code> из <code>iconv(3)</code> .

Значение	Версия	Дата	Релиз
1100070	<a href="#">281613</a>	16 апреля 2015	11.0-CURRENT после перемещения ALTQ из contrib в net/altq.
1100071	<a href="#">282256</a>	29 апреля 2015	11.0-CURRENT после изменения API/ABI в <a href="#">smb(4)</a> (rev <a href="#">281985</a> ).
1100072	<a href="#">282319</a>	1 мая 2015	11.0-CURRENT после добавления <a href="#">reallocarray(3)</a> в libc (изменение <a href="#">282314</a> ).
1100073	<a href="#">282650</a>	8 мая 2015	11.0-CURRENT после увеличения максимального количества разрешённых РСМ-каналов в РСМ-потоке до 127 и уменьшения максимального количества подканалов до 1.
1100074	<a href="#">283526</a>	25 мая 2015	11.0-CURRENT после добавления предварительной поддержки бинарных файлов Linux для x86-64 (rev <a href="#">283424</a> ) и обновления clang и LLVM до версии 3.6.1.
1100075	<a href="#">283623</a>	27 мая 2015	11.0-CURRENT после <a href="#">dounmount()</a> , требующей ссылки на переданную структуру mount (изменение rev <a href="#">283602</a> ).
1100076	<a href="#">283983</a>	4 июня 2015	11.0-CURRENT после отключения генерации записей в устаревших форматах баз данных паролей по умолчанию.

Значение	Версия	Дата	Релиз
1100077	<a href="#">284233</a>	10 июня 2015	11.0-CURRENT после изменений API в <a href="#">lim_cur</a> , <a href="#">lim_max</a> и <a href="#">lim_rlimit</a> (rev <a href="#">284215</a> ).
1100078	<a href="#">286672</a>	12 августа 2015	11.0-CURRENT после изменений <a href="#">crunchgen(1)</a> в ревизиях <a href="#">284356</a> до <a href="#">285986</a> .
1100079	<a href="#">286874</a>	18 августа 2015	11.0-CURRENT после импорта jemalloc 4.0.0 (ревизия <a href="#">286866</a> ).
1100080	<a href="#">288943</a>	5 октября 2015	11.0-CURRENT после обновления clang, llvm, lldb, compiler-rt и libc++ до версии 3.7.0.
1100081	<a href="#">289415</a>	16 октября 2015	11.0-CURRENT после обновления ZFS для поддержки возобновляемой отправки/приёмки (rev <a href="#">289362</a> ).
1100082	<a href="#">289594</a>	19 октября 2015	11.0-CURRENT после обновлений Linux KPI.
1100083	<a href="#">289749</a>	October 22, 2015	11.0-CURRENT после переименования linuxapi.ko в linuxkpi.ko.
1100084	<a href="#">290135</a>	29 октября 2015	11.0-CURRENT после перемещения модуля LinuxKPI в стандартную сборку ядра.
1100085	<a href="#">290207</a>	30 октября 2015	11.0-CURRENT после импорта OpenSSL 1.0.2d.
1100086	<a href="#">290275</a>	2 ноября 2015	11.0-CURRENT после изменения макросов <a href="#">figpar(3)</a> для большей уникальности.

Значение	Версия	Дата	Релиз
1100087	<a href="#">290479</a>	7 ноября 2015	11.0-CURRENT после изменения ABI <a href="#">sysctl_add_oid(9)</a> .
1100088	<a href="#">290495</a>	7 ноября 2015	11.0-CURRENT после переработки сортировки строк и локалей.
1100089	<a href="#">290505</a>	7 ноября 2015	11.0-CURRENT после изменения API в <a href="#">sysctl_add_oid(9)</a> (rev <a href="#">290475</a> ).
1100090	<a href="#">290715</a>	10 ноября 2015	11.0-CURRENT после изменения API для макроса <code>callout_stop</code> ; (ссылка на ревизию: <a href="https://svnweb.freebsd.org/changeset/base/290664">https://svnweb.freebsd.org/changeset/base/290664</a> [290664]).
1100091	<a href="#">291537</a>	30 ноября 2015	11.0-CURRENT после изменения интерфейса между модулями <code>nfsd.ko</code> и <code>nfsccommon.ko</code> в <a href="#">291527</a> .
1100092	<a href="#">292499</a>	19 декабря 2015	11.0-CURRENT после удаления <code>vm_pageout_grow_cache</code> (изменение: <a href="https://svnweb.freebsd.org/changeset/base/292469">https://svnweb.freebsd.org/changeset/base/292469</a> [292469]).
1100093	<a href="#">292966</a>	30 декабря 2015	11.0-CURRENT после удаления <code>sys/crypto/sha2.h</code> (изменение <a href="#">292782</a> ).
1100094	<a href="#">294086</a>	15 января 2016	11.0-CURRENT после изменений LinuxKPI PCI (rev <a href="#">294086</a> ).
1100095	<a href="#">294327</a>	19 января 2016	11.0-CURRENT после оптимизаций LRO.

Значение	Версия	Дата	Релиз
1100096	<a href="#">294505</a>	21 января 2016	11.0-CURRENT после добавления LinuxKPI <code>idr_*</code> .
1100097	<a href="#">294860</a>	26 января 2016	11.0-CURRENT после изменения API в <a href="#">dprv(3)</a> .
1100098	<a href="#">295682</a>	16 февраля 2016	11.0-CURRENT после изменения API в <code>gmap</code> (rev <a href="#">294883</a> ).
1100099	<a href="#">295739</a>	18 февраля 2016	11.0-CURRENT после разрешения драйверам устанавливать лимит агрегации сегментов ТСП АСК/данных.
1100100	<a href="#">296136</a>	26 февраля 2016	11.0-CURRENT после добавления API <a href="#">bus_alloc_resource_any(9)</a> .
1100101	<a href="#">296417</a>	5 марта 2016	11.0-CURRENT после обновления копий clang, llvm, lldb и compiler-rt до релиза 3.8.0.
1100102	<a href="#">296749</a>	12 марта 2016	11.0-CURRENT после исправления кросс-эндианности <code>libelf</code> в ревизии <a href="#">296685</a> .
1100103	<a href="#">297000</a>	18 марта 2016	11.0-CURRENT после использования <code>uintmax_t</code> для диапазонов <code>gmap</code> .
1100104	<a href="#">297156</a>	21 марта 2016	11.0-CURRENT после отслеживания использования <code>filemon</code> через указатель <code>proc.p_filemon</code> вместо собственных списков.

Значение	Версия	Дата	Релиз
1100105	<a href="#">297602</a>	6 апреля 2016	11.0-CURRENT после исправления функций <code>i</code> и <code>a</code> в <code>sed</code> , которые отбрасывали начальные пробелы.
1100106	<a href="#">298486</a>	22 апреля 2016	11.0-CURRENT после исправлений для использования IPv6-адресов с RDMA.
1100107	<a href="#">299090</a>	4 мая 2016	11.0-CURRENT после улучшения производительности и функциональности API <code>bitstring(3)</code> .
1100108	<a href="#">299530</a>	12 мая 2016	11.0-CURRENT после исправления обработки IOCTL в LinuxKPI.
1100109	<a href="#">299933</a>	16 мая 2016	11.0-CURRENT после реализации дополнительных функций, связанных с устройствами Linux, в LinuxKPI.
1100110	<a href="#">300207</a>	19 мая 2016	11.0-CURRENT после добавления поддержки управления дисками с черепичной магнитной записью (Shingled Magnetic Recording, SMR).
1100111	<a href="#">300303</a>	20 мая 2016	11.0-CURRENT после удаления <code>brk</code> и <code>sbrk</code> из arm64.
1100112	<a href="#">300539</a>	23 мая 2016	11.0-CURRENT после добавления <code>bit_count</code> в <code>bitstring(3)</code> API.
1100113	<a href="#">300701</a>	26 мая 2016	11.0-CURRENT после отключения ошибок выравнивания на armv6.

Значение	Версия	Дата	Релиз
1100114	<a href="#">300806</a>	26 мая 2016	11.0-CURRENT после исправления использования <a href="#">crunchgen(1)</a> с <a href="#">MAKEOBJDIRPREFIX</a> .
1100115	<a href="#">300982</a>	30 мая 2016	11.0-CURRENT после добавления флага <code>mbuf</code> для <a href="#">M_HASHTYPE_</a> .
1100116	<a href="#">301011</a>	31 мая 2016	11.0-CURRENT после добавления SHA-512t256 (ревизия <a href="#">300903</a> ) и Skein (ревизия <a href="#">300966</a> ) в <code>libmd</code> , <code>libsrupt</code> , ядро и ZFS (ревизия <a href="#">301010</a> ).
1100117	<a href="#">301892</a>	6 июня 2016	11.0-CURRENT после синхронизации <code>libram</code> с основной версией <a href="#">301602</a> , что привело к увеличению версии библиотеки.
1100118	<a href="#">302071</a>	21 июня 2016	11.0-CURRENT после нарушения бинарной совместимости структуры <code>disk</code> <a href="#">302069</a> .
1100119	<a href="#">302150</a>	23 июня 2016	11.0-CURRENT после перевода <code>geom_disk</code> на использование мьютекса пула.
1100120	<a href="#">302153</a>	23 июня 2016	11.0-CURRENT после добавления запасных элементов в <code>struct ifnet</code> .
1100121	<a href="#">303979</a>	12 августа 2015	11-STABLE после того, как ветка <a href="#">releng/11.0</a> отделилась от 11-STABLE (изменение: <a href="https://svnweb.freebsd.org/change-set/base/303975[303975]">https://svnweb.freebsd.org/change-set/base/303975[303975]</a> ).

Значение	Версия	Дата	Релиз
1100500	<a href="#">303979</a>	12 августа 2016	11.0-STABLE добавлена ветвленная <a href="#">link: 303976</a> .
1100501	<a href="#">304609</a>	22 августа 2016	11.0-STABLE после добавления поддержки <code>thread_local</code> в C++11.
1100502	<a href="#">304865</a>	26 августа 2016	11.0-STABLE после исправления <code>LC_*_MASK</code> .
1100503	<a href="#">305733</a>	12 сентября 2016	11.0-STABLE после устранения взаимоблокировки между <code>device_detach()</code> и <code>usbd_do_request_flags(9)</code> .
1100504	<a href="#">307330</a>	14 октября 2016	11.0-STABLE после объединения ZFS.
1100505	<a href="#">307590</a>	19 октября 2016	11.0-STABLE после изменения <code>struct fb_info</code> .
1100506	<a href="#">308048</a>	28 октября 2016	11.0-STABLE после установки заголовочных файлов, необходимых для разработки с <code>libzfs_core</code> .
1100507	<a href="#">310120</a>	15 декабря 2016	11.0-STABLE после добавления члена <code>ki_moretdname</code> в структуры <code>struct kinfo_proc</code> и <code>struct kinfo_proc32</code> для экспорта полного имени потока в пользовательские утилиты.

Значение	Версия	Дата	Релиз
1100508	<a href="#">310618</a>	26 декабря 2016	11.0-STABLE после обновления копий clang, lvm, lldb, compiler-rt и libc++ до версии 3.9.1, а также добавления lld 3.9.1.
1100509	<a href="#">311186</a>	3 января 2017	11.0-STABLE после исправления META_MODE в <a href="#">crunchgen(1)</a> (изменение <a href="#">311185</a> ).
1100510	<a href="#">315312</a>	15 марта 2017	11.0-STABLE после MFC изменений, связанных с <a href="#">fget_cap</a> , <a href="#">getsock_cap</a> и другими.
1100511	<a href="#">316423</a>	2 апреля 2017	11.0-STABLE после нескольких MFC, обновляющих clang, lvm, lld, lldb, compiler-rt и libc++ до версии 4.0.0.
1100512	<a href="#">316498</a>	4 апреля 2017	11.0-STABLE после того, как блокировка CAM SIM стала опциональной (реvisions <a href="#">315673</a> , <a href="#">315674</a> ).
1100513	<a href="#">318197</a>	11 мая 2017	11.0-STABLE после объединения добавления заголовочного файла <code>&lt;dev/mmc/mmc_ioctl.h&gt;</code> .
1100514	<a href="#">319279</a>	31 мая 2017	11.0-STABLE после нескольких MFC для <a href="#">librcap</a> , <a href="#">WITHOUT_INET6</a> и нескольких других незначительных изменений.
1101000	<a href="#">320486</a>	June 30, 2017	<a href="#">releng/11.1</a> отделился от <a href="#">stable/11</a> .

Значение	Версия	Дата	Релиз
1101001	<a href="#">320763</a>	June 30, 2017	11.1-RC1 После объединения добавления флага <code>MAP_GUARD</code> в <code>mmap(2)</code> .
1101500	<a href="#">320487</a>	June 30, 2017	11-STABLE после ветвления <code>releng/11.1</code> .
1101501	<a href="#">320666</a>	5 июля 2017	11-STABLE после объединения добавления флага <code>MAP_GUARD</code> в <code>mmap(2)</code> .
1101502	<a href="#">321688</a>	29 июля 2017	11-STABLE после объединения поддержки принудительного демонтажирования клиента NFS с добавлением <code>umount -N</code> .
1101503	<a href="#">323431</a>	11 сентября 2017	11-STABLE после объединения изменений, сделавших инструкцию <code>WRFSBASE</code> работоспособной на <code>amd64</code> .
1101504	<a href="#">324006</a>	26 сентября 2017	11-STABLE после слияния <code>libm</code> из <code>head</code> , что добавляет <code>cacoshl(3)</code> , <code>cacosl(3)</code> , <code>casinhl(3)</code> , <code>casinl(3)</code> , <code>catanl(3)</code> , <code>catanhl(3)</code> , <code>sincos(3)</code> , <code>sincosf(3)</code> и <code>sincosl(3)</code> .
1101505	<a href="#">324023</a>	26 сентября 2017	11-STABLE после объединения <code>clang</code> , <code>llvm</code> , <code>lld</code> , <code>lldb</code> , <code>compiler-rt</code> и <code>libc++</code> версии 5.0.0.

Значение	Версия	Дата	Релиз
1101506	<a href="#">325003</a>	25 октября 2017	11-STABLE после слияния <a href="#">324281</a> , добавления поля <code>value.u16</code> в структуру <code>struct diocgattr_arg</code> .
1101507	<a href="#">328379</a>	24 января 2018	11-STABLE после слияния с <a href="#">325028</a> , исправление <code>ptrace()</code> для корректной очистки события нужного потока при возобновлении.
1101508	<a href="#">328386</a>	24 января 2018	11-STABLE после объединения изменений <a href="#">316648</a> , переименование <code>smp_no_rendevous_barrier()</code> в <code>smp_no_rendezvous_barrier()</code> .
1101509	<a href="#">328653</a>	1 февраля 2018	11-STABLE после обратного переноса (overwrite merge) LinuxKPI из FreeBSD-head.
1101510	<a href="#">329450</a>	17 февраля 2018	11-STABLE после того, как макрос <code>smpxchg()</code> стал полностью функциональным в LinuxKPI.
1101511	<a href="#">329981</a>	25 февраля 2018	11-STABLE после завершения недавних обновлений, связанных с LinuxKPI.
1101512	<a href="#">331219</a>	19 марта 2018	11-STABLE после объединения поддержки <code>retpoline</code> из вышестоящих веток llvm, clang и lld 5.0.

Значение	Версия	Дата	Релиз
1101513	<a href="#">331838</a>	31 марта 2018	11-STABLE после объединения clang, llvm, lld, lldb, compiler-rt и libc++ версии 6.0.0, а также нескольких последующих исправлений.
1101514	<a href="#">332089</a>	5 апреля 2018	11-STABLE после объединения изменений <a href="#">328331</a> , добавляющего новую и несовместимую интерпретацию <code>\${name}_limits</code> в rc-скриптах.
1101515	<a href="#">332363</a>	10 апреля 2018	11-STABLE после отмены изменений из <a href="#">331880</a> , удаляющих новую и несовместимую интерпретацию <code>\${name}_limits</code> в rc-скриптах.
1101516	<a href="#">334392</a>	30 мая 2018	11-STABLE после доработок <code>dwatch(1)</code> .
1102000	<a href="#">334459</a>	1 июня 2018	<code>releng/11.2</code> отделился от <code>stable/11</code> .
1102500	<a href="#">334461</a>	1 июня 2018	11-STABLE после ветвления <code>releng/11.2</code> .
1102501	<a href="#">335436</a>	June 20, 2018	11-STABLE после обновлений LinuxKPI, требующих перекомпиляции внешних модулей ядра.

Значение	Версия	Дата	Релиз
1102502	<a href="#">338617</a>	12 сентября 2018	11-STABLE после добавления сокет-опции SO_TS_CLOCK и исправления системного вызова <code>recvmsg32()</code> для корректного преобразования структуры 64-битных данных в формат, ожидаемый 32-битными приложениями.
1102503	<a href="#">338931</a>	25 сентября 2018	11-STABLE после объединения исправления контрольной суммы TCP в <code>iflib(9)</code> и добавления новых типов носителей в <code>if_media.h</code>
1102504	<a href="#">340309</a>	9 ноября 2018	11-STABLE после нескольких MFC: обновление <code>objcopy(1)</code> для корректной обработки little-endian MIPS64 объектов; исправление теста <code>mips64el</code> для использования заголовка ELF; добавление теста для 64-битного ELF в <code>_libelf_is_mips64el</code> .
1102505	<a href="#">342804</a>	6 января 2019	11-STABLE после слияния исправления поведения <code>linux_destroy_dev()</code> , когда остаются открытые файлы из уничтожаемого <code>cdev</code> .

Значение	Версия	Дата	Релиз
1102506	<a href="#">344220</a>	17 февраля 2019	11-STABLE после объединения нескольких коммитов в lualoader.
1102507	<a href="#">346296</a>	16 апреля 2019	11-STABLE после объединения llvm, clang, compiler-rt, libc++, libunwind, lld, lldb и openmp 8.0.0 финальный релиз r356365.
1102508	<a href="#">346784</a>	27 апреля 2019	11-STABLE после появления <a href="#">ether_gen_addr</a> .
1102509	<a href="#">347212</a>	6 мая 2019	11-STABLE после слияния изменений <a href="#">345303</a> , <a href="#">345658</a> , и частично <a href="#">345305</a> .
1102510	<a href="#">347883</a>	16 мая 2019	11-STABLE после увеличения номеров версий драйверов Mellanox ( <a href="#">mlx4en(4)</a> ; <a href="#">mlx5en(4)</a> ).
1103000	<a href="#">349026</a>	14 июня 2019	<a href="#">releng/11.3</a> отделился от <a href="#">stable/11</a> .
1103500	<a href="#">349027</a>	14 июня 2019	11-STABLE после ветвления <a href="#">releng/11.3</a> .
1103501	<a href="#">354598</a>	10 ноября 2019	11-STABLE после исправления потенциальной проблемы безопасности ООВ-чтения в libc++.
1103502	<a href="#">354614</a>	11 ноября 2019	11-STABLE после добавления функций создания/удаления sysfs, обрабатывающих несколько файлов за один вызов в LinuxKPI.

Значение	Версия	Дата	Релиз
1103503	<a href="#">354615</a>	11 ноября 2019	11-STABLE после улучшений LinuxKPI sysfs.
1103504	<a href="#">354616</a>	11 ноября 2019	11-STABLE после включения атрибутов группы классов устройств в LinuxKPI.
1103505	<a href="#">355899</a>	19 декабря 2019	11-STABLE после добавления расширений <code>sigsetop</code> , которые часто встречаются в musl libc и glibc.
1103506	<a href="#">356395</a>	6 января 2020	11-STABLE после изменения статистики USB для каждого устройства вместо каждой шины.
1103507	<a href="#">356680</a>	13 января 2020	11-STABLE после добавления собственного счетчика для отмененных USB-передач.
1103508	<a href="#">357613</a>	6 февраля 2020	11-STABLE после недавних изменений в LinuxKPI.
1103509	<a href="#">359958</a>	15 апреля 2020	11-STABLE после перемещения <code>id_mapped</code> в конец структуры <code>bus_dma_impl</code> для сохранения KPI.
1103510	<a href="#">360658</a>	5 мая 2020	11-STABLE после обновления llvm, clang, compiler-rt, libc++, libunwind, lld, lldb и openmp до финального релиза 9.0.0 r372316.

Значение	Версия	Дата	Релиз
1103511	<a href="#">360784</a>	7 мая 2020	11-STABLE после обновления llvm, clang, compiler-rt, libc++, libunwind, lld, lldb и openmp до версии 10.0.0.
1104000	<a href="#">360804</a>	8 мая 2020	<a href="#">releng/11.4</a> ответвился от <a href="#">stable/11</a> .
1104001	<a href="#">360822</a>	8 мая 2020	11.4-BETA1 после обновления llvm, clang, compiler-rt, libc++, libunwind, lld, lldb и openmp до версии 10.0.0.
1104500	<a href="#">360805</a>	8 мая 2020	11-STABLE после ветвления releng/11.4.
1104501	<a href="#">362320</a>	18 июня 2020	11-STABLE после реализации макроса функции <a href="#">__is_constexpr()</a> в LinuxKPI.
1104502	<a href="#">362919</a>	4 июля 2020	11-STABLE после перевода liblzma на использование реализации SHA256 из libmd.
1104503	<a href="#">363496</a>	24 июля 2020	11-STABLE после обновления llvm, clang, compiler-rt, libc++, libunwind, lld, lldb и openmp до версии 10.0.1.
1104504	<a href="#">363792</a>	3 августа 2020	11-STABLE после реализации функции <a href="#">array_size()</a> в LinuxKPI.
1104505	<a href="#">364391</a>	19 августа 2020	11-STABLE после изменения для клонирования полей структуры задачи, связанных с RCU.

Значение	Версия	Дата	Релиз
1104506	<a href="#">365471</a>	8 сентября 2020	11-STABLE после добавления атомарных функций и функций <code>bswap</code> в <code>libcompiler_rt</code> .
1104507	<a href="#">365661</a>	12 сентября 2020	11-STABLE после последующих коммитов в <code>libcompiler_rt</code> .
1104508	<a href="#">366879</a>	20 октября 2020	11-STABLE после заполнения поля контекста получения в <code>ww_mutex</code> в LinuxKPI.
1104509	<a href="#">366889</a>	20 октября 2020	11-STABLE после добавлений в список RCU LinuxKPI.
1104510	<a href="#">367513</a>	9 ноября 2020	11-STABLE после добавления <code>ptsname_r</code> .

## 18.6. Версии FreeBSD 10

Таблица 58. Значения `__FreeBSD_version` в FreeBSD 10

Значение	Версия	Дата	Релиз
1000000	<a href="#">225757</a>	26 сентября 2011	10.0-CURRENT.
1000001	<a href="#">227070</a>	4 ноября 2011	10-CURRENT после добавления системного вызова <code>posix_fadvise(2)</code> .
1000002	<a href="#">228444</a>	12 декабря 2011	10-CURRENT после определения булевых значений true/false в <code>sys/types.h</code> , размер <code>sizeof(bool)</code> мог измениться (ревизия <a href="#">228444</a> ). 10-CURRENT после введения <code>xlocale.h</code> (ревизия <a href="#">227753</a> ).

Значение	Версия	Дата	Релиз
1000003	<a href="#">228571</a>	16 декабря 2011	10-CURRENT после значительных изменений в <a href="#">carp(4)</a> , изменения размера структур <a href="#">in_aliasreq</a> , <a href="#">in6_aliasreq</a> (rev <a href="#">228571</a> ) и упрощения проверки аргументов <a href="#">SIOCAIFADDR</a> (rev <a href="#">228574</a> ).
1000004	<a href="#">229204</a>	1 января 2012	10-CURRENT после удаления <a href="#">skpc()</a> и добавления <a href="#">memchr(9)</a> (изменение: <a href="https://svnweb.freebsd.org/change-set/base/229200[229200]">https://svnweb.freebsd.org/change-set/base/229200[229200]</a> ).
1000005	<a href="#">230207</a>	16 января 2012	10-CURRENT после удаления поддержки ioctl's <a href="#">SIOCSIFADDR</a> , <a href="#">SIOCSIFNETMASK</a> , <a href="#">SIOCSIFBRDADDR</a> , <a href="#">SIOCSIFDSTADDR</a> .
1000006	<a href="#">230590</a>	26 января 2012	10-CURRENT после внедрения асинхронного уведомления о данных пропускной способности чтения в слое <a href="#">cam(4)</a> .
1000007	<a href="#">231025</a>	5 февраля 2012	10-CURRENT после введения новых параметров сокета <a href="#">tcp(4)</a> : <a href="#">TCP_KEEPIKIT</a> , <a href="#">TCP_KEEPIKIDLE</a> , <a href="#">TCP_KEEPIKINTVL</a> и <a href="#">TCP_KEEPIKICNT</a> .

Значение	Версия	Дата	Релиз
1000008	<a href="#">231505</a>	11 февраля 2012	10-CURRENT после введения нового расширяемого интерфейса <a href="#">sysctl(3)</a> NET_RT_IFLISTL для запроса списков адресов.
1000009	<a href="#">232154</a>	25 февраля 2012	10-CURRENT после импорта libarchive 3.0.3 (rev <a href="#">232153</a> ).
1000010	<a href="#">233757</a>	31 марта 2012	10-CURRENT после очистки <a href="#">xlocale</a> .
1000011	<a href="#">234355</a>	16 апреля 2012	Импорт LLVM/Clang 3.1 из 10-CURRENT, ссылка на ревизию: <a href="#">154661</a> (ревизия <a href="#">234353</a> ).
1000012	<a href="#">234924</a>	2 мая 2012	10-CURRENT импорт jemalloc.
1000013	<a href="#">235788</a>	22 мая 2012	10-CURRENT после импорта <a href="#">byacc</a> .
1000014	<a href="#">237631</a>	27 июня 2012	10-CURRENT после того, как BSD sort стал сортировкой по умолчанию (rev <a href="#">237629</a> ).
1000015	<a href="#">238405</a>	12 июля 2012	10-CURRENT после импорта OpenSSL 1.0.1c.
(не изменено)	<a href="#">238429</a>	July 13, 2012	10-CURRENT после исправления регрессии в LLVM/Clang 3.1.
1000016	<a href="#">239179</a>	8 августа 2012	10-CURRENT после изменения KBI в <a href="#">ucom(4)</a> .
1000017	<a href="#">239214</a>	8 августа 2012	10-CURRENT после добавления функции потоков в стек USB.
1000018	<a href="#">240233</a>	8 сентября 2012	10-CURRENT после значительной переработки <a href="#">pf(4)</a> .

Значение	Версия	Дата	Релиз
1000019	<a href="#">241245</a>	6 октября 2012	10-CURRENT после изменения KBI/KPI в <a href="#">pfil(9)</a> для передачи пакетов в порядке байтов сети к хукам фильтрации AF_INET.
1000020	<a href="#">241610</a>	16 октября 2012	10-CURRENT после изменения KPI клонирования сетевых интерфейсов и структура <a href="#">if_clone</a> стала непрозрачной.
1000021	<a href="#">241897</a>	22 октября 2012	10-CURRENT после удаления поддержки не-MPSAFE файловых систем и добавления поддержки FUSEFS (rev <a href="#">241519</a> ).
1000022	<a href="#">241913</a>	22 октября 2012	10-CURRENT после перевода всего стека IPv4 на сетевой порядок байтов для хранения заголовков IP-пакетов.
1000023	<a href="#">242619</a>	5 ноября 2012	10-CURRENT после буфера джиттера в общем коде драйвера USB-последовательного порта, для временного хранения символов, если буфер TTY заполнен. Добавлены сигналы остановки и возобновления потока при возникновении такой ситуации.
1000024	<a href="#">242624</a>	5 ноября 2012	10-CURRENT после того, как clang стал компилятором по умолчанию для i386 и amd64.

Значение	Версия	Дата	Релиз
1000025	<a href="#">243443</a>	17 ноября 2012	10-CURRENT после того, как переменная-член <code>sin6_score_id</code> в структуре <code>sockaddr_in6</code> была изменена таким образом, что ядро заполняет её перед передачей структуры в пользовательское пространство через <code>sysctl</code> или сокет маршрутизации. Это означает, что специфичный для KAME встроенный идентификатор области в <code>sin6_addr.s6_addr[2]</code> всегда очищается в пользовательских приложениях.
1000026	<a href="#">245313</a>	11 января 2013	10-CURRENT после установки получил флаг <code>-N</code> . Также может использоваться для указания наличия <code>nmtree</code> .
1000027	<a href="#">246084</a>	29 января 2013	10-CURRENT после того, как команда <code>cat</code> получила флаг <code>-l</code> (rev <a href="#">246083</a> ).
1000028	<a href="#">246759</a>	13 февраля 2013	10-CURRENT после перемещения USB в структуру драйверов, требующую пересборки всех модулей USB.

Значение	Версия	Дата	Релиз
1000029	<a href="#">247821</a>	4 марта 2013	10-CURRENT после внедрения бестиковой системы отложенных вызовов, которая также изменила структуру struct callout (rev <a href="#">247777</a> ).
1000030	<a href="#">248210</a>	12 марта 2013	10-CURRENT после нарушения KPI, внесённого в подсистему VM для поддержки блокировок чтения/записи (rev <a href="#">248084</a> ).
1000031	<a href="#">249943</a>	26 апреля 2013	10-CURRENT после изменения параметра <code>dst</code> метода <code>if_output</code> в <code>ifnet</code> , чтобы он принимал квалификатор <code>const</code> (ревизия <a href="#">249925</a> ).
1000032	<a href="#">250163</a>	1 мая 2013	10-CURRENT после введения системных вызовов <code>accept4(2)</code> (rev <a href="#">250154</a> ) и <code>pipe2(2)</code> (rev <a href="#">250159</a> ).
1000033	<a href="#">250881</a>	21 мая 2013	10-CURRENT после импорта flex 2.5.37.
1000034	<a href="#">251294</a>	3 июня 2013	10-CURRENT после добавления следующих функций в <code>libm</code> : <code>cacos(3)</code> , <code>cacosf(3)</code> , <code>cacosh(3)</code> , <code>cacoshf(3)</code> , <code>casin(3)</code> , <code>casinf(3)</code> , <code>casinh(3)</code> , <code>casinhf(3)</code> , <code>catan(3)</code> , <code>catanf(3)</code> , <code>catanh(3)</code> , <code>catanhf(3)</code> , <code>logl(3)</code> , <code>log2l(3)</code> , <code>log10l(3)</code> , <code>log1pl(3)</code> , <code>expm1(3)</code> .

Значение	Версия	Дата	Релиз
1000035	<a href="#">251527</a>	8 июня 2013	10-CURRENT после введения системного вызова <code>aio_mlock(2)</code> (изменение <a href="#">251526</a> ).
1000036	<a href="#">253049</a>	9 июля 2013	10-CURRENT после добавления новой функции в интерфейс вызовов функций модуля ядра GSSAPI.
1000037	<a href="#">253089</a>	9 июля 2013	10-CURRENT после миграции структур статистики на RCPU-счетчики. Измененные структуры включают: <code>ahstat</code> , <code>arpstat</code> , <code>espstat</code> , <code>icmp6_ifstat</code> , <code>icmp6stat</code> , <code>in6_ifstat</code> , <code>ip6stat</code> , <code>ipcompstat</code> , <code>ipipstat</code> , <code>ipsecstat</code> , <code>mrt6stat</code> , <code>mrtstat</code> , <code>pfkeystat</code> , <code>rim6stat</code> , <code>rimstat</code> , <code>rip6stat</code> , <code>udpstat</code> (rev <a href="#">253081</a> ).
1000038	<a href="#">253396</a>	16 июля 2013	10-CURRENT после установки <code>ARM EABI</code> в качестве ABI по умолчанию для архитектур <code>arm</code> , <code>armeb</code> , <code>armv6</code> и <code>armv6eb</code> .
1000039	<a href="#">253549</a>	22 июля 2013	10-CURRENT после изменений в сканировании драйверов <code>CAM</code> и <code>mps(4)</code> .
1000040	<a href="#">253638</a>	24 июля 2013	10-CURRENT после добавления файлов <code>pkgconf</code> для <code>libusb</code> .
1000041	<a href="#">253970</a>	5 августа 2013	10-CURRENT после изменения с <code>time_second</code> на <code>time_uptime</code> в <code>PF_INET6</code> .

Значение	Версия	Дата	Релиз
1000042	<a href="#">254138</a>	9 августа 2013	10-CURRENT после изменения подсистемы VM для объединения механизмов мягкой и жесткой занятости.
1000043	<a href="#">254273</a>	13 августа 2013	10-CURRENT после того, как <code>WITH_ICONV</code> включён по умолчанию. Новая опция <code>src.conf(5)</code> , <code>WITH_LIBICONV_COMPAT</code> (выключена по умолчанию), добавляет <code>libiconv_open</code> для обеспечения совместимости с портом <code>converters/libiconv</code> .
1000044	<a href="#">254358</a>	15 августа 2013	10-CURRENT после преобразования <code>libc.so</code> в скрипт <code>ld(1)</code> (изменение <code>rev 251668</code> ).
1000045	<a href="#">254389</a>	15 августа 2013	10-CURRENT после изменения программного интерфейса <code>devfs</code> путем замены флага <code>D_UNMAPPED_IO</code> в <code>cdevsw</code> на флаг <code>SI_UNMAPPED</code> в структуре <code>cdev</code> .
1000046	<a href="#">254537</a>	19 августа 2013	10-CURRENT после добавления <code>M_PROTO[9-12]</code> и удаления <code>M_FRAG</code>
<code>M_FIRSTFRAG</code>   <code>M_LASTFRAG</code> ` флаги <code>mbuf</code> (версии <a href="#">254524</a> , <a href="#">254526</a> ).	1000047	<a href="#">254627</a>	21 августа 2013

Значение	Версия	Дата	Релиз
10-CURRENT после обновления <code>stat(2)</code> , позволяющего сохранять некоторые атрибуты файлов Windows/DOS и CIFS в виде флагов <code>stat(2)</code> .	1000048	<a href="#">254672</a>	August 22, 2013
10-CURRENT после изменения структуры <code>xsctp_inpcb</code> .	1000049	<a href="#">254760</a>	24 августа 2013
10-CURRENT после поддержки <code>physio(9)</code> для устройств, которые работают некорректно с разделённым вводом-выводом, таких как <code>sa(4)</code> .	1000050	<a href="#">254844</a>	24 августа 2013
10-CURRENT после изменений структуры <code>mbuf</code> (rev <a href="#">254780</a> , <a href="#">254799</a> , <a href="#">254804</a> , <a href="#">254807</a> <a href="#">254842</a> ).	1000051	<a href="#">254887</a>	25 августа 2013
10-CURRENT после импорта драйвера Radeon KMS (ревизия <a href="#">254885</a> ).	1000052	<a href="#">255180</a>	3 сентября 2013
10-CURRENT после импорта NetBSD <code>libexecinfo</code> подключен к сборке.	1000053	<a href="#">255305</a>	6 сентября 2013
10-CURRENT после изменений API и ABI в рамках Capsicum.	1000054	<a href="#">255321</a>	6 сентября 2013
10-CURRENT после того, как <code>gcc</code> и <code>libstdc++</code> больше не собираются по умолчанию.	1000055	<a href="#">255449</a>	6 сентября 2013
10-CURRENT после добавления флага <code>MMAP_32BIT</code> в <code>mmap(2)</code> (rev <a href="#">255426</a> ).	1000100	<a href="#">259065</a>	December 7, 2013

Значение	Версия	Дата	Релиз
<a href="#">releng/10.0</a> ответвился от <a href="#">stable/10.</a>	1000500	<a href="#">256283</a>	10 октября 2013
10-STABLE после ветвления от <a href="#">head/</a> .	1000501	<a href="#">256916</a>	22 октября 2013
10-STABLE после добавления поддержки <a href="#">rc(8)</a> при первой загрузке.	1000502	<a href="#">258398</a>	20 ноября 2013
10-STABLE после удаления символов <a href="#">iconv</a> из <a href="#">libc.so.7</a> .	1000510	<a href="#">259067</a>	December 7, 2013
<a href="#">releng/10.0</a> <a href="#">__FreeBSD_version</a> обновлён, чтобы предотвратить уменьшение значения.	1000700	<a href="#">259069</a>	December 7, 2013
10-STABLE после ветки <a href="#">releng/10.0</a> .	1000701	<a href="#">259447</a>	15 декабря 2013
10.0-STABLE после исправления кодирования Heimdal.	1000702	<a href="#">260135</a>	31 декабря 2013
10-STABLE после исправлений <a href="#">MAP_STACK</a> .	1000703	<a href="#">262801</a>	5 марта 2014
10-STABLE после обновления <a href="#">libc++</a> до версии 3.4.	1000704	<a href="#">262889</a>	7 марта 2014
10-STABLE после слияния из ветки <a href="#">vt(4)</a> драйвера (ревизия <a href="#">262861</a> ).	1000705	<a href="#">263508</a>	21 марта 2014
10-STABLE после обновления <a href="#">llvm/clang</a> до версии 3.4.	1000706	<a href="#">264214</a>	6 апреля 2014
10-STABLE после удаления поддержки GCC для определения <a href="#">__block</a> .	1000707	<a href="#">264289</a>	8 апреля 2014

Значение	Версия	Дата	Релиз
10-STABLE после FreeBSD-SA- 14:06.openssl.	1000708	<a href="#">265122</a>	30 апреля 2014
10-STABLE после FreeBSD-SA- 14:07.devfs, FreeBSD- SA-14:08.tcp и FreeBSD- SA-14:09.openssl.	1000709	<a href="#">265946</a>	13 мая 2014
10-STABLE после добавления поддержки протокола UDP-Lite (RFC 3828).	1000710	<a href="#">267465</a>	13 июня 2014
10-STABLE после изменений в <a href="#">strcasecmp(3)</a> , переноса <a href="#">strcasecmp_l(3)</a> и <a href="#">strncasecmp_l(3)</a> из <string.h> в <strings.h> для соответствия POSIX 2008.	1000711	<a href="#">268442</a>	8 июля 2014
10-STABLE после FreeBSD-SA- 14:17.kmem (ревизия: <a href="https://svnweb.freebsd.org/changeset/base/268432[268432]">https://svnwe b.freebsd.org/changeset /base/268432[268432]</a> ).	1000712	<a href="#">269400</a>	1 августа 2014
10-STABLE после слияния <a href="#">nfsd(8)</a> 4.1 (rev <a href="#">269398</a> ).	1000713	<a href="#">269484</a>	3 августа 2014
10-STABLE после обновления библиотеки <a href="#">regex(3)</a> для добавления разделителей ">" и "<".	1000714	<a href="#">270174</a>	3 августа 2014
10-STABLE после исправления ошибки <a href="#">SOCK_DGRAM</a> (rev <a href="#">269490</a> ).	1000715	<a href="#">271341</a>	9 сентября 2014
10-STABLE после FreeBSD-SA-14:18 (rev <a href="#">269686</a> ).	1000716	<a href="#">271686</a>	16 сентября 2014

Значение	Версия	Дата	Релиз
10-STABLE после FreeBSD-SA-14:19 (ревизия <a href="#">271667</a> ).	1000717	<a href="#">271816</a>	18 сентября 2014
10-STABLE после добавления поддержки аппаратного контекста i915.	1001000	<a href="#">272463</a>	2 октября 2014
10.1-RC1 после ветки releng/10.1.	1001500	<a href="#">272464</a>	2 октября 2014
10-STABLE после ветки releng/10.1.	1001501	<a href="#">273432</a>	21 октября 2014
10-STABLE после исправлений уязвимостей FreeBSD-SA-14:20, FreeBSD-SA-14:22 и FreeBSD-SA-14:23 (ссылка на ревизию: <a href="#">273411</a> ).	1001502	<a href="#">274162</a>	4 ноября 2014
10-STABLE после FreeBSD-SA-14:23, FreeBSD-SA-14:24 и FreeBSD-SA-14:25.	1001503	<a href="#">275040</a>	25 ноября 2014
10-STABLE после объединения новых библиотек/утилит ( <a href="#">dpr(1)</a> , <a href="#">dpr(3)</a> и <a href="#">figpar(3)</a> ) для визуализации пропускной способности данных.	1001504	<a href="#">275742</a>	13 декабря 2014
10-STABLE после объединения важного исправления в векторизатор LLVM, которое в некоторых случаях могло приводить к переполнению буфера.	1001505	<a href="#">276633</a>	3 января 2015

Значение	Версия	Дата	Релиз
10-STABLE после объединения некоторых констант ARM в <a href="#">276312</a> .	1001506	<a href="#">277087</a>	12 января 2015
10-STABLE после объединения обновления максимального размера таблицы для уасс.	1001507	<a href="#">277790</a>	27 января 2015
10-STABLE после изменений в обратном вызове туннелирования UDP для предоставления указателя контекста и исходного <code>sockaddr</code> .	1001508	<a href="#">278974</a>	18 февраля 2015
10-STABLE после добавления типа запроса <code>CDAI_TYPE_EXT_INQ</code> .	1001509	<a href="#">279287</a>	25 февраля 2015
10-STABLE после FreeBSD-EN-15:01.vt, FreeBSD-EN-15:02.openssl, FreeBSD-EN-15:03.freebsd-update, FreeBSD-SA-15:04.igmp и FreeBSD-SA-15:05.bind.	1001510	<a href="#">279329</a>	26 февраля 2015
10-STABLE после MFC ревизии <a href="#">278964</a> .	1001511	<a href="#">280246</a>	19 марта 2015
10-STABLE после переименования <code>sys/capability.h</code> в <code>sys/capsicum.h</code> (изменение: <a href="https://svnweb.freebsd.org/changeset/base/280224/[280224/]">https://svnweb.freebsd.org/changeset/base/280224/[280224/]</a> ).	1001512	<a href="#">280438</a>	24 марта 2015
10-STABLE после добавления новых <code>ioctl</code> в <code>mtio(4)</code> и <code>sa(4)</code> .	1001513	<a href="#">281955</a>	24 апреля 2015

Значение	Версия	Дата	Релиз
10-STABLE после начала процесса удаления использования устаревшего флага "M_FLOWID" из сетевого кода.	1001514	<a href="#">282275</a>	30 апреля 2015
10-STABLE после MFC исправлений <a href="#">iconv(3)</a> .	1001515	<a href="#">282781</a>	11 мая 2015
10-STABLE после возврата <a href="#">M_FLOWID</a> .	1001516	<a href="#">283341</a>	24 мая 2015
10-STABLE после переноса (MFC) множества изменений, связанных с USB.	1001517	<a href="#">283950</a>	3 июня 2015
10-STABLE после слияния изменений, связанных со звуком.	1001518	<a href="#">284204</a>	10 июня 2015
10-STABLE после MFC исправлений <a href="#">vfs</a> для <a href="#">zfs</a> (rev <a href="#">284203</a> ).	1001519	<a href="#">284720</a>	23 июня 2015
10-STABLE после отмены увеличения <a href="#">MAXCPU</a> на amd64.	1002000	<a href="#">285830</a>	24 июля 2015
<a href="#">rele/10.2</a> отделился от 10-STABLE.	1002500	<a href="#">285831</a>	24 июля 2015
10-STABLE после того, как ветка <a href="#">rele/10.2</a> отделилась от 10-STABLE.	1002501	<a href="#">289005</a>	8 октября 2015
10-STABLE после объединения изменений ZFS, затронувших внутренний интерфейс структуры <a href="#">zfeature_info</a> (rev <a href="#">288572</a> ).	1002502	<a href="#">291243</a>	24 ноября 2015

Значение	Версия	Дата	Релиз
10-STABLE после объединения изменений устройств дампа, которые затронули аргументы <code>g_dev_setdumpdev()</code> (rev <a href="#">291215</a> ).	1002503	<a href="#">292224</a>	14 декабря 2015
10-STABLE после объединения изменений во внутренний интерфейс между модулями <code>nfsd.ko</code> и <code>nfscommon.ko</code> , что требует их совместного обновления (rev <a href="#">292223</a> ).	1002504	<a href="#">292589</a>	22 декабря 2015
10-STABLE после слияния хз 5.2.2 (поддержка многопоточности) (rev <a href="#">292588</a> ).	1002505	<a href="#">292908</a>	30 декабря 2015
10-STABLE после объединения изменений в <code>pci(4)</code> (rev <a href="#">292907</a> ).	1002506	<a href="#">293476</a>	9 января 2016
10-STABLE после объединения <code>utimensat(2)</code> (изменение <a href="#">293473</a> ).	1002507	<a href="#">293610</a>	9 января 2016
10-STABLE после объединения изменений в <code>linux(4)</code> (rev <a href="#">293477</a> через <a href="#">293609</a> ).	1002508	<a href="#">293619</a>	9 января 2016
10-STABLE после объединения изменений в типы/макросы <code>figpar(3)</code> (rev <a href="#">290275</a> ).	1002509	<a href="#">295107</a>	1 февраля 2016

Значение	Версия	Дата	Релиз
10-STABLE после объединения изменения API в <a href="#">dprv(3)</a> .	1003000	<a href="#">296373</a>	4 марта 2016
<a href="#">releng/10.3</a> ответвился от 10-STABLE.	1003500	<a href="#">296374</a>	4 марта 2016
10-STABLE после того, как ветка <a href="#">releng/10.3</a> отделилась от 10-STABLE.	1003501	<a href="#">298299</a>	19 июня 2016
10-STABLE после добавления опции -P для <a href="#">kdbcontrol</a> (rev <a href="#">298297</a> ).	1003502	<a href="#">299966</a>	19 июня 2016
10-STABLE после того, как <a href="#">libcrypto.so</a> стала позиционно-независимой.	1003503	<a href="#">300235</a>	19 июня 2016
10-STABLE после разрешения переопределений <a href="#">MK_</a> (изменение <a href="#">300233</a> ).	1003504	<a href="#">302066</a>	21 июня 2016
10-STABLE после переноса изменений <a href="#">filemon</a> из 11-CURRENT.	1003505	<a href="#">302228</a>	27 июня 2016
10-STABLE после замены в <a href="#">sed</a> на использование <a href="#">REG_STARTEND</a> , с исправлением проблемы в Mesa.	1003506	<a href="#">304611</a>	22 августа 2016
10-STABLE после добавления поддержки <a href="#">thread_local</a> в C++11.	1003507	<a href="#">304864</a>	26 августа 2016
10-STABLE после исправления <a href="#">LC_*_MASK</a> .	1003508	<a href="#">305734</a>	12 сентября 2016

Значение	Версия	Дата	Релиз
10-STABLE после устранения взаимоблокировки между <code>device_detach()</code> и <code>usbd_do_request_flags(9)</code> .	1003509	<a href="#">307331</a>	14 октября 2016
10-STABLE после слияния с ZFS.	1003510	<a href="#">308047</a>	28 октября 2016
10-STABLE после установки заголовочных файлов, необходимых для разработки с <code>libzfs_core</code> .	1003511	<a href="#">310121</a>	15 декабря 2016
10-STABLE после экспорта полного имени потока в <code>kinfo_proc</code> (rev <a href="#">309676</a> ).	1003512	<a href="#">315730</a>	22 марта 2017
10-STABLE после изменений в <code>libmd</code> (rev <a href="#">314143</a> ).	1003513	<a href="#">316499</a>	4 апреля 2017
10-STABLE после того, как блокировка CAM SIM стала опциональной (ревизии <a href="#">315673</a> , <a href="#">315674</a> ).	1003514	<a href="#">318198</a>	11 мая 2017
10-STABLE после объединения добавления заголовочного файла <code>&lt;dev/mmc/mmc_ioctl.h&gt;</code> .	1003515	<a href="#">321222</a>	19 июля 2017
10-STABLE после добавления функций освобождения памяти с указанием размера из C14 в <code>libc</code> .	1003516	<a href="#">321717</a>	30 июля 2017
10-STABLE после объединения добавления флага <code>MAP_GUARD</code> в <code>mmap(2)</code> .	1004000	<a href="#">323604</a>	15 сентября 2017

Значение	Версия	Дата	Релиз
<a href="#">releng/10.4</a> ответвился от 10-STABLE.	1004500	<a href="#">323605</a>	15 сентября 2017
10-STABLE после того, как ветка <a href="#">releng/10.4</a> отделилась от 10-STABLE.	1004501	<a href="#">328379</a>	24 января 2018
10-STABLE после слияния изменений <a href="#">325028</a> , исправляющего <a href="#">ptrace()</a> , чтобы всегда очищать правильное событие потока при возобновлении.	1004502	<a href="#">356396</a>	6 января 2020
10-STABLE после изменения статистики USB для каждого устройства вместо каждой шины.	1004503	<a href="#">356681</a>	13 января 2020

## 18.7. Версии FreeBSD 9

Таблица 59. Значения `__FreeBSD_version` в FreeBSD 9

Значение	Версия	Дата	Релиз
900000	<a href="#">196432</a>	22 августа 2009	9.0-CURRENT.
900001	<a href="#">197019</a>	8 сентября 2009	9.0-CURRENT после импорта x86emu, программного эмулятора процессора x86 в реальном режиме из OpenBSD.
900002	<a href="#">197430</a>	23 сентября 2009	9.0-CURRENT после реализации функциональности фильтра <a href="#">kevent EVFILT_USER</a> .
900003	<a href="#">200039</a>	2 декабря 2009	9.0-CURRENT после добавления <a href="#">sigpause(2)</a> и поддержки PIE в <a href="#">csu</a> .

Значение	Версия	Дата	Релиз
900004	<a href="#">200185</a>	6 декабря 2009	9.0-CURRENT после добавления <code>libulog</code> и его совместимого интерфейса <code>libutempter</code> .
900005	<a href="#">200447</a>	December 12, 2009	9.0-CURRENT после добавления <code>sleepq_sleepcnt(9)</code> , который может использоваться для запроса количества ожидающих в конкретной очереди ожидания.
900006	<a href="#">201513</a>	4 января 2010	9.0-CURRENT после изменения прототипов <code>scandir(3)</code> и <code>alphasort(3)</code> для соответствия SUSv4.
900007	<a href="#">202219</a>	13 января 2010	9.0-CURRENT после удаления <code>utmp(5)</code> и добавления <code>utmpx</code> (см. <code>getutxent(3)</code> ) для улучшенного логирования входов пользователей и системных событий.
900008	<a href="#">202722</a>	20 января 2010	9.0-CURRENT после импорта BSDL bc/dc и устаревания GNU bc/dc.
900009	<a href="#">203052</a>	26 января 2010	9.0-CURRENT после добавления <code>ioctl</code> <code>SIOCGIFDESCR</code> и <code>SIOCSIFDESCR</code> к сетевым интерфейсам. Эти <code>ioctl</code> могут использоваться для управления описанием интерфейса, по аналогии с OpenBSD.

Значение	Версия	Дата	Релиз
900010	<a href="#">205471</a>	22 марта 2010	9.0-CURRENT после импорта <code>zlib 1.2.4</code> .
900011	<a href="#">207410</a>	April 24, 2010	9.0-CURRENT после добавления журналирования <code>soft-updates</code> .
900012	<a href="#">207842</a>	10 мая 2010	9.0-CURRENT после добавления <code>liblzma</code> , <code>xz</code> , <code>xzdec</code> и <code>lzmainfo</code> .
900013	<a href="#">208486</a>	24 мая 2010	9.0-CURRENT после внесения исправлений для USB в <a href="#">linux(4)</a> .
900014	<a href="#">208973</a>	10 июня 2010	9.0-CURRENT после добавления Clang.
900015	<a href="#">210390</a>	22 июля 2010	9.0-CURRENT после импорта BSD <code>grep</code> .
900016	<a href="#">210565</a>	28 июля 2010	9.0-CURRENT после добавления <code>mti_zone</code> в структуру <code>malloc_type_internal</code> .
900017	<a href="#">211701</a>	23 августа 2010	9.0-CURRENT после возврата к GNU <code>grep</code> по умолчанию и добавления параметра <code>WITH_BSD_GREP</code> .
900018	<a href="#">211735</a>	24 августа 2010	9.0-CURRENT после того, как сигнал, сгенерированный через <code>pthread_kill(3)</code> , идентифицируется как <code>SI_LWP</code> в <code>si_code</code> . Ранее <code>si_code</code> имел значение <code>SI_USER</code> .
900019	<a href="#">211937</a>	28 августа 2010	9.0-CURRENT после добавления флага <code>MAP_PREFAULT_READ</code> в <a href="#">mmap(2)</a> .

Значение	Версия	Дата	Релиз
900020	<a href="#">212381</a>	9 сентября 2010	9.0-CURRENT после добавления функциональности drain в <code>sbufs</code> , что также изменило структуру struct <code>sbuf</code> .
900021	<a href="#">212568</a>	13 сентября 2010	9.0-CURRENT после того, как DTrace обзавелся поддержкой трассировки в пользовательском пространстве.
900022	<a href="#">213395</a>	2 октября 2010	9.0-CURRENT после добавления утилит man под лицензией BSDL и удаления утилит man под лицензией GNU/GPL.
900023	<a href="#">213700</a>	11 октября 2010	9.0-CURRENT после обновления xz до снимка git 20101010.
900024	<a href="#">215127</a>	11 ноября 2010	9.0-CURRENT после замены libgcc.a на libcompiler_rt.a.
900025	<a href="#">215166</a>	12 ноября 2010	9.0-CURRENT после внедрения модульной системы управления перегрузкой.
900026	<a href="#">216088</a>	30 ноября 2010	9.0-CURRENT после введения сквозной передачи Serial Management Protocol (SMP) и CAM <code>CCBs</code> XPT_SMP_IO и XPT_GDEV_ADVINFO.
900027	<a href="#">216212</a>	5 декабря 2010	9.0-CURRENT после добавления log2 в libm.

Значение	Версия	Дата	Релиз
900028	<a href="#">216615</a>	21 декабря 2010	9.0-CURRENT после добавления Nhook (Helper Hook), Khelp (Kernel Helpers) и KPI Object Specific Data (OSD).
900029	<a href="#">216758</a>	28 декабря 2010	9.0-CURRENT после модификации стека TCP для разрешения модулям Khelp взаимодействовать с ним через точки подключения вспомогательных функций и хранить данные для каждого соединения в блоке управления TCP.
900030	<a href="#">217309</a>	12 января 2011	9.0-CURRENT после обновления libdialog до версии 20100428.
900031	<a href="#">218414</a>	7 февраля 2011	9.0-CURRENT после добавления <a href="#">pthread_getthreadid_np(3)</a> .
900032	<a href="#">218425</a>	8 февраля 2011	9.0-CURRENT после удаления прототипа и символа <a href="#">uio_yield</a> .
900033	<a href="#">218822</a>	18 февраля 2011	9.0-CURRENT после обновления binutils до версии 2.17.50.
900034	<a href="#">219406</a>	8 марта 2011	9.0-CURRENT после изменений в структуре <a href="#">sysvec</a> ( <a href="#">sv_schedtail</a> ).
900035	<a href="#">220150</a>	29 марта 2011	9.0-CURRENT после обновления базового gcc и libstdc++ до последней ревизии, лицензированной под GPLv2.

Значение	Версия	Дата	Релиз
900036	<a href="#">220770</a>	18 апреля 2011	9.0-CURRENT после удаления libobjc и поддержки Objective-C из базовой системы.
900037	<a href="#">221862</a>	13 мая 2011	9.0-CURRENT после импорта библиотеки <a href="#">libprocstat(3)</a> и утилиты <a href="#">fuser(1)</a> в базовую систему.
900038	<a href="#">222167</a>	22 мая 2011	9.0-CURRENT после добавления аргумента флага блокировки к <a href="#">VFS_FHTOVP(9)</a> .
900039	<a href="#">223637</a>	28 июня 2011	9.0-CURRENT после импорта pf из OpenBSD 4.5.
900040	<a href="#">224217</a>	19 июля 2011	Увеличить значение MAXCPU по умолчанию для FreeBSD до 64 на amd64 и ia64 и до 128 для XLP (mips).
900041	<a href="#">224834</a>	13 августа 2011	Версия 9.0-CURRENT после реализации возможностей Capsicum; <a href="#">fget(9)</a> получает аргумент rights.
900042	<a href="#">225350</a>	28 августа 2011	Увеличьте номера версий общих библиотек для библиотек, чей ABI изменился в рамках подготовки к 9.0.
900043	<a href="#">225350</a>	2 сентября 2011	Добавить автоматическое обнаружение USB-накопителей, которые не поддерживают команду SCSI "no synchronize cache".

Значение	Версия	Дата	Релиз
900044	<a href="#">225469</a>	10 сентября 2011	Переработка автоматического определения особенностей оборудования (auto-quirk). 9.0-RELEASE.
900045	<a href="#">229285</a>	2 января 2012	9-STABLE после MFC значения true/false из 1000002.
900500	<a href="#">229318</a>	2 января 2012	9.0-STABLE.
900501	<a href="#">229723</a>	6 января 2012	9.0-STABLE после объединения добавления системного вызова <a href="#">posix_fadvise(2)</a> .
900502	<a href="#">230237</a>	16 января 2012	9.0-STABLE после слияния gperf 3.0.3
900503	<a href="#">231768</a>	15 февраля 2012	9.0-STABLE после введения нового расширяемого интерфейса <a href="#">sysctl(3)</a> NET_RT_IFLISTL для запроса списков адресов.
900504	<a href="#">232728</a>	3 марта 2012	9.0-STABLE после изменений, связанных с монтированием файловой системы внутри клетки.
900505	<a href="#">232945</a>	13 марта 2012	9.0-STABLE после введения новых параметров сокета <a href="#">tcp(4)</a> : TCP_KEEPIKIT, TCP_KEEPIKLE, TCP_KEEPIKTVL и TCP_KEEPCNT.

Значение	Версия	Дата	Релиз
900506	<a href="#">235786</a>	22 мая 2012	9.0-STABLE после введения функции <a href="#">quick_exit</a> и связанных изменений, необходимых для C++11.
901000	<a href="#">239082</a>	5 августа 2012	9.1-RELEASE.
901500	<a href="#">239081</a>	6 августа 2012	9.1-STABLE после ветвления releng/9.1 (RELENG_9_1).
901501	<a href="#">240659</a>	11 ноября 2012	9.1-STABLE после <a href="#">LIST_PREV(3)</a> добавлен в queue.h (изменение rev <a href="#">242893</a> ) и изменения KBI в USB последовательных устройствах.
901502	<a href="#">243656</a>	28 ноября 2012	9.1-STABLE после того, как буфер дрожания USB serial требует пересборки модулей устройств USB serial.
901503	<a href="#">247090</a>	21 февраля 2013	9.1-STABLE после перемещения USB в структуру драйверов, что потребовало пересборки всех модулей USB. Также указывает на наличие nmtree.
901504	<a href="#">248338</a>	15 марта 2013	9.1-STABLE после установки получил флаги -l, -M, -N и связанные с ними, а cat получил опцию -l.
901505	<a href="#">251687</a>	13 июня 2013	9.1-STABLE после исправлений в начальной загрузке <a href="#">ctfmerge</a> (rev <a href="#">249243</a> ).
902001	<a href="#">253912</a>	3 августа 2013	<a href="#">releng/9.2</a> ответвился от <a href="#">stable/9</a> .

Значение	Версия	Дата	Релиз
902501	<a href="#">253913</a>	2 августа 2013	9.2-STABLE после создания ветки <a href="#">releng/9.2</a> .
902502	<a href="#">254938</a>	26 августа 2013	9.2-STABLE после включения флага запроса пути CAM <a href="#">PIM_RESCAN</a> .
902503	<a href="#">254979</a>	27 августа 2013	9.2-STABLE после включения флага <a href="#">SI_UNMAPPED</a> для cdev.
902504	<a href="#">256917</a>	22 октября 2013	9.2-STABLE после добавления поддержки скриптов "первой загрузки" <a href="#">rc(8)</a> .
902505	<a href="#">259448</a>	12 декабря 2013	9.2-STABLE после исправления кодировки Heimdal.
902506	<a href="#">260136</a>	31 декабря 2013	9-STABLE после исправлений MAP_STACK (rev <a href="#">260082</a> ).
902507	<a href="#">262801</a>	5 марта 2014	9-STABLE после обновления libc++ до версии 3.4.
902508	<a href="#">263171</a>	14 марта 2014	9-STABLE после объединения драйвера Radeon KMS (rev <a href="#">263170</a> ).
902509	<a href="#">263509</a>	21 марта 2014	9-STABLE после обновления llvm/clang до версии 3.4.
902510	<a href="#">263818</a>	27 марта 2014	9-STABLE после слияния драйвера <a href="#">vt(4)</a> .
902511	<a href="#">264289</a>	27 марта 2014	9-STABLE после FreeBSD-SA-14:06.openssl.
902512	<a href="#">265123</a>	30 апреля 2014	9-STABLE после FreeBSD-SA-14:08.tcp.
903000	<a href="#">267656</a>	20 июня 2014	9-RC1 ветка <a href="#">releng/9.3</a> .

Значение	Версия	Дата	Релиз
903500	<a href="#">267657</a>	20 июня 2014	9.3-STABLE ветка <a href="#">releng/9.3</a> .
903501	<a href="#">268443</a>	8 июля 2014	9-STABLE после FreeBSD-SA-14:17.kmem (изменение: <a href="https://svnweb.freebsd.org/changeset/base/268433">https://svnweb.freebsd.org/changeset/base/268433</a> [268433]).
903502	<a href="#">270175</a>	19 августа 2014	9-STABLE после исправления ошибки <a href="#">SOCK_DGRAM</a> (rev <a href="#">269789</a> ).
903503	<a href="#">271341</a>	9 сентября 2014	9-STABLE после FreeBSD-SA-14:18 (ссылка на ревизию: <a href="https://svnweb.freebsd.org/changeset/base/269687">https://svnweb.freebsd.org/changeset/base/269687</a> [269687]).
903504	<a href="#">271686</a>	16 сентября 2014	9-STABLE после FreeBSD-SA-14:19 (rev <a href="#">271668</a> ).
903505	<a href="#">273432</a>	21 октября 2014	9-STABLE после исправлений FreeBSD-SA-14:20, FreeBSD-SA-14:21 и FreeBSD-SA-14:22 (rev <a href="#">273412</a> ).
903506	<a href="#">274162</a>	4 ноября 2014	9-STABLE после FreeBSD-SA-14:23, FreeBSD-SA-14:24 и FreeBSD-SA-14:25.
903507	<a href="#">275742</a>	13 декабря 2014	9-STABLE после объединения важного исправления в векторизатор LLVM, которое в некоторых случаях могло приводить к переполнению буфера.

Значение	Версия	Дата	Релиз
903508	<a href="#">279287</a>	25 февраля 2015	9-STABLE после FreeBSD-EN-15:01.vt, FreeBSD-EN-15:02.openssl, FreeBSD-EN-15:03.freebsd-update, FreeBSD-SA-15:04.igmp и FreeBSD-SA-15:05.bind.
903509	<a href="#">296219</a>	29 февраля 2016	9-STABLE после увеличения значения по умолчанию <code>compat.linux.osrelease</code> до <b>2.6.18</b> для поддержки портов <code>linux-c6-*</code> без дополнительной настройки.
903510	<a href="#">300236</a>	19 мая 2016	9-STABLE после того, как страница System Binary Interface (SBI) была перемещена в последней версии Berkeley Boot Loader (BBL) из-за увеличения размера кода в <a href="#">300234</a> .
903511	<a href="#">305735</a>	12 сентября 2016	9-STABLE после разрешения взаимоблокировки между <code>device_detach()</code> и <code>usbd_do_request_flags(9)</code> .

## 18.8. Версии FreeBSD 8

Таблица 60. Значения `__FreeBSD_version` в FreeBSD 8

Значение	Версия	Дата	Релиз
800000	<a href="#">172531</a>	11 октября 2007	8.0-CURRENT. Разделение <code>ctype</code> на широкие и однобайтовые символы.

Значение	Версия	Дата	Релиз
800001	<a href="#">172688</a>	16 октября 2007	8.0-CURRENT после импорта <code>librcar 0.9.8</code> и <code>tcpdump 3.9.8</code> .
800002	<a href="#">172841</a>	21 октября 2007	8.0-CURRENT после переименования <code>kthread_create(9)</code> и связанных функций в <code>kproc_create(9)</code> и т.д.
800003	<a href="#">172932</a>	24 октября 2007	8.0-CURRENT после того, как была добавлена обратная совместимость ABI с версиями FreeBSD 4/5/6 для IOCTL <code>PCIOCGETCONF</code> , <code>PCIOCREAD</code> и <code>PCIOCWRITE</code> , что потребовало снова нарушить ABI IOCTL <code>PCIOCGETCONF</code>
800004	<a href="#">173573</a>	12 ноября 2007	8.0-CURRENT после перемещения драйвера <code>agr(4)</code> из <code>src/sys/pci</code> в <code>src/sys/dev/agr</code>
800005	<a href="#">174261</a>	4 декабря 2007	8.0-CURRENT после изменений в аллокаторе джамбо-фреймов (rev <a href="#">174247</a> ).
800006	<a href="#">174399</a>	7 декабря 2007	8.0-CURRENT после добавления функциональности захвата <code>callgraph</code> в <code>hwpmc(4)</code> .
800007	<a href="#">174901</a>	25 декабря 2007	8.0-CURRENT после <code>kdb_enter()</code> получает аргумент "why".
800008	<a href="#">174951</a>	28 декабря 2007	8.0-CURRENT после удаления опции <code>LK_EXCLUPGRADE</code> .
800009	<a href="#">175168</a>	9 января 2008	8.0-CURRENT после введения <code>lockmgr_disown(9)</code>

Значение	Версия	Дата	Релиз
800010	<a href="#">175204</a>	10 января 2008	8.0-CURRENT после изменения прототипа <a href="#">vn_lock(9)</a> .
800011	<a href="#">175295</a>	13 января 2008	8.0-CURRENT после изменений прототипов <a href="#">VOP_LOCK(9)</a> и <a href="#">VOP_UNLOCK(9)</a> .
800012	<a href="#">175487</a>	19 января 2008	8.0-CURRENT после введения <a href="#">lockmgr_recursed(9)</a> , <a href="#">BUF_RECURSED(9)</a> и <a href="#">BUF_ISLOCKED(9)</a> , а также удаления <a href="#">BUF_REFCNT()</a> .
800013	<a href="#">175581</a>	23 января 2008	8.0-CURRENT после введения кодировки "ASCII".
800014	<a href="#">175636</a>	24 января 2008	8.0-CURRENT после изменения прототипа <a href="#">lockmgr(9)</a> и удаления <a href="#">lockcount()</a> и <a href="#">LOCKMGR_ASSERT()</a> .
800015	<a href="#">175688</a>	26 января 2008	8.0-CURRENT после расширения типов структур <a href="#">fts(3)</a> .
800016	<a href="#">175872</a>	1 февраля 2008	8.0-CURRENT после добавления аргумента в <a href="#">MEXTADD(9)</a>
800017	<a href="#">176015</a>	6 февраля 2008	8.0-CURRENT после введения опций LK_NODUP и LK_NOWITNESS в пространстве <a href="#">lockmgr(9)</a> .
800018	<a href="#">176112</a>	8 февраля 2008	8.0-CURRENT после добавления <a href="#">m_collapse</a> .

Значение	Версия	Дата	Релиз
800019	<a href="#">176124</a>	9 февраля 2008	8.0-CURRENT после добавления поддержки текущего рабочего каталога, корневого каталога и каталога клетки в <code>sysctl kern.proc.filedesc</code> .
800020	<a href="#">176251</a>	13 февраля 2008	8.0-CURRENT после добавления <a href="#">lockmgr_assert(9)</a> и функций <code>BUF_ASSERT</code> .
800021	<a href="#">176321</a>	15 февраля 2008	8.0-CURRENT после введения <a href="#">lockmgr_args(9)</a> и удаления флага <code>LK_INTERNAL</code> .
800022	<a href="#">176556</a>	(отменено)	8.0-CURRENT после изменения стандартной системной <code>ar</code> на BSD <a href="#">ar(1)</a> .
800023	<a href="#">176560</a>	25 февраля 2008	8.0-CURRENT после изменения прототипов <a href="#">lockstatus(9)</a> и <a href="#">VOP_ISLOCKED(9)</a> ; а именно удаления аргумента <code>struct thread</code> .
800024	<a href="#">176709</a>	1 марта 2008	8.0-CURRENT после удаления функций <code>lockwaiters</code> и <code>BUF_LOCKWAITERS</code> , изменения возвращаемого значения <code>brrelvp</code> с <code>void</code> на <code>int</code> и добавления новых флагов для <a href="#">lockinit(9)</a> .
800025	<a href="#">176958</a>	8 марта 2008	8.0-CURRENT после добавления команды <code>F_DUP2FD</code> в <a href="#">fcntl(2)</a> .

Значение	Версия	Дата	Релиз
800026	<a href="#">177086</a>	12 марта 2008	8.0-CURRENT после изменения параметра приоритета на <code>cv_broadcastpri</code> , где 0 означает отсутствие приоритета.
800027	<a href="#">177551</a>	24 марта 2008	8.0-CURRENT после изменения ABI мониторинга <code>brf</code> при добавлении буферов <code>brf</code> с <code>zergosopu</code> .
800028	<a href="#">177637</a>	26 марта 2008	8.0-CURRENT после добавления <code>l_sysid</code> в структуру <code>flock</code> .
800029	<a href="#">177688</a>	March 28, 2008	8.0-CURRENT после повторного включения функции <code>BUF_LOCKWAITERS</code> и добавления <code>lockmgr_waiters(9)</code> .
800030	<a href="#">177844</a>	1 апреля 2008	8.0-CURRENT после введения функций <code>rw_try_rlock(9)</code> и <code>rw_try_wlock(9)</code> .
800031	<a href="#">177958</a>	6 апреля 2008	8.0-CURRENT после введения функций <code>lockmgr_gw</code> и <code>lockmgr_args_gw</code> .
800032	<a href="#">178006</a>	8 апреля 2008	8.0-CURRENT после реализации системных вызовов <code>openat</code> и связанных с ними, введения флага <code>O_EXEC</code> для <code>open(2)</code> и предоставления соответствующих системных вызовов совместимости с Linux.

Значение	Версия	Дата	Релиз
800033	<a href="#">178017</a>	8 апреля 2008	8.0-CURRENT после добавления поддержки <code>write(2)</code> для <code>psm(4)</code> на уровне нативной работы. Теперь произвольные команды можно записывать в <code>/dev/psm%d</code> , а статус — считывать из него.
800034	<a href="#">178051</a>	10 апреля 2008	8.0-CURRENT после введения функции <code>memchr</code> .
800035	<a href="#">178256</a>	16 апреля 2008	8.0-CURRENT после введения функции <code>fdopendir</code> .
800036	<a href="#">178362</a>	20 апреля 2008	8.0-CURRENT после перехода на поддержку multi-bss в беспроводных сетях 802.11 (также известную как <code>vaps</code> ).
800037	<a href="#">178892</a>	9 мая 2008	8.0-CURRENT после добавления поддержки нескольких таблиц маршрутизации (также известных как <code>setfib(1)</code> , <code>setfib(2)</code> ).
800038	<a href="#">179316</a>	26 мая 2008	8.0-CURRENT после удаления <code>netatm</code> и <code>ISDN4BSD</code> . Также добавлены инструменты Compaq C Type (CTF).
800039	<a href="#">179784</a>	14 июня 2008	8.0-CURRENT после удаления <code>sgtty</code> .
800040	<a href="#">180025</a>	26 июня 2008	8.0-CURRENT с клиентом <code>lockd</code> для NFS в ядре.

Значение	Версия	Дата	Релиз
800041	<a href="#">180691</a>	22 июля 2008	8.0-CURRENT после добавления <a href="#">arc4random_buf(3)</a> и <a href="#">arc4random_uniform(3)</a> .
800042	<a href="#">181439</a>	8 августа 2008	8.0-CURRENT после добавления <a href="#">cructl(4)</a> .
800043	<a href="#">181694</a>	13 августа 2008	8.0-CURRENT после изменения <a href="#">bpf(4)</a> для использования единого узла устройства вместо клонирования устройств.
800044	<a href="#">181803</a>	17 августа 2008	8.0-CURRENT после внесения изменений, связанных с первым этапом проекта VIMAGE, переименованы глобальные переменные, подлежащие виртуализации, с добавлением префикса <code>V_</code> и макросов для их обратного отображения на глобальные имена.
800045	<a href="#">181905</a>	20 августа 2008	8.0-CURRENT после интеграции MPSAFE TTY слоя, включая изменения в различных драйверах и утилитах, взаимодействующих с ним.
800046	<a href="#">182869</a>	8 сентября 2008	8.0-CURRENT после разделения GDT для каждого процессора на архитектуре amd64.

Значение	Версия	Дата	Релиз
800047	<a href="#">182905</a>	10 сентября 2008	8.0-CURRENT после удаления VSVTX, VSGID и VSUID.
800048	<a href="#">183091</a>	16 сентября 2008	8.0-CURRENT после преобразования кода монтирования ядра NFS для поддержки отдельных опций монтирования в <code>iovec nmount(2)</code> , а не только одной большой структуры <code>nfs_args</code> .
800049	<a href="#">183114</a>	17 сентября 2008	8.0-CURRENT после удаления <code>suser(9)</code> и <code>suser_cred(9)</code> .
800050	<a href="#">184099</a>	20 октября 2008	8.0-CURRENT после изменения API кэша буфера.
800051	<a href="#">184205</a>	23 октября 2008	8.0-CURRENT после удаления макросов <code>MALLOC(9)</code> и <code>FREE(9)</code> .
800052	<a href="#">184419</a>	28 октября 2008	8.0-CURRENT после введения <code>accmode_t</code> и переименования аргумента <code>a_mode</code> в <code>a_accmode</code> в <code>VOP_ACCESS</code> .
800053	<a href="#">184555</a>	2 ноября 2008	8.0-CURRENT после изменения прототипа <code>vfs_busy(9)</code> и введения флагов <code>MBF_NOWAIT</code> и <code>MBF_MNTLSTLOCK</code> .

Значение	Версия	Дата	Релиз
800054	<a href="#">185162</a>	22 ноября 2008	8.0-CURRENT после добавления <code>buf_ring</code> , барьеров памяти и функций <code>ifnet</code> для поддержки нескольких аппаратных очередей передачи для карт, которые их поддерживают, а также реализации кольцевого буфера без блокировок, чтобы позволить драйверам более эффективно управлять очередями пакетов.
800055	<a href="#">185363</a>	27 ноября 2008	8.0-CURRENT после добавления поддержки Intel™ Core, Core2 и Atom в <code>hwpmc(4)</code> .
800056	<a href="#">185435</a>	29 ноября 2008	8.0-CURRENT после введения многопользовательских/без IPv4/v6 клеток.
800057	<a href="#">185522</a>	1 декабря 2008	8.0-CURRENT после перехода на исходный код <code>hal</code> для <code>ath</code> .
800058	<a href="#">185968</a>	12 декабря 2008	8.0-CURRENT после введения операции <code>VOP_VPTOCNP</code> .
800059	<a href="#">186119</a>	15 декабря 2008	8.0-CURRENT включает новую переработанную версию <code>arp-v2</code> .
800060	<a href="#">186344</a>	19 декабря 2008	8.0-CURRENT после добавления <code>makefs</code> .
800061	<a href="#">187289</a>	15 января 2009	8.0-CURRENT после TCP Appropriate Byte Counting.

Значение	Версия	Дата	Релиз
800062	<a href="#">187830</a>	28 января 2009	8.0-CURRENT после удаления <code>minor()</code> , <code>minor2unit()</code> , <code>unit2minor()</code> и т.д.
800063	<a href="#">188745</a>	18 февраля 2009	8.0-CURRENT после изменения конфигурации GENERIC для использования стека USB2, а также добавления <code>fdevname(3)</code> .
800064	<a href="#">188946</a>	23 февраля 2009	8.0-CURRENT после переноса стека USB2 и замены <code>dev/usb</code> .
800065	<a href="#">189092</a>	26 февраля 2009	8.0-CURRENT после переименования всех функций в <code>libmp(3)</code> .
800066	<a href="#">189110</a>	27 февраля 2009	8.0-CURRENT после изменения обработки и структуры USB <code>devfs</code> .
800067	<a href="#">189136</a>	28 февраля 2009	8.0-CURRENT после добавления <code>getdelim()</code> , <code>getline()</code> , <code>stpncpy()</code> , <code>strlen()</code> , <code>wcsnlen()</code> , <code>wscasemp()</code> , and <code>wcsncasemp()</code> .
800068	<a href="#">189276</a>	2 марта 2009	8.0-CURRENT после переименования класса устройств <code>ushub</code> в <code>uhub</code> .
800069	<a href="#">189585</a>	9 марта 2009	8.0-CURRENT после переименования <code>libusb20.so.1</code> в <code>libusb.so.1</code> .
800070	<a href="#">189592</a>	9 марта 2009	8.0-CURRENT после объединения IGMPv3 и Source-Specific Multicast (SSM) в стек IPv4.

Значение	Версия	Дата	Релиз
800071	<a href="#">189825</a>	14 марта 2009	8.0-CURRENT после того, как gcc был исправлен для использования семантики встраивания C99 в режимах c99 и gnu99.
800072	<a href="#">189853</a>	March 15, 2009	8.0-CURRENT после удаления флага IFF_NEEDSGIANT; неподдерживаемые MPSAFE драйверы сетевых устройств больше не поддерживаются.
800073	<a href="#">190265</a>	18 марта 2009	8.0-CURRENT после реализации подстановки динамических строковых токенов для gpath и необходимых путей.
800074	<a href="#">190373</a>	24 марта 2009	8.0-CURRENT после импорта tcprdump 4.0.0 и libpcap 1.0.0.
800075	<a href="#">190787</a>	6 апреля 2009	8.0-CURRENT после изменения структуры структур vnet_net, vnet_inet и vnet_ipfw.
800076	<a href="#">190866</a>	9 апреля 2009	8.0-CURRENT после добавления профилей задержки в dummpynet.
800077	<a href="#">190914</a>	14 апреля 2009	8.0-CURRENT после удаления <code>VOP_LEASE()</code> и <code>vop_vector.vop_lease</code> .

Значение	Версия	Дата	Релиз
800078	<a href="#">191080</a>	15 апреля 2009	8.0-CURRENT после добавления полей <code>rt_weight</code> в структуры <code>rt_metrics</code> и <code>rt_metrics_lite</code> , что изменило расположение полей в структуре <code>rt_metrics_lite</code> . Версия <code>RTM_VERSION</code> была увеличена, но затем отменена.
800079	<a href="#">191117</a>	15 апреля 2009	8.0-CURRENT после добавления указателей на структуру <code>lentry</code> в структуры <code>route</code> и <code>route_in6</code> .
800080	<a href="#">191126</a>	15 апреля 2009	8.0-CURRENT после изменения структуры <code>inpcb</code> .
800081	<a href="#">191267</a>	19 апреля 2009	8.0-CURRENT после изменения структуры <code>malloc_type</code> .
800082	<a href="#">191368</a>	21 апреля 2009	8.0-CURRENT после изменения структуры <code>struct ifnet</code> и с <code>if_ref()</code> и <code>if_rele()</code> для подсчета ссылок ( <code>refcounting</code> ) <code>ifnet</code> .
800083	<a href="#">191389</a>	22 апреля 2009	8.0-CURRENT после реализации низкоуровневого API HCI Bluetooth.
800084	<a href="#">191672</a>	29 апреля 2009	8.0-CURRENT после изменений в IPv6 SSM и MLDv2.
800085	<a href="#">191688</a>	30 апреля 2009	8.0-CURRENT после включения поддержки сборки ядра с VIMAGE с одним активным образом.

Значение	Версия	Дата	Релиз
800086	<a href="#">191910</a>	8 мая 2009	8.0-CURRENT после добавления поддержки строк ввода произвольной длины в <a href="#">patch(1)</a> .
800087	<a href="#">191990</a>	11 мая 2009	8.0-CURRENT после некоторых изменений в VFS KPI. Аргумент потока был удален из частей FSD в VFS. Функциям <a href="#">VFS_*</a> больше не нужен контекст, так как он всегда относится к <a href="#">curthread</a> . В некоторых особых случаях старое поведение сохранено.
800088	<a href="#">192470</a>	20 мая 2009	8.0-CURRENT после изменений режима монитора в net80211.
800089	<a href="#">192649</a>	23 мая 2009	8.0-CURRENT после добавления поддержки блока управления UDP.
800090	<a href="#">192669</a>	23 мая 2009	8.0-CURRENT после виртуализации клонирования интерфейсов.
800091	<a href="#">192895</a>	27 мая 2009	8.0-CURRENT после добавления иерархических клеток и удаления глобального <code>securelevel</code> .

Значение	Версия	Дата	Релиз
800092	<a href="#">193011</a>	29 мая 2009	8.0-CURRENT после изменения KPI <code>sx_init_flags()</code> . <code>SX_ADAPTIVESPIN</code> упрямднён, и введён новый флаг <code>SX_NOADAPTIVE</code> для обработки обратной логики.
800093	<a href="#">193047</a>	29 мая 2009	8.0-CURRENT после добавления <code>mnt_xflag</code> в структуру <code>mount</code> .
800094	<a href="#">193093</a>	30 мая 2009	8.0-CURRENT после добавления <code>VOP_ACCESSX(9)</code> .
800095	<a href="#">193096</a>	30 мая 2009	8.0-CURRENT после изменения KPI для <code>polling</code> . Обработчики <code>polling</code> теперь возвращают количество обработанных пакетов. Также введена новая возможность <code>IFCAP_POLLING_NOCOUNT</code> , которая указывает, что возвращаемое значение не является значимым и подсчёт следует пропустить.
800096	<a href="#">193219</a>	1 июня 2009	8.0-CURRENT после обновления до новой реализации <code>netisr</code> и после изменения способа хранения и доступа к FIB.
800097	<a href="#">193731</a>	8 июня 2009	8.0-CURRENT после введения хуков и инфраструктуры деструктора <code>vnet</code> .

Значение	Версия	Дата	Релиз
(не изменено)	<a href="#">194012</a>	11 июня 2009	8.0-CURRENT после введения в netgraph обнаружения вызовов пути из исходящего во входящий и организации очередей, что также изменило структуру struct thread.
800098	<a href="#">194210</a>	14 июня 2009	8.0-CURRENT после импорта OpenSSL 0.9.8k.
800099	<a href="#">194675</a>	22 июня 2009	8.0-CURRENT после обновления NGROUPS и переноса виртуализации маршрутов в собственный модуль VImage.
800100	<a href="#">194920</a>	24 июня 2009	8.0-CURRENT после изменения ABI SYSVIPIC.
800101	<a href="#">195175</a>	29 июня 2009	8.0-CURRENT после удаления символьных устройств /dev/net/* для каждого интерфейса.
800102	<a href="#">195634</a>	12 июля 2009	8.0-CURRENT после добавления заполнения к структурам sackhint, tcpcb и tcpstat.
800103	<a href="#">195654</a>	13 июля 2009	8.0-CURRENT после замены структуры tcpopt на структуру тоеopt в интерфейсе драйвера TOE к syncache TCP.

Значение	Версия	Дата	Релиз
800104	<a href="#">195699</a>	14 июля 2009	8.0-CURRENT после добавления распределителя на основе наборов компоновщика для каждого vnet.
800105	<a href="#">195767</a>	19 июля 2009	8.0-CURRENT после увеличения версии для всех разделяемых библиотек, у которых не включено управление версиями символов.
800106	<a href="#">195852</a>	24 июля 2009	8.0-CURRENT после введения типа объекта VM OBJT_SG.
800107	<a href="#">196037</a>	2 августа 2009	8.0-CURRENT после освобождения подсистемы newbus от Giant путем добавления <code>sxlock</code> в newbus и 8.0-RELEASE.
800108	<a href="#">199627</a>	21 ноября 2009	8.0-STABLE после реализации фильтра <code>kevent EVFILT_USER</code> .
800500	<a href="#">201749</a>	7 января 2010	8.0-STABLE после увеличения <code>__FreeBSD_version</code> , чтобы <code>pkg_add -r</code> использовал пакеты-8-stable.
800501	<a href="#">202922</a>	24 января 2010	8.0-STABLE после изменения прототипов <code>scandir(3)</code> и <code>alphasort(3)</code> для соответствия SUSv4.
800502	<a href="#">203299</a>	31 января 2010	8.0-STABLE после добавления <code>sigpause(2)</code> .

Значение	Версия	Дата	Релиз
800503	<a href="#">204344</a>	25 февраля 2010	8.0-STABLE после добавления ioctl SIOCGIFDESCR и SIOCSIFDESCR к сетевым интерфейсам. Эти ioctl могут использоваться для управления описанием интерфейса, по аналогии с OpenBSD.
800504	<a href="#">204546</a>	1 марта 2010	8.0-STABLE после MFC импорта x86emu, программного эмулятора реального режима x86 CPU из OpenBSD.
800505	<a href="#">208259</a>	18 мая 2010	8.0-STABLE после MFC добавления liblzma, xz, xzdec и lzmainfo.
801000	<a href="#">209150</a>	14 июня 2010	8.1-RELEASE
801500	<a href="#">209146</a>	14 июня 2010	8.1-STABLE после 8.1-RELEASE.
801501	<a href="#">214762</a>	3 ноября 2010	8.1-STABLE после изменения KBI в структуре <code>sysentvec</code> , а также реализации <code>PL_FLAG_SCE/SCX/EXEC/SI</code> и <code>pl_siginfo</code> для <code>ptrace(PT_LWPINFO)</code> .
802000	<a href="#">216639</a>	22 декабря 2010	8.2-RELEASE
802500	<a href="#">216654</a>	22 декабря 2010	8.2-STABLE после 8.2-RELEASE.
802501	<a href="#">219107</a>	28 февраля 2011	8.2-STABLE после объединения изменений DTrace, включая поддержку трассировки пользовательского пространства.

Значение	Версия	Дата	Релиз
802502	<a href="#">219324</a>	6 марта 2011	8.2-STABLE после объединения log2 и log2f в libm.
802503	<a href="#">221275</a>	1 мая 2011	8.2-STABLE после обновления gcc до последней версии GPLv2 из ветки FSF gcc-4_2-branch.
802504	<a href="#">222401</a>	28 мая 2011	8.2-STABLE после введения KPI и поддерживаемой инфраструктуры для модульного управления перегрузкой.
802505	<a href="#">222406</a>	28 мая 2011	8.2-STABLE после введения KPIs Nhook и Khelp.
802506	<a href="#">222408</a>	28 мая 2011	8.2-STABLE после добавления OSD в структуру tcrsb.
802507	<a href="#">222741</a>	6 июня 2011	8.2-STABLE после импорта ZFS v28.
802508	<a href="#">222846</a>	8 июня 2011	8.2-STABLE после удаления обработчика событий <code>schedtail</code> и добавления метода <code>sv_schedtail</code> в структуру <code>sysvec</code> .
802509	<a href="#">224017</a>	14 июля 2011	8.2-STABLE после объединения поддержки SSSE3 в binutils.
802510	<a href="#">224214</a>	19 июля 2011	8.2-STABLE после добавления флага RFTSIGZMB для <code>rfork(2)</code> .

Значение	Версия	Дата	Релиз
802511	<a href="#">225458</a>	9 сентября 2011	8.2-STABLE после добавления автоматического обнаружения USB-накопителей, которые не поддерживают команду SCSI "no synchronize cache".
802512	<a href="#">225470</a>	10 сентября 2011	8.2-STABLE после объединения рефакторинга auto-quirk.
802513	<a href="#">226763</a>	25 октября 2011	8.2-STABLE после объединения флага MAP_PREFAULT_READ в <a href="#">mmap(2)</a> .
802514	<a href="#">227573</a>	16 ноября 2011	8.2-STABLE после объединения добавления системного вызова <a href="#">posix_fallocate(2)</a> .
802515	<a href="#">229725</a>	6 января 2012	8.2-STABLE после объединения добавления системного вызова <a href="#">posix_fadvise(2)</a> .
802516	<a href="#">230239</a>	16 января 2012	8.2-STABLE после объединения gperf 3.0.3
802517	<a href="#">231769</a>	15 февраля 2012	8.2-STABLE после введения нового расширяемого интерфейса <a href="#">sysctl(3)</a> NET_RT_IFLISTL для запроса списков адресов.
803000	<a href="#">232446</a>	3 марта 2012	8.3-RELEASE.
803500	<a href="#">232439</a>	3 марта 2012	8.3-STABLE после ветвления releng/8.3 (RELENG_8_3).

Значение	Версия	Дата	Релиз
803501	<a href="#">247091</a>	21 февраля 2013	8.3-STABLE после слияния двух исправлений для USB (ссылки на ревизии: <a href="#">246616</a> и <a href="#">246759</a> ).
804000	<a href="#">248850</a>	28 марта 2013	8.4-RELEASE.
804500	<a href="#">248819</a>	28 марта 2013	8.4-STABLE после 8.4-RELEASE.
804501	<a href="#">259449</a>	16 декабря 2013	8.4-STABLE после MFC исправления кодировки из вышестоящего Heimdal.
804502	<a href="#">265123</a>	30 апреля 2014	8.4-STABLE после FreeBSD-SA-14:08.tcp.
804503	<a href="#">268444</a>	9 июля 2014	8.4-STABLE после FreeBSD-SA-14:17.kmem.
804504	<a href="#">271341</a>	9 сентября 2014	8.4-STABLE после FreeBSD-SA-14:18 (ссылка на ревизию: <a href="https://svnweb.freebsd.org/changeset/base/271305">https://svnweb.freebsd.org/changeset/base/271305</a> [271305]).
804505	<a href="#">271686</a>	16 сентября 2014	8.4-STABLE после FreeBSD-SA-14:19 (ревизия <a href="#">271668</a> ).
804506	<a href="#">273432</a>	21 октября 2014	8.4-STABLE после FreeBSD-SA-14:21 (ссылка на ревизию: <a href="https://svnweb.freebsd.org/changeset/base/273413">https://svnweb.freebsd.org/changeset/base/273413</a> [273413]).
804507	<a href="#">274162</a>	4 ноября 2014	8.4-STABLE после FreeBSD-SA-14:23, FreeBSD-SA-14:24 и FreeBSD-SA-14:25.

Значение	Версия	Дата	Релиз
804508	<a href="#">279287</a>	25 февраля 2015	8-STABLE после FreeBSD-EN-15:01.vt, FreeBSD-EN-15:02.openssl, FreeBSD-EN-15:03.freebsd-update, FreeBSD-SA-15:04.igmp и FreeBSD-SA-15:05.bind.
804509	<a href="#">305736</a>	12 сентября 2016	8-STABLE после устранения взаимоблокировки между <code>device_detach()</code> и <code>usbdo_request_flags(9)</code> .

## 18.9. Версии FreeBSD 7

Таблица 61. Значения `__FreeBSD_version` для FreeBSD 7

Значение	Версия	Дата	Релиз
700000	<a href="#">147925</a>	11 июля 2005	7.0-CURRENT.
700001	<a href="#">148341</a>	23 июля 2005	7.0-CURRENT после увеличения версий всех общих библиотек, которые не изменялись с RELENG_5.
700002	<a href="#">149039</a>	13 августа 2005	7.0-CURRENT после добавления аргумента учетных данных в обработчик события <code>dev_clone</code> .
700003	<a href="#">149470</a>	25 августа 2005	7.0-CURRENT после добавления <code>memmem(3)</code> в <code>libc</code> .
700004	<a href="#">151888</a>	30 октября 2005	7.0-CURRENT после изменения аргументов ядра <code>solisten(9)</code> для принятия параметра <code>backlog</code> .

Значение	Версия	Дата	Релиз
700005	<a href="#">152296</a>	11 ноября 2005	7.0-CURRENT после изменения <code>IFP2ENADDR()</code> для возврата указателя на <code>IF_LLADDR()</code> .
700006	<a href="#">152315</a>	11 ноября 2005	7.0-CURRENT после добавления члена <code>if_addr</code> в <code>struct ifnet</code> и удаления <code>IFP2ENADDR()</code> .
700007	<a href="#">153027</a>	2 декабря 2005	7.0-CURRENT после включения скриптов из каталогов <code>local_startup</code> в базовый <code>rcorder(8)</code> .
700008	<a href="#">153107</a>	5 декабря 2005	7.0-CURRENT после удаления опции монтирования <code>MNT_NODEV</code> .
700009	<a href="#">153519</a>	19 декабря 2005	7.0-CURRENT после изменений типа ELF-64 и версионирования символов.
700010	<a href="#">153579</a>	20 декабря 2005	7.0-CURRENT после добавления драйверов <code>hostb</code> и <code>vgarci</code> , добавления <code>pci_find_extcap()</code> и изменения драйверов AGP, чтобы они больше не отображали апертуру.
700011	<a href="#">153936</a>	31 декабря 2005	7.0-CURRENT после того, как <code>tv_sec</code> был изменён на <code>time_t</code> на всех платформах, кроме Alpha.
700012	<a href="#">154114</a>	8 января 2006	7.0-CURRENT после изменения <code>ldconfig_local_dirs</code> .

Значение	Версия	Дата	Релиз
700013	<a href="#">154269</a>	12 января 2006	7.0-CURRENT после изменений в /etc/rc.d/abi для поддержки /compat/linux/etc/ld.so.cache в виде символьной ссылки в файловой системе только для чтения.
700014	<a href="#">154863</a>	26 января 2006	7.0-CURRENT после импорта pts.
700015	<a href="#">157144</a>	26 марта 2006	7.0-CURRENT после введения версии 2 ABI <a href="#">hwpmc(4)</a> .
700016	<a href="#">157962</a>	22 апреля 2006	7.0-CURRENT после добавления <a href="#">fcloseall(3)</a> в libc.
700017	<a href="#">158513</a>	13 мая 2006	7.0-CURRENT после удаления ip6fw.
700018	<a href="#">160386</a>	15 июля 2006	7.0-CURRENT после импорта snd_emu10kx.
700019	<a href="#">160821</a>	29 июля 2006	7.0-CURRENT после импорта OpenSSL 0.9.8b.
700020	<a href="#">161931</a>	3 сентября 2006	7.0-CURRENT после добавления функции <a href="#">bus_dma_get_tag</a>
700021	<a href="#">162023</a>	4 сентября 2006	7.0-CURRENT после импорта libpcap 0.9.4 и tcpdump 3.9.4.
700022	<a href="#">162170</a>	9 сентября 2006	7.0-CURRENT после изменения <a href="#">dlsym</a> , чтобы искать запрошенный символ как в указанном DSO, так и в его неявных зависимостях.
700023	<a href="#">162588</a>	23 сентября 2006	7.0-CURRENT после добавления новых звуковых IOCTL для API микшера OSSv4.

Значение	Версия	Дата	Релиз
700024	<a href="#">162919</a>	28 сентября 2006	7.0-CURRENT после импорта OpenSSL 0.9.8d.
700025	<a href="#">164190</a>	11 ноября 2006	7.0-CURRENT после добавления libelf.
700026	<a href="#">164614</a>	26 ноября 2006	7.0-CURRENT после значительных изменений в звуковых sysctls.
700027	<a href="#">164770</a>	30 ноября 2006	7.0-CURRENT после добавления особенности Wi-Spy.
700028	<a href="#">165242</a>	15 декабря 2006	7.0-CURRENT после добавления вызовов <code>sctp</code> в <code>libc</code>
700029	<a href="#">166259</a>	26 января 2007	7.0-CURRENT после того, как реализация GNU <code>gzip(1)</code> была заменена на версию с лицензией BSD, портированную из NetBSD.
700030	<a href="#">166549</a>	7 февраля 2007	7.0-CURRENT после удаления инкапсуляции туннеля IPIP (VIFF_TUNNEL) из кода переадресации IPv4 multicast.
700031	<a href="#">166907</a>	23 февраля 2007	7.0-CURRENT после изменения <code>bus_setup_intr()</code> (newbus).
700032	<a href="#">167165</a>	2 марта 2007	7.0-CURRENT после включения firmware для <code>ipw(4)</code> и <code>iwi(4)</code> .
700033	<a href="#">167360</a>	9 марта 2007	7.0-CURRENT после включения поддержки широких символов ncurses.

Значение	Версия	Дата	Релиз
700034	<a href="#">167684</a>	19 марта 2007	7.0-CURRENT после изменений в работе <code>insmntque()</code> , <code>getnewvnode()</code> и <code>vfs_hash_insert()</code> .
700035	<a href="#">167906</a>	26 марта 2007	7.0-CURRENT после добавления механизма уведомлений об изменениях частоты CPU.
700036	<a href="#">168413</a>	6 апреля 2007	7.0-CURRENT после импорта файловой системы ZFS.
700037	<a href="#">168504</a>	8 апреля 2007	7.0-CURRENT после добавления периферийного устройства CAM 'SG', реализующего подмножество API сквозного устройства SCSI SG в Linux.
700038	<a href="#">169151</a>	30 апреля 2007	7.0-CURRENT после изменения <code>getenv(3)</code> , <code>putenv(3)</code> , <code>setenv(3)</code> и <code>unsetenv(3)</code> для соответствия стандарту POSIX.
700039	<a href="#">169190</a>	1 мая 2007	7.0-CURRENT после отмены изменений в 700038.
700040	<a href="#">169453</a>	10 мая 2007	7.0-CURRENT после добавления <code>flopen(3)</code> в <code>libutil</code> .
700041	<a href="#">169526</a>	13 мая 2007	7.0-CURRENT после включения версионирования символов и изменения библиотеки потоков по умолчанию на <code>libthr</code> .

Значение	Версия	Дата	Релиз
700042	<a href="#">169758</a>	19 мая 2007	7.0-CURRENT после импорта gss 4.2.0.
700043	<a href="#">169830</a>	21 мая 2007	7.0-CURRENT после увеличения версий всех общих библиотек, которые не изменялись с RELENG_6.
700044	<a href="#">170395</a>	7 июня 2007	7.0-CURRENT после изменения аргумента для <code>vp_open()</code> / <code>VOP_OPEN()</code> с индекса файлового дескриптора на указатель на структуру <code>file *</code> .
700045	<a href="#">170510</a>	10 июня 2007	7.0-CURRENT после изменения <code>ram_nologin(8)</code> для предоставления функции управления учетными записями вместо функции аутентификации в рамках RAM.
700046	<a href="#">170530</a>	11 июня 2007	7.0-CURRENT после обновления поддержки беспроводных сетей 802.11.
700047	<a href="#">170579</a>	11 июня 2007	7.0-CURRENT после добавления возможностей интерфейса TCP LRO.

Значение	Версия	Дата	Релиз
700048	<a href="#">170613</a>	12 июня 2007	7.0-CURRENT после добавления поддержки API RFC 3678 в стек IPv4. Устаревшее поведение RFC 1724 для ioctl IP_MULTICAST_IF теперь удалено; 0.0.0.0/8 больше нельзя использовать для указания индекса интерфейса. Вместо этого используйте структуру <code>ipmreqn</code> .
700049	<a href="#">171175</a>	3 июля 2007	7.0-CURRENT после импорта pf из OpenBSD 4.1
(не изменено)	<a href="#">171167</a>		7.0-CURRENT после добавления поддержки IPv6 для FAST_IPSEC, удаления KAME IPSEC и переименования FAST_IPSEC в IPSEC.
700050	<a href="#">171195</a>	4 июля 2007	7.0-CURRENT после преобразования вызовов setenv/putenv/etc. из традиционного BSD в POSIX.
700051	<a href="#">171211</a>	4 июля 2007	7.0-CURRENT после добавления новых системных вызовов mmap/lseek и др.
700052	<a href="#">171275</a>	6 июля 2007	7.0-CURRENT после перемещения заголовков I4B в include/i4b.
700053	<a href="#">172394</a>	30 сентября 2007	7.0-CURRENT после добавления поддержки доменов PCI

Значение	Версия	Дата	Релиз
700054	<a href="#">172988</a>	25 октября 2007	7.0-STABLE после переноса изменений (MFC) разделения широких и однобайтовых стур.
700055	<a href="#">173104</a>	28 октября 2007	7.0-RELEASE и 7.0-CURRENT после того, как обратная совместимость ABI с версиями FreeBSD 4/5/6 для IOCTL PCIOGETCONF, PCIOCREAD и PCIOCWRITE была перенесена в стабильную ветку (MFC), что потребовало снова нарушить ABI IOCTL PCIOGETCONF
700100	<a href="#">174864</a>	22 декабря 2007	7.0-STABLE после 7.0-RELEASE
700101	<a href="#">176111</a>	8 февраля 2008	7.0-STABLE после MFC <a href="#">m_collapse()</a> .
700102	<a href="#">177735</a>	30 марта 2008	7.0-STABLE после MFC <a href="#">kdb_enter_why()</a> .
700103	<a href="#">178061</a>	10 апреля 2008	7.0-STABLE после добавления <a href="#">l_sysid</a> в структуру flock.
700104	<a href="#">178108</a>	11 апреля 2008	7.0-STABLE после MFC <a href="#">procstat(1)</a> .
700105	<a href="#">178120</a>	11 апреля 2008	7.0-STABLE после MFC функций <a href="#">umtx</a> .
700106	<a href="#">178225</a>	15 апреля 2008	7.0-STABLE после MFC поддержки <a href="#">write(2)</a> в <a href="#">psm(4)</a> .
700107	<a href="#">178353</a>	20 апреля 2008	7.0-STABLE после MFC команды F_DUP2FD в <a href="#">fcntl(2)</a> .

Значение	Версия	Дата	Релиз
700108	<a href="#">178783</a>	5 мая 2008	7.0-STABLE после некоторых изменений в <a href="#">lockmgr(9)</a> , что делает необходимым включение <code>sys/lock.h</code> для использования <a href="#">lockmgr(9)</a> .
700109	<a href="#">179367</a>	27 мая 2008	7.0-STABLE после MFC функции <a href="#">memrchr(3)</a> .
700110	<a href="#">181328</a>	5 августа 2008	7.0-STABLE после MFC клиента <code>lockd</code> ядра NFS.
700111	<a href="#">181940</a>	20 августа 2008	7.0-STABLE после добавления поддержки физически непрерывных больших кадров (jumbo frame).
700112	<a href="#">182294</a>	27 августа 2008	7.0-STABLE после переноса изменений (MFC) поддержки DTrace в ядре.
701000	<a href="#">185315</a>	25 ноября 2008	7.1-RELEASE
701100	<a href="#">185302</a>	25 ноября 2008	7.1-STABLE после 7.1-RELEASE.
701101	<a href="#">187023</a>	10 января 2009	7.1-STABLE после слияния <a href="#">strndup(3)</a> .
701102	<a href="#">187370</a>	17 января 2009	7.1-STABLE после добавления поддержки <a href="#">cpuctl(4)</a> .
701103	<a href="#">188281</a>	7 февраля 2009	7.1-STABLE после объединения клеток с поддержкой multi-/no-IPv4/v6.

Значение	Версия	Дата	Релиз
701104	<a href="#">188625</a>	14 февраля 2009	7.1-STABLE после сохранения владельца приостановки в структуре mount и добавления метода <code>vfs_susp_clean</code> в структуру <code>vfsops</code> .
701105	<a href="#">189740</a>	12 марта 2009	7.1-STABLE после несовместимого изменения <code>sysctl kern.ipc.shmsegs</code> для выделения больших сегментов разделяемой памяти SysV на 64-битных архитектурах.
701106	<a href="#">189786</a>	14 марта 2009	7.1-STABLE после объединения исправления для операций ожидания семафоров POSIX.
702000	<a href="#">191099</a>	15 апреля 2009	7.2-RELEASE
702100	<a href="#">191091</a>	15 апреля 2009	7.2-STABLE после 7.2-RELEASE.
702101	<a href="#">192149</a>	15 мая 2009	7.2-STABLE после изменения <a href="#">ichsmb(4)</a> для использования выравнивания по левому краю вторичной адресации, чтобы соответствовать другим драйверам контроллеров SMBus.
702102	<a href="#">193020</a>	28 мая 2009	7.2-STABLE после слияния из ветки <code>man</code> функции <code>fdopendir(3)</code> .
702103	<a href="#">193638</a>	6 июня 2009	7.2-STABLE после MFC <code>PmcTools</code> .
702104	<a href="#">195694</a>	14 июля 2009	7.2-STABLE после MFC системного вызова <a href="#">closefrom(2)</a> .

Значение	Версия	Дата	Релиз
702105	<a href="#">196006</a>	31 июля 2009	7.2-STABLE после слияния изменения ABI SYSVIPC.
702106	<a href="#">197198</a>	14 сентября 2009	7.2-STABLE после слияния изменений (MFC) улучшений PAT для x86 и добавления <code>d_mmap_single()</code> и типа объекта VM со списком scatter/gather.
703000	<a href="#">203740</a>	9 февраля 2010	7.3-RELEASE
703100	<a href="#">203742</a>	9 февраля 2010	7.3-STABLE после 7.3-RELEASE.
704000	<a href="#">216647</a>	22 декабря 2010	7.4-RELEASE
704100	<a href="#">216658</a>	22 декабря 2010	7.4-STABLE после 7.4-RELEASE.
704101	<a href="#">221318</a>	2 мая 2011	7.4-STABLE после MFC gcc в ревизии <a href="#">221317</a> .

## 18.10. Версии FreeBSD 6

Таблица 62. Значения `__FreeBSD_version` в FreeBSD 6

Значение	Версия	Дата	Релиз
600000	<a href="#">133921</a>	18 августа 2004	6.0-CURRENT
600001	<a href="#">134396</a>	27 августа 2004	6.0-CURRENT после постоянного включения PFIL_HOOKS в ядре.
600002	<a href="#">134514</a>	30 августа 2004	6.0-CURRENT после первоначального добавления <code>ifi_epoch</code> в структуру <code>if_data</code> . Отменено через несколько дней. Не используйте это значение.

Значение	Версия	Дата	Релиз
600003	<a href="#">134933</a>	8 сентября 2004	6.0-CURRENT после повторного добавления члена <code>ifi_epoch</code> в структуру <code>if_data</code> .
600004	<a href="#">135920</a>	29 сентября 2004	6.0-CURRENT после добавления аргумента <code>struct inpcb</code> в API <code>pfil</code> .
600005	<a href="#">136172</a>	5 октября 2004	6.0-CURRENT после добавления аргумента <code>"-d DESTDIR"</code> в <code>newsyslog</code> .
600006	<a href="#">137192</a>	4 ноября 2004	6.0-CURRENT после добавления опций заполнения в стиле <code>glibc</code> для <code>strftime(3)</code> .
600007	<a href="#">138760</a>	12 декабря 2004	6.0-CURRENT после добавления обновлений для фреймворка 802.11.
600008	<a href="#">140809</a>	25 января 2005	6.0-CURRENT после изменений в функциях <code>VOP_*VOBJECT()</code> и введения флага <code>MNTK_MPSAFE</code> для файловых систем, работающих без Giant.
600009	<a href="#">141250</a>	4 февраля 2005	6.0-CURRENT после добавления фреймворка <code>crifreq</code> и драйверов.
600010	<a href="#">141394</a>	6 февраля 2005	6.0-CURRENT после импорта <code>nc(1)</code> из OpenBSD.
600011	<a href="#">141727</a>	12 февраля 2005	6.0-CURRENT после удаления подобия поддержки <code>matherr()</code> из SVID2.

Значение	Версия	Дата	Релиз
600012	<a href="#">141940</a>	15 февраля 2005	6.0-CURRENT после увеличения размера стеков потоков по умолчанию.
600013	<a href="#">142089</a>	19 февраля 2005	6.0-CURRENT после исправлений в <code>&lt;src/include/stdbool.h&gt;</code> и <code>&lt;src/sys/i386/include/_types.h&gt;</code> для обеспечения совместимости с GCC компилятора Intel C/C++.
600014	<a href="#">142184</a>	21 февраля 2005	6.0-CURRENT после исправления проверок EOVERFLOW в <code>vswprintf(3)</code> .
600015	<a href="#">142501</a>	25 февраля 2005	6.0-CURRENT после изменения члена структуры <code>if_data</code> , <code>ifi_epoch</code> , с времени настенных часов на время работы системы.
600016	<a href="#">142582</a>	26 февраля 2005	6.0-CURRENT после изменения формата диска LC_STYPE.
600017	<a href="#">142683</a>	27 февраля 2005	6.0-CURRENT после изменения формата диска каталогов NLS.
600018	<a href="#">142686</a>	27 февраля 2005	6.0-CURRENT после изменения формата диска LC_COLLATE.
600019	<a href="#">142752</a>	28 февраля 2005	Установка <code>asprisa</code> включает файлы в <code>/usr/include</code> .
600020	<a href="#">143308</a>	9 марта 2005	Добавление флага MSG_NOSIGNAL в API <code>send(2)</code> .
600021	<a href="#">143746</a>	17 марта 2005	Добавление полей в <code>cdevsw</code>

Значение	Версия	Дата	Релиз
600022	<a href="#">143901</a>	21 марта 2005	Удален gtar из базовой системы.
600023	<a href="#">144980</a>	13 апреля 2005	Добавлены параметры сокета LOCAL_CREDS, LOCAL_CONNWAIT в <a href="#">unix(4)</a> .
600024	<a href="#">145565</a>	19 апреля 2005	<a href="#">hwpmc(4)</a> и связанные инструменты добавлены в 6.0-CURRENT.
600025	<a href="#">145565</a>	26 апреля 2005	Структура <a href="#">icmphdr</a> добавлена в 6.0-CURRENT.
600026	<a href="#">145843</a>	3 мая 2005	pf обновлен до версии 3.7.
600027	<a href="#">145966</a>	6 мая 2005	Добавлены libalias в ядре и <a href="#">ng_nat</a> .
600028	<a href="#">146191</a>	13 мая 2005	POSIX <a href="#">ttyname_r(3)</a> , доступный через unistd.h и libc.
600029	<a href="#">146780</a>	29 мая 2005	6.0-CURRENT после обновления librcar до v0.9.1 alpha 096.
600030	<a href="#">146988</a>	5 июня 2005	6.0-CURRENT после импорта <a href="#">if_bridge(4)</a> из NetBSD.
600031	<a href="#">147256</a>	10 июня 2005	6.0-CURRENT после того, как структура ifnet была вынесена из <a href="#">softcs</a> драйвера.
600032	<a href="#">147898</a>	11 июля 2005	6.0-CURRENT после импорта librcar v0.9.1.
600033	<a href="#">148388</a>	25 июля 2005	6.0-STABLE после увеличения версий всех общих библиотек, которые не изменялись с RELENG_5.

Значение	Версия	Дата	Релиз
600034	<a href="#">149040</a>	13 августа 2005	6.0-STABLE после добавления аргумента <code>credential</code> в обработчик события <code>dev_clone</code> . 6.0-RELEASE.
600100	<a href="#">151958</a>	1 ноября 2005	6.0-STABLE после 6.0-RELEASE
600101	<a href="#">153601</a>	21 декабря 2005	6.0-STABLE после включения скриптов из каталогов <code>local_startup</code> в базовый <code>rcorder(8)</code> .
600102	<a href="#">153912</a>	30 декабря 2005	6.0-STABLE после обновления типов и констант ELF.
600103	<a href="#">154396</a>	15 января 2006	6.0-STABLE после переноса изменений (MFC) API <code>pidfile(3)</code> .
600104	<a href="#">154453</a>	17 января 2006	6.0-STABLE после MFC изменений <code>ldconfig_local_dirs</code> .
600105	<a href="#">156019</a>	26 февраля 2006	6.0-STABLE после поддержки каталога NLS в <code>csch(1)</code> .
601000	<a href="#">158330</a>	6 мая 2006	6.1-RELEASE
601100	<a href="#">158331</a>	6 мая 2006	6.1-STABLE после 6.1-RELEASE.
601101	<a href="#">159861</a>	22 июня 2006	6.1-STABLE после импорта <code>csup</code> .
601102	<a href="#">160253</a>	11 июля 2006	6.1-STABLE после обновления <code>iwi(4)</code> .
601103	<a href="#">160429</a>	17 июля 2006	6.1-STABLE после обновления резолвера до BIND9 и добавления реентерабельной версии функций <code>netdb</code> .

Значение	Версия	Дата	Релиз
601104	<a href="#">161098</a>	8 августа 2006	6.1-STABLE после включения поддержки DSO (динамически разделяемых объектов) в OpenSSL.
601105	<a href="#">161900</a>	2 сентября 2006	6.1-STABLE после исправлений 802.11 изменил API для ioctl IEEE80211_IOC_STA_INFO.
602000	<a href="#">164312</a>	15 ноября 2006	6.2-RELEASE
602100	<a href="#">162329</a>	15 сентября 2006	6.2-STABLE после 6.2-RELEASE.
602101	<a href="#">165122</a>	12 декабря 2006	6.2-STABLE после добавления особенности Wi-Spy.
602102	<a href="#">165596</a>	28 декабря 2006	6.2-STABLE после добавления <code>pci_find_extcap()</code> .
602103	<a href="#">166039</a>	16 января 2007	6.2-STABLE после MFC изменения <code>dlsym</code> для поиска запрошенного символа как в указанном DSO, так и в его неявных зависимостях.
602104	<a href="#">166314</a>	28 января 2007	6.2-STABLE после слияния изменений (MFC) узлов <code>netgraph</code> <code>ng_deflate(4)</code> и <code>ng_pred1(4)</code> , а также новых режимов сжатия и шифрования для узла <code>ng_ppp(4)</code> .
602105	<a href="#">166840</a>	20 февраля 2007	6.2-STABLE после переноса (MFC) версии <code>gzip(1)</code> под лицензией BSD из NetBSD.

Значение	Версия	Дата	Релиз
602106	<a href="#">168133</a>	31 марта 2007	6.2-STABLE после слияния изменений (MFC) поддержки PCI MSI и MSI-X.
602107	<a href="#">168438</a>	6 апреля 2007	6.2-STABLE после слияния изменений (MFC) ncurses 5.6 с поддержкой широких символов.
602108	<a href="#">168611</a>	11 апреля 2007	6.2-STABLE после слияния изменений (MFC) для периферийного устройства CAM 'SG', реализующего подмножество API сквозного устройства SCSI SG в Linux.
602109	<a href="#">168805</a>	17 апреля 2007	6.2-STABLE после MFC набора исправлений readline 5.2 patch-set 002.
602110	<a href="#">169222</a>	2 мая 2007	6.2-STABLE после слияния изменений (MFC) функций <code>mpar_invalidate_cache()</code> , <code>mpar_change_attr()</code> , <code>mpar_marpbios()</code> , <code>mpar_mapdev_attr()</code> и <code>mpar_unmarpbios()</code> для архитектур amd64 и i386.
602111	<a href="#">170556</a>	11 июня 2007	6.2-STABLE после слияния изменений VOP_BDFLUSH, что привело к нарушению KVI модулей файловой системы.
602112	<a href="#">172284</a>	21 сентября 2007	6.2-STABLE после <a href="#">libutil(3)</a> MFC's.

Значение	Версия	Дата	Релиз
602113	<a href="#">172986</a>	25 октября 2007	6.2-STABLE после слияния изменений (MFC) разделения широких и однобайтовых символов стуре. Вновь скомпилированные двоичные файлы, ссылающиеся на стуре.h, могут требовать новый символ <code>__mb_sb_limit</code> , который недоступен в старых системах.
602114	<a href="#">173170</a>	30 октября 2007	6.2-STABLE после восстановления прямой совместимости ABI стуре.
602115	<a href="#">173794</a>	21 ноября 2007	6.2-STABLE после отмены разделения широких и однобайтовых символов стуре.
603000	<a href="#">173897</a>	25 ноября 2007	6.3-RELEASE
603100	<a href="#">173891</a>	25 ноября 2007	6.3-STABLE после 6.3-RELEASE.
(не изменено)	<a href="#">174434</a>	7 декабря 2007	6.3-STABLE после исправления поддержки поддержки многобайтовых типов в макросе <code>bit</code> .
603102	<a href="#">178459</a>	24 апреля 2008	6.3-STABLE после добавления <code>l_sysid</code> в структуру <code>flock</code> .
603103	<a href="#">179367</a>	27 мая 2008	6.3-STABLE после MFC функции <code>memrchr(3)</code> .
603104	<a href="#">179810</a>	15 июня 2008	6.3-STABLE после MFC поддержки модификатора переменной <code>:u</code> в <code>make(1)</code> .

Значение	Версия	Дата	Релиз
604000	183583	4 октября 2008	6.4-RELEASE
604100	183584	4 октября 2008	6.4-STABLE после 6.4-RELEASE.

## 18.11. Версии FreeBSD 5

Таблица 63. Значения `__FreeBSD_version` в FreeBSD 5

Значение	Версия	Дата	Релиз
500000	58009	13 марта 2000	5.0-CURRENT
500001	59348	18 апреля 2000	5.0-CURRENT после добавления дополнительных полей заголовка ELF и изменения метода маркировки ELF-бинарников.
500002	59906	2 мая 2000	5.0-CURRENT после изменений метаданных kld.
500003	60688	18 мая 2000	5.0-CURRENT после изменений в buf/bio.
500004	60936	26 мая 2000	5.0-CURRENT после обновления binutils.
500005	61221	3 июня 2000	5.0-CURRENT после объединения кода libxprg4 с libc и после введения интерфейса TASKQ.
500006	61500	10 июня 2000	5.0-CURRENT после добавления интерфейсов AGP.
500007	62235	29 июня 2000	5.0-CURRENT после обновления Perl до версии 5.6.0
500008	62764	7 июля 2000	5.0-CURRENT после обновления кода KAME до исходников от 2000/07.

Значение	Версия	Дата	Релиз
500009	<a href="#">63154</a>	14 июля 2000	5.0-CURRENT после изменений в <code>ether_ifattach()</code> и <code>ether_ifdetach()</code> .
500010	<a href="#">63265</a>	16 июля 2000	5.0-CURRENT после изменения настроек <code>mtree</code> обратно на исходный вариант, с добавлением <code>-L</code> для следования по символьным ссылкам.
500011	<a href="#">63459</a>	18 июля 2000	5.0-CURRENT после изменения API <code>kqueue</code> .
500012	<a href="#">65353</a>	2 сентября 2000	5.0-CURRENT после переноса <code>setproctitle(3)</code> из <code>libutil</code> в <code>libc</code> .
500013	<a href="#">65671</a>	10 сентября 2000	5.0-CURRENT после первого коммита <code>SMPng</code> .
500014	<a href="#">70650</a>	4 января 2001	5.0-CURRENT после перемещения <code>&lt;sys/select.h&gt;</code> в <code>&lt;sys/selinfo.h&gt;</code> .
500015	<a href="#">70894</a>	10 января 2001	5.0-CURRENT после объединения <code>libgcc.a</code> и <code>libgcc_r.a</code> , а также связанных изменений в компоновке <code>GCC</code> .
500016	<a href="#">71583</a>	24 января 2001	5.0-CURRENT после изменения, разрешающего совместную линковку <code>libc</code> и <code>libc_r</code> , с объявлением устаревшим параметра <code>-pthread</code> .

Значение	Версия	Дата	Релиз
500017	<a href="#">72650</a>	18 февраля 2001	5.0-CURRENT после перехода со структуры <b>uc</b> red на структуру <b>xu</b> cred для стабилизации API, экспортируемого ядром, для mountd и других.
500018	<a href="#">72975</a>	24 февраля 2001	5.0-CURRENT после добавления переменной сборки CRUTYPE для управления оптимизациями под конкретный процессор.
500019	<a href="#">77937</a>	9 июня 2001	5.0-CURRENT после перемещения machine/ioctl_fd.h в sys/fdcio.h
500020	<a href="#">78304</a>	15 июня 2001	5.0-CURRENT после переименования названий локалей.
500021	<a href="#">78632</a>	22 июня 2001	5.0-CURRENT после импорта Vzip2. Также означает удаление S/Key.
500022	<a href="#">83435</a>	12 июля 2001	5.0-CURRENT после поддержки SSE.
500023	<a href="#">83435</a>	14 сентября 2001	5.0-CURRENT после второго этапа KSE.
500024	<a href="#">84324</a>	1 октября 2001	5.0-CURRENT после <b>d_thread_t</b> и перемещение UUCP в порты.
500025	<a href="#">84481</a>	4 октября 2001	5.0-CURRENT после изменения ABI для передачи дескрипторов и <b>creds</b> на 64-битных платформах.

Значение	Версия	Дата	Релиз
500026	<a href="#">84710</a>	9 октября 2001	5.0-CURRENT после перехода на XFree86 4 по умолчанию для сборки пакетов и после добавления новой функции <code>strnstr()</code> в библиотеку <code>libc</code> .
500027	<a href="#">84743</a>	10 октября 2001	5.0-CURRENT после добавления новой функции <code>strcasestr()</code> в библиотеку <code>libc</code> .
500028	<a href="#">87879</a>	14 декабря 2001	5.0-CURRENT после импорта компонентов пользовательского пространства <code>smbfs</code> .
(не изменено)			5.0-CURRENT после добавления новых целочисленных типов фиксированной ширины C99.
500029	<a href="#">89938</a>	29 января 2002	5.0-CURRENT после изменения возвращаемого значения <code>sendfile(2)</code> .
500030	<a href="#">90711</a>	15 февраля 2002	5.0-CURRENT после введения типа <code>fflags_t</code> , который имеет подходящий размер для флагов файлов.
500031	<a href="#">91203</a>	24 февраля 2002	5.0-CURRENT после переименования элемента структуры <code>usb</code> .
500032	<a href="#">92453</a>	16 марта 2002	5.0-CURRENT после внедрения Perl 5.6.1.

Значение	Версия	Дата	Релиз
500033	<a href="#">93722</a>	3 апреля 2002	5.0-CURRENT после того, как переменная <code>sendmail_enable</code> из <code>rc.conf(5)</code> стала принимать значение <code>NONE</code> .
500034	<a href="#">95831</a>	30 апреля 2002	5.0-CURRENT после того, как <code>mtx_init()</code> получил третий аргумент.
500035	<a href="#">96498</a>	13 мая 2002	5.0-CURRENT с Gcc 3.1.
500036	<a href="#">96781</a>	17 мая 2002	5.0-CURRENT без Perl в <code>/usr/src</code>
500037	<a href="#">97516</a>	29 мая 2002	5.0-CURRENT после добавления <code>dlfunc(3)</code>
500038	<a href="#">100591</a>	24 июля 2002	5.0-CURRENT после изменения типов некоторых членов структуры <code>sockbuf</code> и её переупорядочивания.
500039	<a href="#">102757</a>	1 сентября 2002	5.0-CURRENT после импорта GCC 3.2.1. Также после того, как заголовки перестали использовать <code>BSD_FOO_T</code> и начали использовать <code>_FOO_T_DECLARED</code> . Это значение также можно использовать как консервативную оценку начала поддержки пакета <code>bzip2(1)</code> .

Значение	Версия	Дата	Релиз
500040	<a href="#">103675</a>	20 сентября 2002	5.0-CURRENT после внесения различных изменений в функции работы с дисками, направленных на устранение зависимости от внутренней структуры <code>disklabel</code> .
500041	<a href="#">104250</a>	1 октября 2002	5.0-CURRENT после добавления <a href="#"><code>getopt_long(3)</code></a> в <code>libc</code> .
500042	<a href="#">105178</a>	15 октября 2002	5.0-CURRENT после обновления <code>Binutils 2.13</code> , которое включило новую эмуляцию FreeBSD, <code>vec</code> и формат вывода.
500043	<a href="#">106289</a>	1 ноября 2002	5.0-CURRENT после добавления слабых заглушек <code>pthread_XXX</code> в <code>libc</code> , что сделало устаревшей <code>libXThrStub.so</code> . 5.0-RELEASE.
500100	<a href="#">109405</a>	17 января 2003	5.0-CURRENT после ветвления для <code>RELENG_5_0</code>
500101	<a href="#">111120</a>	19 февраля 2003	<code>&lt;sys/dkstat.h&gt;</code> пустой. Не включайте его.
500102	<a href="#">111482</a>	25 февраля 2003	5.0-CURRENT после изменения интерфейса <code>d_mmap_t</code> .
500103	<a href="#">111540</a>	26 февраля 2003	5.0-CURRENT после изменения <code>taskqueue_sw</code> для работы без <code>Giant</code> и добавления <code>taskqueue_sw_giant</code> для работы с <code>Giant</code> .

Значение	Версия	Дата	Релиз
500104	<a href="#">111600</a>	27 февраля 2003	<code>cdevsw_add()</code> и <code>cdevsw_remove()</code> больше не существуют. Появление средства выделения <code>MAJOR_AUTO</code> .
500105	<a href="#">111864</a>	4 марта 2003	5.0-CURRENT после новой инициализации метода <code>cdevsw</code> .
500106	<a href="#">112007</a>	8 марта 2003	<code>devstat_add_entry()</code> был заменён на <code>devstat_new_entry()</code>
500107	<a href="#">112288</a>	15 марта 2003	Изменение интерфейса <code>devstat</code> ; см. <code>sys/sys/param.h</code> 1.149
500108	<a href="#">112300</a>	15 марта 2003	Изменения в интерфейсе Token-Ring.
500109	<a href="#">112571</a>	25 марта 2003	Добавление <code>vm_paddr_t</code> .
500110	<a href="#">112741</a>	28 марта 2003	5.0-CURRENT после того, как <code>realpath(3)</code> стал потокобезопасным
500111	<a href="#">113273</a>	9 апреля 2003	5.0-CURRENT после синхронизации <code>usbhid(3)</code> с NetBSD
500112	<a href="#">113597</a>	17 апреля 2003	5.0-CURRENT после новой реализации NSS и добавления функций POSIX.1 <code>getpw*_r</code> , <code>getgr*_r</code>
500113	<a href="#">114492</a>	2 мая 2003	5.0-CURRENT после удаления старой системы <code>rs</code> .
501000	<a href="#">115816</a>	4 июня 2003	5.1-RELEASE.
501100	<a href="#">115710</a>	2 июня 2003	5.1-CURRENT после ветвления для <code>RELENG_5_1</code> .

Значение	Версия	Дата	Релиз
501101	<a href="#">117025</a>	29 июня 2003	5.1-CURRENT после исправления семантики <a href="#">sigtimedwait(2)</a> и <a href="#">sigwaitinfo(2)</a> .
501102	<a href="#">117191</a>	3 июля 2003	5.1-CURRENT после добавления полей <a href="#">lockfunc</a> и <a href="#">lockfuncarg</a> в <a href="#">bus_dma_tag_create(9)</a> .
501103	<a href="#">118241</a>	31 июля 2003	5.1-CURRENT после интеграции снимка GCC 3.3.1-pre 20030711.
501104	<a href="#">118511</a>	5 августа 2003	5.1-CURRENT Изменения API Zware в twe.
501105	<a href="#">119021</a>	17 августа 2003	5.1-CURRENT динамически связанные /bin и /sbin поддержка и перемещение библиотек в /lib.
501106	<a href="#">119881</a>	8 сентября 2003	5.1-CURRENT после добавления поддержки ядра для Coda 6.x.
501107	<a href="#">120180</a>	17 сентября 2003	5.1-CURRENT после того, как константы UART 16550 были перемещены из <dev/sio/sioreg.h> в <dev/ic/ns16550.h>. Также когда функциональность libmap стала безусловно поддерживаться rtd.
501108	<a href="#">120386</a>	23 сентября 2003	5.1-CURRENT после обновления API PFIL_HOOKS
501109	<a href="#">120503</a>	27 сентября 2003	5.1-CURRENT после добавления <a href="#">kiconv(3)</a>

Значение	Версия	Дата	Релиз
501110	<a href="#">120556</a>	28 сентября 2003	5.1-CURRENT после изменения операций по умолчанию для open и close в cdevsw
501111	<a href="#">121125</a>	16 октября 2003	5.1-CURRENT после изменения структуры cdevsw
501112	<a href="#">121129</a>	16 октября 2003	5.1-CURRENT после добавления множественного наследования kobj
501113	<a href="#">121816</a>	31 октября 2003	5.1-CURRENT после изменения if_xname в структуре ifnet
501114	<a href="#">122779</a>	16 ноября 2003	5.1-CURRENT после изменения /bin и /sbin на динамически линкуемые
502000	<a href="#">123198</a>	7 декабря 2003	5.2-RELEASE
502010	<a href="#">126150</a>	23 февраля 2004	5.2.1-RELEASE
502100	<a href="#">123196</a>	7 декабря 2003	5.2-CURRENT после ветвления для RELENG_5_2
502101	<a href="#">123677</a>	19 декабря 2003	5.2-CURRENT после добавления функций cxa_atexit/cxa_finalize в libc.
502102	<a href="#">125236</a>	30 января 2004	5.2-CURRENT после изменения стандартной библиотеки потоков с libc_r на libpthread.
502103	<a href="#">126083</a>	21 февраля 2004	5.2-CURRENT после масштабного патча API драйверов устройств.
502104	<a href="#">126208</a>	25 февраля 2004	5.2-CURRENT после добавления getopt_long_only().

Значение	Версия	Дата	Релиз
502105	<a href="#">126644</a>	5 марта 2004	5.2-CURRENT после того, как NULL заменён на ((void *)0) для C, что вызывает больше предупреждений.
502106	<a href="#">126757</a>	8 марта 2004	5.2-CURRENT после подключения rf к сборке и установке.
502107	<a href="#">126819</a>	10 марта 2004	5.2-CURRENT после изменения <code>time_t</code> на 64-битное значение на <code>sparc64</code> .
502108	<a href="#">126891</a>	12 марта 2004	5.2-CURRENT после поддержки компилятора Intel C/C++ в некоторых заголовочных файлах и изменений в <a href="#">execve(2)</a> для более строгого соответствия POSIX.
502109	<a href="#">127312</a>	22 марта 2004	5.2-CURRENT после введения API <code>bus_alloc_resource_any</code>
502110	<a href="#">127475</a>	27 марта 2004	5.2-CURRENT после добавления локалей UTF-8
502111	<a href="#">128144</a>	11 апреля 2004	5.2-CURRENT после удаления API <a href="#">getvfsent(3)</a>
502112	<a href="#">128182</a>	13 апреля 2004	5.2-CURRENT после добавления директивы <code>.warning</code> для <code>make</code> .
502113	<a href="#">130057</a>	4 июня 2004	5.2-CURRENT после того, как <code>ttyioctl()</code> стал обязательным для драйверов последовательных портов.

Значение	Версия	Дата	Релиз
502114	<a href="#">130418</a>	13 июня 2004	5.2-CURRENT после импорта инфраструктуры ALTQ.
502115	<a href="#">130481</a>	14 июня 2004	5.2-CURRENT после изменения <a href="#">sema_timedwait(9)</a> для возврата 0 при успехе и ненулевого кода ошибки при сбое.
502116	<a href="#">130585</a>	16 июня 2004	5.2-CURRENT после изменения типа <a href="#">dev_t</a> в ядре на указатель на структуру <a href="#">cdev *</a> .
502117	<a href="#">130640</a>	17 июня 2004	5.2-CURRENT после изменения ядра <a href="#">udev_t</a> на <a href="#">dev_t</a> .
502118	<a href="#">130656</a>	17 июня 2004	5.2-CURRENT после добавления поддержки <a href="#">CLOCK_VIRTUAL</a> и <a href="#">CLOCK_PROF</a> в <a href="#">clock_gettime(2)</a> и <a href="#">clock_getres(2)</a> .
502119	<a href="#">130934</a>	22 июня 2004	5.2-CURRENT после изменения переработки клонирования сетевых интерфейсов.
502120	<a href="#">131429</a>	2 июля 2004	5.2-CURRENT после обновления инструментов пакетов до ревизии 20040629.
502121	<a href="#">131883</a>	9 июля 2004	5.2-CURRENT после пометки кода Bluetooth как не специфичного для i386.

Значение	Версия	Дата	Релиз
502122	<a href="#">131971</a>	11 июля 2004	5.2-CURRENT после внедрения фреймворка отладчика KDB, преобразования DDB в бэкенд и добавления бэкенда GDB.
502123	<a href="#">132025</a>	12 июля 2004	5.2-CURRENT после изменения, чтобы VFS_ROOT принимал аргумент struct thread, как и vflush. Структура <code>kinfo_proc</code> теперь содержит указатель на пользовательские данные. Переключение реализации X по умолчанию на <code>xorg</code> также произошло в это время.
502124	<a href="#">132597</a>	24 июля 2004	5.2-CURRENT после изменения, разделяющего способ запуска rc.d портов и устаревших скриптов.
502125	<a href="#">132726</a>	28 июля 2004	5.2-CURRENT после отмены предыдущего изменения.
502126	<a href="#">132914</a>	31 июля 2004	5.2-CURRENT после удаления <code>kmem_alloc_pageable()</code> и импорта gcc 3.4.2.
502127	<a href="#">132991</a>	2 августа 2004	5.2-CURRENT после изменения UMA API ядра для разрешения ошибок в ctors/inits.

Значение	Версия	Дата	Релиз
502128	<a href="#">133306</a>	8 августа 2004	5.2-CURRENT после изменения сигнатуры <code>vfs_mount</code> , а также глобальной замены <code>PRISON_ROOT</code> на <code>SUSER_ALLOWJAIL</code> для API <code>suser(9)</code> .
503000	<a href="#">134189</a>	23 августа 2004	5.3-BETA/RC до изменения API <code>pfil</code>
503001	<a href="#">135580</a>	22 сентября 2004	5.3-RELEASE
503100	<a href="#">136595</a>	16 октября 2004	5.3-STABLE после ветвления для <code>RELENG_5_3</code>
503101	<a href="#">138459</a>	3 декабря 2004	5.3-STABLE после добавления опций заполнения <code>strftime(3)</code> в стиле <code>glibc</code> .
503102	<a href="#">141788</a>	13 февраля 2005	5.3-STABLE после импорта <code>nc(1)</code> из OpenBSD MFC.
503103	<a href="#">142639</a>	27 февраля 2005	5.4-PRERELEASE после MFC исправлений в <code>&lt;src/include/stdbool.h&gt;</code> и <code>&lt;src/sys/i386/include/_types.h&gt;</code> для обеспечения совместимости с GCC компилятора Intel C/C++.
503104	<a href="#">142835</a>	28 февраля 2005	5.4-PRERELEASE после MFC изменения <code>ifi_epoch</code> с времени реального мира на время работы системы.
503105	<a href="#">143029</a>	2 марта 2005	5.4-PRERELEASE после переноса исправления проверки <code>E_OVERFLOW</code> в <code>vswprintf(3)</code> .

Значение	Версия	Дата	Релиз
504000	<a href="#">144575</a>	3 апреля 2005	5.4-RELEASE.
504100	<a href="#">144581</a>	3 апреля 2005	5.4-STABLE после ветвления для RELENG_5_4
504101	<a href="#">146105</a>	11 мая 2005	5.4-STABLE после увеличения размеров стеков потоков по умолчанию
504102	<a href="#">504101</a>	24 июня 2005	5.4-STABLE после добавления sha256
504103	<a href="#">150892</a>	3 октября 2005	5.4-STABLE после слияния изменений (MFC) if_bridge
504104	<a href="#">152370</a>	13 ноября 2005	5.4-STABLE после слияния изменений (MFC) bsdiff и portsnap
504105	<a href="#">154464</a>	17 января 2006	5.4-STABLE после MFC изменений ldconfig_local_dirs.
505000	<a href="#">158481</a>	12 мая 2006	5.5-RELEASE.
505100	<a href="#">158482</a>	12 мая 2006	5.5-STABLE после ветвления для RELENG_5_5

## 18.12. Версии FreeBSD 4

Таблица 64. Значения `__FreeBSD_version` в FreeBSD 4

Значение	Версия	Дата	Релиз
400000	<a href="#">43041</a>	22 января 1999	4.0-CURRENT после ветки 3.4
400001	<a href="#">44177</a>	20 февраля 1999	4.0-CURRENT после изменения в обработке динамического компоновщика
400002	<a href="#">44699</a>	13 марта 1999	4.0-CURRENT после изменения порядка конструкторов/деструкторов C++

Значение	Версия	Дата	Релиз
400003	<a href="#">45059</a>	27 марта 1999	4.0-CURRENT после функционирования <a href="#">dladdr(3)</a>
400004	<a href="#">45321</a>	5 апреля 1999	4.0-CURRENT после исправления ошибки динамического компоновщика <a href="#">__deregister_frame_info</a> (также 4.0-CURRENT после интеграции EGCS 1.1.2)
400005	<a href="#">46113</a>	27 апреля 1999	4.0-CURRENT после изменения API <a href="#">suser(9)</a> (также 4.0-CURRENT после newbus)
400006	<a href="#">47640</a>	31 мая 1999	4.0-CURRENT после изменения регистрации <code>cdevsw</code>
400007	<a href="#">47992</a>	17 июня 1999	4.0-CURRENT после добавления <code>so_cred</code> для учётных данных на уровне сокета
400008	<a href="#">48048</a>	20 июня 1999	4.0-CURRENT после добавления обёртки системного вызова <code>poll</code> в <code>libc_r</code>
400009	<a href="#">48936</a>	20 июля 1999	4.0-CURRENT после изменения типа <code>dev_t</code> ядра на указатель <code>struct specinfo</code>
400010	<a href="#">51649</a>	25 сентября 1999	4.0-CURRENT после исправления уязвимости в <a href="#">jail(2)</a>
400011	<a href="#">51791</a>	29 сентября 1999	4.0-CURRENT после изменения типа данных <code>sigset_t</code>
400012	<a href="#">53164</a>	15 ноября 1999	4.0-CURRENT после перехода на компилятор GCC 2.95.2

Значение	Версия	Дата	Релиз
400013	<a href="#">54123</a>	4 декабря 1999	4.0-CURRENT после добавления подключаемых обработчиков ioctl в режиме linux
400014	<a href="#">56216</a>	18 января 2000	4.0-CURRENT после импорта OpenSSL
400015	<a href="#">56700</a>	27 января 2000	4.0-CURRENT после изменения ABI C++ в GCC 2.95.2 с -fvtable -thunks на -fno-vtable -thunks по умолчанию
400016	<a href="#">57529</a>	27 февраля 2000	4.0-CURRENT после импорта OpenSSH
400017	<a href="#">58005</a>	13 марта 2000	4.0-RELEASE
400018	<a href="#">58170</a>	17 марта 2000	4.0-STABLE после 4.0-RELEASE
400019	<a href="#">60047</a>	5 мая 2000	4.0-STABLE после введения отложенных контрольных сумм.
400020	<a href="#">61262</a>	4 июня 2000	4.0-STABLE после объединения кода libxpg4 в libc.
400021	<a href="#">62820</a>	8 июля 2000	4.0-STABLE после обновления Binutils до 2.10.0, изменения маркировки ELF и tcsh в базовой системе.
410000	<a href="#">63095</a>	14 июля 2000	4.1-RELEASE
410001	<a href="#">64012</a>	29 июля 2000	4.1-STABLE после 4.1-RELEASE
410002	<a href="#">65962</a>	16 сентября 2000	4.1-STABLE после перемещения <a href="#">setproctitle(3)</a> из libutil в libc.
411000	<a href="#">66336</a>	25 сентября 2000	4.1.1-RELEASE
411001			4.1.1-STABLE после 4.1.1-RELEASE

Значение	Версия	Дата	Релиз
420000	<a href="#">68066</a>	31 октября 2000	4.2-RELEASE
420001	<a href="#">70895</a>	10 января 2001	4.2-STABLE после объединения libgcc.a и libgcc_r.a, а также связанных изменений в компоновке GCC.
430000	<a href="#">73800</a>	6 марта 2001	4.3-RELEASE
430001	<a href="#">76779</a>	18 мая 2001	4.3-STABLE после введения <code>wint_t</code> .
430002	<a href="#">80157</a>	22 июля 2001	4.3-STABLE после объединения API управления состоянием питания PCI.
440000	<a href="#">80923</a>	1 августа 2001	4.4-RELEASE
440001	<a href="#">85341</a>	23 октября 2001	4.4-STABLE после введения <code>d_thread_t</code> .
440002	<a href="#">86038</a>	4 ноября 2001	4.4-STABLE после изменений в структуре монтирования (затрагивает модули файловых систем klds).
440003	<a href="#">88130</a>	18 декабря 2001	4.4-STABLE после импорта компонентов пользовательского пространства smbfs.
450000	<a href="#">88271</a>	20 декабря 2001	4.5-RELEASE
450001	<a href="#">91203</a>	24 февраля 2002	4.5-STABLE после переименования элемента структуры <code>usb</code> .
450002	<a href="#">92151</a>	12 марта 2002	4.5-STABLE после изменений локали.
450003			(Никогда не создавался)

Значение	Версия	Дата	Релиз
450004	<a href="#">94840</a>	16 апреля 2002	4.5-STABLE после того, как переменная <code>sendmail_enable</code> из <code>rc.conf(5)</code> стала принимать значение <code>NONE</code> .
450005	<a href="#">95555</a>	27 апреля 2002	4.5-STABLE после перехода на XFree86 4 по умолчанию для сборки пакетов.
450006	<a href="#">95846</a>	1 мая 2002	4.5-STABLE после исправления фильтрации ассерт, чтобы он больше не был подвержен простой DoS-атаке.
460000	<a href="#">97923</a>	21 июня 2002	4.6-RELEASE
460001	<a href="#">98730</a>	21 июня 2002	4.6-STABLE <code>sendfile(2)</code> исправлен для соответствия документации, чтобы не учитывать отправленные заголовки в объеме данных, отправляемых из файла.
460002	<a href="#">100366</a>	19 июля 2002	4.6.2-RELEASE
460100	<a href="#">98857</a>	26 июня 2002	4.6-STABLE
460101	<a href="#">98880</a>	26 июня 2002	4.6-STABLE после MFC <code>sed -i</code> .
460102	<a href="#">102759</a>	1 сентября 2002	4.6-STABLE после MFC множества новых функций <code>pkg_install</code> из HEAD.
470000	<a href="#">104655</a>	8 октября 2002	4.7-RELEASE
470100	<a href="#">104717</a>	9 октября 2002	4.7-STABLE

Значение	Версия	Дата	Релиз
470101	<a href="#">106732</a>	10 ноября 2002	Начинать генерировать ссылки <code>std{in,out,err}p</code> вместо <code>F</code> . Это изменяет <code>std{in,out,err}</code> с выражения времени компиляции на выражение времени выполнения.
470102	<a href="#">109753</a>	23 января 2003	4.7-STABLE после MFC изменений <code>mbuf</code> для замены <code>m_aux</code> <code>mbuf</code> на <code>m_tag</code>
470103	<a href="#">110887</a>	14 февраля 2003	4.7-STABLE получает OpenSSL 0.9.7
480000	<a href="#">112852</a>	30 марта 2003	4.8-RELEASE
480100	<a href="#">113107</a>	5 апреля 2003	4.8-STABLE
480101	<a href="#">115232</a>	22 мая 2003	4.8-STABLE после того, как <code>realpath(3)</code> стал потокобезопасным
480102	<a href="#">118737</a>	10 августа 2003	4.8-STABLE Изменения API <code>3ware</code> в <code>twe</code> .
490000	<a href="#">121592</a>	27 октября 2003	4.9-RELEASE
490100	<a href="#">121593</a>	27 октября 2003	4.9-STABLE
490101	<a href="#">124264</a>	8 января 2004	4.9-STABLE после добавления <code>e_sid</code> в структуру <code>kinfo_eproc</code> .
490102	<a href="#">125417</a>	4 февраля 2004	4.9-STABLE после MFC функциональности <code>libmap</code> для <code>rtld</code> .
491000	<a href="#">129700</a>	25 мая 2004	4.10-RELEASE
491100	<a href="#">129918</a>	1 июня 2004	4.10-STABLE
491101	<a href="#">133506</a>	11 августа 2004	4.10-STABLE после слияния изменения из ревизии 20040629 пакета <code>tools</code>

Значение	Версия	Дата	Релиз
491102	<a href="#">137786</a>	16 ноября 2004	4.10-STABLE после исправления VM, связанного с обработкой размонтирования фиктивных страниц
492000	<a href="#">138960</a>	17 декабря 2004	4.11-RELEASE
492100	<a href="#">138959</a>	17 декабря 2004	4.11-STABLE
492101	<a href="#">157843</a>	18 апреля 2006	4.11-STABLE после добавления каталогов libdata/ldconfig в файлы mtree.

## 18.13. Версии FreeBSD 3

Таблица 65. Значения `__FreeBSD_version` для FreeBSD 3

Значение	Версия	Дата	Релиз
300000	<a href="#">22917</a>	19 февраля 1996	3.0-CURRENT до изменения <a href="#">mount(2)</a>
300001	<a href="#">36283</a>	24 сентября 1997	3.0-CURRENT после изменения <a href="#">mount(2)</a>
300002	<a href="#">36592</a>	2 июня 1998	3.0-CURRENT после изменения <a href="#">semctl(2)</a>
300003	<a href="#">36735</a>	7 июня 1998	3.0-CURRENT после изменений аргументов <code>ioctl</code>
300004	<a href="#">38768</a>	3 сентября 1998	3.0-CURRENT после преобразования в ELF
300005	<a href="#">40438</a>	16 октября 1998	3.0-RELEASE
300006	<a href="#">40445</a>	16 октября 1998	3.0-CURRENT после 3.0-RELEASE
300007	<a href="#">43042</a>	22 января 1999	3.0-STABLE после ветвления 3/4
310000	<a href="#">43807</a>	9 февраля 1999	3.1-RELEASE
310001	<a href="#">45060</a>	27 марта 1999	3.1-STABLE после 3.1-RELEASE
310002	<a href="#">45689</a>	14 апреля 1999	3.1-STABLE после изменения порядка конструкторов/деструкторов C++

Значение	Версия	Дата	Релиз
320000			3.2-RELEASE
320001	<a href="#">46742</a>	8 мая 1999	3.2-STABLE
320002	<a href="#">50563</a>	29 августа 1999	3.2-STABLE после бинарно-несовместимых изменений в IPFW и сокетах
330000	<a href="#">50813</a>	2 сентября 1999	3.3-RELEASE
330001	<a href="#">51328</a>	16 сентября 1999	3.3-STABLE
330002	<a href="#">53671</a>	24 ноября 1999	3.3-STABLE после добавления <a href="#">mkstemp(3)</a> в libc
340000	<a href="#">54166</a>	5 декабря 1999	3.4-RELEASE
340001	<a href="#">54730</a>	17 декабря 1999	3.4-STABLE
350000	<a href="#">61876</a>	20 июня 2000	3.5-RELEASE
350001	<a href="#">63043</a>	12 июля 2000	3.5-STABLE

## 18.14. Версии FreeBSD 2.2

Таблица 66. Значения `__FreeBSD_version` в FreeBSD 2.2

Значение	Версия	Дата	Релиз
220000	<a href="#">22918</a>	19 февраля 1997	2.2-RELEASE
(не изменено)			2.2.1-RELEASE
(не изменено)			2.2-STABLE после 2.2.1-RELEASE
221001	<a href="#">24941</a>	15 апреля 1997	2.2-STABLE после texinfo-3.9
221002	<a href="#">25325</a>	30 апреля 1997	2.2-STABLE после обновления
222000	<a href="#">25851</a>	16 мая 1997	2.2.2-RELEASE
222001	<a href="#">25921</a>	19 мая 1997	2.2-STABLE после 2.2.2-RELEASE
225000	<a href="#">30053</a>	2 октября 1997	2.2.5-RELEASE
225001	<a href="#">31300</a>	20 ноября 1997	2.2-STABLE после 2.2.5-RELEASE
225002	<a href="#">32019</a>	27 декабря 1997	2.2-STABLE после слияния ldconfig -R

Значение	Версия	Дата	Релиз
226000	<a href="#">34445</a>	24 марта 1998	2.2.6-RELEASE
227000	<a href="#">37803</a>	21 июля 1998	2.2.7-RELEASE
227001	<a href="#">37809</a>	21 июля 1998	2.2-STABLE после 2.2.7-RELEASE
227002	<a href="#">39489</a>	19 сентября 1998	2.2-STABLE после изменения <a href="#">semctl(2)</a>
228000	<a href="#">41403</a>	29 ноября 1998	2.2.8-RELEASE
228001	<a href="#">41418</a>	29 ноября 1998	2.2-STABLE после 2.2.8-RELEASE



Обратите внимание, что 2.2-STABLE иногда идентифицирует себя как "2.2.5-STABLE" после выпуска 2.2.5-RELEASE. Ранее использовался шаблон "год-месяц", но сообщество решило изменить его на более простую систему "основной/второстепенный", начиная с версии 2.2. Это связано с тем, что параллельная разработка нескольких веток сделала невозможным классифицировать выпуски только по датам их фактического выхода. Не беспокойтесь о старых версиях -CURRENT; они приведены здесь только для справки.

## 18.15. FreeBSD 2 Версии до 2.2-RELEASE

Таблица 67. FreeBSD 2 До версии 2.2-RELEASE Значения `__FreeBSD_version`

Значение	Версия	Дата	Релиз
119411			2.0-RELEASE
199501	<a href="#">7153</a>	19 марта 1995	2.1-CURRENT
199503	<a href="#">7310</a>	24 марта 1995	2.1-CURRENT
199504	<a href="#">7704</a>	9 апреля 1995	2.0.5-RELEASE
199508	<a href="#">10297</a>	26 августа 1995	2.2-CURRENT до 2.1
199511	<a href="#">12189</a>	10 ноября 1995	2.1.0-RELEASE
199512	<a href="#">12196</a>	10 ноября 1995	2.2-CURRENT до 2.1.5
199607	<a href="#">17067</a>	10 июля 1996	2.1.5-RELEASE
199608	<a href="#">17127</a>	12 июля 1996	2.2-CURRENT до 2.1.6
199612	<a href="#">19358</a>	15 ноября 1996	2.1.6-RELEASE
199612			2.1.7-RELEASE